Representation and learning schemes for argument stance mining.

CLOS, J.

2019

Representation and Learning Schemes for Argument Stance Mining

CLOS, JÉRÉMIE

A thesis submitted in partial fulfilment
of the requirements of Robert Gordon University
for the degree of Doctor of Philosophy

This research programme was carried out
in collaboration with the Department of Computer Science, University of Glasgow

June 2019

Abstract

Argumentation is a key part of human interaction. Used introspectively, it searches for the truth by laying down argument for and against positions. As a mediation tool, it can be used to search for compromise between multiple human agents. For this purpose, theories of argumentation have been in development since the Ancient Greeks in order to formalise the process and therefore remove the human imprecision from it. From this practice the process of argument mining has emerged. As human interaction has moved from the small scale of one-to-one (or few-to-few) debates to large scale discussions where tens of thousands of participants can express their opinion in real time, the importance of argument mining has grown while its feasibility in a manual annotation setting has diminished and relied mainly on a human-defined heuristics to process the data. This underlines the importance of a new generation of computational tools that can automate this process on a larger scale.

In this thesis we study argument stance detection, one of the steps involved in the argument mining workflow. We demonstrate how we can use data of varying reliability in order to mine argument stance in social media data. We investigate a spectrum of techniques, from completely unsupervised classification of stance using a sentiment lexicon, automated computation of a regularised stance lexicon, automated computation of a lexicon with modifiers, and the use of a lexicon with modifiers as a temporal feature model for more complex classification algorithms.

We find that the addition of contextual information enhances unsupervised stance classification, within reason, and that multi-strategy algorithms that combine multiple heuristics by ordering them from the precise to the general tend to outperform other approaches by a large margin. Focusing then on building a stance lexicon, we find that optimising such lexicons using a empirical risk minimisation framework allows us to regularise them to a higher degree than competing probabilistic techniques, which helps us learn better lexicons from noisy data. We also conclude that adding local context (neighbouring words) information during the learning phase of the lexicons tends to produce more accurate results at the cost of robustness, since part of the weights is distributed from the words with a class valence to the contextual words. Finally, when investigating the use of lexicons to build feature models for traditional machine learning techniques, simple lexicons (without context) seem to perform overall as well as more complex ones, and better than purely semantic representations. We also find that word-level feature models tend to outperform sentence and instance-level representations, but that they do not benefit as much from being augmented by lexicon knowledge.

Keywords: Argument Stance Mining ; Natural Language Processing ; Social Media Mining ; Lexicon-Based Methods

Declaration of Authorship

I declare that I am the sole author of this thesis and that all verbatim extracts contained in the thesis have been identified as such and all sources of information have been specifically acknowledged in the bibliography. Parts of the work presented in this thesis have appeared in the following publications:

- J. Clos, N. Wiratunga, J. Jose, S. Massie, and G. Cabanac: Towards Argumentative Opinion Mining in Online Discussions. In: Proceedings of the SICSA Workshop on Argument Mining (2014)
  (**Chapter 5**)
- J. Clos, N. Wiratunga, S. Massie, and G. Cabanac: Shallow techniques for argument mining. In: ECA'15: Proceedings of the European Conference on Argumentation (2016)
  (**Chapter 6**)
- J. Clos and N. Wiratunga: Neural Induction of a Lexicon for Fast and Interpretable Stance Classification. In: International Conference on Language, Data and Knowledge (2017)
  (**Chapter 6**)
- J. Clos and N. Wiratunga: Lexicon Induction from Text for Interpretable Classification. In: 21 International Conference on Theory and Practice of Digital Libraries (2017)
  (**Chapter 6**)
- J. Clos, N. Wiratunga, and S. Massie: Towards Explainable Text Classification by Jointly Learning Lexicon and Modifier Terms. In: IJCAI Workshop on Explainable Artificial Intelligence (2017)
  (**Chapter 6**)

Acknowledgements

Due to the long overdue nature of this thesis, the list of people that I want to thank and acknowledge has grown beyond reason.

The Scottish Informatics and Computer Science Alliance (SICSA) for providing the initial funding for this research, and Prof. Susan Craw for helping me secure it ;

My very patient supervisory team, Prof. Nirmalie Wiratunga, Dr. Stewart Massie and Prof. Joemon Jose for their wisdom and advice, with specific mention to my long-lasting external advisor and friend Dr. Guillaume Cabanac for his mentoring and support ;

My wonderful partner Yoke Yie Chen ;

My extremely supportive family, including my parents Marie-Pascale and Jean-Pierre, my brother Dimitri, as well as the only one of my cats who lived long enough to see me finish this thesis ;

The very helpful people I have met along the way, including those who employed me and helped me support myself. While this research was firstly funded by a SICSA scholarship, it was allowed to continue thanks to the generosity of the people who trusted me and allowed me to mature as a professional and an academic:

– from the Robert Gordon University, Prof. John McCall ;
– from Cognitive Geology, Luke and Fiona Johnson and Alex Shiell ;
– from the University of Nottingham, Dr. Steven Bagley, Dr. Rong Qu, Dr. Jason Atkin and Prof. Robert John.

And finally, last but not least, the very helpful people from the Robert Gordon University who allowed me to push the boundaries of how many times one can ask for an extension (Dr. Eyad Elyan) as well as the colleagues without whom I would be writing this from the cold padded cell of an asylum: Anil, Ahmed, Sadiq, Kyle, Pam, Anjana, Reginald, Ike, Adamu, Pat, Rustam, and many others.

Contents

List of Figures

List of Tables

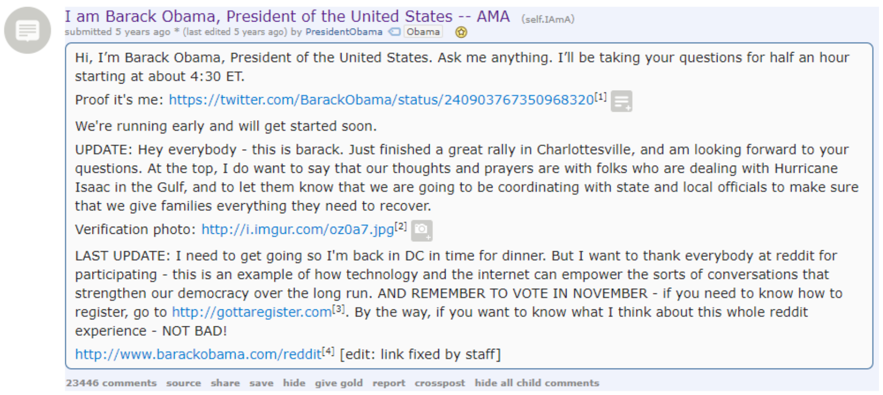
I dedicate this to my parents, my brother, and my wonderful partner.

Chapter 1

Introduction

The advent of the social web has transformed the way we communicate on the Internet. While the World Wide Web was created as a mean to disseminate knowledge so as to prevent information loss in CERN (Berners-Lee, 1989) ; the several improvements in networking infrastructure and the invention of web technologies that were able to make use of the faster speed paved the way for complex web platforms that allowed for not only producer-to-consumer communication, but also for user-to-user interaction. With the popularisation of social websites, the richness of human interactions soon were adapted to their virtual equivalent, bringing casual discussions, debates, and all out profanity-rich "shouting matches" to the comment sections of many websites. However, the architecture of the World Wide Web allowed for bringing these activities to scales that were never explored before, with sometimes tens of thousands communicating with each other on a common topic of interest. For example, the social link aggregator Reddit.com held an informal interview (AMA, for Ask Me Anything, in the Reddit terminology) with Barack Obama which contained more than 20,000 comments (figure 1.1).

This increase in communication has consequentially led to an increase in the data generated by these interactions, which has made it more and more difficult for both human users and algorithms to make use of such data. This had led to a significant amount of research effort being dedicated at automated techniques that are able to make sense of these quantities of data, whether it is mining the political opinions of Twitter users (Pak and Paroubek, 2010), detecting customer tastes to improve product recommendation using Amazon reviews (Chen et al., 2014) or trying to predict the emotional reaction to Facebook posts (Clos et al., 2017).



I am Barack Obama, President of the United States -- AMA (self.IAmA)
submitted 5 years ago * (last edited 5 years ago) by PresidentObama Obama

Hi, I'm Barack Obama, President of the United States. Ask me anything. I'll be taking your questions for half an hour starting at about 4:30 ET.

Proof it's me: https://twitter.com/BarackObama/status/240903767350968320[1]

We're running early and will get started soon.

UPDATE: Hey everybody - this is barack. Just finished a great rally in Charlottesville, and am looking forward to your questions. At the top, I do want to say that our thoughts and prayers are with folks who are dealing with Hurricane Isaac in the Gulf, and to let them know that we are going to be coordinating with state and local officials to make sure that we give families everything they need to recover.

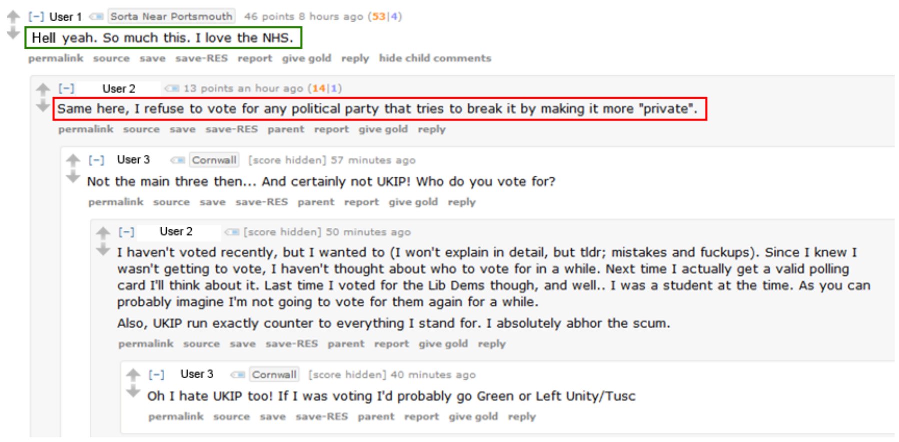
Verification photo: http://i.imgur.com/oz0a7.jpg[2]

LAST UPDATE: I need to get going so I'm back in DC in time for dinner. But I want to thank everybody at reddit for participating - this is an example of how technology and the internet can empower the sorts of conversations that strengthen our democracy over the long run. AND REMEMBER TO VOTE IN NOVEMBER - if you need to know how to register, go to http://gottaregister.com[3]. By the way, if you want to know what I think about this whole reddit experience - NOT BAD!

http://www.barackobama.com/reddit[4] [edit: link fixed by staff]

23446 comments source share save hide give gold report crosspost hide all child comments

FIGURE 1.1: Barack Obama's AMA (Ask Me Anything) on Reddit contained more than 20,000 comments over the few hours of the event.



[–] User 1 Sorta Near Portsmouth 46 points 8 hours ago (53|4)
Hell yeah. So much this. I love the NHS.
permalink source save save-RES report give gold reply hide child comments

[–] User 2 13 points an hour ago (14|1)
Same here, I refuse to vote for any political party that tries to break it by making it more "private".
permalink source save save-RES parent report give gold reply

[–] User 3 Cornwall [score hidden] 57 minutes ago
Not the main three then... And certainly not UKIP! Who do you vote for?
permalink source save save-RES parent report give gold reply

[–] User 2 [score hidden] 50 minutes ago
I haven't voted recently, but I wanted to (I won't explain in detail, but tldr; mistakes and fuckups). Since I knew I wasn't getting to vote, I haven't thought about who to vote for in a while. Next time I actually get a valid polling card I'll think about it. Last time I voted for the Lib Dems though, and well.. I was a student at the time. As you can probably imagine I'm not going to vote for them again for a while.
Also, UKIP run exactly counter to everything I stand for. I absolutely abhor the scum.
permalink source save save-RES parent report give gold reply

[–] User 3 Cornwall [score hidden] 40 minutes ago
Oh I hate UKIP too! If I was voting I'd probably go Green or Left Unity/Tusc
permalink source save save-RES parent report give gold reply

FIGURE 1.2: A discussion tree from Reddit where users discuss United Kingdom politics.

Online discussion forums (Figure 1.2) typically manifest into tree-like structures that are reminiscent of argument trees. Whilst these discussion forums contain a wealth of information related to people's opinions they also include implicit argumentation information. However unlike argument trees any relationship between posts in a discussion tree remains implicit. In recent years there has been considerable interest in harnessing opinionated knowledge that is buried in discussion forums in a variety of domains (e. g., retail, health and politics).

However to-date the main focus of opinion mining has been on mining raw opinion, sentiment and emotion without paying much regard to some of the relational characteristics of user comments. Consider the discussion excerpt presented in Figure 1.2. We consider the outcome of applying the SMARTSA state-of-the-art sentiment analysis system (Muhammad et al., 2013) to argumentative content. The aggregated sentiment of comment 1 (circled in green) is positive with a value of 0.18, while the aggregated sentiment of comment 2 (circled in red) is negative with a value of

-0.23. A casual observation would suggest that the opinion expressed in both comments are not opposed but aligned.

However, a closer examination shows us that there is indeed a discrepancy between the actual opinions expressed, which are similar, and what a sentiment analysis algorithm tells us, which is that comment 1 is positive while comment 2 is negative. This is due to the fact that sentiment analysis algorithms, such as SMARTSA, generally rely only on the aggregated sentiment mined from the terms contained in the sentences, thus considering user comments in a vacuum.

In this thesis, we tackle the challenge of argument stance mining by exploring the use of argumentation constructs for argumentative stance mining in social media. We explore a series of techniques that make a progressively heavier use of context and machine supervision, drawing from unsupervised heuristics to recurrent neural networks.

1.1 Applications of argumentative stance mining

Debating and arguing has always been a fundamental process of human cognition and interaction. Only recently have those debates moved from the small scale of panel discussions and expert-driven arguments to the large-scale, comparatively messy process of online argumentation. Traditionally, when a user seeks to understand a debate, it is a trivial matter for them to simply ask multiple participants of the debate for their point of view, and the facts they are basing them on, before synthesising their own conclusion. That process does not work when the participants are counted in the thousands and the facts, reasoning patterns and opinions in the tens of thousands.

Considering that online user discussions are already structured as interaction graphs, being able to instantly know how much and where people are agreeing and disagreeing would provide systems with valuable information to summarise a debate. One could, for instance, use the average number of agreement and average number of disagreement as a social signal that allows the system to rank and organise user comments by level of controversiality. Conversely, it would allow, given enough data to begin with, for a user to know the likely response of a community to a given idea, argument or statement by simply looking at similar cases from the recent past.

Finally, being able to provide an argumentative stance between pairs of argument would simply allow experts to better understand the finer points of a debates. This has led to the creation of multiple software tools to analyse arguments in different types of text in a guided way: Araucaria (Reed and Rowe, 2004), OVA+ (Reed et al., 2014) and the Argument Analysis Wall (Bex et al., 2013), both as a way to study argumentation in the wild and to build better systems able to analyse arguments in more standard documents.

1.2 Related research fields

Argument stance mining draws upon the research of many neighbouring fields that have dealt with data of the same nature, such as argumentation mining, sentiment analysis, text classification (supervised and unsupervised) and information retrieval. This section focuses on relating all these fields to argument stance mining and contextualising them with respect to this thesis.

Argument mining Argument mining as a field tends to focus on detecting the presence of arguments in formal or informal text. As such it is often used as a precursor of argument stance mining. Traditional approaches pioneered by Palau and Moens (2009) make heavy use of supervised classifiers such as the Maximum Entropy Classifier (detailed in Jaakkola et al. (2000) and Support Vector Machines (detailed in Hearst et al. (1998)) trained on legal documents, as well as context-free grammars (detailed in Figures 1.3 and 1.4) to detect the internal structure of arguments before classifying them.

In this thesis we focus on informal arguments. In order to make the task feasible, and because informal argumentation does not tend to focus on following a strict representation, we make the simplifying assumption of considering the entire posts as potential arguments, and then attempt to detect their argumentative stance with respect to their parent post.

Sentiment analysis Sentiment analysis is the field focusing on detecting the strength and polarity of sentiment in informal or formal text. It is often used in social media analysis in order to provide a high level overview of what users think about a topic, whether it is a political topic/idea, a product, or an event. Sentiment and stance are two highly related topics, with

$$T \Rightarrow A^{+}D$$

$$A \Rightarrow \{A^{+}C|A^{*}CnP^{+}|Cns|A^{*}sr_{c}C|P^{+}\}$$

$$D \Rightarrow r_{c}f\{v_{c}s|.\}^{+}$$

$$P \Rightarrow \{P_{verbP}|P_{art}|PP_{sup}|PP_{ag}|sP_{sup}|sP_{ag}\}$$

$$P_{verbP} = sv_{p}s$$

$$P_{art} = sr_{art}s$$

$$P_{sup} = \{r_{s}\}\{s|P_{verbP}|P_{art}|P_{sup}|P_{ag}\}$$

$$P_{ag} = \{r_{a}\}\{s|P_{verbP}|P_{art}|P_{sup}|P_{ag}\}$$

$$C = \{r_{c}|r_{s}\}\{s|C|r_{c}P_{verbP}\}$$

$$C = s^{*}v_{c}s$$

FIGURE 1.3: Rules of the Context Free Grammars in Palau and Moens (2009).

T	General argumentative structure of legal case.
A	Argumentative structure that leads to a final decision of the factfinder $A = \{a_1, ..., a_n\}$, each a_i is an argument from the argumentative structure.
D	The final decision of the factfinder $D = \{d_1, ..., d_n\}$, each d_i is a sentence of the final decision.
P	One or more premises $P = \{p_1, ..., p_n\}$, each p_i is a sentence classified as premise.
C	Sentence with a conclusive meaning.
n	Sentence, clause or word that indicates one or more premises will follow.
s	Sentence, clause or word neither classified as a conclusion nor as a premise ($s! = \{C\|P\}$).
r_c	Conclusive rhetorical marker (e.g. therefore, thus, ...).
r_s	Support rhetorical marker (e.g. moreover, furthermore, also, ...).
r_a	Contrast rhetorical marker (e.g. however, although, ...).
r_{art}	Article reference (e.g. terms of article, art. para. ...).
v_p	Verb related to a premise (e.g. note, recall, state,...).
v_c	Verb related to a conclusion (e.g. reject, dismiss, declare, ...).
f	The entity providing the argumentation (e.g. court, jury, commission, ...).

FIGURE 1.4: Symbols of the Context Free Grammars in Palau and Moens (2009).

sentiment being often seen as a proxy for stance towards a global topic. In this thesis, we explore the use of sentiment as a proxy for stance using a series of heuristics. We also use sentiment as a feature model for supervised stance classification.

Supervised text classification Supervised text classification focuses on building a supervised model that is able to learn from data and classify new data. Its main drawback is the need for a lot of training data. Its advantage is how powerful and accurate it can be given the right representation and the right algorithm. In this thesis we explore supervised classifiers to classify the argument stance of informal posts and focus on representation schemes in order to enhance argument stance classification.

Unsupervised text classification Unsupervised text classification is similar to supervised text classification but replaces the training data with heuristic rules that have been designed by humans. Its advantage is that it is not affected by the absence of data, and its drawback is that human heuristics are plagued by human bias and are never as powerful as statistical correlations. In this thesis we explore unsupervised classifiers to classify the argument stance of informal posts, particularly unsupervised techniques that make use of heuristics to detect the target of local sentiment such as the Sentiment Surface algorithm.

Information retrieval Information Retrieval (IR) focuses on the storage, indexing and retrieval of documents given an information need formulated by a user (implicitly or explicitly). A large part of the focus of information retrieval as a field is term weighting, in order to represent documents not by all of their terms in an equal manner, which would give an unwarranted over-importance to documents which are very long, but by their most representative terms (both representative at a document level and at a corpus level). In this thesis we investigate IR-inspired term weighting techniques used to enhance supervised and unsupervised classification of argument stance in social media posts.

1.3 Scope of the research

In this section we detail the scope of our research, first by detailing our research motivation and how it fits in the scientific landscape of social media mining and debate analysis, and then moving on to the research objectives that stem from this motivation.

1.3.1 Research motivation

The classification pipeline (illustrated in figure 1.5) is a multi-step process. First, the complete dataset is split into two datasets: a hold-out evaluation dataset, typically composed of 20% of the data (using a stratified sampling strategy to not bias the evaluation), and a training dataset that is used for tuning model parameters. We then use $K-$Fold cross-validation (with K typically being 5 or 10) to select the optimal parameters for our model. This process works on the following steps: first, the training dataset is split into Kpartitions using a stratified sampling strategy. Then, every one of these partitions is used as a test set with a model that is trained on the remaining $K-1$ partitions. This process produces K performance scores, which are averaged to produce an overall score for the given hyperparameters. Then, either the parameters which are issued from the fold that produced the best performance score or a weighted combination of the parameters of all folds using the performance metrics as weights is used to produce the optimal model, which is finally evaluated against the hold-out evaluation dataset that was held out in the first step.

This process requires that the data be represented in a way that allows the learning algorithm to perform the training stage in an efficient and effective way. This is typically called the feature engineering phase.

Traditional stance classification feature engineering techniques lean in either one of the following extremes: on one hand, a complex representation model that uses up to a few hundreds manually crafted features in order to enhance the learning process using human common sense, and on the other hand an overly simplistic feature model that is built only on raw term frequencies (bag of word models) which are fed to a complex classifier. Both of those approaches present crucial flaws:

- Complex feature models are highly platform specific, and thus any knowledge learned from them is specific to the data on which is it used and cannot be ported to alternative platforms.
- Bag of word models fed into complex classifiers are turned into black boxes in which it is virtually impossible to inspect a classification a posteriori and justify it by looking at the model (explanation by transparency).

In this thesis we aim to bridge that gap by developing a model that is simple, knowledge-light (in that it doesn't require specially hand-crafted features) and competitive with common black box models.

1.3.2 Research questions

In order to bridge this gap, we derive a list of research questions that will guide our investigation throughout this thesis. Each of those research questions builds upon the potential findings of the previous one.

RQ1. Can we use the direct conversational context to improve classification accuracy by helping a naive classifier focus on relevant terms? Concretely, can we use our knowledge of the

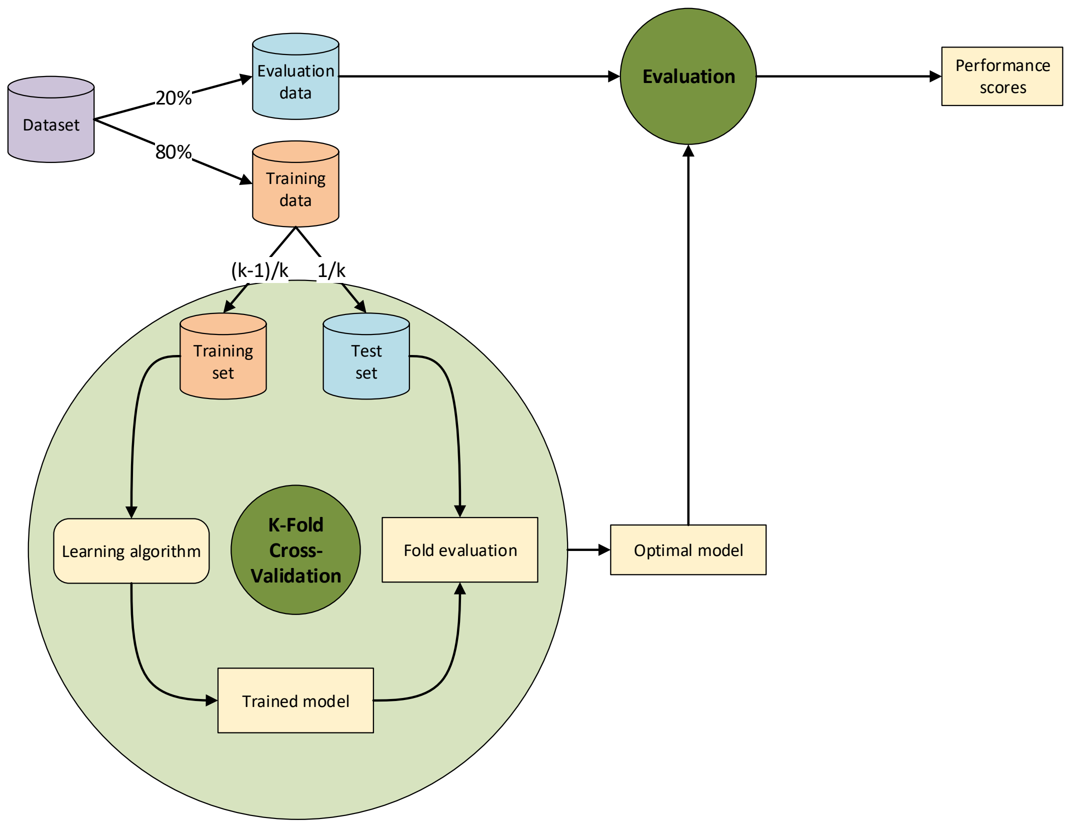


FIGURE 1.5: The supervised classification process

direct context of a conversation to improve the performance of a naive classifier? This research question is addressed in chapter 5.

RQ2. Can we use empirical risk minimisation techniques to learn a regularised stance classification lexicon? This research question is addressed in chapter 6.

RQ3. Can we leverage the neighbourhood of words to learn a context-sensitive classification lexicon? This research question is also addressed in chapter 6.

RQ4. Can class-sensitive features learned on a corpus enrich an existing term representation and allow it to better represent argumentative content? This is addressed in chapter 7.

RQ5. At what level of granularity (words, sentences, or entire instance) does lexicon enhancement performs better when applied to standard representation schemes? This question is also answered in chapter 7.

These research questions lead us to formulate a set of research objectives which are detailed in the following section.

1.3.3 Research objectives

We formulate the following four research objectives in order to answer our research questions.

- ◇ To develop an unsupervised classification algorithm that leverages conversational context and existing resources from the field of opinion mining.
- ◇ To develop a novel approach to learn an argument lexicon from weakly labelled data.
- ◇ To develop a novel learning algorithm to jointly learn an argument lexicon and its modifiers.
- ◇ To develop a representation scheme using the argument lexicon as a feature extractor that can leverage the temporal nature of text.

1.4 Contributions

The contributions in this thesis are multiple, and target different research questions and research objectives mentioned in the previous sections.

Firstly, we contribute a novel stance classification dataset built on a large scale mining of the Reddit social link aggregation website, and a distant labelling strategy. This dataset is used to experiment on the transfer learning capabilities of regularised lexicons and lexicon-enhanced representations.

Secondly, we contribute a series of unsupervised stance classification algorithms. Those algorithms allow us to investigate the use of complex heuristics in order to approximate stance classification using approaches inspired from the already well explored field of opinion mining. We explore a set of techniques that make use of opinion as a way to approximate stance by augmenting it with additional heuristics, either by using simple lexicon classification or by utilising natural language processing heuristics.

Thirdly, we contribute two approaches to learn regularised lexicons from data of varying quality. LEXICNET is the first of those approaches, using a regularised gradient-based optimisation process in order to learn a lexicon that is robust to errors in the training data. It is then extended in the form of ReLexNet, which adds contextual terms in order to offer a richer way to model argumentative language.

Finally, we offer a series of representation schemes that use the argument stance lexicons previously computed (LexicNet and ReLexNet) in order to enrich semantic and/or statistical representations of the data, before feeding it into traditional machine learning algorithms. Those representation schemes operate at multiple levels of granularity (representing words, representing sentences, and representing the whole instance) and use different lexicons, in order to provide a comparative study of those effects.

1.5 Overview of the thesis

In this section we will detail an overview of the thesis and the mapping between each chapter and the corresponding contributions.

In chapter 2, we present a review of the literature in argument stance mining, starting with argumentation in social media, algorithms to detect and process arguments and modern methods for stance classification. The first part of this chapter is dedicated to explaining the main theories of argumentation and how they relate to stance detection. The second part of this chapter is divided into two sections. The first section focuses on the different corpora built by the stance detection and social media mining communities; the second section gives a detailed overview of the literature in methods for stance classification, separating the related works into unsupervised, supervised shallow, and supervised deep methods to illustrate the evolution of the field from heuristic-based approaches to deep neural networks.

In chapter 3, we introduce the background knowledge necessary for the rest of the thesis. We begin with a detailed explanation of the general field of text classification by explaining its theoretical underpinnings, the standard methodologies for evaluating its algorithms, and an explanation of two types of methods popular in the social media mining literature: supervised classifiers and lexicon-based methods. We finally present standard evaluation methodologies that will be used to validate our research objectives against the state of the art.

In chapter 4 we present our evaluation methodology, starting with the datasets collected for this research and finishing with the statistical testing procedure used to formally test the significance of our results.

In chapter 5, we present our first contribution: three novel unsupervised argument stance mining algorithms using sentiment-as-a-proxy heuristic-based methods. The first one, SP-ADAPT, relies on existing sentiment analysis resources along with adaptive thresholding to approximate the stance of the user. The second one, USS+, extends the existing Unsupervised Sentiment Surface with different failure modes to account for malformed data that is typical of social media corpora. Finally the third one, SP-COSTS, uses relational knowledge between a post and its parent post to perform term selection and improve the classification stage, while relying on different failure modes for when the heuristics do not apply.

In chapter 6, we present our second contribution: an approach to learning lexicons, which we extend to joint learning of lexicon and modifier terms. We formalise lexicon learning in an empirical risk minimisation framework, and then apply gradient-based optimisation in order to induce optimal lexicon weights from very small datasets (the LEXICNET algorithm). This allows us to apply different regularisation methods that make our lexicon more robust to overfitting. We extend our lexicon learning method to take into account modifier terms using a simple additive gating model (the RELEXNET algorithm) along with a recurrent network-inspired architecture.

In chapter 7, we present our third and final contribution: a representation scheme that preserves the temporal structure of posts using lexicons as feature extractors. Our representation models the temporal nature of posts by keeping the structure of discourse, which allows us to train a supervised classification algorithm not only on the semantic vectors of terms (traditional embeddings) but also on class vectors of terms. We build a set of representations at multiple levels of granularity in order to evaluate its usefulness on a diverse class of datasets.

Finally in chapter 8, we present the conclusion to this thesis. We re-state our research questions and the interpretation of our experimental results and use them to outline potential future directions of our research.

Chapter 2

Literature Review

Argumentation theory is a highly interdisciplinary domain, spanning a period of time starting with the study of rhetoric in ancient Greece to, more recently, the use of abstract argumentation for collaborative decision-making systems. This multidisciplinary status allows us to examine it under multiple lenses. Philosophers, linguists, and computer scientists have all built their own theories of argumentation with the goal of modelling specific phenomena (e. g., multi-agent negotiation). In this thesis we focus on the role of argumentation as a formalisation of the interaction between multiple users in a social media context, and therefore select Pragmatic Argumentation Theory (van Eemeren, 1993, Hutchby, 2013) as our underlying target.

Pragmatic Argumentation Theory defines an argument as an opinionated piece of text which can arise in the presence of two elements: (1) a **target**, being some other action by another actor, has been called out as being problematic ; (2) a **stance**, i.e. whether it is supporting or attacking the target regarding the problematic element, has been stated. This simplification of the argumentation representation helps with applying those mappings to social media content, by relaxing the definition of an argument. Indeed, users of social media tend to prefer a highly informal language and as such are less likely to worry about formulating a well-formed argument.

We thus study the classification of argument stance in a Pragmatic Argumentation Theory setting, where the dialogue between users can be viewed as a graph, and more specifically a tree, and the stance of each contribution can be seen as an edge that is present between two vertices of that graph (local stance), an edge that is present between one vertex and the root of the tree (global

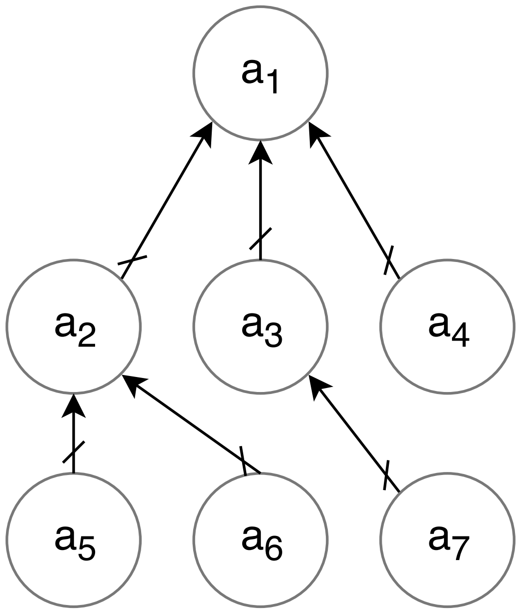


FIGURE 2.1: Graphical example of a debate graph. The vertices are arguments, while the arcs are interactions between arguments, where a crossed arc between an argument a_i and an argument a_j means that argument a_i attacks argument a_j, and therefore an observer who believes a_i to be true must believe a_j to be false in order to maintain consistency.

stance). Argument stance detection thus consists in the classification of such edges, and lies on existing research in stance detection, sentiment analysis, and general text classification.

A detailed analysis of the relevant theories can be found in appendix B.

2.1 Methods for Argument stance detection

In this section we will detail datasets and methods for stance classification, both in the context of argument mining and in more general terms as it relates to this thesis. Being bound to the study of dialogical argumentation, the task of stance classification has multiple definitions, according to the use case it is attached to. We can broadly divide them into two different cases: (1) global stance classification, and (2) local stance classification.

Global stance classification is the study of the stance of a claim with respect to an overarching topic. While it is possible that global stance can be inferred from the network of local stances, the difference in granularity makes it almost impossible: the root topic can contain a multitude of details, and any further comment can agree or disagree with any arbitrary subset of those details. For example, in Figure 2.1 depicting a graphical example of a debate where a_1 is the root argument, which can be thought of as the core topic of the discussion (e. g., , "I am for LGBTQ rights" is both an argument in favour of LGBTQ rights and a topic of discussion), the global stance of argument a_5 is the stance of a_5 with respect to the argument a_1. Because a_5 attacks argument a_2, which itself is attacking argument a_1, we can infer that it is likely that a_5 has a positive stance towards a_1 and therefore the topic at hand.

Local stance classification is the study of the stance of a claim with respect to the claim it is responding to. For example, in figure 2.1 depicting a graphical example of a debate where a_1 is the core topic of the discussion, the local stance of argument a_5 is the stance of a_5 with respect to the its most immediate parent a_2, and is therefore a negative stance.

In this section we will go into more details about the methods that have been used in the literature to perform stance classification. This section is organised in the following way: first, we will give an overview of unsupervised approaches that can be applied to stance classification. We will then give more details on supervised approaches to stance classification, starting with shallow classifiers and feature engineering-based approaches and ending with newer deep learning-based methods.

2.1.1 Unsupervised approaches to stance classification

Because resources for unsupervised stance classification are scarce, this section will merge both original approaches from the stance classification literature as well as methods adapted from the closely related field of sentiment analysis.

Unsupervised stance classifiers The sentiment surface feature representation was built as a baseline algorithm to extract significant features in Wang and Cardie (2014), based on previous work by Hassan et al. (2010) where the feature extractor extracts the minimum, mean and maximum distance (in number of words) between words of negative polarity and second person pronouns. Those features are then fed into a support vector machine to provide the classification. Wang and Cardie (2014) find that the sentiment surface representation is a strong baseline, especially in identifying the neutral class where it is barely significantly beaten by the approaches of the authors. Because of the scarcity of unsupervised stance classification algorithms, we leverage the sentiment surface heuristic as a baseline algorithm along with sentiment analysis-inspired methods.

Ghosh et al. (2018) take a different approach to detect the stance of a statement, focusing on syntactic rules and aspect extraction in order to provide a representation for each person's position. Once they have determined the positions with respect to each aspect of the conversation, they formulate the classification of the overall stance as an integer linear programming problem and optimise an objective function in order to provide a consistent stance classification for the entire set of statements. Their final approach, which combines an improved lexicon, a co-reference resolution, an aspect importance weighting and a negation handling procedure outperforms several baselines on the MPQA political debate dataset (Somasundaran and Wiebe, 2010a).

Adapting sentiment analysis for stance classification Sentiment analysis is the computational task of determining the underlying sentiment (positive, negative, or neutral) within an extract of text (Pang and Lee, 2008). Unsupervised approaches to sentiment analysis focus on the use of sentiment lexicons. Due to the relatedness between sentiment and stance, they can be adapted to the context of stance classification. Lexicons are a classification model where instead of using every word as a feature, only certain words which are deemed to be sentiment-bearing are considered. A list of those words is called a sentiment lexicon. After detecting sentiment-bearing words within an arbitrary textual content, a computational aggregation rule is used to determined whether the overall content is deemed positive or negative.

There are a number of freely available sentiment lexicons, the three most popular being:

- The **General Inquirer lexicon** (Stone et al., 1966) is composed of 1,915 positive words and 2,291 negative words.
- The **MPQA Subjectivity lexicon** (Wilson et al., 2005) is composed of 2,718 positive and 4,912 negative words from multiple sources such as the General Inquirer lexicon, a list of sentiment-bearing adjectives (Hatzivassiloglou and McKeown, 1997), and a list of hand-labelled subjective words and phrases (Riloff and Wiebe, 2003).

- The **SentiWordNet lexicon** (Baccianella et al., 2010) is a lexicon induced from the WordNet (Miller, 1995) ontology. It contains over 38,000 words from WordNet to which it assigns a positive, negative and objective score.

Several additional works focus on optimising the use of such lexicons, either by fine-tuning them, or by modifying their scores based on contextual clues. For example, Muhammad et al. (2016) enhance an existing general purpose lexicon (SentiWordNet) for a domain-specific application by adding additional vocabulary and sentiment scores.

Sentiment lexicons typically output scores based on which the classification is made, they can be trivially adapted to stance classification by changing the threshold at which the classification is made.

2.1.1.1 Conclusions from the unsupervised stance classification literature

We can see that there is no clear unsupervised stance classification literature to speak of, where most of the unsupervised techniques are either built from external resources such as the graphical structure of the discussion, or directly ported from the sentiment analysis literature. This points towards a need for more robust and specialised approaches that can deal with a variety of datasets without requiring a learning phase, which we will develop in chapter 4.

2.1.2 Supervised approaches to stance classification

Supervised text classification is the process of automatically discovering a function that maps a piece of text, formally represented as a set of numerical features, to one or more classes. It does so by observing a dataset composed of a set of training instances X and a set of their corresponding labels Y, and iteratively refining its model (the mapping function) by minimising its propensity to commit classification errors. Supervised approaches to stance classification can be put into two broad categories, which we will discuss in this section: a) **feature engineering** and **shallow classifier-based stance classifiers** focus on manually extracting an enormous number of relevant features and using state-of-the-art shallow classifiers such as Support Vector Machines and the Maximum Entropy Classifier in order to build a model ; b) **deep learning-based classifiers** focus on jointly learning features and a classifier from the data itself using deep neural networks.

2.1.2.1 Feature engineering and shallow stance classifiers

Because of the novelty of deep learning models, most of the literature of the past decade has focused on traditional shallow models for stance classification, which will be explored in this section. Early work has been focused on structured debates (Janin et al., 2003, Carletta et al., 2005, McCowan et al., 2005), which have the distinct advantage of being easy to label. However, such data does not easily map to modern social media conversations, in two ways:

- Transcribed structured debates typically involve a reduced set of participants in a long-form debate on a pre-chosen topic of discussion. Conversely, social media debates typically involve a large number of participants who often do not follow up on their statements, which creates a large number of branches for each argument.
- Transcribed structured debates typically use a more formal vocabulary than social media debates.

Later work on stance classification involved more and more use of social media as a source of data, from Twitter and Weibo debates Mohammad et al. (2016), Xu et al. (2016) to debate-focused internet forums Walker et al. (2012a), Ferreira and Vlachos (2016), Rosenthal and McKeown (2015). This has shifted the focus on the research from static techniques that rely on mining the correct structure of arguments to a pervasive use of feature models, natural language processing techniques, and machine learning.

Anand et al. (2011) use a set of arbitrary text-focused features (unigrams, bigrams, punctuation marks, syntactic dependencies and dialogic structure) to build a rule-based classifier. However the difference between systems using only unigrams and systems using more complex features such as generalised dependencies was not found to be significant. Walker et al. (2012b) propose the use of dialogic relations in order to enrich the content-based features using the MaxCut algorithm, which is used to exploit the structure of the reply graph. While the core of their approach is feature engineering-based, they show a significant improvement over a purely content-based approach by exploiting this graphical structure.

Somasundaran and Wiebe (2010a) and Faulkner (2014) focus on detecting stance-taking language as a way of generating features for their classifiers. While Somasundaran and Wiebe (2010a) take the route of pairing stance-taking expressions to expressions of sentiment (positive or negative) using a lexicon of argument trigger expressions in order to build a feature model for their Support Vector Classifier, Faulkner (2014) feed the stance language directly into their classifier (Support Vector Machines and Multinomial Naive Bayes).

Ahmed and Xing (2010) use a Latent Dirichlet Allocation inspired approach to model the generative process that generated each word as mixture of two distributions: a distribution over topics and a distribution over ideologies. They then use collapsed Gibbs sampling to infer the parameters of the mixture, leading to discover the most likely ideological stance with respect to the subject at hand.

Some works, such as Hasan and Ng (2013a), put constraints on consecutive posts in order to transform the problem from a classification task to a sequence labelling task. This approach is similar to Rosenthal and McKeown (2015), who choose to study the role of conversational structure using a set of features related to the thread structure: whether the post is the root of the discussion (Boolean feature), whether the reply was by the same author (Boolean feature), the distance in number of posts from the root of the discussion (numerical feature) and the number of sentences in the post (numerical features).

Because of the SemEval Task 6 challenge on stance detection (Mohammad et al., 2016), a spike of works on the topic of stance detection on a Twitter corpus appeared in 2016. In those approaches, the most successful used complex deep learning models, which we will discuss in section 2.1.2.2, while other high performing ones focused on clever feature engineering along with very diverse classifiers such as Logistic Regression, Multinomial Naive Bayes, Decision Trees, Maximum Entropy classification, Support Vector Machines, and Ensemble learners. Most of those approaches relied on Twitter specific-features such as hashtags (Zhang and Lan, 2016, Tutek et al., 2016, Krejzl and Steinberger, 2016), social media-specific features such as emoticons (Zhang and Lan, 2016, Tutek et al., 2016), but also on more complex features such as latent topics discovered through Brown clustering (Zhang and Lan, 2016), matrix factorisation (Elfardy and Diab, 2016) or frequency analysis (Wojatzki and Zesch, 2016a) and features derived from popular word embedding approaches such as Word2Vec (Zhang and Lan, 2016, Tutek et al., 2016), Sentiment2Vec (Zhang and Lan, 2016) and GloVe (Igarashi et al., 2016, Bøhler et al., 2016). In the case of Igarashi et al. (2016), the authors found that a Support Vector Machine model

N-grams/ tokens	Unigrams	Hillard et al. (2003) ; Moens et al. (2007) ; Germesin and Wilson (2009) ; Palau and Moens (2009) ; Wang et al. (2011) ; Abbott et al. (2011) ; Hassan et al. (2012) ; Walker et al. (2012b) ; Yin et al. (2012a) ; Hasan and Ng (2013b) ; Hasan and Ng (2013a) ; Misra and Walker (2017) ; Wang and Cardie (2014) ; Rosenthal and McKeown (2015) ; Walker et al. (2012c) ; Sridhar et al. (2015) ; Elfardy and Diab (2016) ; Zhang and Lan (2016) ; Sun et al. (2016) ; Liu et al. (2016) ; Patra et al. (2016) ; Misra et al. (2016) ; Li et al. (2016) ; Mohammad et al. (2017) ; Tutek et al. (2016) ; Igarashi et al. (2016) ; Krejzl and Steinberger (2016)
	Bigrams	Moens et al. (2007) ; Palau and Moens (2009) ; Abbott et al. (2011) ; Hassan et al. (2012) ; Walker et al. (2012b) ; Yin et al. (2012a) ; Hasan and Ng (2013b) ; Hasan and Ng (2013a) ; Misra and Walker (2017) ; Wang and Cardie (2014) ; Rosenthal and McKeown (2015) ; Walker et al. (2012c) ; Sridhar et al. (2015) ; Elfardy and Diab (2016) ; Zhang and Lan (2016) ; Sun et al. (2016) ; Bøhler et al. (2016) ; Wojatzki and Zesch (2016a) ; Misra et al. (2016) ; Li et al. (2016) ; Mohammad et al. (2017) ; Tutek et al. (2016)
	Trigrams	Moens et al. (2007) ; Palau and Moens (2009) ; Misra and Walker (2017) ; Rosenthal and McKeown (2015) ; Elfardy and Diab (2016) ; Zhang and Lan (2016) ; Sun et al. (2016) ; Wojatzki and Zesch (2016a) ; Mohammad et al. (2017) ; Tutek et al. (2016)
	4-grams	Zhang and Lan (2016)
	2-char	Mohammad et al. (2017) ; Tutek et al. (2016)
	3-char	Bøhler et al. (2016) ; Tutek et al. (2016)
	{3,4,5}-char	Zhang and Lan (2016) ; Mohammad et al. (2017)
	Word pairings	Moens et al. (2007) ; Palau and Moens (2009)
	Punctuation	Moens et al. (2007) ; Abbott et al. (2011) ; Yin et al. (2012a) ; Misra and Walker (2017) ; Rosenthal and McKeown (2015) ; Zhang and Lan (2016) ; Wojatzki and Zesch (2016a) ; Mohammad et al. (2017)
	Language model perplexity	Hillard et al. (2003) ; Hahn et al. (2006)

TABLE 2.1: Features from the stance classification literature: n-grams and tokens

provided with a carefully crafted feature model could perform extremely close to a Convolutional Neural Network. However, considering the amount of work in crafting such feature models and then removing a significant portion of them through feature selection, it could be argued that the computational advantages of a shallow algorithm are not that significant.

For the sake of completeness, we illustrate the set of features that were the most used in the literature in tables 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, and 2.7, categorising them along 13 different themes:

- **N-grams/tokens** (Table 2.1) refers to features which are extracted by an analysis of the raw text. They are often followed by a feature selection step which trims the least used

Hashtags	Hashtag unigrams	Zhang and Lan (2016) ; Mohammad et al. (2017) ; Tutek et al. (2016) ; Krejzl and Steinberger (2016)
	Hashtag bigrams	Zhang and Lan (2016) ; Krejzl and Steinberger (2016)
Initial n-gram	First unigram	Hillard et al. (2003) ; Abbott et al. (2011) ; Hasan and Ng (2013a) ; Walker et al. (2012c) ; Sridhar et al. (2015) ; Krejzl and Steinberger (2016)
	First bigram	Abbott et al. (2011) ; Hasan and Ng (2013a) ; Walker et al. (2012c) ; Sridhar et al. (2015) ; Krejzl and Steinberger (2016)
	First trigram	Abbott et al. (2011) ; Hasan and Ng (2013a) ; Walker et al. (2012c) ; Sridhar et al. (2015) ; Krejzl and Steinberger (2016)
Final n-gram	Last unigram	Germesin and Wilson (2009) ; Wang and Cardie (2014)
	Last bigram	Germesin and Wilson (2009) ; Wang and Cardie (2014)
	Last trigram	Germesin and Wilson (2009) ; Wang and Cardie (2014)

TABLE 2.2: Features from the stance classification literature: hashtags and positional markers.

Syntactic parsing	Dependencies	Moens et al. (2007) ; Palau and Moens (2009) ; Abbott et al. (2011) ; Hassan et al. (2012) ; Walker et al. (2012b) ; Hasan and Ng (2013b) ; Hasan and Ng (2013a) ; Wang and Cardie (2014) ; Walker et al. (2012c) ; Sridhar et al. (2015) ; Zhang and Lan (2016) ; Sun et al. (2016) ; Patra et al. (2016) ; Misra et al. (2016) ; Igarashi et al. (2016)
	Part of speech tagging	Wang and Cardie (2014) ; Rosenthal and McKeown (2015) ; Zhang and Lan (2016) ; Sun et al. (2016) ; Misra et al. (2016) ; Mohammad et al. (2017) ; Igarashi et al. (2016)
	POS-generalised dependencies	Abbott et al. (2011) ; Hassan et al. (2012) ; Walker et al. (2012b) ; Hasan and Ng (2013b) ; Hasan and Ng (2013a) ; Walker et al. (2012c) ; Sridhar et al. (2015)
Durational	# sentences	Misra and Walker (2017) ; Rosenthal and McKeown (2015) ; Walker et al. (2012c) ; Sridhar et al. (2015)
	# words/tokens	Hillard et al. (2003) ; Galley et al. (2004) ; Hahn et al. (2006) ; Germesin and Wilson (2009) ; Wang et al. (2011) ; Yin et al. (2012a) ; Misra and Walker (2017) ; Wang and Cardie (2014) ; Walker et al. (2012c) ; Sridhar et al. (2015) ; Sun et al. (2016) ; Bøhler et al. (2016) ; Krejzl and Steinberger (2016)
	# characters	Walker et al. (2012b) ; Yin et al. (2012a) ; Misra and Walker (2017) ; Walker et al. (2012c) ; Sridhar et al. (2015) ; Bøhler et al. (2016)

TABLE 2.3: Features from the stance classification literature: dependency-parsing-based and durational features

Counts	Repeated punctuation	Walker et al. (2012b) ; Hasan and Ng (2013b) ; Wang and Cardie (2014) ; Walker et al. (2012c) ; Sridhar et al. (2015) ; Bøhler et al. (2016)
	# Repeated words	Germesin and Wilson (2009)
	# Punctuation marks	Palau and Moens (2009) ; Wang et al. (2011)
	Word lengthening	Rosenthal and McKeown (2015) ; Zhang and Lan (2016) ; Bøhler et al. (2016) ; Mohammad et al. (2017)
	# Misspelled words	Tutek et al. (2016)
	Repeated vowels	Tutek et al. (2016)
	Upper-case words	Wang and Cardie (2014) ; Zhang and Lan (2016) ; Bøhler et al. (2016) ; Mohammad et al. (2017) ; Tutek et al. (2016)
	Average word length	Palau and Moens (2009) ; Walker et al. (2012b) ; Walker et al. (2012c) ; Tutek et al. (2016)
	Sentence length	Palau and Moens (2009) ; Wang et al. (2011) ; Walker et al. (2012c)

TABLE 2.4: Features from the stance classification literature: specific counts and text statistics

n-grams in order to reduce the dimensionality of the classification problem. It also contains character-based n-grams, such as 2-char (sequences of 2 characters), 3-char (sequences of 3 characters) and {2,3,4}-char (sequences of 2, 3 or 4 characters), as well as exhaustive word pairings (an exhaustive combination of all possible bigrams, followed by feature selection), treatment of punctuation signs as special tokens, and statistics derived from the language model derived from the text.

- **Hashtags** (Table 2.2) refers to special n-grams which are marked by the user/debater as being topical, using a specific symbol (#). They are treated either as unigrams (the entire hashtag is a token) or as bigrams (if multiple hashtags follow each other).
- **Initial n-grams** (Table 2.2) is a positional n-gram, where the first 1, 2 or 3 terms are extracted and given special importance. It is built on the assumption that some indicators of the argumentative stance can be found in the beginning of the text.

Lexicons and patterns	Generic lexicon, keywords	Hillard et al. (2003) ; Moens et al. (2007) ; Germesin and Wilson (2009) ; Palau and Moens (2009) ; Walker et al. (2012b) ; Hasan and Ng (2013b) ; Misra and Walker (2017)
	Negation	Wang et al. (2011) ; Walker et al. (2012b) ; Wang and Cardie (2014) ; Rosenthal and McKeown (2015) ; Zhang and Lan (2016) ; Bøhler et al. (2016) ; Wojatzki and Zesch (2016a)
	Acquiescence	Wang et al. (2011)
	Subjectivity lexicon	Misra and Walker (2017)
	Stance lexicon	Rosenthal and McKeown (2015) ; Wojatzki and Zesch (2016a) ; Krejzl and Steinberger (2016)
	Modal verbs	Wojatzki and Zesch (2016a)
	Emoticons	Yin et al. (2012a) ; Rosenthal and McKeown (2015) ; Zhang and Lan (2016) ; Mohammad et al. (2017) ; Tutek et al. (2016)
	Foul words	Yin et al. (2012a)
	Hedging {1,2,3}-grams	Misra and Walker (2017) ; Wang and Cardie (2014)
	LIWC lexicon	Abbott et al. (2011) ; Walker et al. (2012b) ; Hasan and Ng (2013b) ; Rosenthal and McKeown (2015) ; Walker et al. (2012c) ; Sridhar et al. (2015) ; Elfardy and Diab (2016) ; Misra et al. (2016)
	Adverbs	Moens et al. (2007) ; Palau and Moens (2009)
	Verbs	Moens et al. (2007) ; Palau and Moens (2009)
	Main verb tense/type	Palau and Moens (2009)
	Sentence patterns	Palau and Moens (2009) ; Wang et al. (2011)
	Subject type	Palau and Moens (2009)
	Accommodation	Rosenthal and McKeown (2015)
	Ends in question mark	Rosenthal and McKeown (2015)
	Ends in exclamation mark	Bøhler et al. (2016)

TABLE 2.5: Features from the stance classification literature: extracted from lexicons and lists of patterns

- **Final n-grams** (Table 2.2) is the converse of initial n-grams, where we assume that the last 1, 2 or 3 terms are more indicative of the stance than the rest of the text, and therefore we extract them as specific features.

- **Syntactic parsing** (Table 2.3) are features which can be extracted by using a dependency parser, such as the Stanford parser (Chen and Manning, 2014). It contains special annotations such as dependencies, part-of-speech tags, and generalised dependencies.

- **Durational** features (Table 2.3) are features indicative of the length of the text. It can be measured in different ways, e. g., characters, terms or sentences.

- **Counts**-based features (Table 2.4) are features which depend on the triggering of specific patterns, such as repeated punctuation, abuse of upper-case, or use of word lengthening

Citation	Reference to scripture	Tutek et al. (2016) ; Krejzl and Steinberger (2016)
	Reference to scientific article	Palau and Moens (2009)
Word vectors	Word2Vec-based features	Zhang and Lan (2016) ; Tutek et al. (2016)
	GloVe-based features	Bøhler et al. (2016) ; Liu et al. (2016)
Sentiment	Sentiment of subject (I, we)	Igarashi et al. (2016)
	Sentiment of target	Igarashi et al. (2016)
	Sentiment2Vec-based feature	Zhang and Lan (2016)
	Opinion-generalised dependencies	Abbott et al. (2011) ; Hassan et al. (2012) ; Walker et al. (2012b) ; Hasan and Ng (2013a) ; Wang and Cardie (2014) ; Walker et al. (2012c) ; Sridhar et al. (2015) ; Misra et al. (2016)
	Sentiment lexicon-based features	Hillard et al. (2003) ; Hahn et al. (2006) ; Germesin and Wilson (2009) ; Wang et al. (2011) ; Hassan et al. (2012) ; Yin et al. (2012a) ; Hasan and Ng (2013a) ; Misra and Walker (2017) ; Wang and Cardie (2014) ; Rosenthal and McKeown (2015) ; Elfardy and Diab (2016) ; Zhang and Lan (2016) ; Sun et al. (2016) ; Bøhler et al. (2016) ; Patra et al. (2016) ; Mohammad et al. (2017) ; Igarashi et al. (2016) ; Krejzl and Steinberger (2016)

TABLE 2.6: Features from the stance classification literature: references, word vectors and sentiment-based features

(e. g., “terrrrrrible”). It also contains some text statistics such as average word length and average sentence length.

- **Lexicons and patterns**-based features (Table 2.5) are features which are triggered by lexicons, which are predefined lists of terms or patterns. They include sentiment lexicons (positive or negative), subjectivity lexicons (objective or subjective), argument/stance lexicons (argumentative or non-argumentative) but also different markers such as modal verbs, emoticons, presence of negation or acquiescence, marks of accommodation, etc.

- **Citation**-based features (Table 2.6) are features based on the text citing external resources, which can be evidence of a specific stance. For instance, making references to scripture in a debate on abortion could be very indicative of a negative stance towards the subject.

- **Word vectors**-based features (Table 2.6) are features extracted from pre-computed word embeddings using Word2Vec (Mikolov et al., 2013a) or GloVe (Pennington et al., 2014).

Context	Child-Parent matching	Galley et al. (2004) ; Abbott et al. (2011) ; Hasan and Ng (2013a) ; Wang and Cardie (2014) ; Rosenthal and McKeown (2015) ; Sridhar et al. (2015) ; Walker et al. (2012c) ; Zhang and Lan (2016)
	Conversational structure constraints	Hassan et al. (2012) ; Rosenthal and McKeown (2015)
	Strict reference to target	Galley et al. (2004) ; Wang et al. (2011) ; Hassan et al. (2012) ; Walker et al. (2012b) ; Wang and Cardie (2014) ; Mohammad et al. (2017)
	Time between posts	Abbott et al. (2011) ; Yin et al. (2012a)
	Author constraints	Hasan and Ng (2013b)
	User-interaction constraints	Galley et al. (2004) ; Hasan and Ng (2013b)
	Ideology constraints	Hasan and Ng (2013b)
	Distance from root post	Rosenthal and McKeown (2015)
	Number of debaters	Galley et al. (2004)
Topics and concepts	Latent semantics from matrix factorisation	Elfardy and Diab (2016)
	Latent semantics from Brown clustering	Zhang and Lan (2016)
	Latent concepts from frequency analysis	Wojatzki and Zesch (2016a)
	Topic features (word, sentence, instance level)	Li et al. (2016) ; Zhang and Lan (2016)
	Frame-semantic features	Hasan and Ng (2013a) ; Elfardy and Diab (2016)

TABLE 2.7: Features from the stance classification literature: features extracted from conversational context and topic analysis

- **Sentiment**-based features (Table 2.6) are extracted using a sentiment lexicon, such as opinion-generalised dependencies or average sentiment as a single numerical feature.
- **Context**-based features (Table 2.7) are extracted by observing the context of the current text. For example, strict references to the target of the stance, distance from the beginning of the conversation in number of posts, number of debaters involved in the discussion, or some measure of similarity between the current stance-bearing text being analysed and the one that it is directly replying to.
- **Topics and concepts** (Table 2.7) are features which are computed by analysing latent topics in the text. Such analysis can be done for example with Brown clustering over a collection of stance-bearing items, or simply by using an external resource to analyse the frame semantics of the terminology used.

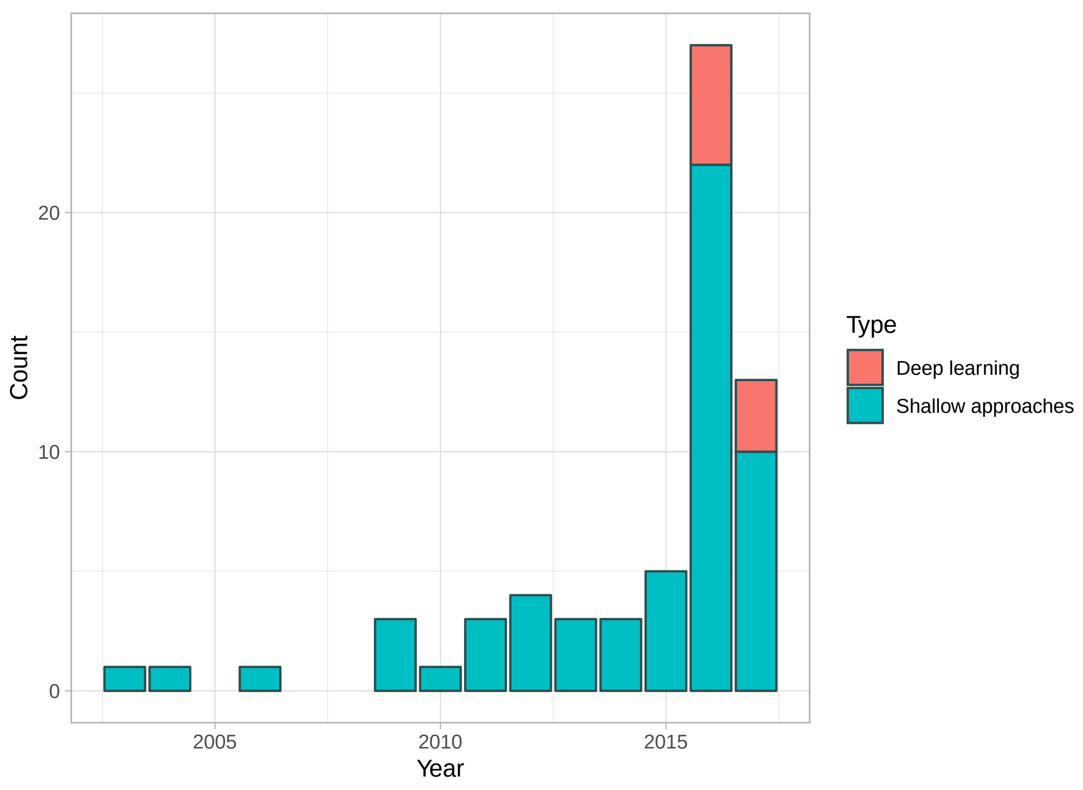


FIGURE 2.2: Evolution of shallow versus deep learning approaches for stance classification

We can see from Tables 2.1 to 2.7 that besides some generic text features such as n-grams, features extracted from sentiment lexicons and word vectors and features extracted from topic analysis, most of the literature has not focused on the production of robust features but instead on the production of as many features as possible in order to obtain an edge in classification accuracy.

Learning algorithms While there are some exceptions, most of the work done with shallow algorithms and feature models in stance classification revolves around a handful of key algorithms: Support Vector Machines, Logistic Regression, Decision Trees, and Maximum Entropy. We list them in Table 2.8, where it is easy to see that a majority of works focus on Support Vector Machines, and illustrate in Figure 2.2 the evolution of shallow versus deep classifiers. We can see that from 2016 onward, a significant percentage of the works on stance classifiers focused on the use of deep neural networks in order to forego the difficult part of feature engineering and build more robust algorithms. We also note that at the time of the redaction of this thesis,

Algorithm	Papers
Support Vector Machines	Hahn et al. (2006) ; Somasundaran and Wiebe (2010a) ; Hassan et al. (2012) ; Hasan and Ng (2013b) ; Sridhar et al. (2014) ; Boltuzic and Šnajder (2014) ; Sobhani et al. (2015) ; Patra et al. (2016) ; Wojatzki and Zesch (2016a) ; Elfardy and Diab (2016) ; Dias and Becker (2016) ; Patra et al. (2016) ; Sun et al. (2016) ; Mohammad et al. (2017) ; Grčar et al. (2017) ; Wang and Cardie (2016)
Logistic Regression	Yin et al. (2012b) ; Sridhar et al. (2015) ; Zhang and Lan (2016) ; Augenstein et al. (2016a)
Decision Tree	Hillard et al. (2003) ; Germesin and Wilson (2009) ; Wojatzki and Zesch (2016b) ; Misra and Walker (2017)
Maximum Entropy	Galley et al. (2004) ; Palau and Moens (2009) ; Rosenthal and McKeown (2015) ; Krejzl and Steinberger (2016)
Ensemble	Bøhler et al. (2016) ; Liu et al. (2016) ; Tutek et al. (2016) ; Mourad et al. (2018)

TABLE 2.8: Learning algorithms used in the literature

2.1.2.2 Deep learning methods for stance classification

Recent trends in machine learning have been leaning towards deep learning algorithms, i. e., neural network architectures which use a stack of successive hidden layers to simultaneously learn a better representation of the data and a better linear classifier on the transformed version of this data. We qualify as deep learning algorithms any neural network with a depth of 3 or more layers. In the context of stance classification, two types of architectures have emerged as the dominant ones in their respective specialized tasks: convolutional neural networks, and gated recurrent neural networks.

- **Convolutional neural networks** (CNN) are neural networks containing at least one convolution layer. A convolution layer is a neural network layer that is bound by specific constraints, and is usually followed by either a fully-connected layer, a pooling layer or another convolution layer. The goal of convolutions and pooling layers is to achieve specific invariances towards certain transformations that make them invaluable in the fields of image and signal processing: translation invariance, rotation invariance, colour shift invariance, and distortion invariance. Unlike traditional fully-connected layers, convolution layers are not connected to every unit in the previous layer, but only to the ones within their receptive field, which allows them to focus on specific local features and specialize in the detection of hierarchically organized features.

- **Recurrent neural networks** (RNN), unlike convolutional neural networks which use a more conventional structure of layers stacked on top of each others, achieve depth by parsing the input by timestep and using a recurrent relationship that links the hidden layer of the network to itself in the next timestep. By doing this, the output at each timestep becomes dependent on the processing that was done on the previous timestep, and during the backpropagation phase the error signal can be distributed over all timesteps in a weighted manner. Recurrent neural networks are explored in more depth in section 3.4.4.1. Two popular RNN architectures are Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU) networks.

Zubiaga et al. (2018) evaluate multiple sequential classifiers on a rumour stance classification task, where they attempt to classify whether a rumour is pro or against a specific target opinion. As part of their evaluation, Long Short Term Memory networks are shown to outperform other kinds of classifiers (Hawkes Processes, Linear-Chain Conditional Random Fields, and Tree-Structured Conditional Random Fields) with a reduced set of features. Another outcome of their research is that sequential classifiers that make use of discourse properties (the conversational threads) outperform non-sequential classifiers, which only consider one specific item at a time. The specific implementation used in this paper, called branch-LSTM, uses a mean embedding representation as discussed in 3.3.2.2, which allow for representing each item as a fixed size embedding vector, and therefore to represent each conversational thread, which is composed of multiple items, as a variable size matrix. The branch-LSTM can then focus on tagging each timestep (which represents one item) with its stance. Sequential classifiers are a promising area of research and can be thought of as a special case of collective classifiers, which consider a collection of items to enhance the classification of every one of them. Previous work on collective inference in the context of stance classification has been done by Sridhar et al. (2014), where Support Vector Machines were used as the base classifier and the collective inference part of their approach was done using probabilistic soft logic to encode, in a probabilistically well-founded way, possible interaction rules between the users of a forum as edges in a graphical model.

Zhang et al. (2017) investigate stance classification in the context of alternative medicine debates in social media. They used a Long Short-Term Memory network stacked over a Convolutional Neural Network in order to first identify the debates, and then a Convolutional Neural Network to identify the stance of those social media posts with respect to alternative medicines. Interestingly, no significant difference was found between Convolutional Neural Networks and one of their simpler baselines, a logistic regression. Since the actual classification layer of a Convolutional Neural Network is or behaves similarly to a logistic regression, this would tend to lend evidence to the fact that the representation learning layers of the networks were unnecessary, and the data was as separable as it could be. Another explanation would be that the architectural details of the network were ill-fitted for the problem they were trying to solve.

Augenstein et al. (2016b) focus on using conditional bidirectional encoding of the stance-bearing text along with the stance target in order to take such target into account during the representation step, using a bidirectional LSTM network.

Similarly, Du et al. (2017) go further than examining only one piece of text and instead develop a target-specific neural attention network. Neural attention networks are recurrent neural networks which are concurrently learning where to focus their attention as well as how to process the information they are focusing on, instead of exploring the content linearly in a step-by-step manner. In this case, the neural attention networks are primed with a target, which allows them to focus on the parts of the text which are topically related to the argumentative stance we seek to detect. They report significant improvement over the SemEval-2016 and the NLPCC datasets, showing the portability of their techniques to multiple languages.

Hiray and Duppada (2017) take a different approach by processing in parallel the text input that is to be classified and the target text using two sub-networks that are eventually concatenated and fed into a bigger one. Both target and response are represented through two sets of features: an embedding representation and a lexicon representation. The embedding representation is fed into a bidirectional GRU layer, while the lexicon representation is fed into a dense layer. Those four representations (2 for target and 2 for response) are then concatenated and fed into a more conventional stack of dense layers, leading to the final softmax output layer. The authors report significant improvement over all datasets using their complex representation and learning scheme, both for traditional supervised learning and for a transfer learning task (with partial retraining).

Yu et al. (2016) build their model more conventionally by first learning features through an embedding and a convolutional layer, before feeding its output into a bidirectional LSTM. They report difficulty in achieving high levels of classification accuracy because of ambiguity in the data, which causes issues at the word embedding level.

Vijayaraghavan et al. (2016) train a different model for each dataset, using a combination of character-level and word-level convolutional networks. One particularity of their work is their use of Word2Vec embeddings to augment textual data, by randomly replacing terms with other terms whose word vector is similar. The authors report unequal results, ranging from state-of-the-art on some datasets to very poor on others. They hypothesise that the poor performance might be linked to the amount of training data, and that further work on augmentation is needed.

Igarashi et al. (2016) do a comparative study between a convolutional neural network and a feature-based model. One of the main outputs from their work is that a shallow model with properly engineered features can outperform a deep learning model trained on naive features. However, the feature set they propose requires a significant amount of pre-training and preparation.

Wei et al. (2016) propose a convolutional architecture based on a voting scheme, where a snapshot of the network is kept for each of the 10 epochs of training. A majority vote between those 10 networks is then used at testing time. This strategy, associated to a divide-and-conquer strategy where they learned specific models for each of the target topics, led them to state-of-the-art results, ranking them first on SemEval-2016 Task 6B and second on SemEval-2016 Task 6A.

Zarrella and Marsh (2016) train a more conventional model where the words of the responses to be classified are one-hot encoded (each word is encoded as its own dimension, see section 3.3.1.1 for more details). They are then fed one word at a time into an embedding layer, which transforms the very sparse and high dimensional one-hot representation into a dense non-sparse vector. The output of the embedding layer is then fed into a recurrent layer, which produces the final prediction.

2.1.2.3 Conclusions from the stance classification literature

In this section we reviewed the supervised stance classification literature along two axes: shallow methods, which make use of feature engineering techniques to represent instances in a way that makes them separable by conventional classifiers ; and deep methods, which learn those features from the data itself in an end-to-end fashion (or close to it). We note that while deep learning methods overall outperform feature engineering-based methods, they usually require significant architecture engineering to do so, with some naive deep learning approaches being outperformed by carefully crafted feature models (Igarashi et al., 2016). We also note that through this extensive work on feature representation, whether it is designed or learned, there is a large part of these features sets which are not intelligible from a user perspective, and thus cannot be interpreted.

2.2 Corpora for stance classification

Multiple corpora collected from diverse sources (social media, transcribed meetings, etc.) have been made public for the task of stance classification. We list the major ones in this section and give details on their composition because the field of natural language processing research inherently ties novel datasets and the applied techniques that are developed for it. Therefore, this section needs to be considered as part of the literature review. Further details about the datasets that we selected to use in this thesis can be found further in chapter 4, which is dedicated to our evaluation methodology.

Stance classification datasets can be categorised in one of three categories: (1) **transcribed media** are video or audio debates which have been transcribed by human labellers, (2) **social media** corpora are online debates which can be separated in two sub-categories: (2a) **free-form social media** corpora on which no restriction was put in terms of content or length, (2b) **short-text social media** corpora where a length restriction (in terms of number of characters) forces the debaters to be concise, and (3) **non-standard argumentation** corpora contain corpora which are more structured and longer, in the forms of essays or news articles.

2.2.1 Transcribed media corpora

Two datasets, the ICSI corpus and the AMI corpus, compose most of this category. They are of limited size but contain long interactions, meaning that the same actors stay in the debates (unlike social media data, where it is frequent that a user posts a few comments and that another

Dataset	Post count	Agreement	Disagreement	None
ICSI	9,854	886	591	8,377
AMI	706	636	70	0

TABLE 2.9: Descriptive statistics of two transcribed media stance detection datasets.

user picks up the debate and moves it further). Because the content of those datasets has been transcribed, they are devoid of slang, mistakes and typos. However, they also contain metadata which is particular to audio content and might not be present in other applications. Statistics on the datasets can be found in Table 2.9.

We can observe that the classes are not particularly balanced, with a ratio of Agreement to Disagreement going from $3 : 2$ (ICSI) to $9 : 1$ (AMI) and a ratio of stance-bearing to non stance-bearing going from $1 : 6$ (ICSI) to no non stance-bearing at all (AMI).

ICSI corpus The ICSI corpus Janin et al. (2003) was collected from natural meetings occurring at the International Computer Science Institute (ICSI). First recorded in audio, from highly accurate source (head-worn and table-top microphones), it was then transcribed on a text format along with task-specific metadata such as the gender of the speaker as well as an identifier number for each speaker, their qualification (Undergrad, Grad, PhD, Professor or other), their age, and their main language. The dataset was partially automatically labeled with only a seed of 1,800 instances passing through manual inspection. It is also notable that prior to labeling, trivial instances (e.g. spurts only containing “Yeah!” or “yep”) were removed from the positive class and put as backchannel (classified as None in Table 2.9).

AMI corpus The AMI corpus Carletta et al. (2005), McCowan et al. (2005) is built on 100 hours of recorded meetings in the context of the AMI (Augmented Multi-party Interaction) project. It is composed of both real meetings as well as scenario-driven meetings, which are annotated with dialogue acts and summaries.

Dataset	Post count	Agreement	Disagreement	None
ABCD	185,479	38,195	60,991	86,293
IAC	5,940	428	1,236	4,276
AWTP	822	38	148	636

TABLE 2.10: Descriptive statistics of three social media stance detection datasets.

Dataset	Class	
	Agreement	Disagreement
Livejournal	379	80
Wikipedia	141	210

TABLE 2.11: Class distribution in the AWTP dataset.

2.2.2 Free-form social media corpora

Free form social media corpora are extracted from forums and are characterised by the absence of space or form restriction in user-to-user interactions. Unlike transcribed media corpora, they do not contain any information besides the actual text content and some forum-specific metadata (username, dates, times, etc.). They also have the particular feature of being written by the users themselves, which leaves a lot more room for typos, grammatical mistakes and internet slang. Inspection of the statistics for the datasets givens in Table 2.10 indicates a clear class imbalance in all the datasets, with the most balanced of them having a $2 : 3$ class ratio between Agreement and Disagreement. Such imbalance is large enough to affect the quality of learning and require specific data treatment before application of a classifier. However it may be representative of natural argumentation patterns since we notice a similar imbalance in transcribed datasets.

AWTP The AWTP (Agreement in Wikipedia Talk Pages) corpus Andreas et al. (2012) was built from Wikipedia threads and Livejournal blogs. Sentences were extracted and labeled as agreement and disagreement, along with a set of meta information about the annotation process in order to judge the quality of the label. The classes are distributed as illustrated in Table 2.11.

ABCD The ABCD (Agreement By Create-Debaters) corpus Rosenthal and McKeown (2015) was collected on a specialised debating forum called CreateDebate[1]. Debates on this platform

[1]https://www.createdebate.com

can be of three types: open-ended (no side), for/against (two sides), multiple sides (three or more sides). The authors only collected debates of the for/against kind in order to make the task more tractable.

IAC The IAC (Internet Argument Corpus) Walker et al. (2012a) was built from posts collected from the 4forums[2] discussion platform. It was annotated using Amazon Mechanical Turk[3] on a -5 to 5 disagreement to agreement scale.

2.2.3 Short text social media corpora

Short text corpora are social media corpora which have been selected for short form argumentation mining. While proper argumentation requires the formulation of well-formed arguments such as ones defined under the Toulmin model, and social media argumentation bypasses this format by allowing its participants to omit crucial parts of their arguments by relying on common knowledge, short form argumentation compresses social media argumentation even further by limiting it to a specific number of characters per contribution.

SemEval (Twitter) The SemEval corpus Mohammad et al. (2016) was built to provide a platform to study short form argumentation on. It was crawled from Twitter[4] using some specific topical keywords (Atheism, Climate Change is a Real Concern, Feminist Movement, Hillary Clinton, Legalization of Abortion, Donald Trump) and labeled using CrowdFlower[5].

NLPCC-ICCPOL The NLPCC-ICCPOL corpus Xu et al. (2016) was built on the Chinese social media platform Sina Weibo[6] which allows its users to express their opinion towards multiple topics of interest in an explicit or implicit way, similar to the Twitter corpus used for SemEval. Much like SemEval, this dataset was released as part of a shared evaluation task on different text mining fields, one of them being stance detection expressed as a Pro/Against/None

[2]https://4forums.com
[3]https://www.mturk.com/mturk/welcome
[4]https://twitter.com
[5]https://www.crowdflower.com
[6]http://weibo.com

classification problem. Unlike SemEval, this dataset is encoded in Mandarin Chinese except for the labels. An excerpt from the dataset is provided in the example below.

2.2.4 Non-standard argumentation corpora

Non-standard corpora are corpora which are not necessarily built from direct interaction of users.

Emergent Emergent Ferreira and Vlachos (2016) is a dataset composed of news pages. Claims are gathered by journalists from multiple sources and Twitter accounts, on topics such as world news, U.S. news and technology-centered stories. After selecting a claim, articles that mention it are collected and their stance (pro, against, or just observing) is specified and labelled.

2.3 Chapter summary

In this chapter we reviewed the literature related to stance classification from multiple perspectives. We started with the methods for stance classification, analysing unsupervised, supervised shallow and supervised deep approaches for classifying stance in multiple types of environment (forums, long-form social media, Twitter). We then described the different datasets used in the stance classification literature and how they relate to our research. Of the datasets presented, our work focuses on free-form and short text social media corpora, with the exception of the NLPCC-ICCPOL which presents a list of challenges inherent to its language (Mandarin Chinese).

We note from studying the literature that there has been little work done on advancing the state of stance lexicons. Lexicons are useful tools for computational linguistics because of their high degree of interpretability, which makes them ideal for building explainable models. We note that a significant part of the literature was done after our own work, and as such the state of the art has been advanced beyond the scope of what is presented in this thesis.

Chapter 3

Background

The process of supervised learning (illustrated in figure 3.1) contains specific steps which are common to all (or most) problem domains and techniques. Because we make reference to this process throughout the rest of this thesis, we detail that process in this section before going deeper into each step in the following sections. In this chapter we present a background on the machine learning tasks associated to social media mining, their theoretical underpinnings and a discussion of methodologies to evaluate them. We start by giving an overview of the supervised learning process, before going into more detail on each step of that process.

We first address text pre-processing and its multiple steps: tokenisation, annotation, standardisation and filtering. We then move on to text representation, which focuses on taking the output from the pre-processing step and building a representation that can be algorithmically processed. We address both sequential representations (one-hot encoding and word embeddings) and non-sequential representations (bag-of-words and mean embeddings). The next step is model building and consists in using the data output by the representation step in order to build a model of the data with respect to a specific task, e. g., classifying them into specific categories. We explain in sufficient details multiple methods that are used in the context of this thesis: lexicon methods, support vector machines, and neural networks. We leave the final step of model evaluation to its own chapter. It is briefly mentioned in this chapter, but explained in details in chapter 4.

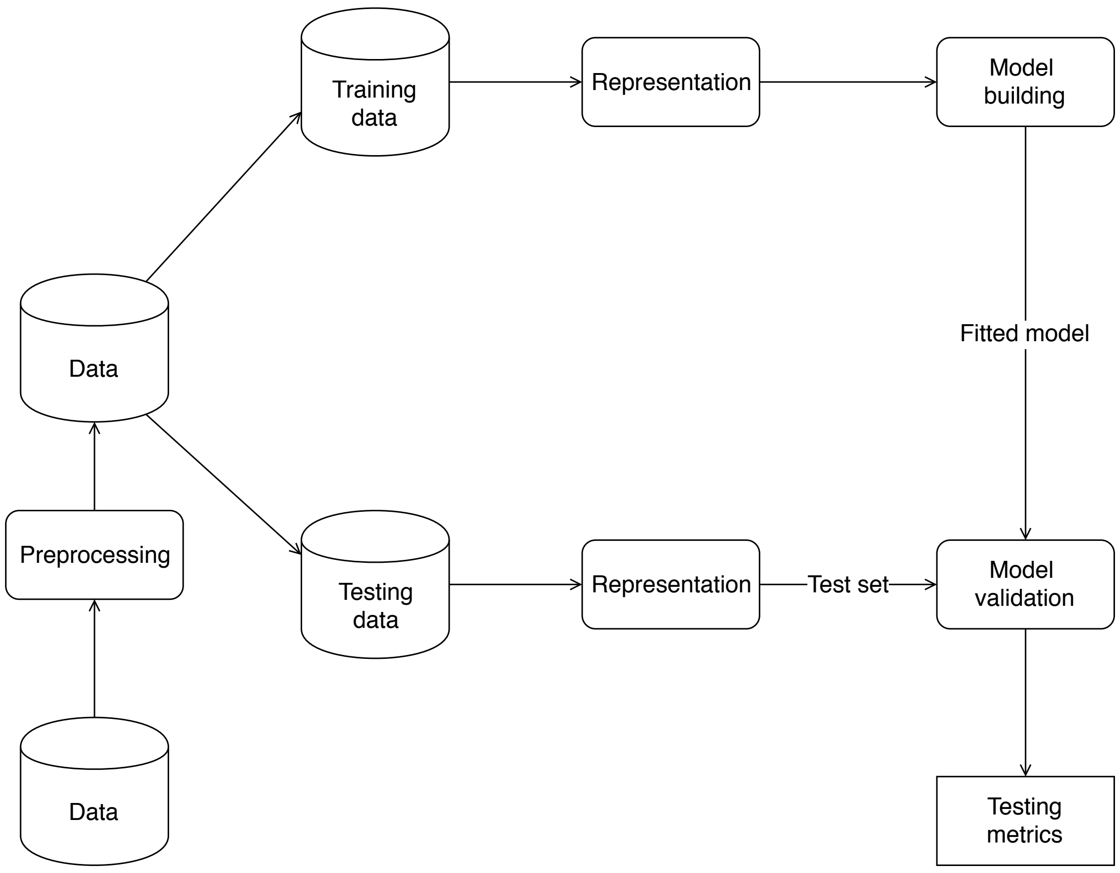


FIGURE 3.1: The machine learning pipeline

3.1 The supervised learning process

The first step is the preprocessing and splitting of the original dataset. The preprocessing consists in turning the text into a format that can be efficiently parsed by a machine learning algorithm, including tokenisation (recognising individual units of interest within the text, whether they are unigrams, bigrams or more complex structures), annotation (adding contextual information to those tokens in order to disambiguate or enrich the text), vocabulariy standardisation (replacing terms with their roots or stems in order to not have to deal with variations of the same words), and finally filtering (removal of stopwords which are deemed to not be useful for the classification process due to their omnipresence. Once that preprocessing is done, the dataset is split into a training set and a test set. The training set will be used for feature selection and model building, while the test test is kept aside in order to evaluate the generalisation capabilities of the model that we learned.

The second step of the process is text representation. Most non-symbolic machine learning algorithms require their input data to be numerical in order to perform model inference. Two common schemes for text data are sequential representation, where text is represented as a tensor of variable length depending on the length of the input, and non-sequential representation, where text is represented as a tensor of fixed size. Representation is a key step in that it defines the vector space in which the learning algorithms will have to learn, and therefore having a few key features can make or break the performance of a machine learning algorithm.

The third step is model building. During this step, the machine learning algorithm uses the processed training set in order to build a model of the data that can be used to produce predictions over unseen data.

Finally, the fourth and final step is the model validation phase. During this step, the model learned in the previous step is used in conjunction with the processed testing set in order to compute performance metrics. Typical performance metrics include the average misclassification rate (when the dependent variable is categorical), the average distance from prediction to ground truth (when the dependent variable is numerical), and probabilistic distance metrics (when the dependent variable is a probability distribution).

In the next sections we will go over each of these steps in more detail.

3.2 Text preprocessing

Pre-processing text is an essential step in all natural language processing tasks due to the mismatch between the state in which text data is typically present, a list of characters, and the state that it needs to be in in order to be a valid input for an algorithm. In addition to that, some of the heuristic pre-processing rules that can be applied during this step of the algorithm can improve the performance of the algorithm by reducing dimensionality of the input (e. g., stemming, lemmatisation, etc.) or transforming the input so that it maintains specific statistical properties (e. g., normalisation, standardisation, etc.). In this section we detail the nature of text pre-processing and the ways it can be done in a text classification setting.

We can see in Figure 3.2 the overall text pre-processing workflow, which we then detail in this section.

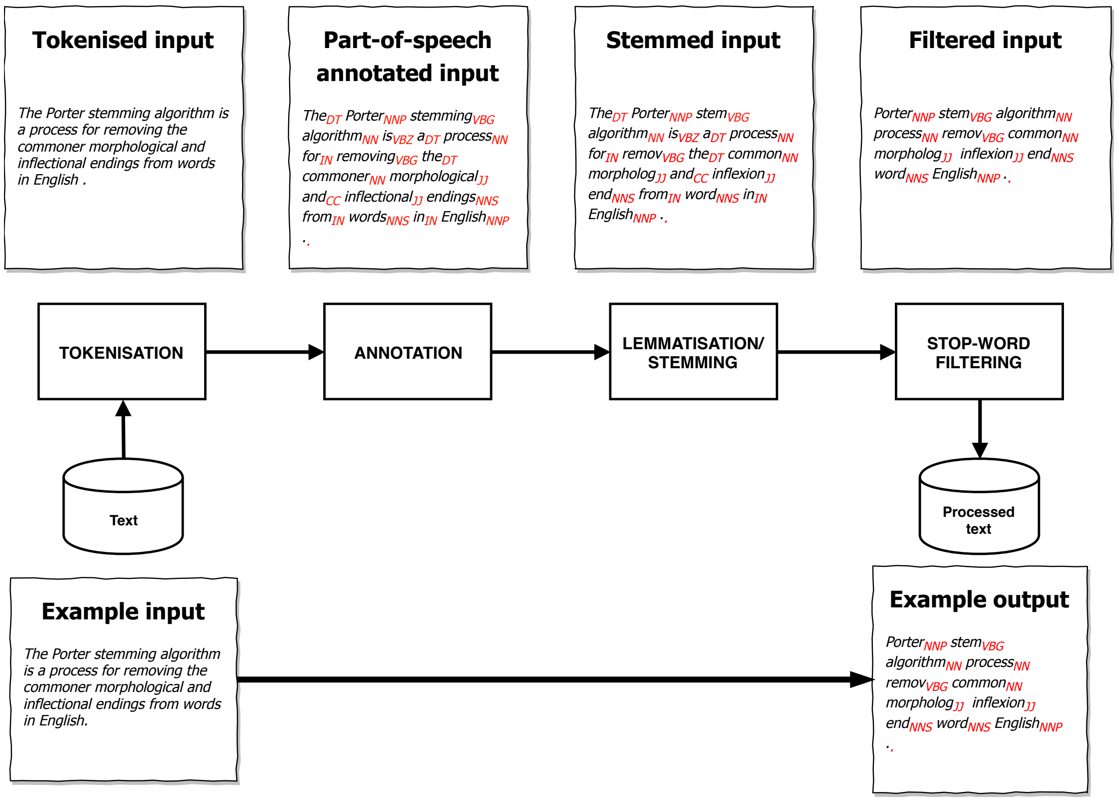

Tokenised input
The Porter stemming algorithm is a process for removing the commoner morphological and inflectional endings from words in English .
Part-of-speech annotated input
The$_{DT}$ Porter$_{NNP}$ stemming$_{VBG}$ algorithm$_{NN}$ is$_{VBZ}$ a$_{DT}$ process$_{NN}$ for$_{IN}$ removing$_{VBG}$ the$_{DT}$ commoner$_{NN}$ morphological$_{JJ}$ and$_{CC}$ inflectional$_{JJ}$ endings$_{NNS}$ from$_{IN}$ words$_{NNS}$ in$_{IN}$ English$_{NNP}$ ..
Stemmed input
The$_{DT}$ Porter$_{NNP}$ stem$_{VBG}$ algorithm$_{NN}$ is$_{VBZ}$ a$_{DT}$ process$_{NN}$ for$_{IN}$ remov$_{VBG}$ the$_{DT}$ common$_{NN}$ morpholog$_{JJ}$ and$_{CC}$ inflexion$_{JJ}$ end$_{NNS}$ from$_{IN}$ word$_{NNS}$ in$_{IN}$ English$_{NNP}$ ..
Filtered input
Porter$_{NNP}$ stem$_{VBG}$ algorithm$_{NN}$ process$_{NN}$ remov$_{VBG}$ common$_{NN}$ morpholog$_{JJ}$ inflexion$_{JJ}$ end$_{NNS}$ word$_{NNS}$ English$_{NNP}$ ..
TOKENISATION
ANNOTATION
LEMMATISATION/ STEMMING
STOP-WORD FILTERING
Text
Processed text
Example input
The Porter stemming algorithm is a process for removing the commoner morphological and inflectional endings from words in English.
Example output
Porter$_{NNP}$ stem$_{VBG}$ algorithm$_{NN}$ process$_{NN}$ remov$_{VBG}$ common$_{NN}$ morpholog$_{JJ}$ inflexion$_{JJ}$ end$_{NNS}$ word$_{NNS}$ English$_{NNP}$ ..

FIGURE 3.2: The pre-processing workflow

3.2.1 Tokenisation

Given a character sequence and a defined document unit, tokenisation is the task of chopping it up into pieces, called tokens Manning et al. (2008). Here is an example of tokenisation:

> **Input**: Seneca was born in Cordoba in Hispania, and raised in Rome, where he was trained in rhetoric and philosophy.
>
> **Output**: Seneca | was | born | in | Cordoba | in | Hispania | , | and | raised | in | Rome | , | where | he | was | trained | in | rhetoric | and | philosophy | .

The key role of tokenisation is to break down the document into pieces that can be processed by the rest of the pre-processing workflow. It is important for instance to process whether hyphenated words, e. g., "state-of-the-art", are one or multiple tokens and should be treated as such. Conversely, some sets of words separated by spaces, e. g., "New York", ought to be treated as one singular token.

3.2.2 Annotation

The step of annotation is a sometimes optional one in that its main focus is not to prepare text into a form that can be processed by the main algorithm, but to enrich it in such a way that the rest of the pre-processing is made easier and the sense of each term can be disambiguated. Two common forms of annotation in argument stance mining are part-of-speech and sentiment tagging.

Part-of-speech tagging Part-of-speech tagging focuses on tagging each term with their grammatical role in the sentence (their part of speech). A part of speech is a category of words which have similar grammatical properties. Words that are assigned to the same part of speech generally display similar syntactic behaviour and play a similar role within the grammatical structure of sentences. In English, the most common parts of speech are noun, verb, adjective, adverb, pronoun, preposition, conjunction, interjection, and sometimes numeral, article or determiner.

Part-of-speech tagging has multiple uses:

- It helps disambiguate terms which have homonyms if their part-of-speech is different, e. g., lead (verb)/lead (noun).
- It provides a generalised representation of the sentence to the algorithm. For instance, "The cat in the hat" would yield the annotation "determiner noun preposition determiner noun" which is a more general representation of the structure of the sentence. This can be powerful when some sentence structures are more correlated to some class of interest, in the classification case.

Sentiment tagging Sentiment tagging focuses on annotating each term with a vector of numerical values or a string representing the sentiment of that term. For instance, "I hate spaghetti" would yield the annotation "neutral negative neutral". This can have profound effects in the context of stance classification, with for example sentiment structures which are indicative of a user's outlook towards another user.

3.2.3 Standardisation of the vocabulary

A crucial step in the preprocessing workflow is the standardisation of the vocabulary. Words tend to appear as inflected forms of a canonical word. For example, the verb "to teach" could appear as "taught", "teach", "teaches", or "teaching" and refer to the same action. Text representation schemes such as bag of words operate at a shallow level and therefore would have "teach" and "taught" as far from each other in the representation space as "teach" and "dog". Lemmatisation and stemming are two techniques that help alleviate this problem by transforming words into a common root that is shared with other words that are semantically related.

Lemmatisation Lemmatisation works by doing a look-up of inflected forms into a dictionary and replacing the forms present in the text by the canonical form. Its main drawback is that it is sensitive to infrequent typos and small variations that a dictionary might not account for, and therefore has a low recall. Its strength is that terms that share a semantic root will be matched systematically, making the vector space in which documents are represented less sparse. We illustrate a concrete example of lemmatisation in Example 1.

Example 1. Original sentence: "I think that cats are nicer than dogs."

Lemmatised sentence: "I think that cat be nice than dog"

Stemming Stemming works by applying a set of unsupervised transformation rules to each term in order to recover their stem. It is highly imprecise and produces stems which are not necessarily identical to the morphological root of the words, but is more efficient than lemmatisation. The most popular and best known stemming algorithm is issued from the information retrieval community and is the Porter Stemmer Porter (1980). We illustrate a concrete example of stemming in example 2.

Example 2. Original sentence: "I think that cats are nicer than dogs."

Stemmed sentence: "I think that cat ar nicer than dog"

In this example, *cats* is stemmed as *cat*, *are* is stemmed as *ar*, and *dogs* is stemmed as *dog*. The resulting sentence is not grammatically correct but it ensures that references to the same nouns

(e. g., , cats) match throughout the entire document, because they will be reduced to the same stem.

3.2.4 Filtering

A stop list is a list of terms that should be ignored by a text processing algorithm. Stop word removal, or filtering, is the step in the text classification pre-processing workflow where terms which are not discriminative enough are removed from the text for the sake of computational efficiency. The stop list is determined by sorting all the terms present in the dataset by increasing inverse document frequency, which can be determined as $\log(\frac{N}{n})$ where N is the number of documents in the dataset and n the number of documents containing a specific term, and selecting the lowest ranked terms.

3.3 Text representation

In order for an algorithm to process text, such text needs to be in a format that the algorithm can take as input. For example, most supervised machine learning algorithms exclusively work with numerical values, which means that any text input needs to be transformed into a vector (or more generally a tensor) of numerical values. We separate two types of representation and detail the most common techniques belonging to them: sequential representations, where word order is preserved, and non-sequential representations, where word order is not preserved.

3.3.1 Sequential representations

Sequential representations of text are representation schemes that do not remove the inherent structure of the text. Because of this, the resulting instances end up having different length, and the learning algorithms need to accommodate this feature in order to fully leverage this representation scheme. The two most common sequential representation schemes are one-hot encodings and word embeddings.

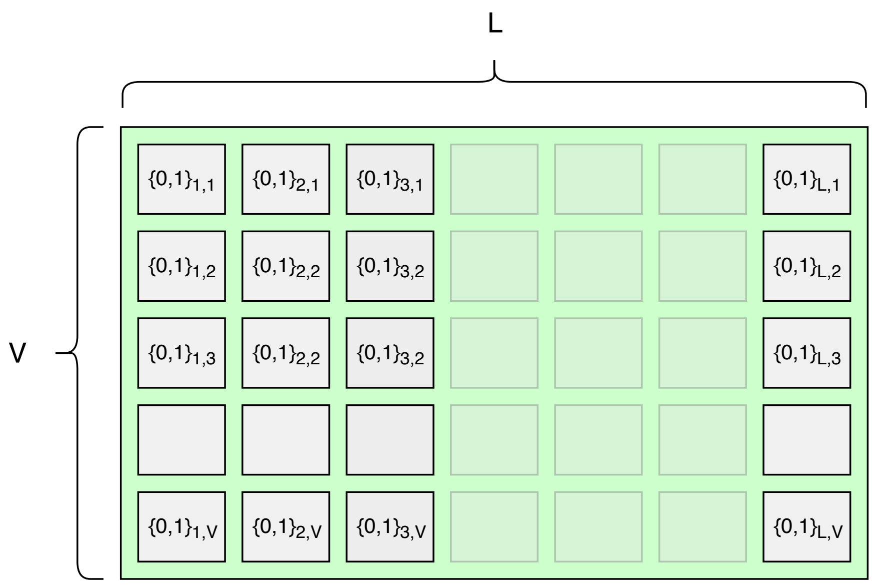


FIGURE 3.3: One hot representation of text. Each dimension can either take the value 0 or 1.

3.3.1.1 One-hot encoding

A one-hot encoding of a text instance encodes each term of the vocabulary on a different dimension of a common vector space. As such, an instance X of length L and a vocabulary of size V would yield a $L \times V$ matrix representation where for each column i, all components would be 0 with the exception of the one corresponding term present at X_i, as illustrated in Figure 3.3, where $\{0, 1\}_{i,j}$ means that dimensions (i, j) can either take value 0 or 1.

We present an example of one-hot encoding on a toy dataset of three small documents.

Example 3. Let us assume that the training dataset D contains the following documents:

- D_1 = “I do not like cats, they are mean.”
- D_2 = “That film was wonderful.”
- D_3 = “Presidents are not as honourable as they used to be.”

The complete vocabulary V_c of the training set would be:

I, do, not, like, cats, they, are, mean, that, film, was, wonderful, presidents, as, honourable, used, to, be

A one-hot encoded version of the training dataset would end up being:

$$D_1 = \begin{bmatrix}
1&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0\\
0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0\\
0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0
\end{bmatrix}
D_2 = \begin{bmatrix}
0&0&0&0\\
0&0&0&0\\
0&0&0&0\\
0&0&0&0\\
0&0&0&0\\
0&0&0&0\\
0&0&0&0\\
0&0&0&0\\
1&0&0&0\\
0&1&0&0\\
0&0&1&0\\
0&0&0&1\\
0&0&0&0\\
0&0&0&0\\
0&0&0&0\\
0&0&0&0\\
0&0&0&0\\
0&0&0&0
\end{bmatrix}
D_3 = \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&1
\end{bmatrix}$$

Advantages and drawbacks of one-hot encoding The main advantage of one-hot encoding is that it makes no assumption about the dimensions of the vocabulary. Since all terms are modelled on separate dimensions, this representation does not take the risk in erroneously considering two terms as being semantically related.

The main drawback of one-hot encoding is the same as its strength: it does not make any assumption about the vocabulary. All terms being modelled as separate dimensions, it does not take the risk of potentially erroneously assuming that two terms are closer than they actually are. This lack of nuance in the way terms are represented is also its biggest flaw. Because each term of the vocabulary is represented as its own dimension in the vector space, there is as much difference between any two terms of the vocabulary and thus no way of representing that, for instance, “cat” is closer to “dog” than to “car”.

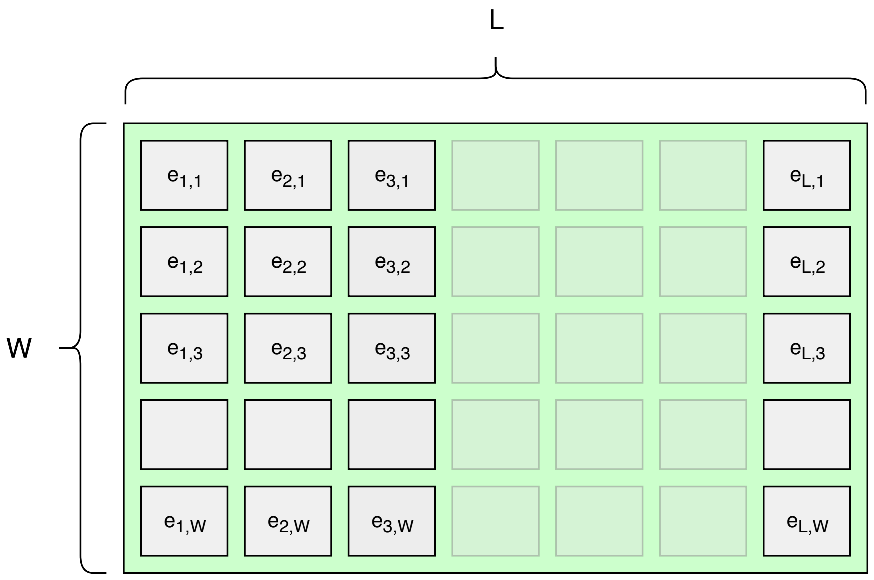


FIGURE 3.4: Embedding representation of size $W \times L$ where W is the size of the embedding (chosen arbitrarily) and L is the size of the instance being represented.

3.3.1.2 Word embeddings

Word embedding methods are a set of language modelling techniques where words, phrases or documents are mapped to vectors of real numbers of fixed size. Their name comes from the fact that it is technically a mathematical embedding from a high dimensional and sparse space where each word is represented by one dimension that is orthogonal to all other words' to a more compact and less sparse space where each word is represented by a smaller set of latent dimensions, illustrated in Figure 3.4 where W represents the set of latent dimensions while L represents the length of the sentence/document being represented. Word embeddings can be computed using a variety of techniques that can extract represent text in a latent space, including probabilistic models Globerson et al. (2007), factorisation of word co-occurrence matrices Lebret and Collobert (2013), Levy and Goldberg (2014), Li et al. (2015), and neural networks Mikolov et al. (2013b).

Example 4. In this example we will compute the word embedding representations of the three documents of Example 3. Let us assume that the training dataset D contains the following documents:

- D_1 = "I do not like cats, they are mean."
- D_2 = "That film was wonderful."

TABLE 3.1: Example of word embedding representation over a latent space of 5 dimensions

Term	**dimension 1**	**dimension 2**	**dimension 3**	**dimension 4**	**dimension 5**
I	-0.0055	0.1148	-0.0595	-0.0035	0.0299
do	-0.0945	0.0239	-0.0391	-0.0925	-0.0812
not	-0.0818	0.0009	-0.0425	-0.0979	0.0140
like	0.0431	-0.0705	-0.0258	0.0427	0.0162
cats,	0.0497	-0.0161	0.0425	0.0602	-0.0399
they	-0.0046	-0.0185	0.0038	0.0756	0.0383
are	-0.0352	-0.04362	-0.0339	-0.0158	-0.0472
mean	-0.0427	-0.0224	-0.0046	-0.0211	-0.0224
That	-0.0259	0.0037	0.0353	-0.0368	0.0101
film	0.0031	-0.0404	-0.0165	-0.0069	-0.0167
was	-0.0587	0.0288	-0.0066	-0.0161	0.0119
wonderful	0.0547	-0.0401	-0.0032	0.0321	-0.0042
Presidents	0.0023	-0.0110	-0.0501	-0.0190	0.0201
as	0.0079	0.0485	0.0540	-0.0127	-0.0378
honourable	0.0034	0.0069	-0.0024	-0.0484	0.0199
used	-0.0517	0.0621	-0.0185	-0.0399	-0.0254
to	-0.0438	0.0316	0.0099	0.0130	-0.1125
be	0.0567	0.0457	0.1122	0.0049	-0.0462

- D_3 = "Presidents are not as honourable as they used to be."

We compute a word embedding representation of this vocabulary using 5 latent dimensions and the Word2Vec algorithm in Table 3.1.

We can see in this example that each term is represented by a vector of 5 values, which allows us to represent D_1, D_2 and D_3 as the following matrices:

$$D_1 = \begin{bmatrix} -0.0055 & -0.0945 & -0.0818 & 0.0431 & 0.0497 & -0.0046 & -0.0352 & -0.0427 \\ 0.1148 & 0.0239 & 0.0009 & -0.0705 & -0.0161 & -0.0185 & -0.0436 & -0.0224 \\ -0.0595 & -0.0391 & -0.0425 & -0.0258 & 0.0425 & 0.0038 & -0.0339 & -0.0046 \\ -0.0035 & -0.0925 & -0.0979 & 0.0427 & 0.0602 & 0.0756 & -0.0158 & -0.0211 \\ 0.0299 & -0.0812 & 0.0140 & 0.0162 & -0.0399 & 0.0383 & -0.0472 & -0.0224 \end{bmatrix}$$

$$D_2 = \begin{bmatrix} -0.0259 & 0.0031 & -0.0587 & 0.0547 \\ 0.0037 & -0.0404 & 0.0288 & -0.0401 \\ 0.0353 & -0.0165 & -0.0066 & -0.0032 \\ -0.0368 & -0.0069 & -0.0161 & 0.0321 \\ 0.0101 & -0.0167 & 0.0119 & -0.0042 \end{bmatrix}$$

$$D_3 = \begin{bmatrix} 0.0023 & -0.0352 & -0.0818 & 0.0079 & 0.0034 & -0.0046 & -0.0517 & -0.0438 & 0.0567 \\ -0.0110 & -0.0436 & 0.0009 & 0.0485 & 0.0069 & -0.0185 & 0.0621 & 0.0316 & 0.0457 \\ -0.0501 & -0.0339 & -0.0425 & 0.0540 & -0.0024 & 0.0038 & -0.0185 & 0.0099 & 0.1122 \\ -0.0190 & -0.0158 & -0.0979 & -0.0127 & -0.0484 & 0.0756 & -0.0399 & 0.0130 & 0.0049 \\ 0.0201 & -0.0472 & 0.0140 & -0.0378 & 0.0199 & 0.0383 & -0.0254 & -0.1125 & -0.0462 \end{bmatrix}$$

We can observe that the resulting representations are more compact than the one-hot encoding seen in example 3 and less sparse, meaning that fewer dimensions are 0. However, it is not possible to interpret the dimensions computed in a word embedding in the same way as a one-hot encoding.

3.3.2 Non-sequential representations

In this section, we review non-sequential representation schemes for text mining. Non sequential representations remove word order, making it impossible to recover the structure of the original text, before feeding an instance to a supervised learning algorithm. Removing this structure has advantages and drawbacks. Its main advantage is that it enforces some form of location-invariance in the resulting features: the learning algorithm is forced to learn how to deal with words without caring about where they are located in the text. The main drawback is that this location can sometimes change the meaning or the class valence of a word, e. g., "I am happy" vs. "I am not happy", where "happy" being placed after a negation changes its meaning to its opposite.

3.3.2.1 Bag-of-words

The bag-of-words model is a representation of documents as vectors in a high dimensional vector space induced by a vocabulary. In order to represent documents in a common representation, the vector space is induced from the union of the vocabulary of all the documents post pre-processing, each dimension of the vector space being one of the terms of that vocabulary and its numerical value being a function of the frequency of that term in the document in question. It is quite common to keep that vocabulary voluntarily restricted, and the two ways to deal with unknown terms observed at test time are to have an unknown token that absorbs all cases of new vocabulary, or more commonly to simply ignore those cases.

The second step of document representation once the features (terms) have been acquired is their transformation and/or filtering in order to represent their actual importance in the training set. This is typically known as feature weighting, or in this case term weighting. The most popular and effective term weighting techniques are the following:

- **binary weighting**: each term is either present or absent from a document as shown in (3.1). This has the advantage of putting large and small documents on the same footing, so that two documents of marginally different size but with a common topic will be represented close to each other in the vector space.

$$\mathrm{bw}(t,d) = \begin{cases} 1, & \text{if } f(x,d) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

- **frequency weighting**: each term is weighted by its raw frequency in the document as shown in (3.2). The main appeal of this weighting technique is that it creates more nuance when representing large documents, where terms which are very frequent in such documents will end up being a lot more important than less frequent terms. Conversely, its main drawback is that this frequency difference can be overstated, where a frequent term might be hundreds of times more frequent than the least frequent term.

$$\mathrm{fw}(t,d) = f(t,d) \tag{3.2}$$

- **log frequency weighting**: each term is weighted by a function of the raw frequency in the document as shown in (3.3). This solves the main drawback of frequency weighting by logarithmically scaling term frequencies, which means that even a very frequent term will never be more than a few times as important as a less frequent one.

$$\mathrm{lfw}(t,d) = \log(1 + f(t,d)) \tag{3.3}$$

- **tf-idf**: each term is weighted using a transformation of the raw frequency multiplied to the inverse document frequency of that term $log(\frac{N}{n})$ where N is the number of documents and n the number of documents that contain that term, as shown in (3.4). The inverse document

	I	not	like	cat	they	(to) be	mean	film	wonderful	president	honourable
D_1	1	1	1	1	1	1	1	0	0	0	0
D_2	0	0	0	0	0	1	0	1	1	0	0
D_3	0	0	0	0	1	2	0	0	0	1	1

TABLE 3.2: Table of logarithmically scaled frequencies

frequency (idf) represents the discriminative power of a term, so that a term that is present in almost every single document will be pulled toward 0. The tf-idf weighting scheme overcomes some of the limitations of log frequency weighting, such as the fact that terms which appear in every single document might have a higher score than terms which only appear in a few despite the fact that appearing in every document means that those terms cannot be used to discriminate between documents, and are thus of no use for text mining.

$$\text{tf-idf}(t, d, n, N) = \log(1 + f(t, d)) \cdot \log(\frac{N}{n}) \tag{3.4}$$

Example 5. We illustrate this using a tf-idf weighting on the three documents of Example 3. Let us assume that the training dataset D contains the following documents:

- D_1 = "I do not like cats, they are mean."
- D_2 = "That film was wonderful."
- D_3 = "Presidents are not as honourable as they used to be."

The complete vocabulary V of the training set, after lemmatising and removing stopwords would be:

I, not, like, cat, they, (to) be, mean, film, wonderful, president, honourable

Now, we can use V to represent all of the documents in D as follows, where the log frequencies are shown in Table 3.2.

We then compute the idf of each term in Table 3.3, where *df* refers to document frequency.

Finally, using Tables 3.2 and 3.3, we can compute the tf-idf scores for the final document representation in Table 3.4.

	df	$\frac{N}{n}$	idf
I	1	3	1.0986
not	1	3	1.0986
like	1	3	1.0986
cat	1	3	1.0986
they	2	1.5	0.4054
(to) be	3	1	0
mean	1	3	1.0986
film	1	3	1.0986
wonderful	1	3	1.0986
president	1	3	1.0986
honourable	1	3	1.0986

TABLE 3.3: Table of inverse document frequencies

	I	not	like	cat	they	(to) be	mean	film	wonderful	president	honourable
D_1	0.33	0.33	0.33	0.33	0.122	0	0.33	0	0	0	0
D_2	0	0	0	0	0	0	0	0.33	0.33	0	0
D_3	0	0	0	0	0.122	0	0	0	0	0.33	0.33

TABLE 3.4: Table of tf-idf scores

We observe that some terms end up having a weight of 0, due to being penalised by the inverse document frequency, even though we could have expected that they would have been considered more important by the term frequency heuristic due to them being more frequent than the other terms.

3.3.2.2 Mean embeddings

Mean embeddings are a compact representation of documents which are built by averaging a word embedding representation, as seen in subsection 3.3.1.2, along the axis of its latent dimensions. It allows us to represent a document using the centroid of the terms present in that document, and is a special case of aggregated embeddings, which were found by De Boom et al. (2016) to perform well at representing short text.

Example 6. In this example we will compute the mean embedding representations of the three documents of Example 3. Let us assume that the training dataset D contains the following documents:

- D_1 = “I do not like cats, they are mean.”
- D_2 = “That film was wonderful.”
- D_3 = “Presidents are not as honourable as they used to be.”

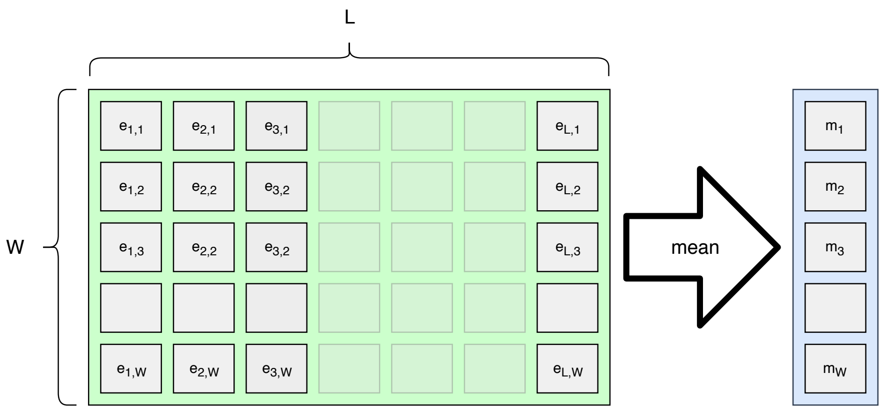


FIGURE 3.5: Mean embedding representation.

$$D_1 = mean(\begin{bmatrix} -0.0055 & -0.0945 & -0.0818 & \dots & -0.0046 & -0.0352 & -0.0427 \\ 0.1148 & 0.0239 & 0.0009 & \dots & -0.0185 & -0.0436 & -0.0224 \\ -0.0595 & -0.0391 & -0.0425 & \dots & 0.0038 & -0.0339 & -0.0046 \\ -0.0035 & -0.0925 & -0.0979 & \dots & 0.0756 & -0.0158 & -0.0211 \\ 0.0299 & -0.0812 & 0.0140 & \dots & 0.0383 & -0.0472 & -0.0224 \end{bmatrix}) = \begin{bmatrix} -0.0214 \\ -0.0039 \\ -0.0199 \\ -0.0065 \\ -0.0115 \end{bmatrix}$$

$$D_2 = mean(\begin{bmatrix} -0.0259 & 0.0031 & -0.0587 & 0.0547 \\ 0.0037 & -0.0404 & 0.0288 & -0.0401 \\ 0.0353 & -0.0165 & -0.0066 & -0.0032 \\ -0.0368 & -0.0069 & -0.0161 & 0.0321 \\ 0.0101 & -0.0167 & 0.0119 & -0.0042 \end{bmatrix}) = \begin{bmatrix} -0.0067 \\ -0.0120 \\ 0.0022 \\ -0.0069 \\ 0.0002 \end{bmatrix}$$

$$D_3 = mean(\begin{bmatrix} 0.0023 & -0.0352 & -0.0818 & \dots & -0.0517 & -0.0438 & 0.0567 \\ -0.0110 & -0.0436 & 0.0009 & \dots & 0.0621 & 0.0316 & 0.0457 \\ -0.0501 & -0.0339 & -0.0425 & \dots & -0.0185 & 0.0099 & 0.1122 \\ -0.0190 & -0.0158 & -0.0979 & \dots & -0.0399 & 0.0130 & 0.0049 \\ 0.0201 & -0.0472 & 0.0140 & \dots & -0.0254 & -0.1125 & -0.0462 \end{bmatrix}) = \begin{bmatrix} -0.0163 \\ 0.0136 \\ 0.0036 \\ -0.0155 \\ -0.0196 \end{bmatrix}$$

3.4 Model building

Model building (also referred to as model fitting, training or learning) is the phase during which the learning algorithm creates a model from the observed training data that is able to generalise to unseen data. We note that this section only makes sense in the context of supervised learning, since unsupervised algorithms are typically not trained on labelled data. In this section we review argument stance mining from a model building perspective.

We start by giving a small taxonomy of machine learning algorithms that allows us to describe each algorithm in terms of degree of supervision and in terms of degree of parametrisation. We then move on to a description of the three main families of algorithms used in the context of this thesis. We start by describing lexicon-based methods, their representation and their learning strategies. We then move on to support vector machines and explain how it is derived from maximum margin classification. Finally, we explain neural networks and more specifically recurrent neural networks, which have been shown to work particularly well with text data.

Under the empirical risk minimisation paradigm, the training process consists of the minimisation of a loss function over a training set, with the assumption that this training set is representative of naturally occurring data. **Stance classification** is a **classification** problem and is linked to our main research objective. Typically, two classification schemes are used: 1. a binary classification scheme such as agreement/disagreement 2. a multi-class classification scheme such as agreement/disagreement/neutral or agreement/disagreement/backchannel, where neutral/backchannel are defined as text which does not take an active stance towards the topic of discussion.

3.4.1 Taxonomy of machine learning algorithms

We consider both unsupervised and supervised techniques in the context of this thesis.

Supervised approaches perform their inference by induction from existing data and require no participation from human experts aside from labelling the data and setting up the hyperparameters of the algorithm;

Unsupervised approaches perform their inference by manual model building from domain experts, requiring those experts to set up heuristics in order for the algorithm to perform well.

Another dimension onto which it is useful to place learning algorithms is their degree of parametrisation. The spectrum of parametrisation considers how strong the assumptions the models are making about the data are, and is in practice determined by the number of parameters required to describe a model. The basic assumptions of a parametric model are the following: 1. We have some knowledge of the probability distribution that the data is drawn from. 2. That probability distribution can be described with some parameters. 3. This number of parameters is finite.

4. There is a procedure that we can use to discover the parameters that were used to generate the data.

The core difference of non-parametric models lies in assumption 3: parametric models can be described using a finite set of parameters, while non-parametric models such as decision trees, k-nearest neighbours or support vector machines cannot due to their ability to shrink or grow their number of parameters with the amount of data available. This has an influence on the inference step of those machine learning algorithms: leaving a non-parametric algorithm to minimise an error function will automatically lead them to overfit on the training data unless some form of regularisation is used. This leads us to distinguish the following classes of supervised classification algorithms:

Non-parametric algorithms are algorithms with a non-finite number of parameters used to describe the model. For example, a support vector machine with a non-linear kernel which has been trained in the dual potentially has a maximum number of parameters that is positively correlated with the number of data points.

Shallow learning algorithms are algorithms with a finite but large enough number of parameters that it would not be reasonable for a human expert to assign them meaningfully. Therefore we require computational procedures such as gradient descent in order to obtain from the data itself the parameters that we deem more likely to generalise well to unseen data. However, we expect to have fewer parameters than training instances. For example, logistic regression has a finite number of parameters that can be found computationally either by optimisation or solved in closed form.

Deep learning algorithms are parametric algorithms with a very high number of parameters, often exceeding the number of data points, making them more likely to suffer from overfitting and requiring special procedures to regularize the resulting models.

3.4.2 Lexicon methods

Lexicons are linguistic tools for the automated analysis of text. Their best known uses are classification and feature extraction Clos et al. (2017), Bandhakavi et al. (2016). They can take many forms, the most common of which is a simple list of terms associated to a certain class of interest. Classification is done by counting the number of terms belonging to each list in a given unlabeled instance, and returning the class associated to the list with the most occurrences. Optionally, the terms can be weighted according to their strength of association with a given class. Some lexicons also contain additional contextual information in order to help their users build more complex models Muhammad et al. (2016), but they all share the same architecture:

Definition 3.1 (formal classification lexicon)**.** A classification lexicon Lex is a tuple $Lex = \langle \mathcal{L}, \mathcal{A}, \mathcal{D} \rangle$ where:

$$\begin{aligned} \mathcal{L}: &\quad T \times C \mapsto \mathbb{R} \\ \mathcal{A}: &\quad \mathbb{R}^n \mapsto \mathbb{R} \\ \mathcal{D}: &\quad \mathbb{R}^n \mapsto \mathbb{R} \end{aligned}$$

For a given dictionary of terms T and set of classes of interest C, $\mathcal{L}$ is a mapping function that assigns an unbounded value to each pair (t, c) where term $t \in T$ and class $c \in C$. The function $\mathcal{A}$ is an aggregation function that accumulates scores and returns one value, and $\mathcal{D}$ is a decision function that selects and returns a single one of these aggregated values. Concretely, the mapping determines an evidence score for each term using a look-up list (the lexicon), propagates it to the aggregation function which aggregates the evidence into one cumulative score per class. Finally, the decision function evaluates each score to select the one that is the most likely.

A core challenge in lexicon-based classification is the lexicon induction problem. The next section reviews techniques traditionally used to solve the lexicon induction problem.

Definition 3.2 (lexicon induction problem)**.** The **lexicon induction problem** is the estimation, given aggregation function $\mathcal{A}$ and decision function $\mathcal{D}$, of the optimal function $\mathcal{L}$ so that the resulting lexicon $Lex = \langle \mathcal{L}, \mathcal{A}, \mathcal{D} \rangle$ minimizes its classification errors on unseen data.

Lexicon induction techniques Research in lexicon induction outlines multiple families of techniques that can be used to produce a computational lexicon. Those techniques are either built on an extensive lexical resource such as an ontology, or on an estimation of strength of association between each term and a class in a reference corpus. Research has shown that merging multiple lexicons produces a reliable feature extractor to augment an existing classifier Wang and Cardie (2014), but using those lexicons for direct classification was not explored.

Traditional hand-crafted lexicons (HCL) Due to the computational cost of building a lexicon from text, early lexicons were hand-crafted by domain experts Stone et al. (1966) and while higher performance in automated classification tasks has been shown using modern techniques, there still exist hand-crafted lexicons in use to this day such as the Linguistic Inquiry and Word Count lexicon Pennebaker et al. (2001). The strengths of these approaches are that they generalize well and are highly interpretable due to their human (and not algorithmic) origin. Conversely their weakness are that they tend to be small due to the human labour involved in generating them, and less effective than other methods due to their focus on human interpretability. However they can provide a common-sense knowledge back-up in hybrid lexicons Muhammad et al. (2014) with some degree of success.

Ontology-based lexicons (OBL) OBL learning techniques use a few human-provided seed words for which the class is known, and leverage some external relationship (typically synonymy, antonymy and hypernymy) in a semantic graph such as WordNet Miller (1995) to propagate class values along that graph Esuli and Sebastiani (2006). Because this family of techniques is extremely foreign to the one we are proposing, we do not evaluate against it and only refer to it for the sake of exhaustiveness.

Corpus-based lexicons (CBL) CBL learning techniques use a labelled corpus of interest in order to learn a domain-specific lexicon. The two main statistics used for this purpose are the conditional probability (equation 3.5) of observing a term given a class, and the point-wise mutual information (PMI, equation 3.6) between the observation of a term and the observation of a class. These approaches are flawed in that they can overemphasize spurious correlations between terms and classes. For example, if a non-class specific term such as “Monday” accidentally co-occurs too often within one class, it will be misconstrued as being indicative of that class, and the lexicon will overfit. Bandhakavi et al. Bandhakavi et al. (2014) describe a method for building conditional probability-based lexicons and Turney Turney (2002) an approach using PMI and an external search engine to compute lexicon scores. Other works Clos et al. (2016) have shown some improvement using the normalized PMI measure (NPMI, equation 3.7) on a stance classification task.

$$P(t;c) = \frac{p(t|c)}{\sum_{i=0}^{|C|} p(t|c_i)} \tag{3.5}$$

$$PMI(t;c) = \frac{\log(p(t;c))}{p(t)p(c)} \tag{3.6}$$

$$NPMI(t;c) = \frac{\frac{\log(p(t;c))}{p(t)p(c)}}{-\log\left[p(t;c)\right]} \tag{3.7}$$

3.4.3 Support vector machines

Support vector machines Hearst et al. (1998) are a powerful family of algorithms built around the notion of simultaneously finding a hyperplane that separates the data from each class as accurately as possible while maximising the distance from this hyperplane to the data points which are closest to it (the support vectors). We illustrate the principle in figure 3.6, where the thick blue line is the separating hyperplane and the thinner light blue lines are the limits of the margin. We can see that on this example the (+) class possesses 2 support vectors which lie on the limits of the margin while the (o) class only possess 1. Because the algorithm attempts to maximise the margin, the data points which are not support vectors are irrelevant to the shape of the separating hyperplane: moving them within their own class boundaries would not change the hyperplane at all. We attempt to build the mathematical intuition of the support vector machine in the rest of this section by giving a quick overview of its different components.

Classification and separating hyperplanes The first step to understand support vector machines is to understand the principles of classifying data using hyperplanes. Hyperplanes are a generalisation of a plane in an arbitrary dimension, and are thus used to classify data in arbitrary dimensions. For instance, a hyperplane in a 3 dimensional space would be a 2 dimensional object (a plane), while a hyperplane in a 4 dimensional space would be a 3 dimensional object (a volume). Hyperplanes are defined in Equation 3.8 for p dimensions.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0 \tag{3.8}$$

From Equation 3.8 we can see whether a data point falls within one side or the other or the hyperplane, as seen in Equations 3.9, 3.10 and 3.11, which gives us the basis for a binary classifier.

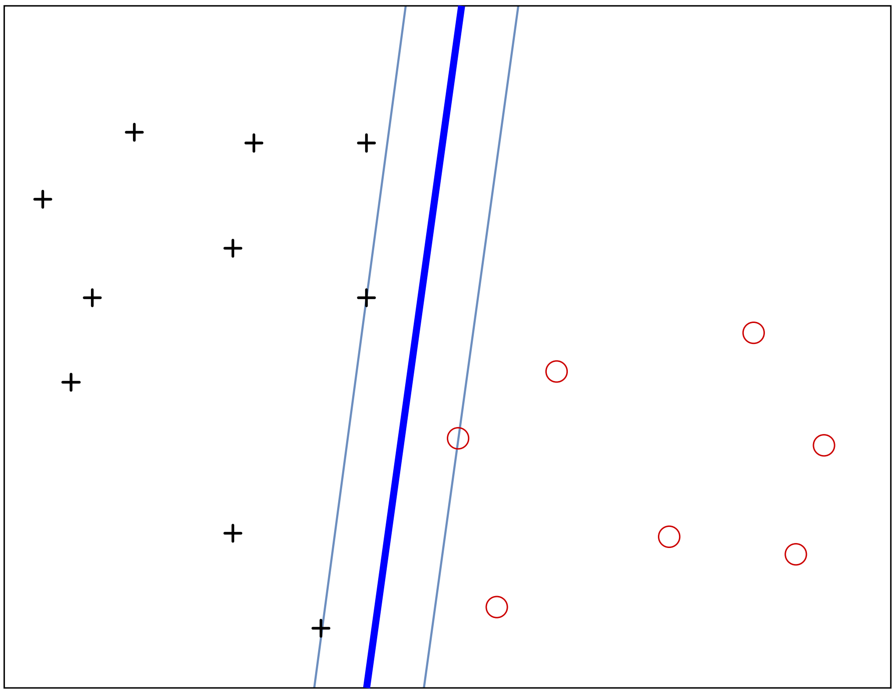


FIGURE 3.6: Example of a binary classification task on a linearly separable dataset with a linear support vector machine. The thick blue line represents the separating hyperplane while the thin blue lines on each side of it represent the limits of the margin which has been maximised during the training process. Objects which appear on the left side of the hyperplane are categorised as the first class (+) and objects appearing on the right side of the hyperplane are categorised as the second class (o).

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0 \text{ if X is on one side,} \tag{3.9}$$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0 \text{ if X is on the other side,} \tag{3.10}$$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0 \text{ if X is on the hyperplane.} \tag{3.11}$$

The β parameter vector which defines the hyperplane can be learned in a variety of ways, which are out of the scope of this thesis. If we encode one class as $+1$ and the other as -1 and consider (x_i, y_i) as one training instance being classified, the following property of a separating hyperplane emerges:

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0 \tag{3.12}$$

This means that for an instance (x_i, y_i) (3.12) will be positive if and only if it is on the correct side of the hyperplane (whichever this side is).

Maximum margin classification The maximum margin classifier can be defined as the hyperplane classifier that maximises the distance between every point and the hyperplane as defined in (3.8). Finding the maximal margin hyperplane consists in solving the following maximisation problem:

$$\text{Maximise }_{\beta_1,\beta_2,\ldots,\beta_p} M \tag{3.13}$$

$$\text{Subject to } \sum_{j=1}^{p} \beta_j^2 = 1, \tag{3.14}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > M, \forall i = 1, \ldots, n \tag{3.15}$$

In this set of equations, Equation 3.13 refers to the maximisation of the margin M which is the distance from the hyperplane to the data points, while Equations 3.14 and 3.15 refer to classifying data points on the correct side of the hyperplane. However, this only works if there is a separating hyperplane that can classify every point correctly as otherwise Equation 3.15 would be violated. In order to overcome this limitation, the support vector classifier is introduced.

Support vector classification In order to overcome the limitation of potentially having data points which cannot be classified properly, support vector classifiers introduce the slack variable ϵ along with the regularisation parameter C. We show the transformed optimisation problem in equations (3.16) to (3.19).

$$\text{Maximise}_{\beta_1,\beta_2,\ldots,\beta_p,\epsilon_1,\epsilon_2,\ldots,\epsilon_n} M \tag{3.16}$$

$$\text{Subject to} \sum_{j=1}^{p} \beta_j^2 = 1, \tag{3.17}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > M(1 - \epsilon_i), \forall i = 1, \ldots, n \tag{3.18}$$

$$\epsilon \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C \tag{3.19}$$

In those equations, ϵ_i refers to the position of observation i with respect to the hyperplane[1], while C refers to the total amount of slack allowed over the entire training set, i. e., the total amount of classification error that is tolerated in order to obtain a large enough margin M. Intuitively, we can see that allowing C to become very large will lead to an increase in classification errors over the training set but will allow the optimisation algorithm to maximise M to a larger degree, while keeping a very low C will lead the algorithm to tolerate fewer mistakes during training but will in consequence of that have less room to maximise the margin M. C is thus a trade-off hyperparameter that needs to be carefully selected depending on the degree of separability of the problem being studied.

Equations 3.16 to 3.19 have the interesting property that the hyperplane is only influenced by a specific subset of data points which are located on the margin. Those data points are called the support vectors and gave their name to the algorithm. Support vector classifiers however are still linear classifiers and are thus plagued with the same limited representation power. Support vector machines use the kernel trick in order to overcome this problem.

Support vector machines A common way to overcome the limitations of linear classifiers is to map the feature into a higher dimensional feature space that allows the classifier to learn a better separating hyperplane. That mapping can be done by building interaction terms (features built from groups of existing features) or transforming each feature individually (e. g., by squaring their value). The kernel trick Hofmann et al. (2008) is a way to do that transformation implicitly by replacing an operation of the solution to support vector classification, the inner product

[1] if $\epsilon > 1$, then the observation is on the wrong side of the hyperplane, if $0 < \epsilon < 1$, then the observation is on the right side of the hyperplane but on the wrong side of the margin, and if $\epsilon = 0$, then the observation is on the right side of the margin.

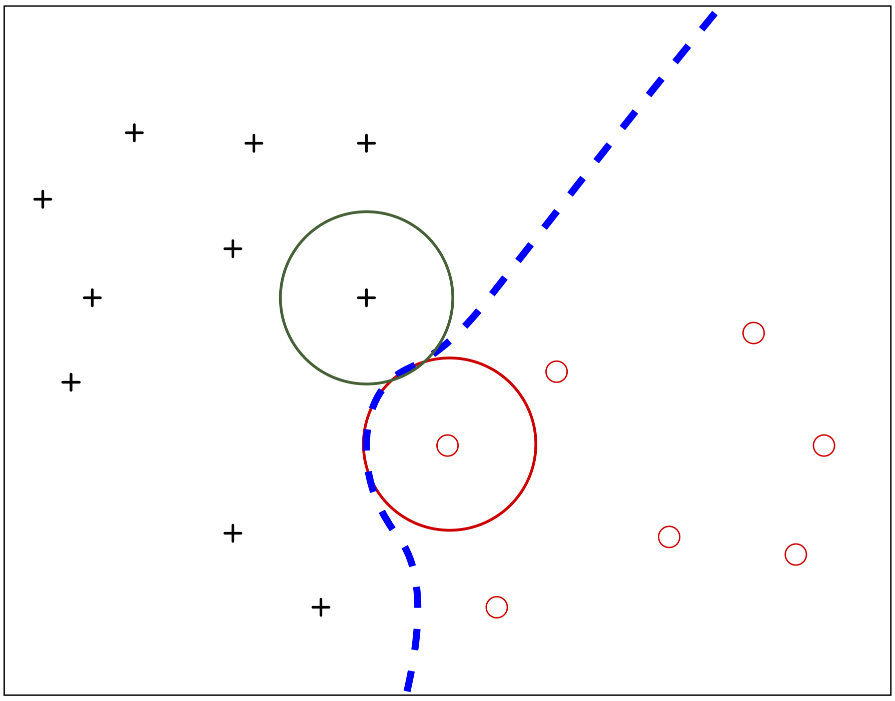


FIGURE 3.7: Example of the influence of 2 support vectors on the shape of the separating hyperplane of a support vector machine.

between data points, by a kernel function that approximates it[2]. Doing so allows us to express the classification rule no more as the sign of the signed distance to the hyperplane but as the sign of a weighted contribution of all support vectors using the kernel function as a similarity metric.

We illustrate in Figure 3.7 the influence of two support vectors on the shape of the decision boundary represented by the separating hyperplane based on such a kernel function. We can intuitively observe that the kernel function induces a non-linearity, which gives kernelised support vector machines their discriminative power.

Support vector machines for N classes Support vector machines are inherently built to handle binary classification problems, but they can be generalised to a multi-class problem. A typical way of handling a multi-class problem is to transform it into a one-vs-all classification, where a N class classification problem is turned into N binary classification problems with one class of interest and one class representing all other classes. We illustrate this process in figure 3.8, where each hyperplane is labelled next to the class it was generated from. The final classification can be done using a weighted vote using the distance to each hyperplane.

[2]In theory such kernel function needs only to satisfy a few specific conditions to be usable in the SVM classifier, known as the Karush-Kuhn-Tucker conditions (Kuhn and Tucker, 1951).

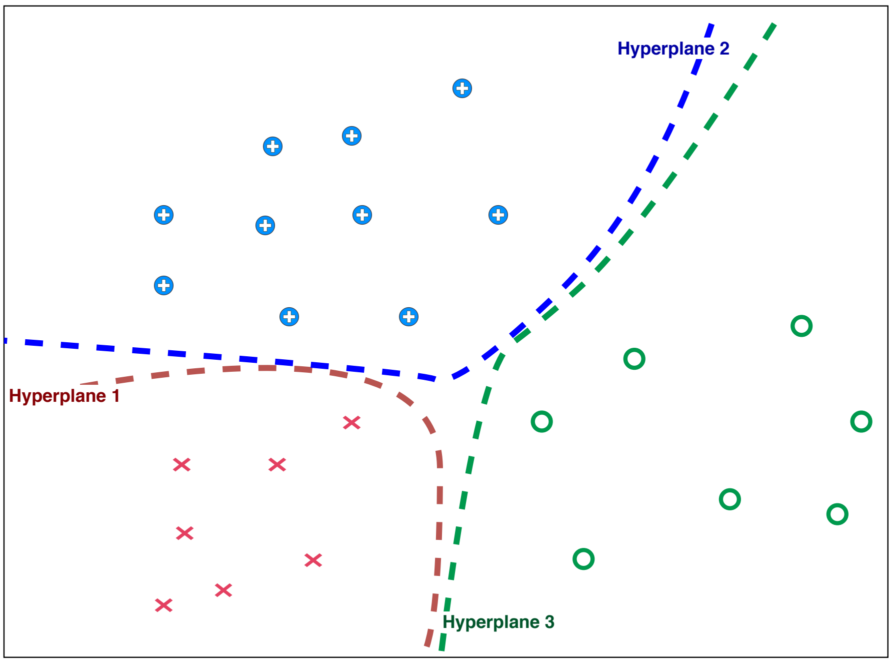


FIGURE 3.8: Example of the different hyperplanes generated by a one-vs-all multi-class SVM.

3.4.4 Neural networks

Neural networks are a family of machine learning algorithms loosely inspired from biological networks of neurons. First popularised by the computational model of nervous activity developed by McCulloch and Pitts (1943), they were developed and extended under the form of the perceptron algorithm Rosenblatt (1958) for supervised machine learning and then the multilayer perceptron (illustrated in Figure 3.9 with 4 input features and 2 layers of 3 neural units), which was trained using the backpropagation algorithm (popularised in the context of neural networks by Rumelhart et al. (1986)). While a single perceptron/neural unit is a linear classifier, the superposition of layers of units creates a non-linear decision boundary where each consecutive layer maps the output of a previous layer into an intermediate space in which the data becomes closer to being linearly separable. The final layer of a neural network is typically a simple linear classifier working on the transformed data of the previous layers.

Deep neural networks Recent advances in hardware have given rise to the use of neural networks with increasing depth. Because the modelling power of neural networks scales exponentially with their depth, it has become more common to use networks of hundreds of layers such as deep residual networks He et al. (2016) for complex classification tasks. This trend towards

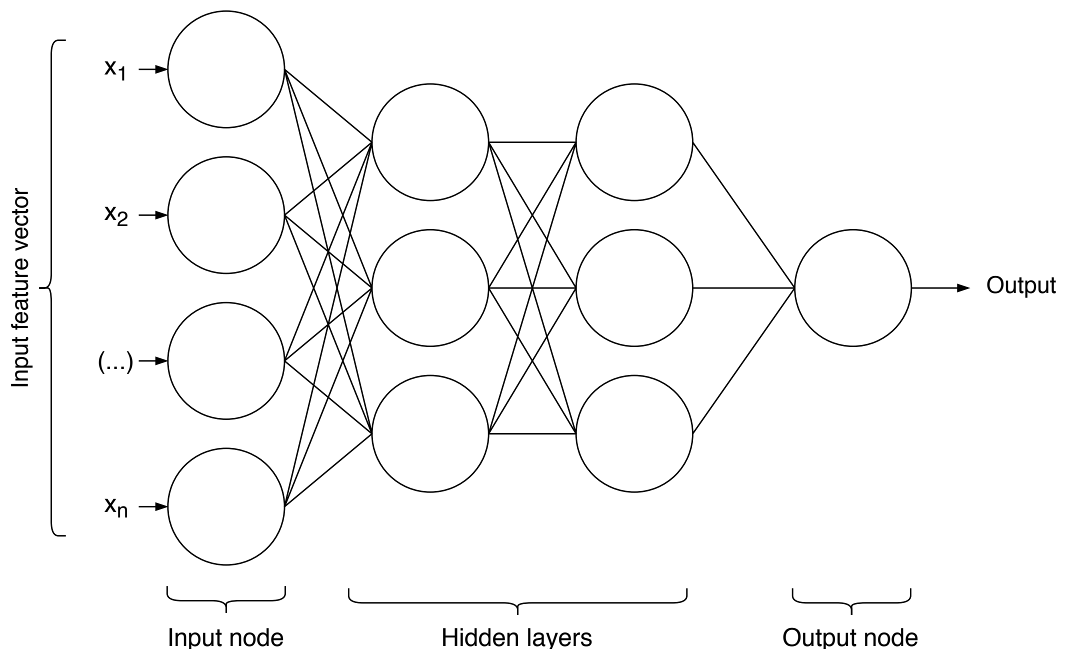


FIGURE 3.9: Example of a 2 layer neural network.

deeper and deeper networks has led to the popularisation of new neural network architectures directed at specific kinds of input, such as recurrent neural networks for sequential data. We introduce recurrent neural networks in the following section, along with their most common implementations: the Elman/Jordan networks and the LSTM/GRU networks.

3.4.4.1 Recurrent neural networks

Recurrent neural networks are neural networks of any shape or form that contain a recurrent relation. This architectural detail means that they are able to parse inputs as sequences, where each step of the sequence is called a timestep. Parsing timesteps one by one, while having a recurrent relationship, means that each timestep has an influence on the following one during forward propagation, and that during the backpropagation step the error signal is propagated through all the timesteps (a process called backpropagation through time, introduced by Mozer (1995)) in order to assign the correct error to each of them. In this section we will give an overview of recurrent networks.

Simple recurrent networks Elman networks and Jordan networks are simple recurrent models popular in signal processing for their simplicity and are sometimes called simple recurrent networks (SRN). Their main distinction lies in the place and form of their recurrent relation.

- Elman networks. The Elman network was conceived by Jeff Elman Elman (1990). The main characteristic of Elman networks was that they were the first model that did not commit to a specific linguistic unit of meaning like phonemes, words, or sentences but instead focused on the smallest potential granularity in order to explore the possibility that these units of meaning could be a simple consequence of a learning process over the latent structure of the input. Elman networks are three-layer neural networks that possess context units, which are units built to contain the context of the current input timestep. The middle layer is connected to those context units with a fixed weight of one in order for backpropagation to not cancel them during the learning process. At each timestep during the parsing of the input, the input is fed-forward, generating an output for that specific timestep. Once the final timestep is parsed, the final output is generated by the network and it can be compared to the real output, allowing us to backpropagate the error through all the different timesteps in a process called backpropagation through time. The connection from contextual units to the middle layer allow the network to maintain an internal state, which make it powerful for modelling sequences such as speech, text, or sensor data. We show an example of Elman network architecture in Figure 3.10, where we can see that m context units have been added as input to the existing n input units. Those context units receive information from the recurrence relation that is formed from the output of the last hidden layer. This architectural feature is not required: the recurrent signal can be fed from the middle or the beginning of the hidden layers.
- Jordan networks. Jordan networks Jordan (1997) are identical to Elman networks with the exception that the context units are fed from the output layer instead of the hidden layer.

Gated recurrent networks Long Short-Term Memory (LSTM) networks are gated architectures, i. e., neural architectures that make use of a gating mechanism in order to control the error propagation and direct it towards the relevant neural units, so that it does not fall prey to vanishing and exploding gradient issues. LSTM networks use a complex 3 gate architecture for each LSTM unit: a **forget gate** which decides whether the current state of the long-term memory should be

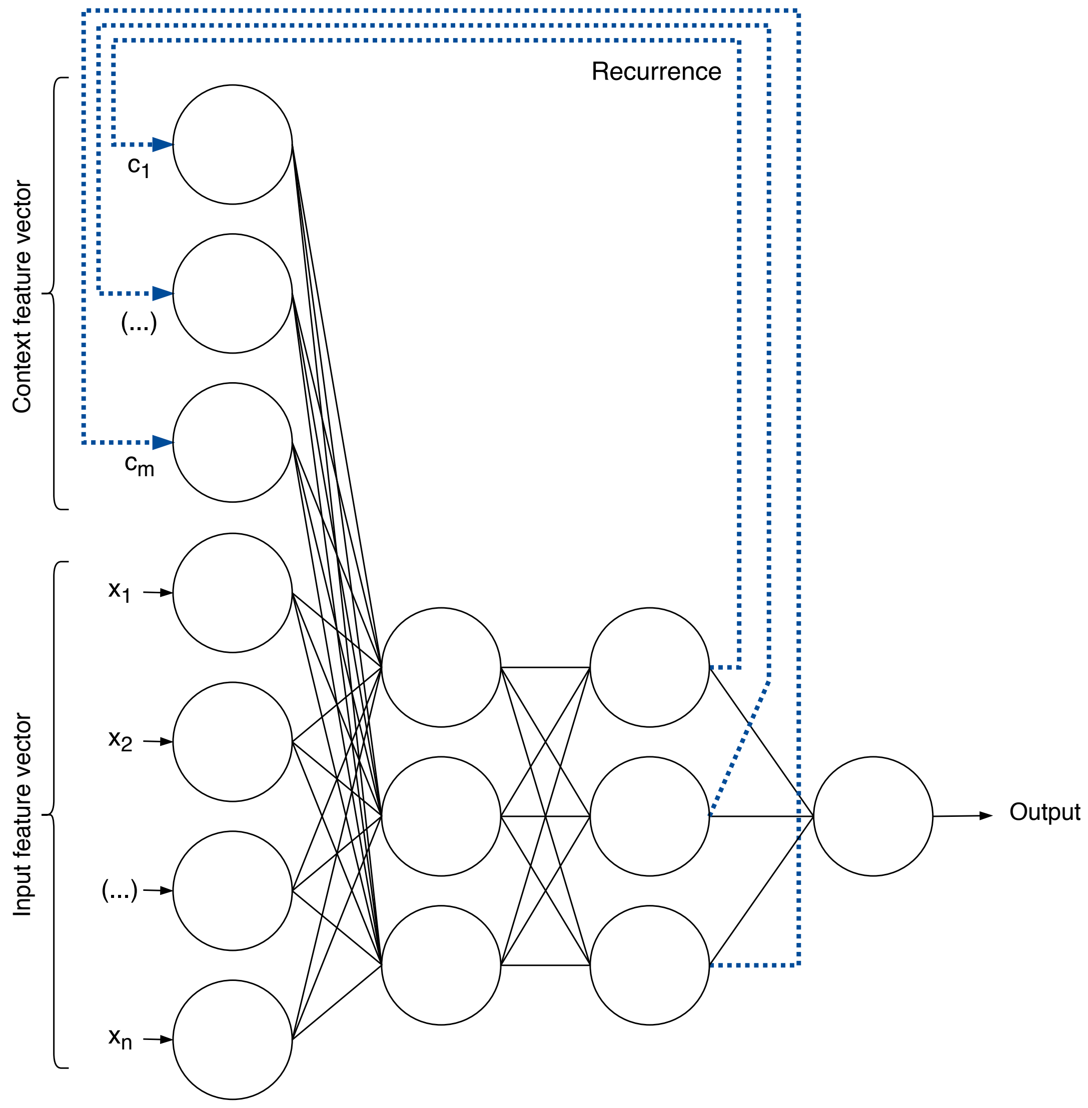


FIGURE 3.10: Example of an Elman network architecture.

reset, an **input gate** that does the transformation of the current timestep into a potential update to the long-term memory, and an **output gate** that takes the long-term memory as well as the transformation of the current timestep as inputs and produces a temporary output corresponding to this timestep.

We can see in Figure 3.11[3] the multiplicative and additive gates controlling the information flow within the LSTM unit. The higher horizontal line, called the cell state, can only be accessed through the additive and multiplicative gates, which makes it easy for information to be propagated through several timesteps without being modified. The lower horizontal line is a more common recurrence relation similar to the one that can be observed in Elman networks.

[3]Adapted from http://colah.github.io/posts/2015-08-Understanding-LSTMs

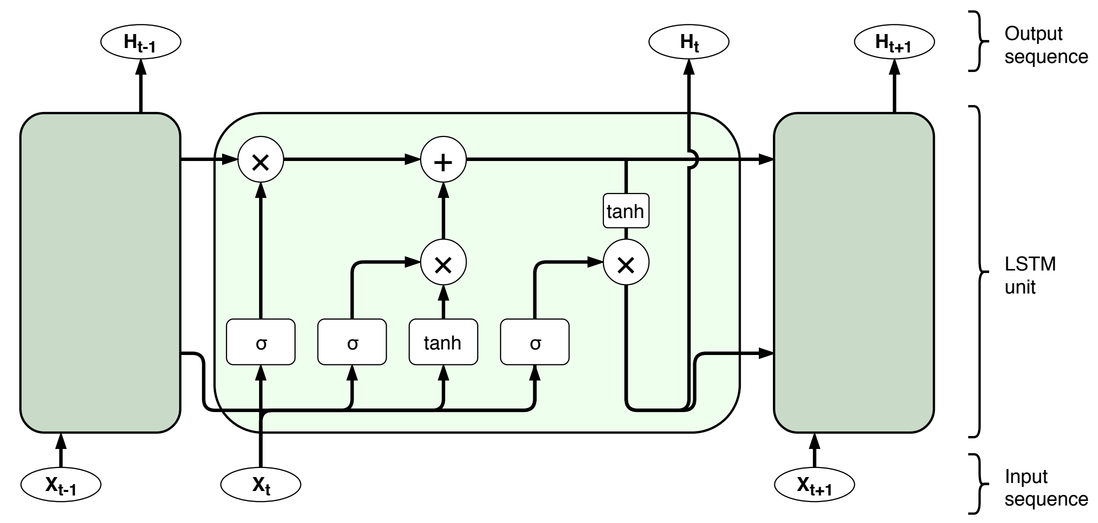


FIGURE 3.11: A LSTM unit.

3.5 Model evaluation

The final phase of the supervised learning process if the evaluation of the model built and trained during the model building phase. The evaluation is handled in details in the first section of chapter 4.

3.6 Chapter Summary

In this chapter we established the background knowledge necessary to understand the rest of this thesis. We introduced text processing techniques necessary for those tasks, before moving on to representation schemes and machine learning models that will be used throughout this thesis.

Chapter 4

Evaluation methodology

In this chapter, we focus on presenting the evaluation methodology that will be used throughout our thesis. We begin by explaining the task of evaluating supervised learning algorithms in a general sense, before moving on to the datasets that will be used in our evaluation. We then discuss the statistical testing procedure that is used to test for statistical significance in our work. Finally, we highlight and discuss the flaws of our procedure, their reason, and the ways they may be remedied in future work.

4.1 Evaluating classifiers

In this section we give an overview of a machine learning evaluation methodology, starting with the K-fold cross-validation methodology before detailing the evaluation metrics that will be used throughout this thesis.

4.1.1 Machine learning evaluation methodology

We perform our evaluation using a procedure called k-fold cross-validation. K-fold cross-validation is a procedure meant to provide an accurate estimate of the generalisation error as follows: 1. if we consider the dataset to be fairly sampled, compute the class proportions in the entire dataset ; 2. divide the original dataset into K partitions where the proportions of each class as previously calculated are respected ; 3. for each of the K partitions, use it as a test set while training the algorithm on the union of the remaining $K - 1$ datasets ; 4. average the accuracy or arbitrary metric of goodness for those K trials.

Figure 4.1 illustrates a 4-fold cross-validation. In this example fold 1 would be used for testing, while folds 2, 3, and 4 would be used for training. The roles are then rotated so that each fold is used at least once for testing.

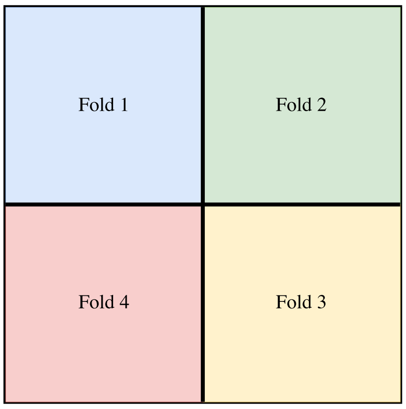


FIGURE 4.1: Standard 4-fold cross-validation split

This procedure allows us to gather an accurate estimate of the generalisation error, which allows us to accurately judge the quality of the hyperparameters used during the training process. However, cross-validation also allows us to obtain an accurate estimate of the performance of an algorithm after tuning its hyperparameters by using the process of nested cross-validation, illustrated in Figure 4.2.

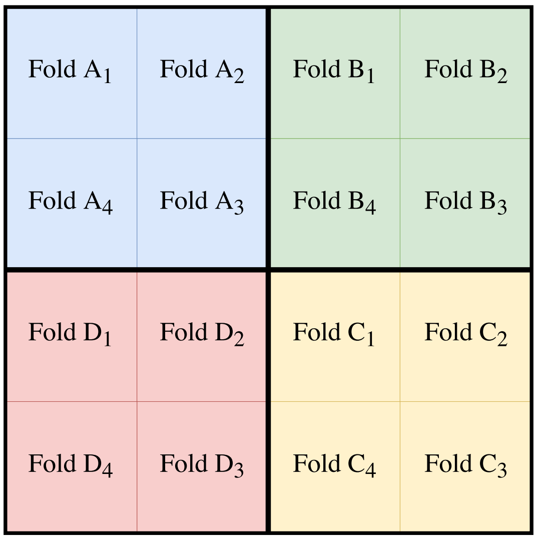


FIGURE 4.2: Nested 4-fold crossvalidation split

		Actual class		
	Total	Class A	Class B	
Predicted class	Class A	True Positives (TP)	False Positives (FP)	Accuracy $= \frac{\text{TP+TN}}{\text{Total}}$
	Class B	False Negatives (FN)	True Negatives (TN)	
		TP Rate $= \frac{\text{TP}}{\text{Class A}}$	FP Rate $= \frac{\text{FP}}{\text{Class B}}$	
		FN Rate $= \frac{\text{FN}}{\text{Class A}}$	TN Rate $= \frac{\text{TN}}{\text{Class B}}$	

TABLE 4.1: An example of confusion matrix for binary classification with classes A and B

When using a nested cross-validation, the inner cross-validation is done in order to tune hyperparameters, while the outer cross-validation is done in order to obtain an estimate of the true error of the algorithm on the data. In this example, fold A would be the first one to be used for testing, so folds B, C and D would be used for the inner cross-validation. In the inner cross-validation, folds B_1, C_1, and D_1 would be used for testing while the other 9 folds would be used for training. Then, folds B_2, C_2, and D_2 would be used for testing while the other 9 folds would be used for training. Then, folds B_3, C_3, and D_3 would be used for testing while the other 9 folds would be used for training. Finally, folds B_4, C_4, and D_4 would be used for testing while the other 9 folds would be used for training. The accuracy of the inner cross-validation is then used to assess the quality of the hyperparameters and tune them accordingly, either manually (using arbitrarily chosen values) or programmatically (using hyperparameter search).

4.1.2 Evaluating classification algorithms

In this section we will discuss the evaluation of classification algorithms. There are multiple metrics that can be used to evaluate classification. Classification results are reported using a confusion matrix, from which those metrics are calculated. We present in Table 4.1 an example of a confusion matrix for a binary classifier. This example shows the way the main evaluation metrics are computed from their core elements: true/false positives, and true/false negatives.

In the case of single class binary classification, the following metrics are typically reported:

Precision is calculated as $\frac{TP}{TP+FP}$. It indicates, for a given class, the ability of the algorithm of not making mistakes when it identifies it, i. e., minimise the false positive rate ;

Recall , or TPR (True Positive Rate), is calculated as $\frac{TP}{ClassA}$. It indicates, for a given class, the ability of the algorithm of not making mistakes when identifying that an object is of a different class, i. e., minimise the false negative rate ;

Accuracy is the inverse of the misclassification rate, and is calculated as $\frac{tp+tn}{tp+tn+fp+fn}$. It is typically calculated as an aggregated measure of goodness for a classifier since it corresponds to the ability of the classifier of not making any mistakes. For certain datasets, weighted accuracy (accuracy which is weighted by the number of instances for each class) can be preferable to accuracy.

F_β**-Score** is a score that aims to balance precision and recall to overcome cases where classes are not balanced, and is calculated as $(1+\beta^2) \cdot \frac{(\text{precision} \cdot \text{recall})}{(\beta^2 + \text{precision}) \cdot \text{recall}}$. The most common form of F-Score uses a balancing parameter β of 1, which puts an equal weight on both classes, becoming $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$.

For classification involving more than two classes, accuracy and F_1-Score are usually reported in order to account for potential class imbalances. Accuracy allows us to have a general idea of how well the algorithm is performing, while F_1-Score allows us to know whether a class is favoured over the others since having a high accuracy in one class and a very low accuracy in another will penalise that score.

The task at hand is the classification of a set of items, called the test set, by a classifier which has been trained using a set of labelled items, called the training set. The predictions produced by the classifier are then compared to the actual labels of the test set, also called the ground truth, to measure a few performance metrics.

- Precision (or specificity) is computed as $\frac{\text{number of true positives}}{\text{number of true positives} + \text{false positives}}$
- Recall (or sensitivity) is computed as $\frac{\text{number of true positives}}{\text{number of true positives} + \text{false negatives}}$
- Accuracy is computed as $\frac{\text{number of true positives} + \text{number of true negatives}}{\text{number of true positives} + \text{number of true negatives} + \text{false negatives} + \text{false positives}}$

These three basic metrics are used to compare classifiers on similar training and test sets.

Dataset	Post count	Agreement	Disagreement	None
ABCD	185,479	38,195	60,991	86,293
IAC	5,940	428	1,236	4,276
SETC	4,163	1,055	2,112	996
RNLC	59,458	39,984	19,474	0

TABLE 4.2: Descriptive statistics of our selected four stance detection datasets.

4.2 Datasets

In this section we present the datasets that are used for the purpose of our evaluation. We show in Table 4.2 descriptive statistics of those datasets.

Free form social media corpora are extracted from forums and are characterised by the absence of restriction in user-to-user interactions. Unlike transcribed media corpora, they do not contain any information besides the actual text content and some forum metadata (username, dates, times, etc.). They also have the particular feature of being written by the users themselves, which leaves a lot more room for typos, grammatical mistakes, and internet slang.

4.2.1 Internet Argument Corpus (IAC)

The IAC (Internet Argument Corpus) (Walker et al., 2012a) was built from posts collected from the 4forums[1] discussion platform. It was annotated using Amazon Mechanical Turk[2] on a -5 to 5 disagreement to agreement scale. The details of the dataset are described in Section 2.2.2.

4.2.2 Agreement By Create-Debaters (ABCD)

The ABCD (Agreement By Create-Debaters) corpus (Rosenthal and McKeown, 2015) was collected on a specialised debating forum called CreateDebate[3]. Debates on this platform can be

[1]https://4forums.com
[2]https://www.mturk.com/mturk/welcome
[3]https://www.createdebate.com

of three types: open-ended (no side), for/against (two sides), multiple sides (three or more sides). Details on the datasets are described in Section 2.2.2.

4.2.3 SemEval Twitter Dataset (SETC)

SemEval (for **Sem**antic **Eval**uation) is an ongoing series of evaluations of computational semantic analysis systems, organized under the umbrella of SIGLEX, the Special Interest Group on the Lexicon of the Association for Computational Linguistics. SemEval has evolved from the SensEval word sense disambiguation evaluation series.

4.2.4 Reddit Noisy-Labelled Corpus (RNLC)

Reddit[4] is a social link aggregation website which allows its users to submit posts or links to other web pages as well as comment on them (top-level comments), or on other comments. This creates a particular tree structure which departs from traditional flat forums in that they indicate explicitly the target of the comment (what it is commenting on) and allow for an easier understanding and structuring of the discussions, allowing them to branch off without polluting the main topic.

The Reddit Noisy-Labelled Corpus is a novel dataset that leverages noisy labelling in order to build a dataset that is significantly larger than most manually labelled ones. As such, it is extremely useful to evaluate attempts at transfer learning by using models learned on an unreliable dataset (RNLC) to classify instances from a reliable dataset. We describe the process of its development in the rest of this section.

Step 1: Collection The source data was collected by crawling all comments posted on Reddit using their JSON endpoint[5] and storing the resulting comments crawled in a MongoDB[6] database. We used data that was collected over 8 months, from December 2013 to July 2014.

[4]`https://www.reddit.com`
[5]For instance, `https://www.reddit.com/r/all/.json`
[6]`https://www.mongodb.com`

Step 2: Filtering Reddit is one of the most visited websites[7] and therefore attracts an international audience and a very variable level of content. Because of this, we then proceeded to filter comments that did met a certain set of criteria:

- Comments that were not written in English[8] ;
- Comments that were shorter than either 200 characters or 20 words, as an arbitrary filter of low-quality content ;
- Comments that contained an indication that they were continued over multiple comments. Because of the technical limitations imposed by Reddit, some comments require being posted either for structural or technical reasons in multiple "episodes". Because of the way the data is collected there is no way to be sure that we have access to the complete comment and therefore it is easier to discard such content. Marks of continuation are sentences such as "Part x of x", or "Continued from previous comment".

Additionally, quotes from a parent post were removed from the text in order to only keep the child comment and not lead the automated labelling to mistakes.

Step 3: Noisy labelling Acquiring user-labelled data is a common issue in machine learning and information retrieval tasks. The quantity and quality of data needed usually require the labelling of large numbers of examples by multiple annotators. One way to go around the problem of finding qualified annotators is to use crowd-sourcing (Wang et al., 2012) (outsourcing to a crowd of anonymous workers, at a low cost), which brings its own problems in terms of quality and consistency of labelling (discussed by Snow et al. (2008)). Another way to circumvent the limitation of human-labelled data is the use of distant supervision (Mintz et al., 2009). Distant supervision consists of using high-precision clues called noisy labels in the examples to automatically label data and use them as labels, before removing them to not bias the classifiers. Go et al. (2009) for example used distant supervision to annotate a tweet dataset using emoticons as noisy labels. We adapt the approach used by Mintz et al. (2009), Go et al. (2009) to our dataset by mining it for highly precise evidence of agreement and disagreement. We extract only the

[7]Alexa ranking

[8]The language detection was done using the LangDetect Python library, available at `https://pypi.org/project/langdetect/`

Id	Agreement expression
a_1	i [adverb] [agreement verb]
a_2	i (don't|do not) [adverb] [disagreement verb]
a_3	i (am not|ain't) [disagreement noun/adjective]
a_4	(you are|you're) [adverb] [agreement noun/adjective]
a_5	you (are not|aren't|ain't) [adverb] [disagreement noun/adjective]
a_6	[this is] [adverb] [agreement NAO] [EOS]
a_7	[this is] not [adverb] [disagreement NAO] [EOS]

TABLE 4.3: High-precision expressions indicative of agreement class

Id	Disagreement expression
d_1	i [adverb] [disagreement verb]
d_2	i (don't|do not) [adverb] [agreement verb]
d_3	i (am not|ain't) of the same (mind|opinion)
d_4	(you are|you're) [adverb] [disagreement NAA]
d_5	you (are not|aren't|ain't) [adverb] [agreement NAA]
d_6	[this is] [adverb] [disagreement NAO] [EOS]
d_7	[this is] not [adverb] [agreement NAO] [EOS]

TABLE 4.4: High-precision expressions indicative of disagreement class

comments which show both evidence of one class and the absence of evidence of the other class, so as to prioritise precision over raw number of examples.

We show in Table 5.2 and 5.3 the expressions used to extract instances of the agreement and disagreement classes (respectively) where [EOS] marks an end-of-sentence marker such as an exclamation mark, a full stop or a comma, [] marks optional expressions or positive/intensifying adverbs such as "extremely", "totally", etc, NAO marks a Noun/Adjective referring to the Object of the discussion and NAA refers to a Noun/Adjective referring to the Author that is being replied to. In tables 4.5, 4.6, 4.7, 4.8, 4.9 and 4.10 we detail the dictionary of words used to feed those expressions.

Agreement verb
agree
concur
am of the same mind
am of the same opinion
share your opinion
share this opinion
sympathise
sympathize
concede
acquiesce

TABLE 4.5: Verbs indicating agreement

Agreement NAA
accurate
veracious
correct
bang on
dead on
on target
right on target
spot on

TABLE 4.6: Nouns/adjectives indicative of agreement class, targeting the author

Agreement NAO
accurate
exact
true
correct
without error
valid
bang on
dead on
on target
right on target
spot on
perfect

TABLE 4.7: Nouns/adjectives indicative of agreement class, targeting the object

Step 4: discussion tree building The last step once the entire dataset is reduced to labelled instances is to filter out the ones that are orphans, i. e., only keep the user comments for which we can retrieve a parent comment. Parent and child comments are then joined together in the final version of the dataset. This enables us to experiment on context-sensitive classification, by allowing algorithms to access the content of the parent comment in order to improve the prediction of the class of the child comment.

Limitations of noisy labelling The use of distant supervision to extract noisy labels allows us to build a large stance classification dataset, but it comes with significant drawbacks:

- confidence in the labels is low, due to the automatic nature of the task. For instance, the automatic labelling from high-precision rules does not cover edge cases such as sarcasm.
- biased sampling of the classes, due to the absence of reliable high precision rules that cover some classes such as neutrality. Additionally, it is sometimes more common for people

Disagreement verb
dissent
object
oppose
differ
disagree
fail to agree
dispute this
challenge this
argue this
take issue with this

TABLE 4.8: Verbs indicating disagreement

Disagreement NAA
incorrect
mistaken
confused
misguided
misinformed
misled
erroneous
untrue
deceived
misleading

TABLE 4.9: Nouns/adjectives indicative of disagreement, targeting the author

Disagreement NAO
incorrect
erroneous
false
inaccurate
inexact
untrue
deluded
delusional
duped
tricked
invalid
fallacious
off target
misleading
illogical
unsound
unfounded
without foundation
misconstrued
unsubstantial
unsubstantiated

TABLE 4.10: Nouns/adjectives indicative of disagreement, targeting the object

expressing agreement to explicitly state their stance, which leads to highly imbalanced classes.

4.3 Statistical testing for classification

We use a consistent evaluation methodology throughout the entire thesis. Firstly, datasets are artificially balanced by subsampling the majority class in order to be able to use accuracy as a reliable metric. Other metrics such as F1-Score are less sensitive to class imbalances but have less applicability to a real-world setting: while it is trivial to set an accuracy threshold to determine an algorithm reliable for real-world applications, doing so with an F1-Score is less obvious.

Secondly, datasets are partitioned into 10 different subsets for the purpose of 10-fold cross-validation. We then create 10 training/testing sets pairs that will be used for evaluation. For unsupervised algorithms, the training set is used in order to empirically decide some parameters such as thresholds. For supervised algorithms, the training set is used to learn the optimal classification model and no hyperparameter tuning is done, removing the need for an external validation set. For transfer learning algorithms, the training set is discarded while the model is learned from a separate dataset. In all cases, the test set is used to compute accuracy metrics.

Thirdly, we perform the statistical analysis of our results in two steps. We start using a one-way ANOVA over each dataset in order to detect whether there is one algorithm that is significantly different from the rest. We then conduct a post hoc Tukey test in order to perform pair comparison and find which algorithm differs significantly from others, while minimising the risk of type 1 error. The tests are done using the base package of the R statistical computing environment[9] (R Core Team, 2014).

4.4 Chapter Summary

In this chapter we briefly introduced our evaluation methodology, starting with the datasets collected for this research and finishing with the statistical testing procedure used to formally test the significance of our results.

[9]`https://www.r-project.org`

Chapter 5

Unsupervised Argument Mining

Unsupervised argument mining makes use of intelligent heuristics in order to provide a classification function that is able to distinguish between argumentative stances without being trained on a corpus. It does so with under-parameterised models (i. e., models with a very restricted set of parameters) which can be tuned by human experts, and therefore do not require any training data. Due to the lack of high quality data in the domain of argument stance mining, it is a gap that unsupervised algorithms can fill with success. In this chapter we investigate unsupervised models for stance mining, which can be derived from existing resources and heuristic rules.

We mostly focus on the approximation of stance by the way of augmenting sentiment lexicons with shallow natural language processing rules. Because sentiment, opinion and stance are closely related concepts, a large part of the literature uses sentiment as a core feature to detect the polarity and strength of argumentative stance. The research question we aim to answer is as follows: can local context be used to improve classification accuracy by helping a naive classifier to focus on relevant terms? Additionally, we aim to build a hierarchical model that can take advantage of the strengths of multiple heuristics in order to improve its classification accuracy.

We introduce five different approaches in order to perform this investigation:

- Two sentiment as proxy approaches: sentiment as proxy with adaptive threshold (SP-ADAPT), and its augmented version: sentiment as proxy with context-sensitive term selection (SP-COSTS) ;

- Two shallow linguistic approaches: unsupervised sentiment surface (USS) and its augmented version (USS+) ;
- A unified approach that organises multiple heuristics in order to improve its classification power (MUSC).

5.1 Sentiment as proxy approaches

In this section we elaborate on computational models of sentiment as an approximation of stance. Sentiment can be seen as a generalisation of stance, and therefore it stands to reason that it can be used as an approximation of it in a natural language processing workflow with reasonable accuracy.

5.1.1 Sentiment as proxy with adaptive threshold (SP-ADAPT)

Sentiment-guided methods provide a means to use emotive context to infer argument stance, by assuming the local stance of a comment to be equivalent to its sentiment valence. We employ a simple sentiment analysis algorithm based on a pre-existing sentiment lexicon Esuli and Sebastiani (2006) to which we will refer as SENTLEX. It operates by looking up positive and negative values of all the terms present in the comment and summing those two components separately into a positive and a negative score. The classification rule is based on a simple comparison: a larger positive strength implies a supportive comment, and a larger negative strength implies an attacking comment. This algorithm uses an additional threshold parameter in order to account for variations in baseline sentiment valence: some topics of discussion might yield a higher natural threshold, meaning that even a slightly disagreeing user comment could have a positive overall sentiment. We estimate the optimal threshold by performing a linear search on a held out dataset, for example by computing the accuracy of the approach using a threshold from -5 to 5 by steps of 0.1, and using the threshold for which classification accuracy was maximised.

In our work we use the SMARTSA algorithm developed by Muhammad et al. (2013). SMARTSA was developed as an extension of traditional lexicon-based sentiment analysis techniques in order to take into account additional linguistic factors such as the presence of contextual valence shifters Polanyi and Zaenen (2006). Contextual valence shifters are terms which alter the class values of terms in their vicinity by exaggerating them (amplifiers), reducing them (diminishers) or inverting their polarity (negators). Example 7 illustrates those modifiers with concrete examples.

Example 7. **Amplifiers**: "I am very happy that you said this", where the positive value of "happy" is amplified by the presence of "very" ;

Diminishers: "I am mildly annoyed by advertisements", where the negative value of "annoyed" is hedged by the presence of "mildly" ;

Negators: "She was not satisfied by her purchase", where the positive value of "satisfied" is negated by the presence of "not".

Sentiment-guided methods assume that global stance of the comment, i. e., the feelings of its author regarding the topic at hand, can be used in place of its local stance, i. e., the feelings of its author towards a specific target. As such, those approaches are flawed whenever (1) those two stances do not align, (2) the stance is expressed in a sentiment-neutral way, or (3) the overall sentiment of the sentence ends up being balanced. Example 8 illustrates those problems.

Example 8. (1) "*I completely disagree with you, this movie was very good and I enjoyed every minute of it.*" Here we can see that the author expresses a positive opinion by disagreeing with the author of the parent comment, thus making an attacking statement.

(2) "*There is nothing in the world that will make me see the situation your way.*" In this comment there is no positive or negative terminology used, while the sentence is clearly written with an attacking stance.

(3) "*I agree that the acting was good, but I am still disappointed that the dialogues were so poorly written.*" Here negative and positive sentiments are equally used, but the stance should be supporting.

SP-ADAPT works by using a sentiment lexicon and tuning its threshold parameter on a training set for argument stance mining. We describe the process in algorithm 1. Concretely, it takes as input a document C and a threshold number α, sums up the positive and sentiment scores of a specific document and applies a simple binary decision rule based on a user-specified threshold.

Corpus	Topic	Optimal α
ABCD	Abortion	1.65
	Gay rights	-1.35
	Marijuana	-0.05
	Obama	-0.10
IAC	All	-0.15

TABLE 5.1: Optimal α thresholds on two datasets with SP-ADAPT

If the positive sentiment outweighs the negative sentiment by an amount superior or equal to that threshold, the agreement class is returned as output. Otherwise, the disagreement class is returned as output.

Algorithm 1 The SP-ADAPT algorithm for binary stance classification.

```
1: function SP-ADAPT($C, \alpha$)                                  ▷ Stance of C with threshold $\alpha$
2:     $Sentiment \leftarrow (0, 0)$          ▷ Initialise the Sentiment vector as an empty vector
3:     for all Term $t$ in $C$ do
4:         $Sentiment[1] \leftarrow Sentiment[1] + PositiveSentiment(t)$
5:         $Sentiment[2] \leftarrow Sentiment[2] + NegativeSentiment(t)$
6:     end for
7:     if $Sentiment[1] - Sentiment[2] \geq \alpha$ then          ▷ Test whether positive sentiment
   outweighs negative by a margin of $\alpha$
8:         Return agreement
9:     else
10:        Return disagreement
11:    end if
12: end function
```

One major flaw of this approach is that a crucial parameter, the threshold α, needs to be determined empirically. A cursory analysis on two stance classification datasets show that the optimal value can vary widely and therefore a parameter search needs to be done for specific types of data. Table 5.1 shows different optimal values of α with respect to classification accuracy for the ABCD corpus (for 4 topics) and the IAC corpus. We can observe a large deviation from dataset to dataset and from topic to topic.

5.1.2 Sentiment as proxy with context-sensitive term selection (SP-CoSTS)

Term vector representation is a convenient way to work with text data because of its simplicity and versatility. It is the required representation for lexicon-based classification as well as more standard supervised classifiers and is thus a standard in text mining. In this section we use conversational context to alter that feature vector by adding or removing some of its components (terms) based on the terms present in the feature vector of the parent post. It is essentially an augmented version of the SP-ADAPT algorithm based on the intuition that selecting terms can help the algorithm know which part of the child post is referring to the parent post, which leaves the sentiment analysis algorithm to detect the child post attitude toward the parent post.

The general methodology is described in algorithm 2. It consists in merging two bag of word vectors by applying an aggregation function f to each dimension of those vectors. We define four variations of the method, by varying the aggregation function f used to build the context-enriched term vector from parent and child comment (p and c respectively). Its inputs are two bag-of-word vectors c and p and an aggregation function $f : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$. Its output is a bag-of-word vector.

Algorithm 2 Context-based term vector enrichment algorithms with input documents c and p, aggregation function f.

```
1: function GENERALCOMBINATION($c, p, f$)
2:     x ← $\langle 0, 0, \ldots, 0\rangle$                    ▷ Initialise an empty bag-of-words
3:     for $i \leftarrow |x|$ do
4:         $x_i \leftarrow f\,(p_i, c_i)$           ▷ element $i$ of $x$ is a mixture from $p$ or $c$
5:     end for
6:     Return $x$
7: end function
```

SP-COSTS$_1$ uses an intersection-based term vector enrichment strategy. In this approach, we build the context-sensitive term vector as the intersection of the parent and the child posts. More formally, the function f is defined as follow:

$$f\,(p_i, c_i) = min\,(p_i, c_i) \tag{5.1}$$

SP-COSTS$_2$ uses an union-based term vector enrichment strategy. We describe the algorithm used to build the term vector as follows:

$$f\left(p_i, c_i\right) = max\left(p_i, c_i\right) \tag{5.2}$$

SP-COSTS$_3$ uses an additive combination model as its term vector enrichment strategy, by adding the frequencies.

$$f\left(p_i, c_i\right) = p_i + c_i \tag{5.3}$$

SP-COSTS$_4$ uses a random mixture-based term vector enrichment strategy, by randomly sampling term frequencies from the parent and the child comments with a given probability. We illustrate it as follows, where t is the threshold over which the term frequency is sampled from the child comment.

$$f\left(p_i, c_i, t\right) = \begin{cases} c_i, & \text{if } \sim B(1,t) \\ \\ p_i, & \text{otherwise} \end{cases} \tag{5.4}$$

In this equation, a random number is sampled from the binomial distribution (parameterised with t) for each term, returning 1 (sample from child comment) or 0 (sample from parent comment).

Algorithm 3 represents the general classification algorithm of SP-COSTS, where the changing part is the way the instance c^+ is created from the child comment c and the parent comment p. The classification is done by first combining the bag of words of both parent and child comments, and then summing the positive and negative sentiment scores of each term of that combination. Finally, if the positive sentiment passes or equals a specific user-specified threshold, the agreement class is returned. Otherwise, the disagreement class is returned.

5.2 Shallow linguistic approximations

Shallow linguistic approximations are a class of methods that use linguistic heuristics to classify text without relying on complex natural language processing techniques. While they lose accuracy

Algorithm 3 The SP-CoSTS algorithm for binary stance classification, with documents C and P, threshold γ and aggregation function f.

```
1: function SP-CoSTS($c, p, \gamma, f$)
2:     $ContextSent \leftarrow ()$
3:     $c^+ \leftarrow$ GeneralCombination($c, p, f$)
4:     for all Term $t$ in $c^+$ do
5:         $ContextSent[1] \leftarrow ContextSent[1] + PositiveSentiment(t)$
6:         $ContextSent[2] \leftarrow ContextSent[2] + NegativeSentiment(t)$
7:     end for
8:     if $ContextSent[1] - ContextSent[2] \geq \gamma$ then
9:         Return agreement
10:    else
11:        Return disagreement
12:    end if
13: end function
```

on properly formed text, they tend to gain on data that is not properly formatted (e. g., social media corpora) due to being more flexible in the way they handle text. Unlike lexicon-based methods such as the ones described in the previous section, shallow linguistic-based methods rely on word order for inference and thus do not use the traditional bag-of-words model to represent data. Instead, they parse text from start to finish and produce a prediction once the entire content has been processed. We explore three methods in this section: the Unsupervised Sentiment Surface (USS) algorithm, the Augmented Unsupervised Sentiment Surface (USS+) algorithm, and the Multi-strategy Unsupervised Stance Classifier (MUSC) method.

5.2.1 Unsupervised sentiment surface (USS)

We explore conversational context-aware methods using the Unsupervised Sentiment Surface algorithm (USS, shown in algorithm 4). USS is extended from Wang and Cardie (2014) and uses a shallow linguistic analysis to compute the average distance between second person pronouns and positive or negative terms (in a sentiment analysis sense). The algorithm parses through the words of a document and keeps track of second person pronouns. For each second person pronoun, it looks for the closest sentiment-bearing expression, and keeps track of the distance (in words) to the nearest positive and the nearest negative terms. It then aggregates those two lists of distances, and return the agreement class if positive expressions are on average closer to second person pronouns, and disagreement otherwise. Its only input is a document C, and its output is a class (agreement or disagreement).

USS works on the intuition that the stance of the comment is contained within explicit references to the parent posts: such references can be analysed by detecting second person pronouns (e.g. “you”, “your”, etc.) and their polarity by searching their grammatical neighbourhood for polarised (positively or negatively) terminology.

Algorithm 4 The Unsupervised Sentiment Surface algorithm.

```
1: function USS($C$)                                              ▷ Stance of C
2:     $Distances \leftarrow ((), ())$
3:     for all Term $t$ in $C$ do
4:         if IsSecondPersonPronoun(t) then
5:             N$_{pos}$ $\leftarrow$ ClosestPositiveTerm(t)
6:             N$_{neg}$ $\leftarrow$ ClosestNegativeTerm(t)
7:             $append$(Distances[1], distance(t, N$_{pos}$)
8:             $append$(Distances[2], distance(t, N$_{neg}$)
9:         end if
10:    end for
11:    $AggregPosDistance \leftarrow aggregate(Distances[1])$
12:    $AggregNegDistance \leftarrow aggregate(Distances[2])$
13:    if $AggregPosDistance < AggregNegDistance$ then
14:        Return agreement
15:    else
16:        Return disagreement
17:    end if
18: end function
```

For example, "*I don’t agree with you and I think your opinion is wrong*" would be interpreted as an attacking statement because of the overwhelming proximity of negative terms ("*don’t*", "*wrong*") near second person pronouns ("*you*", "*your*"). This approach can also give flawed results when there is ambiguity contained in the text in the following: (1) when a positive (respectively negative) term is accidentally closed to a second person pronoun which is semantically linked to a negative (respectively positive) term, (2) when sarcasm is used, (3) when a more complex sentence structure is used where the polarity of a term is implicitly negated, or (4) whenever no second person pronouns or sentiment-bearing terms are used. The following examples illustrate these three cases, where d is defined as a word distance that does not take punctuation into account:

(1) "*I like you, but you are wrong.*" Here we can see that $d(like, you) < d(you, wrong)$ which would classify this instance as a supporting statement.

(2) "*You are a genius, aren't you?*" While the terminology used is positive which would classify this instance as a supporting statement, the high likelihood of it being a sarcastic remark should make it an attacking one.

(3) "*I can barely tolerate that you believe yourself to be right.*" Here the sentence structure puts *yourself* very close to *right*, which will classify the sentence as a supporting statement. However, it is clear under a human eye that the sentence has a disapproving tone.

The USS approach is thus limited to a specific number of cases where the disagreement is expressed as a specific attack on the parent comment. However, in order to take more cases into account we enrich its methodology and build the Augmented unsupervised sentiment surface approach.

5.2.2 Augmented unsupervised sentiment surface (USS+)

In order to overcome this weakness, we designed the augmented unsupervised sentiment surface algorithm (USS+), an extension of the Sentiment Surface method from Wang and Cardie (2014) that uses an additional fall-back heuristic to handle edge-case scenarios where the text being classified does not use explicit attacks on the parent comment using second-person pronouns. The fall-back heuristic uses a general purpose sentiment lexicon to determine the general sentiment expressed within the text and use that as a proxy for stance. The process is described in the flowchart given in Figure 5.1 and in more details in Algorithm 5.

The algorithm takes as inputs a document C, and two threshold numbers α and β, and outputs a class (disagreement or agreement). The first part of the algorithm works exactly as USS, until line 13. A threshold is applied to the result of the distance heuristic taken from USS. If the difference between distances from positive expressions and distances from negative expressions is sufficient (superior or equal to the threshold α), a classification is performed. Otherwise, the algorithm falls back to another heuristic using raw sentiment. If the overall positive sentiment goes over a threshold β, the agreement class is returned. Otherwise, the disagreement class is returned.

We illustrate the process in Algorithm 5: the algorithm firstly attempts to detect explicit attacks using the USS heuristic (lines 2 to 19). It assesses its confidence in the prediction using the

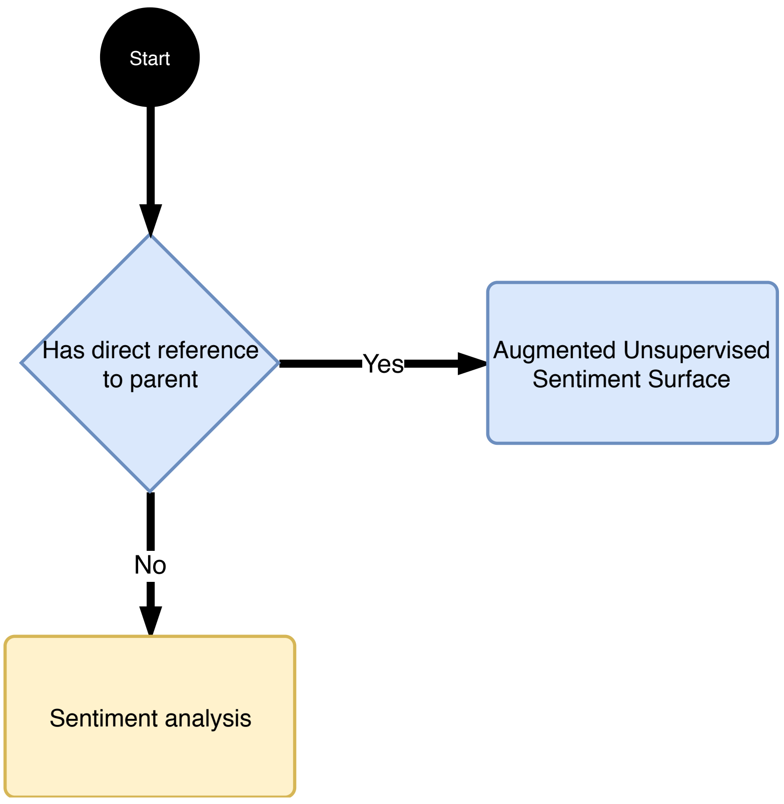

Start
Has direct reference to parent
Yes
Augmented Unsupervised Sentiment Surface
No
Sentiment analysis

FIGURE 5.1: The Augmented Unsupervised Sentiment Surface process.

α threshold. If the prediction has a confidence inferior to α, the USS+ algorithm goes into failure mode (lines 20 to 29) and uses sentiment analysis to determine stance, conditioned on a second confidence threshold β. If once again the prediction confidence falls under that confidence threshold, the algorithm returns a randomly selected class, with the probability being weighted by the frequency of observation of that class in the training set.

Example 9. The following illustrates an example that would be misclassified by USS but correctly classified by USS+: "*I think this is right, and I agree wholeheartedly*". The USS heuristic would miss the stance comment due to the absence of an explicit call out to the parent comment (e. g., "*I think $\underline{\text{you}}$ are right*"), while the failure mode of USS+ would allow it to catch the general sentiment of the comment, and use it as a correct approximation of the stance of the comment. Of course, there can be counter-examples where the sentiment would run counter to the actual stance of the comment, but since those would generally tend to be misclassified by the USS heuristic, there is nothing lost by applying a more general heuristic to it.

Algorithm 5 The Augmented Unsupervised Sentiment Surface algorithm.

```
1:  function AUSS($C, \alpha, \beta$)        ▷ Stance of C with respect to P with thresholds $\alpha, \beta$
2:      $Distances \leftarrow ((), ())$
3:      for all Term $t$ in $C$ do
4:          if IsSecondPersonPronoun(t) then
5:              $N_{pos} \leftarrow$ ClosestPositiveTerm(t)
6:              $N_{neg} \leftarrow$ ClosestNegativeTerm(t)
7:              $append$(Distances[1], distance(t, $N_{pos}$))
8:              $append$(Distances[2], distance(t, $N_{neg}$))
9:          end if
10:     end for
11:     $AggregPosDistance \leftarrow aggregate(Distances[1])$
12:     $AggregNegDistance \leftarrow aggregate(Distances[2])$
13:     if $abs(AggregPosDistance - AggregNegDistance) \geq \alpha$ then
14:         if $AggregPosDistance < AggregNegDistance$ then
15:             Return agreement
16:         else
17:             Return disagreement
18:         end if
19:     end if
20:     $Sentiment \leftarrow ()$
21:     for all Term $t$ in $C$ do
22:         $Sentiment[1] \leftarrow Sentiment[1] + PositiveSentiment(t)$
23:         $Sentiment[2] \leftarrow Sentiment[2] + NegativeSentiment(t)$
24:     end for
25:     if $Sentiment[1] - Sentiment[2] \geq \beta$ then
26:         Return agreement
27:     else
28:         Return disagreement
29:     end if
30: end function
```

5.2.3 Multi-strategy Unsupervised Stance Classifier (MUSC)

The MUSC method is an extension of USS+ that uses a complex, hierarchical decision scheme, illustrated in Figure 5.2 and in more details in Algorithm 6. It combines three strategies, going from the most specific (high precision, low recall) to the least specific (high recall, low precision) in order to capture the maximum number of cases. It takes as inputs two documents (parent and child comments) and four threshold parameters, which define the level of confidence needed to provide a classification and not fall back to a more general heuristic. The threshold α controls whether the algorithm falls back from the high precision expressions to the USS distance heuristic. The threshold β controls whether the algorithm falls back from the USS distance heuristic to the contextual sentiment analysis heuristic. The threshold γ controls whether the algorithm falls back from the contextual sentiment analysis heuristic to the raw sentiment analysis heuristic. Finally, the η heuristic controls the difference between positive and negative sentiment needed to push the classification toward the agreement class. The output of the algorithm is the predicted class of the document C.

Algorithm 6 The Multi-strategy Unsupervised Stance Classification algorithm - Part 1

```
function MUSC($C, P, \alpha, \beta, \gamma, \eta$)          ▷ Stance of C w.r.t. P ; thresholds $\alpha, \beta, \gamma, \eta$
    $HPE \leftarrow (0, 0)$                                   ▷ High Precision Evidence
    for all Agreement sentence AS do
        $HPE[1] \leftarrow HPE[1] + 1$
    end for
    for all Disagreement sentence DS do
        $HPE[2] \leftarrow HPE[2] + 1$
    end for
    if $abs(HPE[1] - HPE[2]) \geq \alpha$ and $HPE[1] + HPE[2] \geq 0$ then
        if $HPE[1] > HPE[2]$ then
            Return agreement
        else
            Return disagreement
        end if
    end if
    $Distances \leftarrow ((), ())$
    for all Term $t$ in $C$ do
        if IsSecondPersonPronoun(t) then
            $N_{pos}$ ← ClosestPositiveTerm(t)
            $N_{neg}$ ← ClosestNegativeTerm(t)
            $append$(Distances[1], distance(t, $N_{pos}$)
            $append$(Distances[2], distance(t, $N_{neg}$)
        end if
    end for
    $AggregPosDistance \leftarrow aggregate(Distances[1])$
    $AggregNegDistance \leftarrow aggregate(Distances[2])$
    if $abs(AggregPosDistance - AggregNegDistance) \geq \beta$ then
        if $AggregPosDistance < AggregNegDistance$ then
            Return agreement
        else
            Return disagreement
        end if
    end if
```

Each strategy is defined in order to favour less precision and more recall than its predecessor.

[A] **High-precision expression analysis** is the detection of specific expressions which are highly indicative of a class. For instance, “I disagree”, “you are wrong”, “I share your

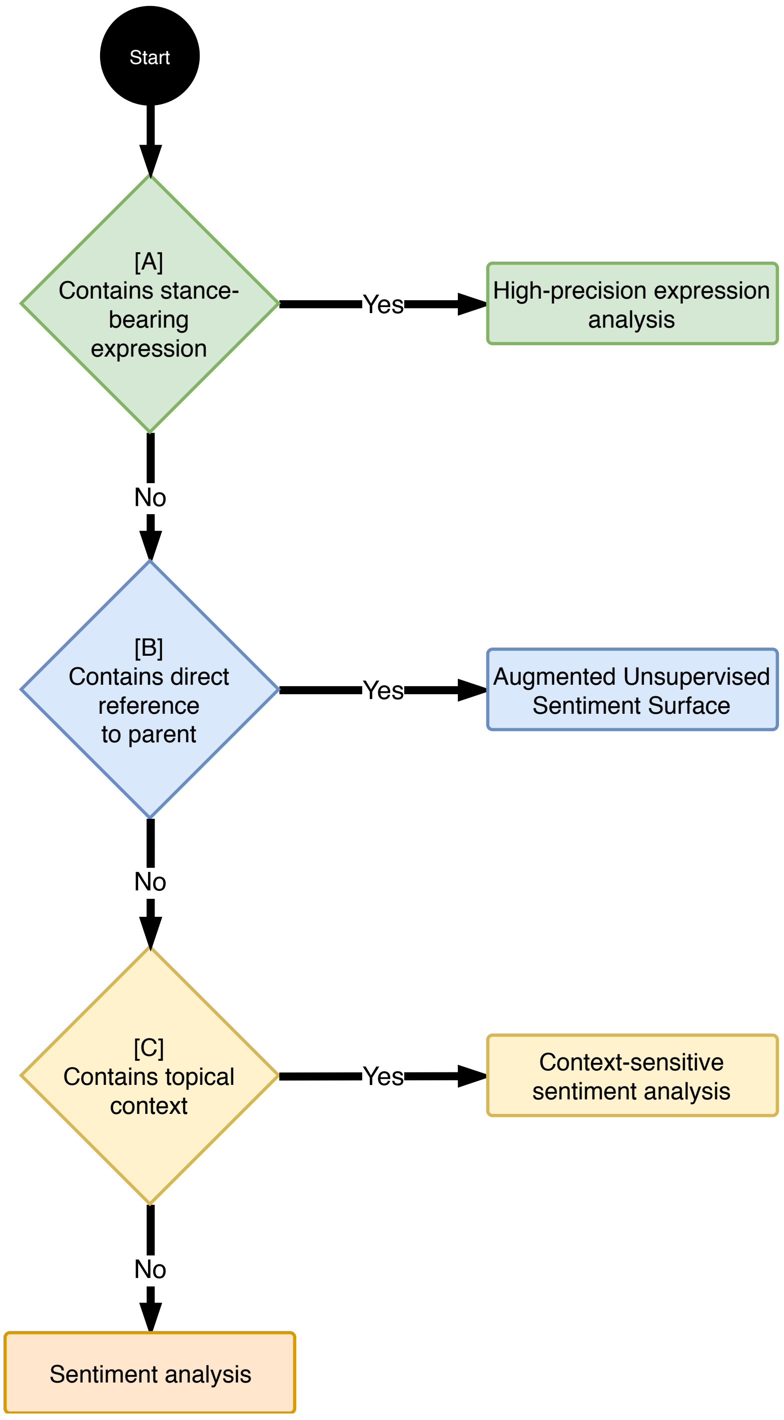

Start
[A] Contains stance-bearing expression
Yes
High-precision expression analysis
No
[B] Contains direct reference to parent
Yes
Augmented Unsupervised Sentiment Surface
No
[C] Contains topical context
Yes
Context-sensitive sentiment analysis
No
Sentiment analysis

FIGURE 5.2: The Multi-strategy Unsupervised Stance Classifier process uses 4 different strategies by decreasing order of specificity in order to maximise prediction accuracy. The first two strategies (A and B) focus on shallow linguistic rules while strategies C and D respectively use sentiment analysis on contextual sentences (sentences with a high number of terms appearing in the parent comment) and the entire content.

Algorithm 7 The Multi-strategy Unsupervised Stance Classification algorithm - Part 2

```
    $ContextSent \leftarrow ()$
    $C^+ \leftarrow \text{combine}(C, P)$
    for all Term $t$ in $C^+$ do
        $ContextSent[1] \leftarrow ContextSent[1] + PositiveSentiment(t)$
        $ContextSent[2] \leftarrow ContextSent[2] + NegativeSentiment(t)$
    end for
    if $abs(ContextSent[1] - ContextSent[2]) \geq \gamma$ then
        if $ContextSent[1] \geq ContextSent[2]$ then
            Return agreement
        else
            Return disagreement
        end if
    end if
    $Sentiment \leftarrow ()$
    for all Term $t$ in $C$ do
        $Sentiment[1] \leftarrow Sentiment[1] + PositiveSentiment(t)$
        $Sentiment[2] \leftarrow Sentiment[2] + NegativeSentiment(t)$
    end for
    if $Sentiment[1] - Sentiment[2] \geq \eta$ then
        Return agreement
    else
        Return disagreement
    end if
end function
```

opinion". The dictionary of expressions used is available in Tables 5.2 and 5.3, reproduced from Chapter 4 where it was used to generate a noisy-labelled dataset ;

[B] **Unsupervised Sentiment Surface**, as described in Section 5.2.1, uses second-person pronouns and negative sentiment-bearing words to identify disagreements ;

[C] **Context-sensitive sentiment analysis** uses sentiment analysis on sentences from the child post which have a high degree of similarity to the sentences of the parent post, and is controlled with a predefined threshold parameter.

[D] **Sentiment analysis** uses standard sentiment analysis techniques on all the sentences of the child post.

The high precision expressions are mined using regular expressions, where [EOS] indicates an end-of-sentence marker such as an exclamation mark, a full stop or a comma, [] marks optional expressions or positive/intensifying adverbs such as "extremely", "totally", etc, NAO marks a Noun/Adjective referring to the Object of the discussion and NAA refers to a Noun/Adjective

Id	Agreement expression
a_1	i [adverb] [agreement verb]
a_2	i (don't|do not) [adverb] [disagreement verb]
a_3	i (am not|ain't) [disagreement NAA]
a_4	(you are|you're) [adverb] [agreement NAA]
a_5	you (are not|aren't|ain't) [adverb] [disagreement NAA]
a_6	[this is] [adverb] [agreement NAO] [EOS]
a_7	[this is] not [adverb] [disagreement NAO] [EOS]

TABLE 5.2: High-precision regular expressions indicative of agreement class

Id	Disagreement expression
d_1	i [adverb] [disagreement verb]
d_2	i (don't|do not) [adverb] [agreement verb]
d_3	i (am not|ain't) of the same (mind|opinion)
d_4	(you are|you're) [adverb] [disagreement NAA]
d_5	you (are not|aren't|ain't) [adverb] [agreement NAA]
d_6	[this is] [adverb] [disagreement NAO] [EOS]
d_7	[this is] not [adverb] [agreement NAO] [EOS]

TABLE 5.3: High-precision regular expressions indicative of disagreement class

referring to the Author that is being replied to. For instance, both "I completely disagree", "I totally disagree" and "I disagree" would trigger expression d_1 and bring evidence of the disagreement class. Similarly, "This is not totally wrong" would trigger a_7 and bring evidence of the agreement class. The goal of this approach is to be able to leverage high precision techniques when they are applicable, and back-off to a slightly more general one when not. The list of agreement and disagreement verbs, nouns and adverbs was collected using multiple thesauri and recouping terms with a higher frequency of apparition in social media text. They are listed in Tables 4.5, 4.6, 4.7, 4.8, 4.9 and 4.10 and are given in Chapter 4 where they were used to generate the RNLC dataset.

Example 10. The following illustrates an example that would be misclassified by USS+ and USS but correctly classified by MUSC: "*I think this is evil and pigheaded, but I totally agree.*". While USS would fail fast by not being able to find any explicit reference to the parent comment, USS+ would also fail in its failure mode since the overall sentiment of that comment is negative (presence of several negative words). However, the fact that the sentence contains a linguistic pattern that is explicitly indicative of a specific class ("*I totally agree*") means that MUSC can correctly identify it.

5.3 Evaluation

In this section we focus on evaluating the approaches developed along the course of this chapter, with two tasks (2 and 3 class classification) based on classification without any form of conversational context and two tasks (2 and 3 class classification) based on classification with a form of conversational context.

5.3.1 Evaluation tasks

We evaluate our approaches using two different types of tasks, context-sensitive and context-less classification, with either 2 or 3 classes, giving us the following tasks:

CL-2. Context-less classification with 2 classes (agreement/disagreement) ;

CL-3. Context-less classification with 3 classes (agreement/disagreement/neutral) ;

CS-2 Context-sensitive classification with 2 classes (agreement/disagreement) ;

CS-3. Context-sensitive classification with 3 classes (agreement/disagreement/neutral).

Unsupervised context-less stance classification refers to the classification in 2 or 3 classes of the stance of a user comment/post without observing the content of the parent, while context-sensitive classification refers to the same but with observing the content of the parent comment/post. This affects the algorithms and the datasets that will be used. Since not all datasets contain direct reference to the context of the conversation, and not all datasets contain a neutral class, we

	CL-2	CL-3	CS-2	CS-3
IAC	✓	✓	✓	✓
ABCD	✓	✓	x	x
SETC	✓	✓	x	x

TABLE 5.4: Task vs dataset. ✓indicates compatibility.

	CL-2	CL-3	CS-2	CS-3
SP-ADAPT	✓	✓	✓	✓
SP-COSTS	x	x	✓	✓
USS	✓	✓	✓	✓
USS+	✓	✓	✓	✓
MUSC	x	x	✓	✓

TABLE 5.5: Task vs algorithm. ✓indicates compatibility.

summarise the datasets for each task in Table 5.4. Similarly, since some algorithms require context, we summarise the algorithms for each task in Table 5.5.

5.3.2 Baselines

Because of the lack of existing work in the domain of corpus-based unsupervised stance classification, we establish two baselines inspired from the field of sentiment analysis.

Sentiment baseline: the SMARTSA algorithm The sentiment baseline is computed by counting the sum of positive and negative sentiments. We bound that sum between -1 and 1 using a Softmax normalisation, and apply a predefined classification rule.

For the binary stance classification tasks, the rule is:

$$\text{Class(X)} = \begin{cases} \text{agreement}, & \text{if sum} > 0.0 \\ \\ \text{disagreement}, & \text{otherwise} \end{cases}$$

For the 3 class stance classification task, the rule becomes:

$$\text{Class(X)} = \begin{cases} \text{agreement,} & \text{if sum} > 0.33 \\ \text{disagreement,} & \text{if sum} < -0.33 \\ \text{neutral,} & \text{otherwise} \end{cases}$$

The thresholds were selected by dividing the range equally.

Lexicon baseline: Stance PMI lexicon Our second baseline is a lazy stance lexicon built from the PMI score of any given term and a background dataset. The background dataset used for each experiment was a fusion of all datasets not directly involved with the same classes represented (e. g., a fusion of IAC, ABCD and RNLC for the SETC dataset on a binary task, etc.). The scores for each term-class association were a weighted average of the PMI scores for each dataset, weighted by the number of instances belonging to that dataset. Lexicon-based classification was elaborated upon in Section 3.4.2.

5.3.3 Results

In this section we present the results of our evaluation with unsupervised stance classification algorithms.

5.3.3.1 Results of task CL-2

The CL-2 task corresponds to a binary stance classification without using context.

The results are presented in Table 5.6 and indicate that USS+ consistently outperforms the rest of the approaches on the three datasets that we are using. We illustrate the results in Figure 5.3. We note that USS+ and PMI-LEX perform very similarly, and that while PMI-LEX displays a higher variance in its results it is very far behind USS+.

	IAC	ABCD	SETC
SP-ADAPT	0.566	0.559	0.558
USS	0.610	0.612	0.554
USS+	**0.634**	**0.637**	**0.590**
SMARTSA	0.498	0.491	0.513
PMI-LEX	0.627	0.623	0.577

TABLE 5.6: Accuracy results of Task CL-2

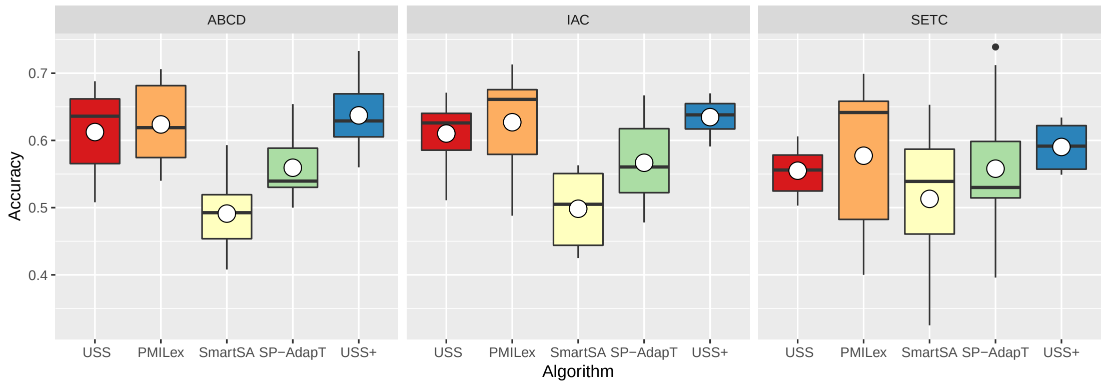


FIGURE 5.3: Box plot of accuracy results on Task CL-2. The white circles indicate the arithmetic means.

Unsurprisingly, the sentiment baseline underperforms compared to the other approaches and performs even worse than random (0.5). However, SP-ADAPT, which is built on a sentiment baseline with a tuned threshold parameter, is relatively competitive with smarter approaches such as PMI-LEX and USS+. We also note that the performances seem to be extremely similar on the SETC dataset. A one-way ANOVA test shown in Table 5.7 confirms that while some algorithms are performing differently on the IAC and ABCD datasets ($p < 0.01$), no such thing happens for the SETC dataset, where the differences in our evaluation have a high chance of being due to chance ($p > 0.05$).

We perform a Tukey test on the results of IAC and ABCD in order to identify the algorithms with a significant difference in their accuracy. There is no need to pursue the analysis of the SETC dataset due to the lack of any significant difference between algorithms, as evidenced by the p-value of 0.3676 shown in Table 5.7.

Dataset	F-Value	P-Value
IAC	9.662	**9.971e-06**†
ABCD	11.01	**2.603e-06**†
SETC	1.1015	0.3676

TABLE 5.7: Results of ANOVA on task CL-2

	IAC		ABCD	
	Difference	Adjusted P-Value	Difference	Adjusted P-Value
USS+ / USS	0.02	0.87	0.02	0.87
PMI-LEX / USS	0.02	0.96	0.01	0.99
SMARTSA / USS	-0.11	**0.00**†	-0.12	**0.00**†
SP-ADAPT / USS	-0.04	0.45	-0.05	0.25
PMI-LEX / USS+	-0.01	1.00	-0.01	0.98
SMARTSA / USS+	-0.14	**0.00**†	-0.15	**0.00**†
SP-ADAPT / USS+	-0.07	0.08	-0.08	**0.03**†
SMARTSA / PMI-LEX	-0.13	**0.00**†	-0.13	**0.00**†
SP-ADAPT / PMI-LEX	-0.06	0.15	-0.06	0.11
SP-ADAPT / SMARTSA	0.07	0.07	0.07	0.08

TABLE 5.8: Tukey test for IAC and ABCD on CL-2. † indicates statistical significance at 0.95 confidence.

The results of Tukey's Honest Significant Difference (HSD) test, shown in table 5.8, reveal that after adjusting the p-values for multiple hypothesis testing in order to avoid increasing the risk of false discovery, very few of the compared pairs of algorithms actually display a significantly different performance from each other. In the IAC dataset, only the SMARTSA baseline underperforms significantly compared to USS+ and PMI-LEX, which are themselves far from significantly different ($p = 1.00$). In the ABCD dataset, USS+ also significantly outperforms SP-ADAPT and once again does not come close to statistical significance when compared to PMI-LEX.

	IAC	ABCD	SETC
SP-ADAPT	0.502	0.466	0.480
USS	0.479	0.510	0.467
USS+	**0.549**	**0.552**	**0.514**
SMARTSA	0.444	0.413	0.390
PMI-LEX	0.522	0.542	0.464

TABLE 5.9: Accuracy results of Task CL-3

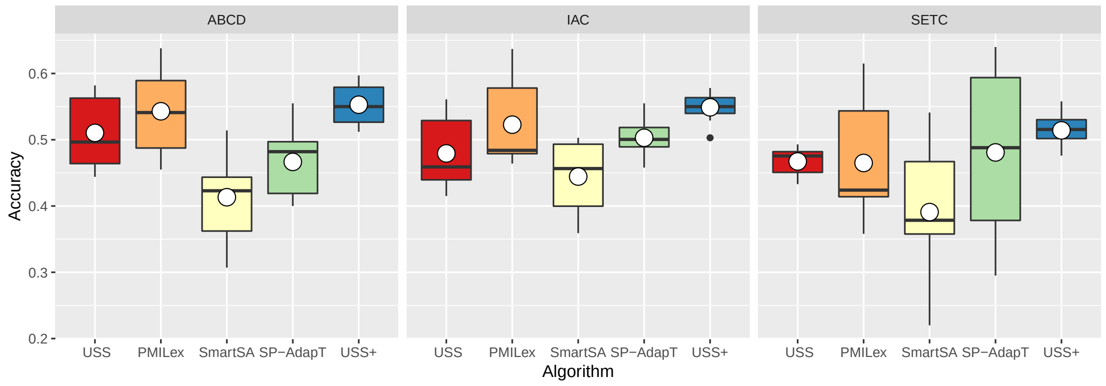


FIGURE 5.4: Box plot of accuracy results on Task CL-3. The white circles indicate the arithmetic means.

5.3.3.2 Results of task CL-3

The CL-3 task corresponds to a three class stance classification without using context. The classes were {Disagreement, Neutral, Agreement} and determined in the case of datasets with numerical labels by grouping those labels uniformly. For example, on IAC which has labels from -5 to 5, the data was separated as Disagreement with -5 to -1.66, Neutral with -1.66 to 1.66, and Agreement with 1.66 to 5.

The results from Table 5.9 show that USS+ outperforms, once again, all other methods; but without attaining statistical significance. We can observe that although USS+ performs better than alternatives, the gap is much closer and baseline techniques such as PMI-LEX come within less than 1% of USS+. We illustrate those scores in Figure 5.4.

A one-way ANOVA test shown in Table 5.10 shows that there is a need for a post hoc test in all datasets, with SETC showing some behaviour slightly above statistical significance ($p < 0.05$).

Dataset	F-Value	P-Value
IAC	6.988	**0.0001829**†
ABCD	11.262	**2.041e-06**†
SETC	2.6805	**0.04348**†

TABLE 5.10: Results of ANOVA on task CL-3. † indicates statistical significance at 0.95 confidence.

	IAC		ABCD		SETC	
	Diff.	Adj. P-Value	Diff.	Adj. P-Value	Diff.	Adj. P-Value
USS+ / USS	0.16	**0.01**†	0.13	0.13	0.14	0.88
PMI-LEX / USS	0.11	0.44	0.08	0.99	0.04	0.38
SMARTSA / USS	0.06	1.00	-0.03	**0.00**†	0.02	0.19
SP-ADAPT / USS	0.10	0.76	0.05	0.93	0.11	1.00
PMI-LEX / USS+	0.02	0.30	0.02	0.34	0.00	0.06
SMARTSA / USS+	-0.03	**0.00**†	-0.09	**0.00**†	-0.01	**0.02**†
SP-ADAPT / USS+	0.01	0.11	-0.01	**0.02**†	0.08	0.91
SMARTSA / PMI-LEX	0.02	0.27	-0.04	**0.00**†	0.09	0.99
SP-ADAPT / PMI-LEX	0.06	0.99	0.04	0.69	0.18	0.33
SP-ADAPT / SMARTSA	0.10	0.56	0.15	**0.02**†	0.20	0.16

TABLE 5.11: Tukey test for IAC, ABCD and SETC on CL-3. † indicates statistical significance at 0.95 confidence.

We therefore perform a Tukey test, as shown in Table 5.11 for all datasets, and see that while a graphical examination of the box plots tends to be in favour of USS+, formal statistical tests show that those differences are not significant and could be due to random chance.

5.3.3.3 Results of task CS-2 and CS-3

The CS-2 and CS-3 tasks refer to context sensitive binary and 3 class stance classification. It involves two novel algorithms that take into account the content of the parent comment as well as the content of the comment being classified, and therefore only the IAC dataset allowed for

	CS-2	CS-3
SP-ADAPT	0.566	0.502
SP-COSTS	0.611	0.564
USS	0.610	0.479
USS+	0.634	0.549
MUSC	**0.646**	**0.604**
SMARTSA	0.498	0.444
PMI-LEX	0.627	0.522

TABLE 5.12: Accuracy results of task CS-2 and CS-3 on the IAC dataset

enough instances to perform a proper comparison of algorithms. This lack of data limits the generalisability of the results of this experiment.

Preliminary experiment: choosing the context-building function Preliminary experiments on a hold out dataset were run in order to determine the context function of the SP-COSTS algorithm by which it builds a new instance from a mixture of parent and child comment. The optimal context building function was determined to be the additive mixture described as SP-COSTS$_3$. SP-COSTS$_3$ will thus be referred as SP-COSTS for the remainder of the thesis.

In Table 5.12 we can see that MUSC outperforms the rest of the algorithms. We illustrate those results in Figure 5.5 and Figure 5.6. SP-COSTS being no more than an extension of SP-ADAPT using contextual information from the parent post, those results present some evidence that knowledge of the context of the discussion, regardless of how naive it is, can improve classification accuracy. Examination of the results of the Tukey test in Table 5.14 however shows that this improvement does not reach statistical significance, due to the high variance of the results corresponding to those techniques.

We show in Table 5.13 the results of an ANOVA test on the results of tasks CS-2 and CS-3, which leads us to investigate the algorithms that display a significantly different behaviour. The results of the Tukey test in Table 5.14 however do not show any algorithm being statistically better than

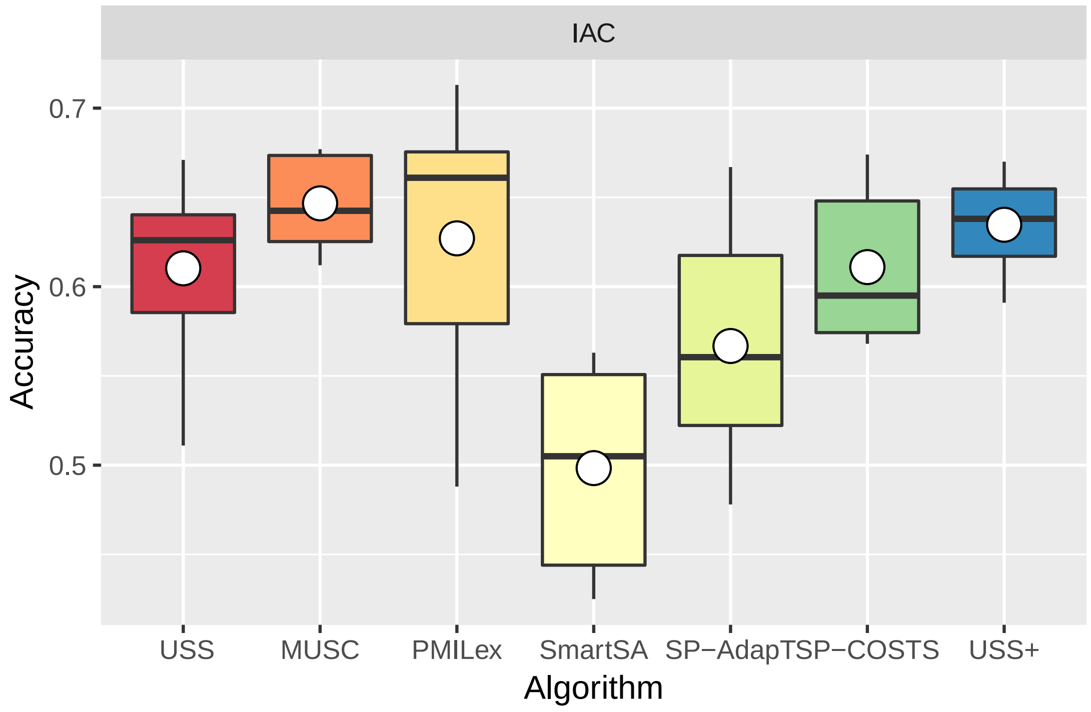


FIGURE 5.5: Box plot of accuracy results on Task CS-2. The white circles indicate the arithmetic means.

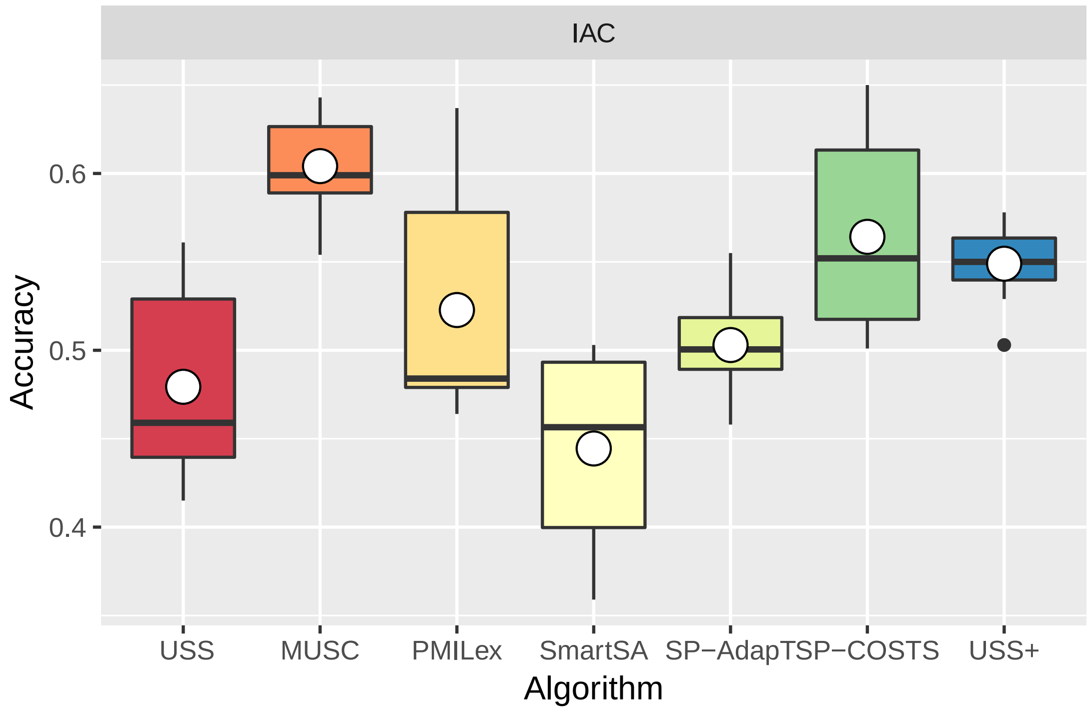


FIGURE 5.6: Box plot of accuracy results on Task CS-3. The white circles indicate the arithmetic means.

	CS-2		CS-3	
Dataset	F-Value	P-Value	F-Value	P-Value
IAC	9.792	**1.313e-07**†	5.3746	**0.0001558**†

TABLE 5.13: Results of ANOVA on tasks CS-2 and CS-3. † indicates statistical significance at 0.95 confidence.

the others, and only that one of the baselines, SMARTSA, performs consistently and significantly worse than any other approach.

5.3.4 Discussion

We can see from the results presented above that more complex heuristics tend to do better than simpler ones, provided they contain integrated back-up heuristics that can perform classification in cases where the complex heuristic does not cover the content of the instances.

We can also see that context can be used to aid in classification, as evidenced by a positive difference between context-sensitive methods such as MUSC and context-insensitive methods such as USS+. In a binary classification setting that difference is more important, but it is not the case when taking into account the neutral stance class. That is possibly due to the separation of the neutral class in the IAC dataset being done by using strict boundaries, which implies that there could be some noise in the form of edge cases (e. g., an instance with the label 1.5 being considered as neutral because the threshold is 1.66).

Finally, a big limitation of our comparison is the poor size of the available datasets. Because statistical tests are often bounded by the amount of data available, the effect size that would have been necessary to achieve statistical significance would have been extremely important and unrealistic in a text classification setting, let alone when working with manually defined heuristics such as the ones presented in this section.

	CS-2		CS-3	
	Difference	Adjusted P-Value	Difference	Adjusted P-Value
USS+/USS	0.09	0.94	0.16	**0.00**†
MUSC/USS	0.11	0.70	0.15	**0.01**†
PMI-LEX/ USS	0.09	0.99	0.11	0.55
SMARTSA/USS	-0.04	**0.00**†	0.06	1.00
SP-ADAPT/USS	0.03	0.50	0.10	0.87
SP-COSTS/USS	0.07	1.00	0.10	0.84
MUSC/USS+	0.08	1.00	0.06	1.00
PMI-LEX/USS+	0.06	1.00	0.02	0.37
SMARTSA/USS+	-0.07	**0.00**†	-0.03	**0.00**†
SP-ADAPT/USS+	0.00	0.07	0.01	0.13
SP-COSTS/USS+	0.05	0.95	0.01	0.15
PMI-LEX/MUSC	0.05	0.98	0.03	0.52
SMARTSA/MUSC	-0.08	**0.00**†	-0.02	**0.00**†
SP-ADAPT/MUSC	-0.01	**0.02**†	0.01	0.21
SP-COSTS/MUSC	0.03	0.72	0.02	0.24
SMARTSA/PMI-LEX	-0.06	**0.00**†	0.02	0.34
SP-ADAPT/PMI-LEX	0.01	0.14	0.06	1.00
SP-COSTS/PMI-LEX	0.05	0.99	0.06	1.00
SP-ADAPT/SMARTSA	0.14	0.06	0.11	0.68
SP-COSTS/SMARTSA	0.18	**0.00**†	0.11	0.64
SP-COSTS/SP-ADAPT	0.11	0.48	0.07	1.00

TABLE 5.14: Tukey test for IAC on CS-2 and CS-3. † indicates statistical significance at 0.95 confidence.

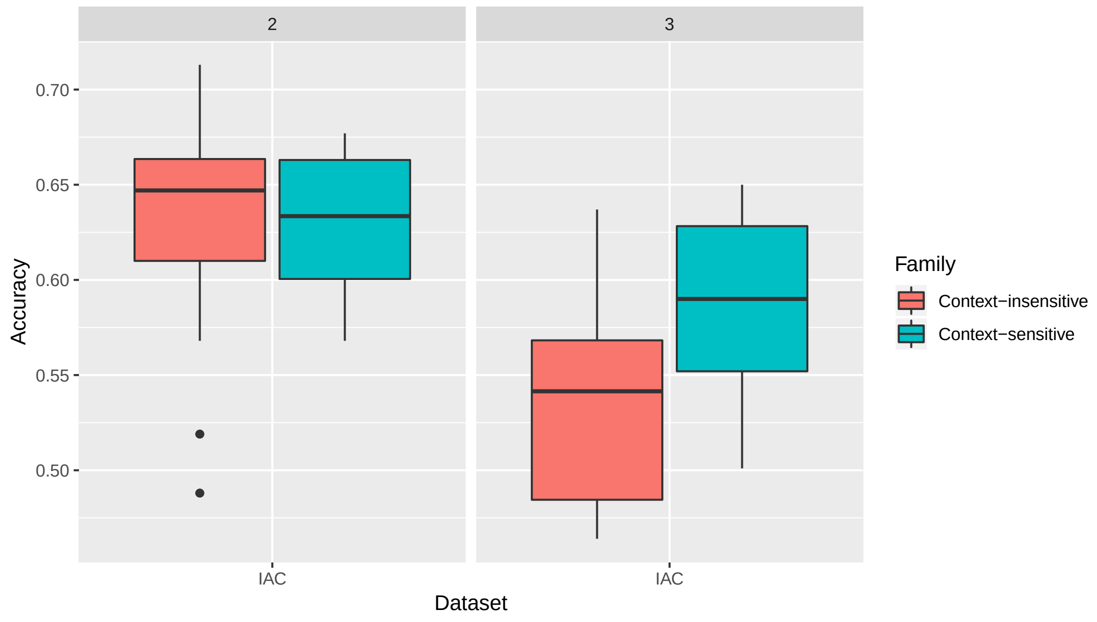


FIGURE 5.7: A comparison of context-sensitive vs context-insensitive methods on the IAC dataset.

Task	2 classes	3 classes
Context-insensitive methods	**0.630**	0.535
Context-sensitive methods	0.628	**0.584**†

TABLE 5.15: Mean accuracies when averaging over types of approaches

5.3.4.1 Using context to improve heuristics

Our initial research question was to investigate whether the use of context could potentially help improve classification accuracy. We do so by creating two groups of algorithms: context-insensitive (USS+, PMI-LEX) and context-sensitive (MUSC, SP-COSTS). We do not use the naive baselines since they would necessarily lower our context-insensitive methods to worse results by their status as naive approaches while not answering the research question.

We average out our result per family and compare them in Figure 5.7 and Table 5.15, showing that the effect seems to differ according to the number of classes. Further testing shown in Table 5.16 confirms this statement with statistical evidence, showing that there is virtually no difference between the overall performance of the family of class-sensitive and class-insensitive algorithms in the binary classification settings which we used for our evaluation, but there is a statistically significant difference at the $p < 0.05$ and $p < 0.01$ thresholds on the 3 class classification

	2 classes		3 classes	
	Diff.	P-Value	Diff.	P-Value
Context-sensitive/Context-insensitive	-0.00205	0.8833585	0.0483	**0.0004789981**†

TABLE 5.16: Differences and p-values of a Tukey HSD test comparing class-sensitive vs class-insensitive methods

problem. This would tend to show that context can be a differentiator on harder classification tasks such as differentiating between 3 classes, but does not necessarily bring much improvement on easier ones.

5.4 Chapter Summary

In this chapter we explored two types of unsupervised techniques to perform stance classification with minimal resources and no training. The first type of technique uses sentiment as a proxy for stance. The second type uses a shallow linguistic analysis using heuristic rules in order to favour precision over recall. Finally, we introduced MUSC, a multi-strategy approach that combines both types of techniques in order to maximise precision if possible and fall back on high-recall methods when necessary. We evaluated our approaches against existing heuristics inspired from the sentiment analysis community and found that while the proposed context-insensitive and context-based approaches USS+ and MUSC tend to perform better, that difference was not steep enough to draw consistent statistical significance in our multiple experiments.

Chapter 6

Supervised Lexicon-Based Methods for Argument Stance Mining

This chapter is focused on lexicon-based argument stance mining and a method to efficiently compute optimal stance lexicons. As a result, we produce two lexicon models, LEXICNET and RELEXNET, as contributions to the state of the art in lexicon-based text classification. We introduce a lexicon learning method based on the backpropagation algorithm, which allows us to use regularisation techniques to improve the generalisation power of the lexicons and outperform existing approaches for corpus-based lexicon learning, such as lexicons based on expectation-maximisation (Bandhakavi et al., 2014). In addition to being competitive to standard machine learning models, lexicon-based methods are highly interpretable on multiple levels.

Interpretability has been defined in machine learning and statistics at different scales and with different goals in mind, but a common thread is that it relates to the degree to which a human can understand the cause behind an algorithmic decision (Miller, 2017), and the model that generated that decision can be understood either at a global or at a local scale (Lipton, 2018). A lexicon model succeeds at both those interpretations, because the cause of a decision can be easily understood by a human participant by looking at the lexicon weights of each term present in an instance, and it is trivial to understand any set of weights in a lexicon as a measure of association between that term and each class. Unlike models such as multilayer neural networks, this association is direct, and the learning process consists only in ensuring that it is not spurious and the model can generalise well.

Our aim is to use a corpus of labelled documents in order to learn optimal weights in a stance lexicon. To do so we model a lexicon as a computational graph and use the backpropagation algorithm to perform probability density estimation using a shallow model, which is then extracted as a lexicon.

Computational graphs A computational graph is a graph structure where nodes are tensor, matrix, vector, or scalar values and edges are function arguments. Computational graphs are directed and typically acyclic. Figure 6.1 shows an example of a computational graph that performs simple operations of addition (σ) on x_1 and x_2 and multiplication (μ) on x_3 and the output of the addition. The two terms x_1 and x_2 of the addition are weighted by w_1 and w_2, while x_3 and the result of the multiplication are weighted by w_3 and w_4.

6.1 Lexicon generation by backpropagation

Backpropagation was introduced as an algorithm based on reverse-mode differentiation to train neural networks by Rumelhart et al. (1988). It works by considering the neural network as a computational graph and iteratively refining the weights of its edges using partial derivatives of the error function with respect to each weight. We detail the mechanics of backpropagation in the next section, before explaining how to apply it to learning a lexicon.

6.1.1 Differentiable architectures and backpropagation

Consider as an example the function $y = (w_1 \cdot x_1 + w_2 \cdot x_2) \cdot (w_3 \cdot x_3)$, represented by the computational graph shown in figure 6.1, and the task of learning the optimal weights (w_1, w_2, w_3) so that, for a given dataset (x_1, x_2, x_3, y) the result $\hat{y}$ is as close as possible from the ground truth result y.

The backpropagation algorithm works by iterating two steps over groups of instances during the learning phase:

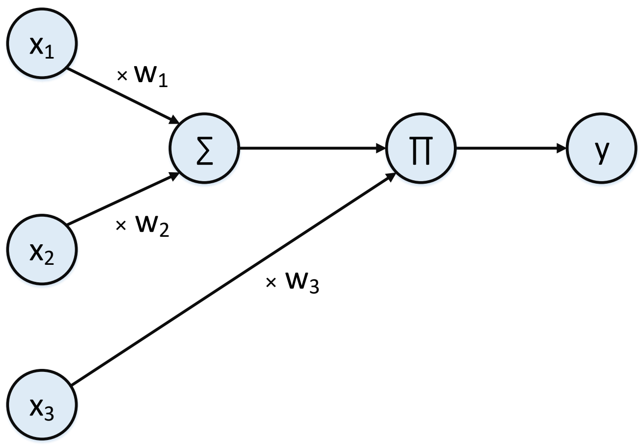


FIGURE 6.1: A simple computational graph

1. **Forward propagation.** For a given set of weights w_1, w_2, w_3 the output is computed as $y = (w_1 \cdot x_1 + w_2 \cdot x_2) \cdot (w_3 \cdot x_3)$ and an error function $J(y, \hat{y})$ is computed. For example, a data point $x = (1, 2, 0.5)$ with $y = 3$ and a initial set of weights $W = (1, 1, 1)$ would yield a prediction $\hat{y} = 1.5$ and assuming an error function $error(y, \hat{y}) = (y - \hat{y})^2$ and a cost function $J(Y, \hat{Y}) = \sum_{i=1}^{n} cost(Y_i, \hat{Y_i})$ we would obtain the error $J(Y, \hat{Y}) = 1.5$

2. **Update.** The partial derivatives $\frac{\partial J}{\partial w_1}$, $\frac{\partial J}{\partial w_2}$ and $\frac{\partial J}{\partial w_3}$ are computed. The weights w_1, w_2 and w_3 are updated according to the update rule outlined in equation 6.1. For example:

   - $w_1 \leftarrow w_1 - \eta \cdot \frac{\partial J}{\partial w_1}$
   - $w_2 \leftarrow w_2 - \eta \cdot \frac{\partial J}{\partial w_2}$
   - $w_3 \leftarrow w_3 - \eta \cdot \frac{\partial J}{\partial w_3}$

The backpropagation weight update uses a learning rate parameter η in order to make several small steps and avoid overstepping the optimal weights while moving in parameter space. The optimisation algorithm used in backpropagation is one of the many variations of gradient descent, where we vary the sampling method (stochastic gradient descent, mini-batch gradient descent or full-batch gradient descent), the update inertia (standard momentum, Nesterov acceleration, historical mean-based learning rate) or the data ordering method (random sampling, scheduled sampling).

$$w_i \leftarrow w_i - \eta \cdot \frac{\partial J}{\partial w_i} \tag{6.1}$$

In the following section we will detail how to model a lexicon as a computational graph and apply backpropagation to induce optimal weights.

6.1.2 Application of backpropagation to lexicons

Assuming $F^{n \times 1}$ is a $n \times 1$ matrix containing the frequencies of each of the n terms in instance x and $W^{c \times n}$ is a $c \times n$ matrix indexed by a class c and containing an association score for each term-class pair, we define the prediction step f of a classification lexicon in the following way:

$$f(x) = ArgMax_c\left(W_c \cdot F\right) \tag{6.2}$$

We can observe that a standard lexicon is a computational graph, i. e., a composition of functions, as shown in Figure 6.2 illustrating the network topology of a binary classification lexicon. However, because $ArgMax$ is not a differentiable function, we need to separate the lexicon architecture in two steps: prediction phase (illustrated in Equation 6.2) and induction phase, using a proxy output that is differentiable. We define the output of the lexicon in the induction phase as a SoftMax-transformed vector of probabilities over the different classes, as we can observe in Equation 6.3) where $\bigoplus$ represents a vectorisation operator such as for instance $\bigoplus_{i=1}^{3}\left(i^2\right) = \langle 1, 4, 9 \rangle$.

$$f(x) = \bigoplus_{j=1}^{|c|} \frac{e^{-(W_j \cdot F)}}{\sum_{m=1}^{|c|} e^{-(W_m \cdot F)}} \tag{6.3}$$

This substitution allows us to use gradient-based learning techniques, such as backpropagation, in order to solve the lexicon induction problem. In the following sections, we explain the details of the network topology and the training protocol.

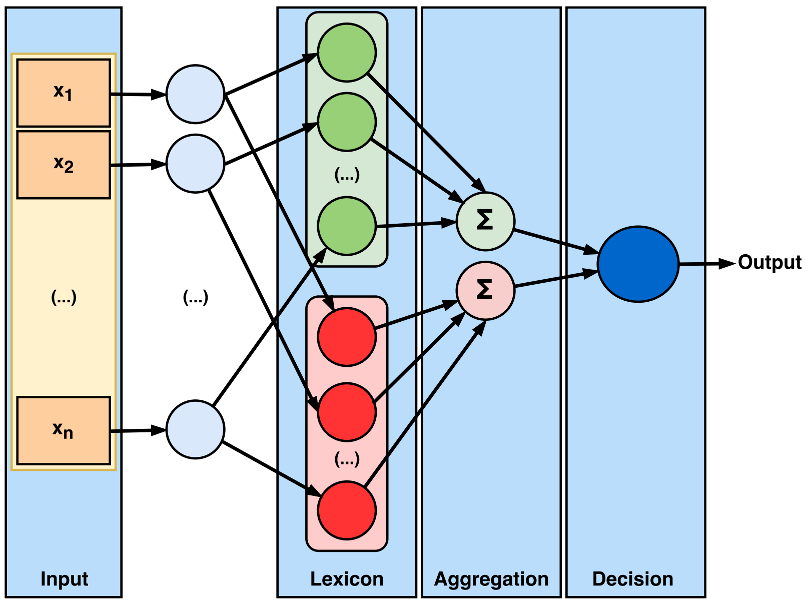


FIGURE 6.2: The LEXICNET network topology

6.1.2.1 The Lexicon network topology

The topology of the lexicon network corresponds to a shallow network where a regression is trained for each class and the output of each regressor (the aggregation layer in Figure 6.2) is fed into a SoftMax normalisation layer (the decision layer in Figure 6.2) so as to produce a probability distribution as a final output, which is necessary to backpropagate the error gradient and find the optimal lexicon weights. The operation of each layer is summarised in Equations 6.4, 6.5, 6.6 and 6.7 respectively and reviewed in the rest of this section.

$$I(t) = \log(1 + \text{RawFrequency}(t)) \tag{6.4}$$

$$L(t, c) = W_{t,c} \cdot I(t) \tag{6.5}$$

$$A(T, c) = \sum_{i=1}^{|T|} L(T_i, c) \tag{6.6}$$

$$O(T) = \bigoplus_{i=1}^{|c|} \frac{e^{-A(T,C_i)}}{\sum_{j=1}^{|c|} e^{-A(T,C_j)}} \tag{6.7}$$

The vocabulary input layer The input layer feeds term frequencies into the network. The output of this layer is a $n \times 1$ matrix F (see Equation 6.2) where n is the number of terms in the lexicon. The inputs are logarithmically scaled to smooth out the differences in input length using the ScaledFrequency function detailed in Equation 6.8, where RawFrequency corresponds to the number of occurrences of a term in the current input.

$$I(t) = \log(1 + \text{RawFrequency}(t)) \tag{6.8}$$

The lexicon layer The lexicon layer maps a term to its respective class-dependent scores. This layer is represented by a $c \times n$ matrix W (see Equation 6.2) where n is the number of terms in the lexicon and c the number of classes.

$$L(t, c) = W_{t,c} \cdot I(t) \tag{6.9}$$

The aggregation layer The aggregation layer adds up evidence towards a class from the previous layer by performing an inner product calculation between $n \times 1$ matrix F and $c \times n$ matrix W. The output of this layer is a $1 \times c$ row vector O containing the aggregated scores for each of the classes.

$$A(T, c) = \sum_{i=1}^{|T|} L(T_i, c) \tag{6.10}$$

The decision layer The decision layer transforms the row vector O into a probability distribution using the $\mathrm{SoftMax}$ function (described in Equation 6.11) and returns it as the output of the network. In the prediction phase, the decision layer returns the ArgMax of the probability distribution in the output in order to compute the accuracy of the current model. In the induction phase, it returns a probability distribution over the classes in order for the network to learn, since the ArgMax function is not differentiable and therefore has no error gradient to backpropagate.

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \forall j \in 1..N \tag{6.11}$$

$$O(T) = \bigoplus_{i=1}^{|c|} \frac{e^{-A(T,C_i)}}{\sum_{j=1}^{|c|} e^{-A(T,C_j)}} \tag{6.12}$$

6.1.2.2 Optimisation

We train our network using backpropagation and the full batch Gradient Descent algorithm with Nesterov momentum (Nesterov, 2007) as described in Equation 6.13, where μ is the velocity scaling parameter and γ is the learning rate. Nesterov momentum works by updating each weight in two steps: firstly using a scaled version of their previous update (conditioned by a fixed velocity parameter), followed by a course correction step using the error gradient calculated after the first update, mimicking the effect of momentum in physical objects.

$$\begin{aligned} w_i &\leftarrow w_i - \mu \times u_i \\ u_i &\leftarrow \frac{\partial J}{\partial w_i} \\ w_i &\leftarrow w_i - \gamma \times u_i \end{aligned} \tag{6.13}$$

6.1.2.3 Example

In this section we will walk through a high level example of lexicon learning on the toy dataset present in Table 6.1. The toy dataset only contains four examples in two different classes in an

TABLE 6.1: Example dataset for stance classification

Agreement	**Disagreement**
We agree! Excellent!	You are wrong on ridiculously many levels it hurts.
People like this guy happen to be right wing republicans.... Isn't that peculiar?	Abortion is like killing a baby. Would you do that? It should be banned...

Term	**Agreement**	**Disagreement**
agree	0.846	2.293
excellent	1.440	1.108
wrong	0.192	1.685
ridiculously	1.323	0.404
many	0.158	1.814
	(...)	

TABLE 6.2: Example of lexicon initialisation

argument stance classification task. They are collected and slightly modified for explanation purpose from real datasets extracted from debate forums.

A standard lexicon is initialised with weights randomly sampled on a Gaussian distribution with parameters $\mu = 1$ and $\sigma = 0.67$, as shown in Table 6.2. The weights are drawn randomly from the distribution $N(1, \frac{2}{(n_{\text{in}}+n_{\text{out}})})$ where n_{in} is the number of incoming connections, i. e., 1 (the frequency of the term), and n_{out} is the number of outgoing connections, i. e., the number of classes. This initialisation scheme is directly inspired by the Glorot initialisation scheme by Glorot and Bengio (2010) to initialise weights in deep neural networks.

During the first forward pass, a batch of randomly sampled instances is fed into the lexicon. The lexicon produces a prediction for each of these instances, which are then used to calculate a loss. For the purpose of demonstration, we will take a mini-batch of size 1. Let us take the following instance, "Ridiculously, all republicans agree, so abortion should be banned" from the agreement class, represented as $x_0 = \langle 1, 0, 0, 1, 0, \ldots \rangle$ and $y_0 = \langle 1, 0 \rangle$ on the vector space induced by the vocabulary of the lexicon.

Forward pass The forward pass yields the aggregated scores of $\langle 4.072, 6.780\rangle$ and the normalised probabilities of $\langle 0.062, 0.937\rangle$.

Cost calculation Since $y_0 = \langle 1, 0\rangle$ and our prediction $\hat{y}_0 = \langle 0.062, 0.937\rangle$, our cross-entropy loss, computed as $CrossEntropy(y, \hat{y}) = -\sum_{c=1}^{M} y_{o,c} \cdot \log(p_{o,c})$ is 1.207.

Backward pass During the backward pass, the lexicon scores of each term is updated by subtracting the derivative of the error with respect to the current scores multiplied by a weighting factor (γ in Equation 6.13). We compute the derivative of the error with respect to the lexicon score using the finite difference method, described below:

1. Compute the error E_1 with the current lexicon score ;
2. Compute the error E_2 after increasing the lexicon score l_w by a small amount h ;
3. Return $\frac{E_2 - E_1}{h}$ as the finite difference approximation of the partial derivative of E with respect to l_w.

The finite difference method has the computational advantage that for a given iteration, step 1 is identical for all lexicon scores and as such only step 2 has to be done for each term of the lexicon. We perform an update using $h = 0.1$ and $\gamma = 0.1$ for the terms “agree” and “ridiculously”, because they are the only terms of the instance that are in our limited (for the sake of explanation) lexicon vocabulary.

In the case of the “agree” term, the hypothetical error after updating the agreement score 0.846 to 0.946 becomes 1.175, yielding an partial derivative of $\frac{1.175-1.207}{0.1} = -0.32$ and therefore an update of $-(-0.32 \times 0.1) = +0.032$. Conversely, the hypothetical error after updating the disagreement score from 2.293 to 2.393 becomes 1.285, yielding a partial derivative of $\frac{1.285-1.207}{0.1} = 0.78$ and therefore an update of $-(0.78 \times 0.1) = -0.078$. Similarly for “ridiculously”, the hypothetical error after updating the agreement score 1.323 to 1.423 becomes 1.227, yielding an partial derivative of $\frac{1.227-1.207}{0.1} = 0.002$ and therefore an update of $-(0.002 \times 0.1) = +0.0002$. Conversely, the hypothetical error after updating the disagreement score from 0.404 to 0.504 becomes 1.197, yielding a partial derivative of $\frac{1.197-1.207}{0.1} = -0.001$ and therefore an update

TABLE 6.3: Example of lexicon after first iteration

	Agreement			Disagreement		
Term	Iteration 1	Update	Iteration 2	Iteration 1	Update	Iteration 2
agree	0.846	+0.032	0.878	2.293	−0.078	2.215
			(...)			
ridiculously	1.323	+0.0002	1.3232	0.404	−0.0001	0.4038
			(...)			

of $-(-0.001 \times 0.1) = -0.0001$. We can see from the size of those updates that “agree” and “ridiculously” become more associated with the “agreement” class as well as less associated with the “disagreement” class due to having been observed in an instance whose label was “agreement”.

6.2 Lexicon regularisation

In order for our lexicon to avoid overfitting, it is necessary to alter the learning process by adding a generalisation term. In this section we will detail the need for generalisation, explain complexity-based regularisation in the context of empirical risk minimisation-based learning, as well as other forms of regularisation that can be applied to lexicon classifiers.

6.2.1 Complexity regularisation with p-norms

The most basic form of model regularisation comes from restricting the p-norm of its parameters, where p is usually 1 or 2. Complexity-based regularisation is applied by adding an additional term to the error function, typically weighted by a parameter λ, which represents the complexity of the model.

The p-norm is defined as follows, where θ is the set of parameters of the model:

$$|\theta|_p = \sqrt[p]{\sum w^p} \tag{6.14}$$

The intuition behind restricting the norm of the parameters of a model stems an analogy with Occam's razor. Stopping the norm of the model parameters from growing out of control is done by modifying the loss function $J(\theta)$ to incorporate the fitting error $\mathfrak{L}$ as well as the complexity of the model $|\theta|_p$, giving us an equation of the form $J(\theta, X, Y) = \mathfrak{L}(\theta, X, Y) + \lambda \cdot |\theta|_p$. Such equation being still mostly differentiable, gradient descent and backpropagation can still be applied and the restriction propagated all the way through the weight update of the lexicon. The ensuing optimisation process during the learning phase is thus forced to find a compromise between a well-fitting model (low error rate) and a simple model (low p-norm), the importance of simplicity being weighted by the mixing parameter λ. In the following subsections we detail different norms and how they impact the learning process.

6.2.1.1 L1-norm

L_1-norm minimisation consists in minimising the sum of absolute values of the parameters of the model. The main effect of minimising the L_1-norm is that it leads to erasing the weights corresponding to the features that do not contribute enough to a good model by setting them to 0, giving those features no importance at all when performing predictions. This has the resulting effect of selecting features during the learning phase, by only giving a non-zero weights to those who have enough influence on the accuracy of the model.

$$|\theta|_1 = \left|\sum w\right| \tag{6.15}$$

6.2.1.2 L2-norm

L_2-norm minimisation consists in minimising the sum of squared values of the parameters of the model. Unlike L_1-norm minimisation, L_2-norm minimisation tends to lead to fewer features being eliminated but instead more feature being kept with a much smaller weight, preserving the original size of the lexicon. Intuitively, the reason for this effect is that given two features with a similar contribution to accuracy, and given the quadratic shape of the L_2-norm, a greater decrease of the loss can be made by decrementing the weight of a feature that is high than by decrementing the weight of a feature that is already close to 0. This is opposed to L_1-norm, where the decrease in loss is the same regardless of the starting weight of the features. We refer to (James et al., 2013, Chapter 6) and (Friedman et al., 2001, Chapter 3) for a more thorough treatment of the matter.

$$|\theta|_2 = \sqrt{\sum w^2} \tag{6.16}$$

6.2.1.3 Elastic-Net

In order to counteract the drawbacks of both L_1 and L_2 regularisation, Elastic-Net combines them into a common objective that is weighted by a combination parameter λ.

$$|\theta|_{en} = \lambda \times \sqrt{\sum w^2} + (1 - \lambda) \times \left|\sum w\right| \tag{6.17}$$

In (6.17) the λ parameter controls the importance of each norm and is typically fixed throughout the training phase.

6.2.2 Drop-out regularisation

Another form of regularisation that can be applied to lexicons is the drop-out Srivastava et al. (2014). The drop-out was designed to regularise deep neural networks where some computational units tended to specialise on certain features, thus building a more fragile predictor in the learning phase and leading to overfitting. In this section we explain the principles behind drop-out regularisation in neural networks before exploring its application on simple computational graphs such as lexicons. The literature suggests that analytically-motivated regularisation techniques such as $p - norm$ minimisation based regularisation can be combined to drop-out Srivastava et al. (2014), but there is no definitive study of their synergistic/antagonistic effects.

6.2.2.1 Drop-out in neural networks

The basic mechanism by which neural networks avoid overfitting is by selecting a set of weights residing between two specific layers of the network and randomly ignoring some of them at training time. We illustrate that process in Figure 6.3, where drop-out is used between layers

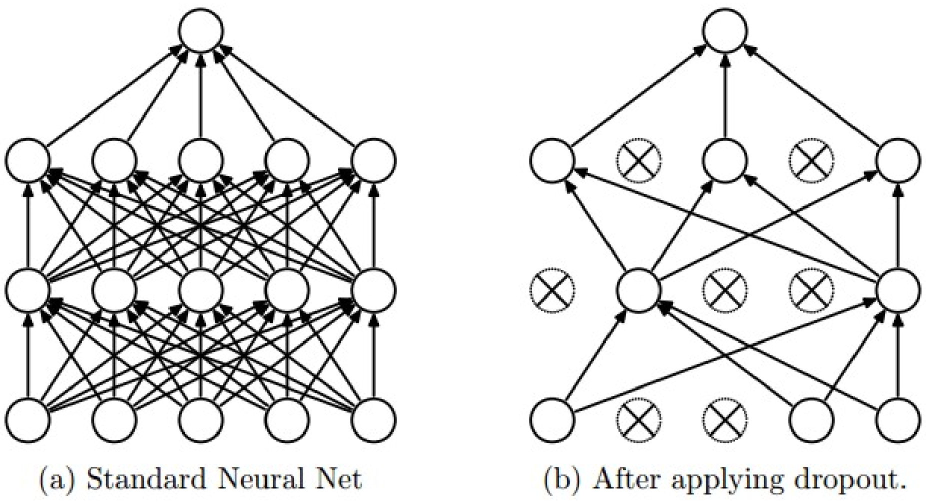


FIGURE 6.3: Visualising drop-out

1 and 2 as well as between layers 2 and 3. By forcing the neural network to restrain itself to a subset of its weights, we force it to build redundancies and avoid specialised units, thus making it more robust to overfitting.

We note that according to Srivastava et al. (2014), drop-out regularisation is not as effective for text classification tasks as for computer vision tasks, but its effect is still deemed statistically significant.

6.2.2.2 Drop-out in lexicons

Because lexicons can be expressed as simple computational graphs, we can apply the same drop-out procedure in order to force it to not put too much emphasis on specific terms which might be highly associated to a class in the training set, but instead be more conservative with its parameters. Applying the drop-out procedure to lexicon learning thus consists in ignoring part of the terms present in the text during the learning phase, and because documents can be of varying size it is important to keep the drop-out rate to lower values than the ones recommended in neural network training in order to not underfit the resulting model.

6.2.2.3 Conclusion on lexicon regularisation

Regularising models is an important part in order to avoid lexicons prioritising spurious correlations over genuine class valence, a problem that is particularly present in lexicons built from simple statistical relationships in corpora, e. g., lexicons built from pointwise mutual information. There are many ways to regularise such models, either during the optimisation phase ($p - norm$ minimisation, drop-out) or during the model selection phase (cross-validation).

6.3 Complex lexicon architectures

While modelling lexicons as computational graphs helps us make them more powerful using regularisation techniques from the machine learning literature, their modelling power remains relatively weak from a linguistic perspective. In this section we elaborate on more complex lexicon architectures that can be used to learn lexicon terms with a class valence as well as term modifiers. Term modifiers are terms which do not possess a class valence but instead have a strengthening or weakening effect on other terms in their vicinity, e. g., quantitative adverbs such as “extremely”, “barely”, or “very”. We name this class of learned models complex lexicons, as opposed to simple lexicons which are only concerned with terms with class valence.

Complex lexicons are learned using gated computational graphs. We present the mechanism of gating in differentiable computational architectures in the next section, before applying it to learning a complex lexicon in the following section.

6.3.1 Gating for information flow control

Gating is based on the use of two arithmetic operations, addition and multiplication, in order to control the flow of information through a computational graph. In a neural network, a multiplicative gate is typically used with a sigmoid unit (see Figure 6.4) in order to control the passage of information. As the sigmoid unit is bounded in $[0, 1]$, it can learn to either block information (by returning a value sufficiently close to 0) or let it pass through (by returning a value sufficiently close to 1). During the backpropagation stage, those gates can then be used to let the error gradient pass through or be stopped, which is a mechanism used in long short-term memory networks (Hochreiter and Schmidhuber, 1997) and gated recurrent units (Chung et al., 2014).

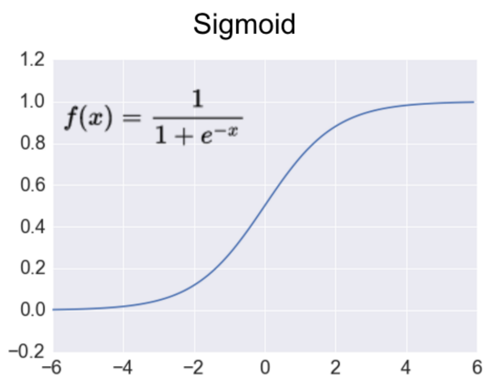


FIGURE 6.4: The Sigmoid function

In these approaches gating is used to learn more complex architectures by creating a notion of memory. These algorithms process inputs as sequences of timesteps, performing a forward phase on each timestep, and use a recurrent connection in the computational graph in order to pass information from one timestep to the next. By using a gating function to moderate this recurrent connection, it allows these architectures to selectively pass or block information from one timestep to another, and conversely to selectively backpropagate or not an error gradient through those timesteps. In the rest of this section we will explore a way to use gating to enhance our lexicon by learning lexicon terms and modifier terms concurrently.

6.3.2 Joint learning of lexicons and modifiers

We define lexicon terms as terms that possess a class valence in a classification scheme of interest (e. g., positive/negative, agreement/disagreement, etc.). We define modifier terms as terms which do not possess an intrinsic class valence, but have the ability of modifying the class valence of other terms in their vicinity. For example, an adverb such as "very" would intensify the strength of a close term such as "happy", while "barely" would diminish it, and "not" would cancel it.

In this section we present the RELEXNET architecture, presented in Figure 6.5. The goal of RELEXNET is to jointly learn lexicon and modifier terms. We expect modifiers to be terms which alter the class valence of neighbouring terms to different degrees, so in order to accelerate the learning we limited the sets of candidate terms to adverbs (e. g., very, etc.) and conjunctions (e. g., and, or, etc.). Instead of representing instances as unordered bag of words, we represent them as sequences of terms which are one-hot encoded, i. e., represented by a vector of size $|V|$

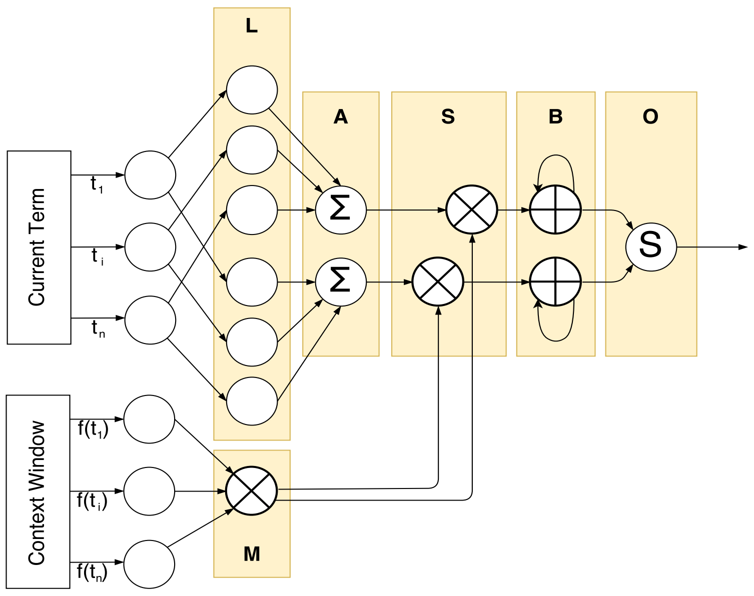


FIGURE 6.5: The RELEXNET architecture

where V is the vocabulary and where the values are all zeroed out except for the dimension of the corresponding term. Each instance of size M is thus represented as a matrix of size $M \times V$. Term are processed in their own timestep, and the current score (aggregating the current timestep with the previous ones) is passed to the next timestep until the final term is processed and a prediction can be output.

Figure 6.5 presents an architecture for binary classification. At the left hand side of the diagram is the standard vocabulary layer (first layer), where each term of the instance is fed one at a time, activating the corresponding lexicon units (second layer, marked **L**) before being summed up in the aggregation unit (third layer, marked **A**), which is then modified by modifier terms nearby (fourth layer, marked **S**) and passed on to the next timestep (next term of the instance) via a summation gate (fifth layer, marked **B**). At the end of the sequence we examine the output of the Softmax function (sixth layer, marked **O**) which determines the probability distribution over classes. At the bottom of the diagram we process a context window of arbitrary size. For a context window of size c and for timestep i, all terms from t_{i-c} to t_{i+c} are fed as a bag of words as the context window ($f(t_i)$ denoting the frequency of term i in the context window). A weighted sum of those terms is then multiplied together before being fed in the top part of the network, to modify the score of the current term.

The forward pass of the graph can be represented in the following way:

$$L_{i,j} = w_{i,j} \times f_1(t_i) \tag{6.18}$$

$$M = \prod_{i=1}^{|t|} (m_i \times f_2(t_i)) \tag{6.19}$$

$$A_j = \sum_{i=1}^{|t|} L_{i,j} \tag{6.20}$$

$$S_{i,j} = A_i \times M \tag{6.21}$$

$$B_{j,t} = B_{j,t-1} + \sum_{i=1}^{|c|} S_{i,j} \tag{6.22}$$

$$O_t = \bigoplus_{j=1}^{|c|} \frac{e^{-B_{j,t}}}{\sum_{m=1}^{|c|} e^{-B_{m,t}}} \tag{6.23}$$

In Equation 6.23 the function $\bigoplus$ takes for input a sequence of real numbers and outputs a vector that contains them in the provided order. The network passes on the data, and at each timestep feeds a one-hot encoding vector of the current term. We detail each part as follows:

- $L_{i,j}$ receives the one-hot encoding vector of the current term i for class j. In Figure 6.5 we can observe that each term is mapped to N lexicon units, where N is the number of classes. It is formalised in Equation 6.18.
- A_j sums the amount of evidence for class j. It is formalised in (6.20).
- M combines all modifiers present within the arbitrary word window into one modification score. It is formalised in Equation 6.19.
- $S_{i,j}$ applies the modification score calculated in (6.19) to the score calculated in (6.20) for term i and class j. It is formalised in (6.21).
- $B_{j,t}$ communicates the output of $S_{i,j}$ at timestep t to the next timestep $t+1$. It is formalised in Equation 6.22.
- O_t receives the output of $B_{j,t}$ (for all classes j) and outputs a probability distribution over the classes at timestep t. It is formalised in Equation 6.23.

6.4 Evaluation

This section focuses on the evaluation of the shallow supervised algorithms presented during the course of this chapter. We evaluate our approaches using two classification tasks: a standard stance classification task (abbreviated as SSC) where a single dataset is separated into a training and test set and therefore the classifiers learned during the process are directly applicable from one to the other, and a transfer learning task (abbreviated as STL), where we train our models on a noisy labelled dataset, RNLC, in order to see whether the knowledge learned on such data can be applied outside of it. Because the RNLC dataset only possesses two classes (agreement and disagreement), only a binary classifier is learned from it.

To summarise, the evaluation tasks are the following:

- SSC-2: stance classification with 2 classes: agreement, disagreement ;
- SSC-3: stance classification with 3 classes: agreement, neutrality, disagreement ;
- STL-2: stance classification with 2 classes from a noisy-labelled external dataset.

6.4.1 Hyperparameter setting

The lexicon models developed in this chapter, LEXICNET and RELEXNET, are subject to a few parameters that need to be set prior to the learning phase, which are commonly called the hyperparameters of the models. Because the goal of our evaluation is not to fine-tune models until they work, we fixed sensible hyperparameters prior to any learning in order to make the models comparable to the baselines described below:

The lexicon size was set to 300, which was found in the literature to be sufficient to capture most terms with a significant class valence ;

The elastic-net regularisation mixing parameter was set to 0.5, which implies that an equally weighted linear combination of the $L1 - norm$ and $L2 - norm$ is minimised during the training phase ;

The drop-out rate was found to not be effective in prior experiments and was set to 0 ;

The training duration was set to 100 epochs, where each epoch is one iteration over the training set.

In the case of LEXICNET, the representation used for each instance was tf-idf weighting, as described in the background section 3.3.2.1 on non-sequential representations.

For both LEXICNET and RELEXNET, the lexicon terms were pre-selected by ranking 300 terms from the training set vocabulary by normalised pointwise mutual information score after removing terms which are known to have no stance valence (articles, determinants, punctuation marks). Additionally for RELEXNET, a set of adverbs is extracted from the training set and ranked by decreasing frequency, and the top 50 was selected as the set of valence modifiers. That number was chosen from empirical observation of the distribution of adverbs in the English language[1].

6.4.2 Baselines

We use a top-performing baseline from the literature, which is a lexicon in emotion analysis from short text (Bandhakavi et al., 2014) and which has a learning phase similar to the LEXICNET and RELEXNET architectures discussed in the context of this thesis. We also select three machine learning baselines: support vector machines (SVM), decision trees (DECISION TREE) and multinomial naive Bayes (NAIVE BAYES), due to their popularity in text analysis. While strictly speaking those machine learning baselines are outside of the scope of comparison of the techniques developed in this chapter due to the fact that they are not lexicons and as such are not as interpretable as lexicon-based methods, they serve as a guide indicating the difficulty of each classification problem. The baselines and their parameters are as follows:

EM-LEX is a stance lexicon built from the expectation maximisation algorithm, as used for emotion classification by Bandhakavi et al. (Bandhakavi et al., 2014). It was built using a vocabulary of 300 terms, similarly to LEXICNET and RELEXNET ;

SVM is a support vector machine (described in Section 3.4.3) with a linear kernel and parameter $C = 1.0$;

[1]https://www.talkenglish.com/vocabulary/top-250-adverbs.aspx

NAIVE BAYES is a multinomial naive Bayes classifier (Manning et al., 2008). Multinomial naive Bayes is a naive Bayesian classifier that assumes a multinomial distribution over the features, which makes it well-fitted and popular for text classification. The model was trained with a Laplace smoothing parameter of 1 and a uniform prior distribution (due to the rebalancing of the datasets) ;

DECISION TREE is a decision tree, built using the C5.0 algorithm, a refined version of Quinlan's C4.5 algorithm (Quinlan, 1990), which is a standard in decision tree learning algorithms. It was trained using the splitting criterion based on Gini impurity metric, no maximal depth, and a requirement of minimum 2 samples in order to split a node.

All those algorithms were used from the Scikit-learn toolkit (Pedregosa et al., 2011), with default parameters (no hyperparameter tuning was done). Where applicable, the instances were represented as vectors of tf-idf weighted frequencies, as described in Section 3.3.2.1 on non-sequential representations.

6.4.3 Results of shallow approaches

In this section we present the results of the shallow approaches for each of the three tasks defined above. We show an example of a lexicon computed on the IAC dataset and LEXICNET in Appendix C. The example lexicon was kept small for the sake of conciseness.

6.4.3.1 Results of task SSC-2

We present the results in Table 6.4. The results are illustrated per dataset in Figure 6.6, and averaged over all datasets in Figure 6.7. We can observe that RELEXNET outperforms the other approaches in two of the three datasets used for testing. An examination of Figure 6.6 tends to show that besides decision trees in the ABCD and IAC datasets, all other classifiers tend to perform very similarly. However, a one-way ANOVA (see Table 6.5) shows that there is a difference worth investigating in all three datasets. We perform a Tukey post hoc test in order to account for the increase in type 1 error inherent to multiple comparison (see Table 6.6) and confirm that besides decision trees under-performing on the IAC and ABCD datasets, and

	IAC	ABCD	SETC
RELEXNET	**0.700**	**0.703**	0.641
LEXICNET	0.670	0.686	0.651
SVM	0.670	0.675	0.666
NAIVE BAYES	0.645	0.661	**0.684**
DECISION TREE	0.606	0.595	0.671
EM-LEX	0.656	0.664	0.650

TABLE 6.4: Accuracy results of Task SSC-2

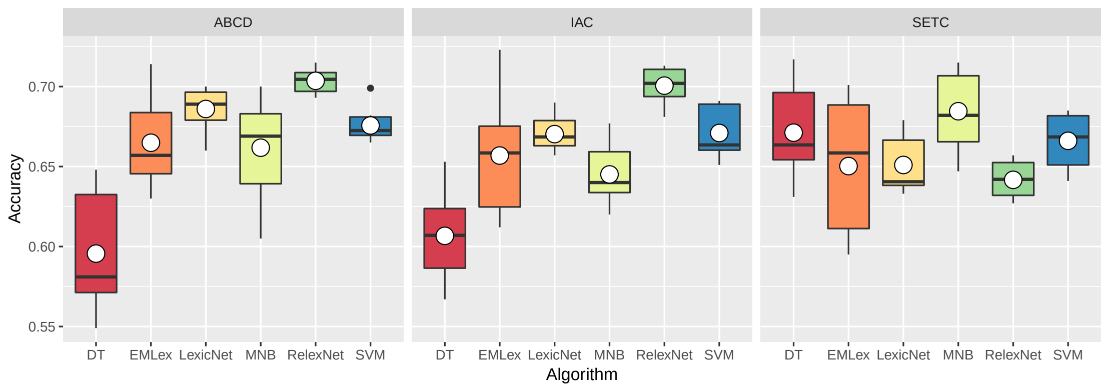


FIGURE 6.6: Box plot of accuracy results of task SSC-2 per dataset. The white circles indicate the arithmetic means.

Dataset	F-Value	P-Value
IAC	19.147	**6.538e-11**†
ABCD	22.991	**2.74e-12**†
SETC	3.685	**0.006**†

TABLE 6.5: Results of ANOVA on task SSC-2. † indicates statistical significance at 0.95 confidence.

RELEXNET performing significantly better than NAIVE BAYES overall and EM-LEX in IAC and ABCD, most other algorithms perform very similarly. While their performance is close, it is worth emphasising that Tukey's Honestly Significant Difference test corrects for multiple comparisons and as such, a large effect size is required to break the threshold of statistical significance.

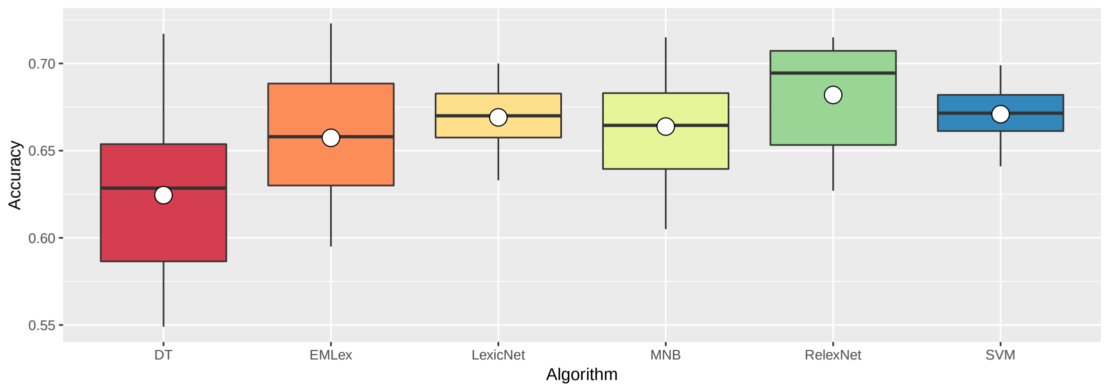


FIGURE 6.7: Box plot of accuracy results of task SSC-2 averaged over all datasets. The white circles indicate the arithmetic means.

	IAC		ABCD		SETC	
	Diff.	Adj. P-Value	Diff.	Adj. P-Value	Diff.	Adj. P-Value
EM-LEX / DECISION TREE	0.05	**0.00**†	0.07	**0.00**†	-0.02	0.49
LEXICNET / DECISION TREE	0.06	**0.00**†	0.09	**0.00**†	-0.02	0.53
NAIVE BAYES / DECISION TREE	0.04	**0.00**†	0.07	**0.00**†	0.01	0.86
RELEXNET / DECISION TREE	0.09	**0.00**†	0.11	**0.00**†	-0.03	0.14
SVM / DECISION TREE	0.06	**0.00**†	0.08	**0.00**†	-0.01	1.00
LEXICNET / EM-LEX	0.01	0.77	0.02	0.40	0.00	1.00
NAIVE BAYES / EM-LEX	-0.01	0.85	-0.00	1.00	0.03	0.05
RELEXNET / EM-LEX	0.04	**0.00**†	0.04	**0.01**†	-0.01	0.98
SVM / EM-LEX	0.01	0.73	0.01	0.92	0.02	0.76
NAIVE BAYES / LEXICNET	-0.03	0.15	-0.02	0.25	0.03	0.06
RELEXNET / LEXICNET	0.03	0.05	0.02	0.60	-0.01	0.97
SVM / LEXICNET	0.00	1.00	-0.01	0.93	0.02	0.79
RELEXNET / NAIVE BAYES	0.06	**0.00**†	0.04	**0.00**†	-0.04	**0.01**†
SVM / NAIVE BAYES	0.03	0.13	0.01	0.80	-0.02	0.62
SVM / RELEXNET	-0.03	0.06	-0.03	0.13	0.02	0.32

TABLE 6.6: Tukey test for IAC and ABCD on SSC-2. † indicates statistical significance at 0.95 confidence.

	IAC	ABCD	SETC
RELEXNET	0.630	0.641	0.618
LEXICNET	0.607	0.615	0.625
SVM	**0.640**	**0.667**	0.632
NAIVE BAYES	0.611	0.618	**0.650**
DECISION TREE	0.600	0.598	0.572
EM-LEX	0.618	0.629	0.622

TABLE 6.7: Accuracy results of Task SSC-3

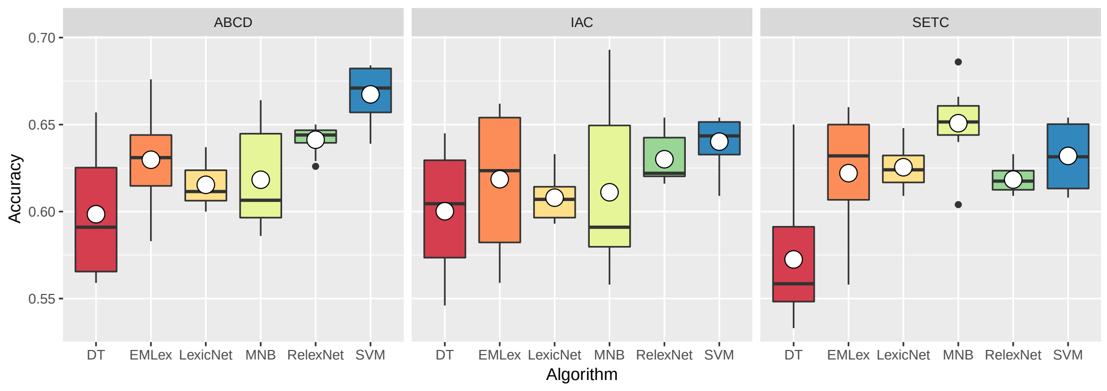


FIGURE 6.8: Box plot of accuracy results of task SSC-3 per dataset. The white circles indicate the arithmetic means.

6.4.3.2 Results of task SSC-3

We present the results of the SSC-3 task in Table 6.7, and illustrate them in the plot given in Figure 6.8 faceted per dataset. A visual inspection of the plots and the table of results show that while LEXICNET and RELEXNET outperform our lexicon baseline, they are outperformed by machine learning baselines (NAIVE BAYES and SVM) on all datasets. Figure 6.9 shows that when averaged over all datasets, the difference becomes much less significant where all algorithms tend to perform around the same with the exception of decision trees, which underperform by a reasonable margin.

We perform an ANOVA significance test in order to detect whether there is any significant difference between algorithm performance per dataset, which can be seen in Table 6.8, and find that only the ABCD and SETC datasets contain a significant level of between group variance.

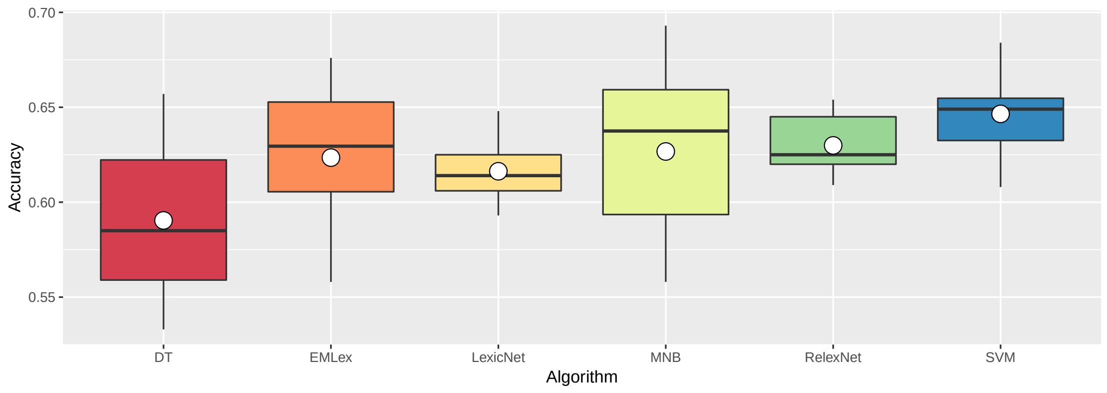


FIGURE 6.9: Box plot of accuracy results of task SSC-3 averaged over all datasets. The white circles indicate the arithmetic means.

Dataset	F-Value	P-Value
IAC	2.244	0.062
ABCD	9.137	**2.416e-06**[†]
SETC	11.269	**1.809e-07**[†]

TABLE 6.8: Results of ANOVA on task SSC-3. [†] indicates statistical significance at 0.95 confidence.

A Tukey HSD test run as a post hoc analysis, which we can see the results of in Table 6.9, shows that the only significant effects on the ABCD and SETC datasets is the poor performance of decision trees in SETC as well as the good performance of SVM in the ABCD dataset. All other algorithms tended to perform similarly enough that no significant difference could be detected.

6.4.3.3 Results of task STL-2

The STL-2 task focuses on applying a model learned from a dataset to another dataset, which is a weak form of transfer learning. Because the source dataset was labelled using an automated method inspired from the distant supervision literature, we only use classes which can be explicitly expressed: agreement and disagreement, rather than classes which show the absence of valence (neutrality). The results are given in Table 6.10 and illustrated in Figure 6.10.

An examination of the results shows that except for the SETC dataset, RELEXNET tends to outperform the other approaches. A visual inspection of Figure 6.11, which shows averages over all datasets confirms that suspicion. This can be explained by the fact that LEXICNET and

	ABCD		SETC	
	Difference	Adjusted P-Value	Difference	Adjusted P-Value
EM-LEX / DECISION TREE	0.03	0.07	0.05	**0.00**†
LEXICNET / DECISION TREE	0.02	0.66	0.05	**0.00**†
NAIVE BAYES / DECISION TREE	0.02	0.49	0.08	**0.00**†
RELEXNET / DECISION TREE	0.04	**0.00**†	0.05	**0.00**†
SVM / DECISION TREE	0.07	**0.00**†	0.06	**0.00**†
LEXICNET / EM-LEX	-0.01	0.79	0.00	1.00
NAIVE BAYES / EM-LEX	-0.01	0.91	0.03	0.11
RELEXNET / EM-LEX	0.01	0.91	-0.00	1.00
SVM / EM-LEX	0.04	**0.02**†	0.01	0.94
NAIVE BAYES / LEXICNET	0.00	1.00	0.03	0.21
RELEXNET / LEXICNET	0.03	0.21	-0.01	0.99
SVM / LEXICNET	0.05	**0.00**†	0.01	0.99
RELEXNET / NAIVE BAYES	0.02	0.32	-0.03	0.05
SVM / NAIVE BAYES	0.05	**0.00**†	-0.02	0.53
SVM / RELEXNET	0.03	0.20	0.01	0.81

TABLE 6.9: Tukey test for ABCD and SETC on SSC-3. † indicates statistical significance at 0.95 confidence.

	IAC	ABCD	SETC
RELEXNET	**0.657**	**0.640**	0.630
LEXICNET	0.639	0.628	**0.646**
SVM	0.628	0.615	0.631
NAIVE BAYES	0.596	0.621	0.624
DECISION TREE	0.591	0.584	0.637
EM-LEX	0.581	0.605	0.581

TABLE 6.10: Accuracy results of Task STL-2

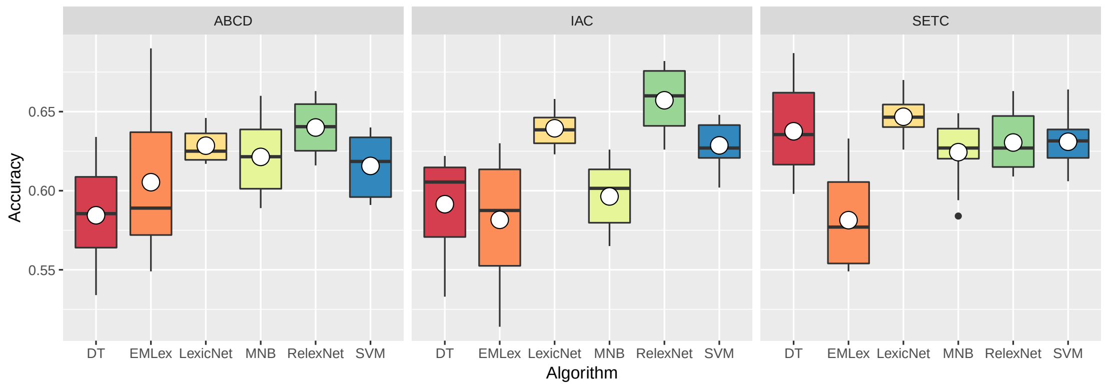


FIGURE 6.10: Box plot of accuracy results of task STL-2 per dataset. The white circles indicate the arithmetic means.

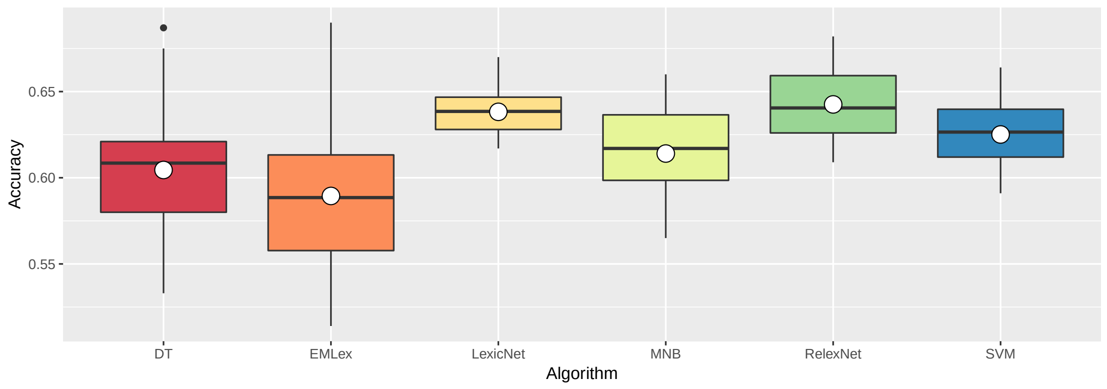


FIGURE 6.11: Box plot of accuracy results of task STL-2 averaged over all datasets. The white circles indicate the arithmetic means.

RELEXNET are both greatly regularised, to the point of often underfitting on some of the most straightforward tasks/datasets. While this is a disadvantage in some situations, it is reasonable that a transfer learning task would be the best suited situation to make use of such features. By being heavily regularised, the lexicon methods avoid picking up and putting too much weight on dataset-specific correlations, thus ensuring that the resulting models are more portable to new tasks. We observe that RELEXNET also outperforms EM-LEX by a reasonable margin, and that LEXICNET outperforms RELEXNET on the SETC dataset. This may be due to the particular nature of microblog data: in the absence of long text, it is harder for a complex architecture such as RELEXNET to leverage the weights of the valence modifiers. This means that the modifier weights learned at training time are not as applicable at test time, and therefore the lexicon weights become less effective at classifying the underlying data.

We perform an analysis of variance in order to pick up on statistical significance of the differences between all groups, shown in Figure 6.11. Our analysis shows that there is a difference worth

Dataset	F-Value	P-Value
IAC	14.175	**7.439e-09**[†]
ABCD	4.412	**0.001**[†]
SETC	9.802	**1.05e-06**[†]

TABLE 6.11: Results of ANOVA on task STL-2. [†] indicates statistical significance at 0.95 confidence.

investigating, which leads us to the Tukey HSD test in Table 6.12. While a visual inspection of results shows a clear difference between all algorithms, this difference was mostly not significant after adjusting for multiple comparisons. Besides the under-performance of EM-LEX in the SETC dataset and DECISION TREE in the IAC dataset, the only statistically significant differences present in the results of the STL-2 task are the dominance of lexicon methods over NAIVE BAYES.

6.4.4 Discussion

The results shown in this section, while looking extremely promising at visual inspection, rarely reach statistical significance when put under the lens of more rigorous testing. However there can be multiple reasons for this, and the lack of effect is only one of them. Indeed, statistical tests are sensitive to multiple parameters including the sample size, which is kept relatively low here (each fold being used as one trial) and as such requires a larger effect size, and the fact that statistical adjustments are made to account for performing multiple comparisons, in order for the Type 1 error rate to not increase dramatically. While our adjustments are not severe, they also require a much larger effect size to display statistical significance. This is a clear limitation of our study: a clearer hypothesis involving fewer algorithms would allow us to focus our evaluation and avoid multiple comparison testing while keeping the power of our tests high.

From the results we can see that lexicon methods can be relatively competitive with typical machine learning algorithms when trained only on raw text, coming often at the same level as algorithms such as SVM and NAIVE BAYES, which are standards of text classification, but with the additional advantage of being highly interpretable (lexicon weights can be interpret). A fair criticism would be that no hyperparameter tuning was done on those algorithms, but similarly the

	IAC		ABCD		SETC	
	Diff.	Adj. P-Value	Diff.	Adj. P-Value	Diff.	Adj. P-Value
EM-LEX / DECISION TREE	-0.01	0.95	0.02	0.60	-0.06	**0.00**†
LEXICNET / DECISION TREE	0.05	**0.00**†	0.04	**0.02**†	0.01	0.95
NAIVE BAYES / DECISION TREE	0.00	1.00	0.04	0.07	-0.01	0.79
RELEXNET / DECISION TREE	0.07	**0.00**†	0.06	**0.00**†	-0.01	0.98
SVM / DECISION TREE	0.04	**0.02**†	0.03	0.17	-0.01	0.99
LEXICNET / EM-LEX	0.06	**0.00**†	0.02	0.49	0.07	**0.00**†
NAIVE BAYES / EM-LEX	0.01	0.78	0.02	0.82	0.04	**0.00**†
RELEXNET / EM-LEX	0.08	**0.00**†	0.03	0.10	0.05	**0.00**†
SVM / EM-LEX	0.05	**0.00**†	0.01	0.97	0.05	**0.00**†
NAIVE BAYES / LEXICNET	-0.04	**0.00**†	-0.01	0.99	-0.02	0.27
RELEXNET / LEXICNET	0.02	0.63	0.01	0.95	-0.02	0.61
SVM / LEXICNET	-0.01	0.93	-0.01	0.92	-0.02	0.64
RELEXNET / NAIVE BAYES	0.06	**0.00**†	0.02	0.71	0.01	0.99
SVM / NAIVE BAYES	0.03	0.07	-0.01	1.00	0.01	0.99
SVM / RELEXNET	-0.03	0.14	-0.02	0.43	0.00	1.00

TABLE 6.12: Tukey test for IAC, ABCD and SETC on STL-2. † indicates statistical significance at 0.95 confidence.

LEXICNET and RELEXNET architectures were not tuned for each task and instead were issued a set of default parameters.

6.5 Chapter Summary

In this chapter we explored LEXICNET, an approach to learn a lexicon from a corpus based on the empirical risk minimisation framework which allows us to apply regularisation techniques to make our lexicons more robust. We extended LEXICNET to take into account neighbouring terms that have an intensification and diminishing effect in the RELEXNET architecture. Finally we evaluated our approaches against a state-of-the-art lexicon (EM-LEX) as well as standard machine learning methods (SVM, NAIVE BAYES, DECISION TREE). We found that a visual inspection of the results tend to show the dominance of RELEXNET and LEXICNET in binary classification problems, but more formal testing using ANOVA with a Tukey post hoc test shows that those differences, while promising, are not statistically significant.

Chapter 7

Lexicon-based Feature Models for Stance Detection

This chapter focuses on the use of lexicons as feature extractors, in order to improve the performance of a supervised classifier over a simple feature baseline. The two fundamental research questions that we seek to answer are the following: (1) can class-sensitive features learned on a corpus enrich an existing term representation and allow it to better represent argumentative content for the purpose of stance classification? and (2) what is the level of granularity (classifying words, classifying sentences, or classifying the entire instance) that is the most compatible with lexicon enhancement? In order to answer these research questions, we separate our feature vector into two parts:

- A semantic part that captures the meaning of the instance ;
- A class-sensitive part that captures information captured by the lexicon about the argumentative value of the instance.

We seek to answer the second research question by investigating the effect of different representations at varying levels of granularity at which we represent instances and observing how class-sensitive knowledge impacts the classification power. We introduce three types of feature models based on RELEXNET (detailed in Chapter 5): the instance-level lexicon feature model (INSTLEX), the sentence-level lexicon feature model (SENTLEX), and the word-level lexicon feature model (WOLLEX) in order to answer this research question, starting by defining lexicon-based feature extraction and then detailing compact and temporal feature models using lexicon-based features.

7.1 Using lexicons for feature extraction

While lexicons can be used as the underlying classification or regression model due to their simplicity and expressiveness, they can also be used to extract features from text in order to build a higher level representation of instances. For example, Somasundaran and Wiebe (2010b) use lexicons in order to detect argument expressions and use these expressions to compute a sentiment with a target, which is then used as a way to represent the text they are trying to classify. In this section we elaborate on multiple ways of generating a representation from a lexicon, using multiple granularity levels:

- At the word level, a lexicon extracts a set of numerical values representing the probability or degree of belonging of this word to multiple classes. This can be seen as a low dimensional embedding of the word in the class space, which can then be used to represent the instance as a sequence of M values (with M being the number of classes) of length L (with L being the number of words in the sentence).
- At the sentence level, a lexicon can extract a set of values for each word which are then averaged per sentence, resulting in an $L \times M$ representation where L is the number of sentences and M the number of classes.
- At the instance level, a lexicon can extract a set of values for each word which are either averaged over the whole instance, or first averaged over sentences then over the whole instance, resulting in an $1 \times M$ representation where M is the number of classes.
  - Over the whole instance give equal importance to each term, but means that one overly long sentence could overpower all others because of the lack of normalisation.
  - Over sentences then instance give equal importance to each sentence, but means that many short sentences which are heavily class-specific (e.g. expletives) could overpower the more moderate or neutral parts of the instance.

This distinction of granularity allows us to differentiate between multiple temporal representations using lexicon-based feature extraction. Given an instance of length L and a lexicon referencing M classes, we focus on the following potential feature models:

- At the word level, a $L \times M$ matrix where L is the number of words in the instance ;
- At the sentence level, a $L \times M$ matrix where L is the number of sentences in the instance ;
- At the instance level, a $S \times M$ matrix where S is the number of statistics that need to be collected over the instance. For example, an instance-level feature extractor could use the mean, minimum, and maximum statistics and extract $3 \times M$ features. Doing so compresses the instance maximally, losing smaller-grained information.

7.2 Instance-level feature model using lexicon features

We first propose an instance-level feature model in the form of a dense matrix of size $5 \times M$ descriptive statistics (mean, median, standard deviation, minimum, and maximum class values) where M is the number of classes, illustrated in Figure 7.1. The goal of this representation is to summarise the instance that we aim to classify and provide the learning algorithm with some form of prior knowledge over that particular instance. By doing so, an instance with a lot of vocabulary tied to one class will need a significant amount of evidence of another class in order for the learning algorithm to overcome that prior knowledge and predict the latter class. In this section we detail two representation schemes that are augmented with this compressed feature model, and the way they are computed.

7.2.1 Instance-level feature model with bag-of-words

The bag-of-words representation (detailed in Chapter 3) is the most common text representation scheme due to its versatility, ease of computation and effectiveness. Considering a vector space induced from a vocabulary of 300 terms, a bag-of-words representation is a 300-dimensional vector of numerical values representing the frequency (or some function of the frequency) of each term in the instance. Common functions of the frequency are raw frequency, Boolean

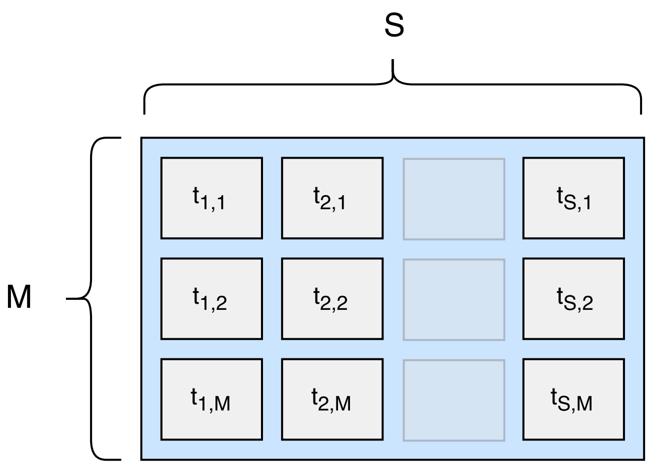


FIGURE 7.1: An instance-level representation scheme using lexicon statistics. M corresponds to the number of classes, while S corresponds to the number of statistics gathered for the representation.

thresholding (1 if frequency is over 0, 0 otherwise) and logarithmic smoothing (natural logarithm of the frequency).

The INSTLEX$_{\text{BOW+LN}}$ and INSTLEX$_{\text{BOW+RLN}}$ representation schemes use the dense vector of summary lexicon statistics (computed respectively from the LEXICNET and RELEXNET lexicon architectures) to extend a bag-of-word model, ending up with $|V| + |S| \times M$ features where V is the vocabulary size, S the set of statistics and M the number of classes. The feature vector illustrated in Figure 7.1 is flattened and appended to the bag-of-words vector, as shown in Figure 7.2.

This representation allows us to offer a dual view of the instance, where the bag-of-words represents the semantics of the text while the lexicon statistics represents class knowledge with which we want to influence the learning algorithm. It is suitable for most machine learning algorithms because the final representation is flat and thus does not necessitate complex algorithms such as recurrent neural networks.

7.2.2 Instance-level feature model with mean embedding

The mean embedding model (detailed in Chapter 3) is a common low dimensional representation scheme where text is represented as a vector where each dimension is an average of the corresponding embedding dimensions of all the terms contained in that instance. The INSTLEX$_{\text{EMB+LN}}$

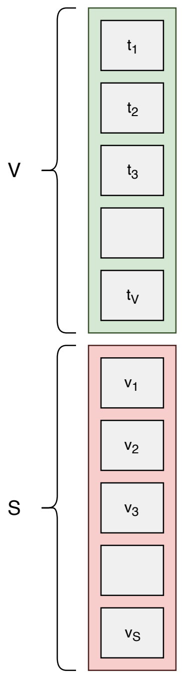


FIGURE 7.2: Instance-level feature model with bag-of-words. V is the size of the vocabulary, S the number of statistics calculated on the lexicon.

and INSTLEX$_{\text{EMB+RLN}}$ representations are built using the dense vector to extend a mean embedding model, ending up with $M + 5 \times N$ features where M is the embedding size and N the number of classes. We illustrate the representation in Figure 7.3.

This representation also offers a dual view of the instance, where the mean embedding is a compact representation of the semantics of the text while the lexicon statistics encodes class knowledge that we want to influence the learning algorithm with. Like previously described representations, it is suitable for most machine learning algorithms. The main difference is that the semantics of the terms are learned outside of our dataset, because we are able to use existing word embeddings learned on much larger datasets.

7.3 Sentence-level feature model using lexicon features

The sentence-level lexicon representation is a series of variable size of lexicon values, giving $S \times M$ values where S is the number of sentences in the current instance and M the number of classes. The lexicon values are first computed at the term level and then averaged over sentences, as illustrated in Figure 7.4. This representation allows us to reduce the differences of long and short sentences and focus on the average meaning of each sentence.

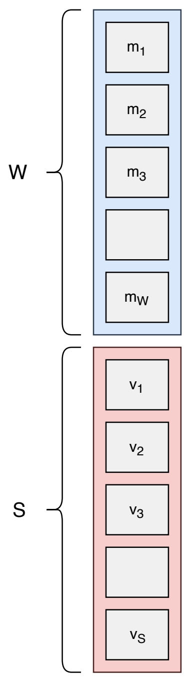


FIGURE 7.3: Instance-level feature model with mean embedding. W is the size of the embedding, S the number of statistics calculated on the lexicon.

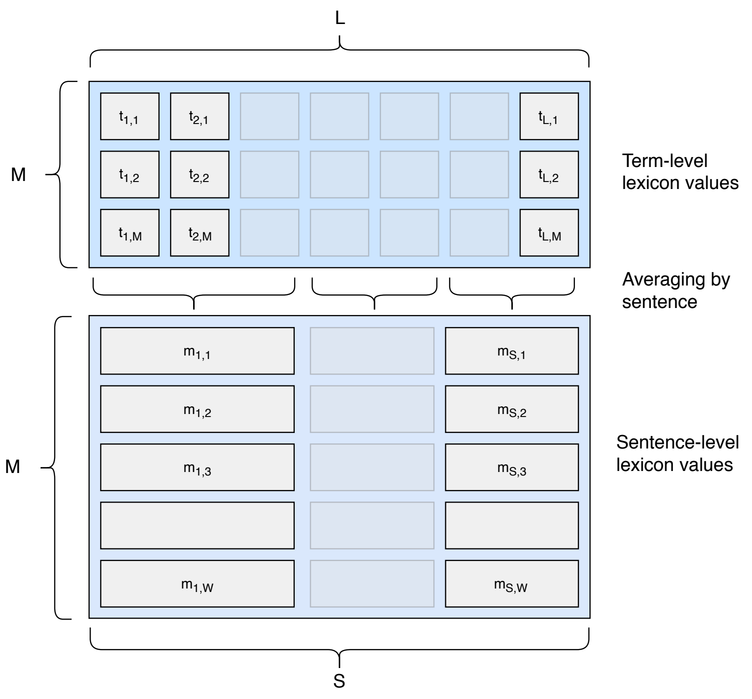


FIGURE 7.4: Lexicon representation of sentence level features. The lexicon values are averaged over sentences to form a more compact class-sensitive representation of the instance that still has a temporal dimension. M represents the number of classes, while L represent the number of sentences.

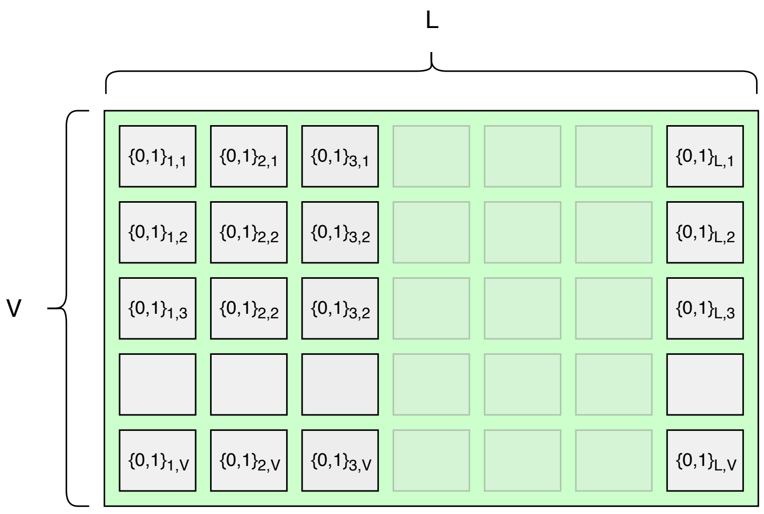


FIGURE 7.5: Sentence level frequency-based representation. V represents the size of the vocabulary, while L represents the number of sentences.

7.3.1 Sentence-level lexicon model with frequency-based sentence encoding.

The SENTLEX$_{\text{BOW+LN}}$ and SENTLEX$_{\text{BOW+RLN}}$ representations are time series of lexicon values (computed respectively from the LEXICNET and RELEXNET lexicon architectures) that are used to augment a frequency-based sentence representation. The semantic part of the representation is built by computing a traditional bag-of-words for at the sentence level, illustrated in Figure 7.5. Having a bag-of-words model at the sentence level allows us to model the flow of topic throughout a user comment, where not all sentences might be related to the topic at hand.

It is then augmented by the class-sensitive part of the representation, which is built by averaging lexicon values at the sentence level. The class-sensitive representation is appended to the semantic representation to form a sentence-level representation, illustrated in Figure 7.6. This allows us to annotate the topic of a sentence, represented by the bag-of-words model, with the overall argumentative stance of the words within it.

7.3.2 Sentence-level lexicon model with sentence embedding.

The SENTLEX$_{\text{EMB+LN}}$ and SENTLEX$_{\text{EMB+RLN}}$ representations are time series of lexicon values that are used to augment an embedding-based sentence representation, computed respectively using the LEXICNET and RELEXNET lexicon architectures. The semantic part of the representation is built using a sentence level mean embedding, illustrated in Figure 7.7. Similarly to

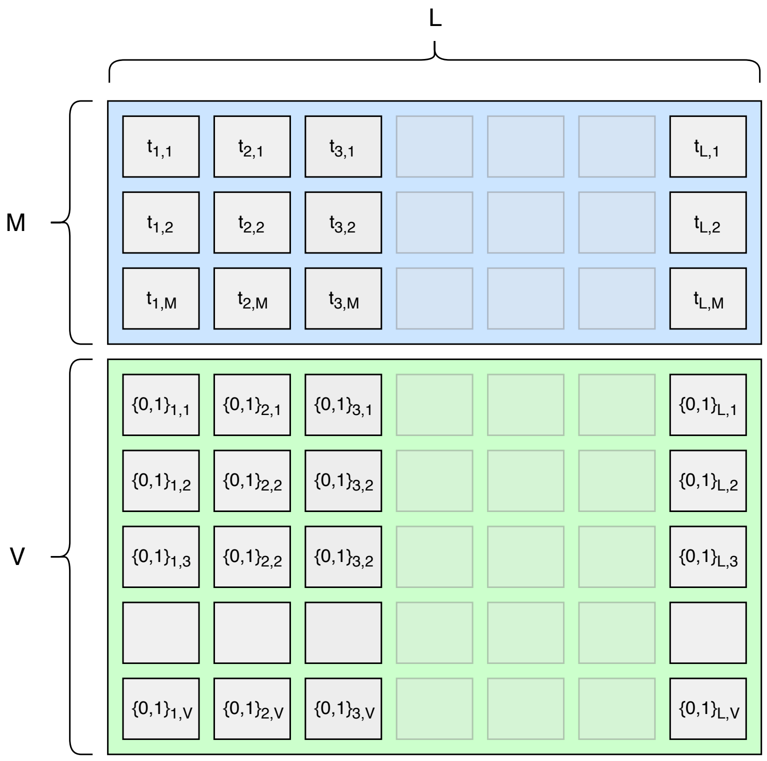


FIGURE 7.6: Augmented sentence level frequency-based representation. M represents the number of classes, V represents the size of the vocabulary, and L represents the number of sentences.

SENTLEX$_{\text{BOW+LN}}$ and SENTLEX$_{\text{BOW+RLN}}$, that mean embedding allows us to model the flow of topic throughout a user comment.

We then augment this representation by a class-sensitive vector, which is built by averaging lexicon values at the sentence level. Similarly to SENTLEX$_{\text{BOW+LN}}$ and SENTLEX$_{\text{BOW+RLN}}$, the class-sensitive representation is appended to the semantic representation to form a sentence-level representation, illustrated in figure 7.8.

7.4 Word-level feature model using lexicon features

The word-level representation is a time series of lexicon values associated to each term contained in an instance, building a $L \times M$ matrix (illustrated in Figure 7.9) where L is the number of terms in the current instance and M the number of classes. This representation allows us to consider the argumentative value of each term separately, instead of blending them into sentence or instance vectors.

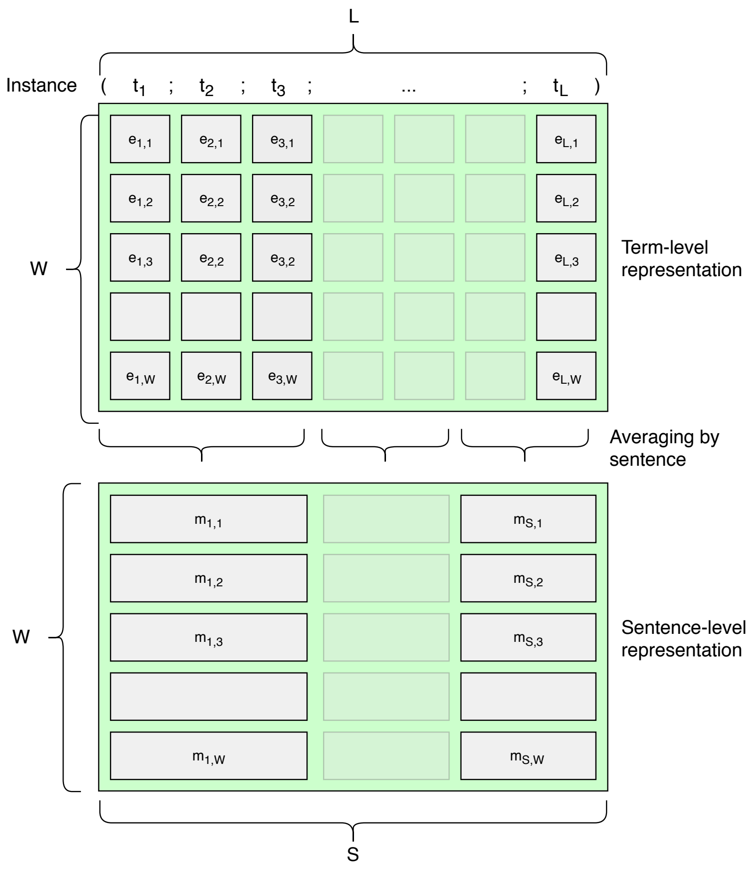


FIGURE 7.7: Lexicon representation of sentence level features. The embedding dimensions are averaged over sentences to form a more compact representation of the instance that still has a temporal dimension. W represents the size of the embedding, while L represents the number of sentences.

7.4.1 Word-level feature model with one-hot word encoding

The WoLLEx$_{\text{OH+LN}}$ and WoLLEx$_{\text{OH+RLN}}$ representations use a one-hot encoding (discussed in more detail in Chapter 3 and illustrated in Figure 3.3) as a way to model the semantic part of the instance. That one hot encoding is augmented by appending the lexicon vector for each term, as illustrated in Figure 7.10.

7.4.2 Word-level feature model with embedding

The WoLLEx$_{\text{EMB+LN}}$ and WoLLEx$_{\text{EMB+RLN}}$ representations use a word embedding (discussed in more detail in Chapter 3 and illustrated in Figure 3.4) as a way to model the semantic part of the instance. The embedding is augmented by appending the lexicon vector for each term, as illustrated in Figure 7.11.

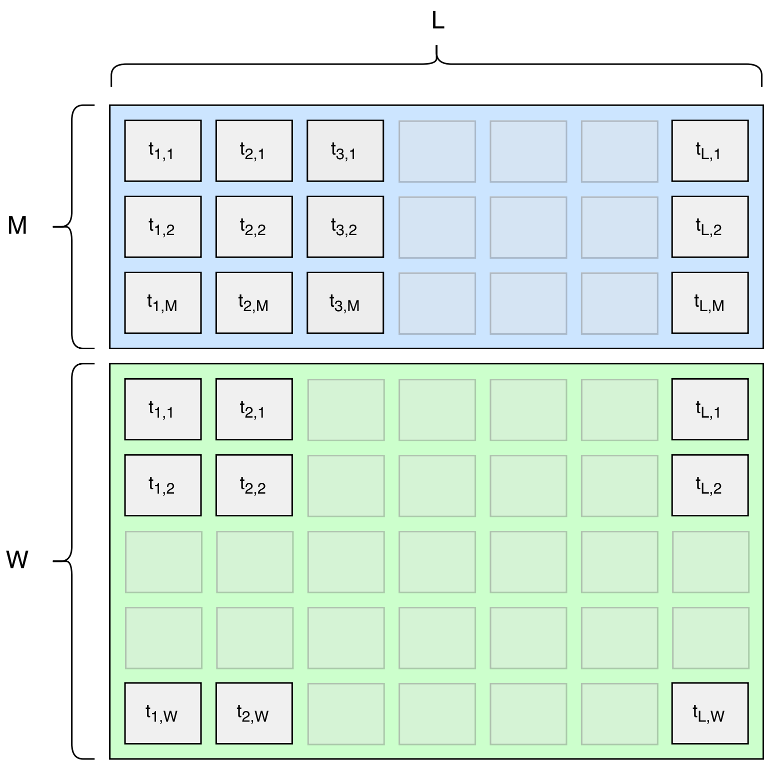

L
M
W
$t_{1,1}$ $t_{2,1}$ $t_{3,1}$ $t_{L,1}$
$t_{1,2}$ $t_{2,2}$ $t_{3,2}$ $t_{L,2}$
$t_{1,M}$ $t_{2,M}$ $t_{3,M}$ $t_{L,M}$
$t_{1,1}$ $t_{2,1}$ $t_{L,1}$
$t_{1,2}$ $t_{2,2}$ $t_{L,2}$
$t_{1,W}$ $t_{2,W}$ $t_{L,W}$

FIGURE 7.8: Augmented sentence level mean embedding representation. M represents the number of classes, W represents the size of the embedding, and L represents the number of sentences.

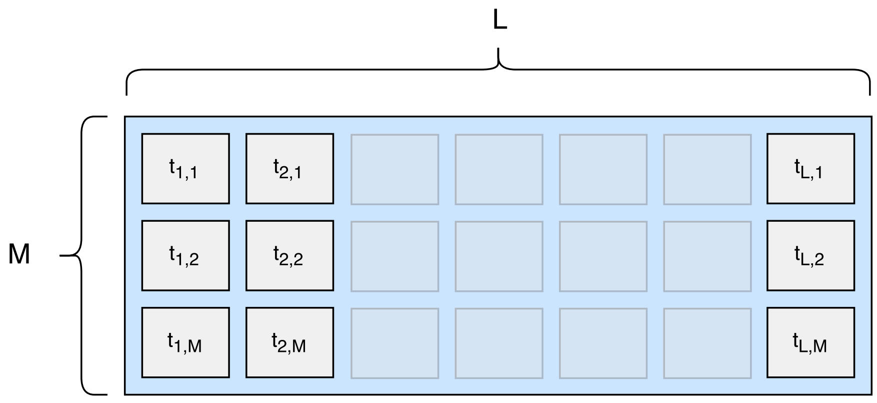

L
M
$t_{1,1}$ $t_{2,1}$ $t_{L,1}$
$t_{1,2}$ $t_{2,2}$ $t_{L,2}$
$t_{1,M}$ $t_{2,M}$ $t_{L,M}$

FIGURE 7.9: Lexicon representation of word level features. M represents the number of classes while L represents the number of words.

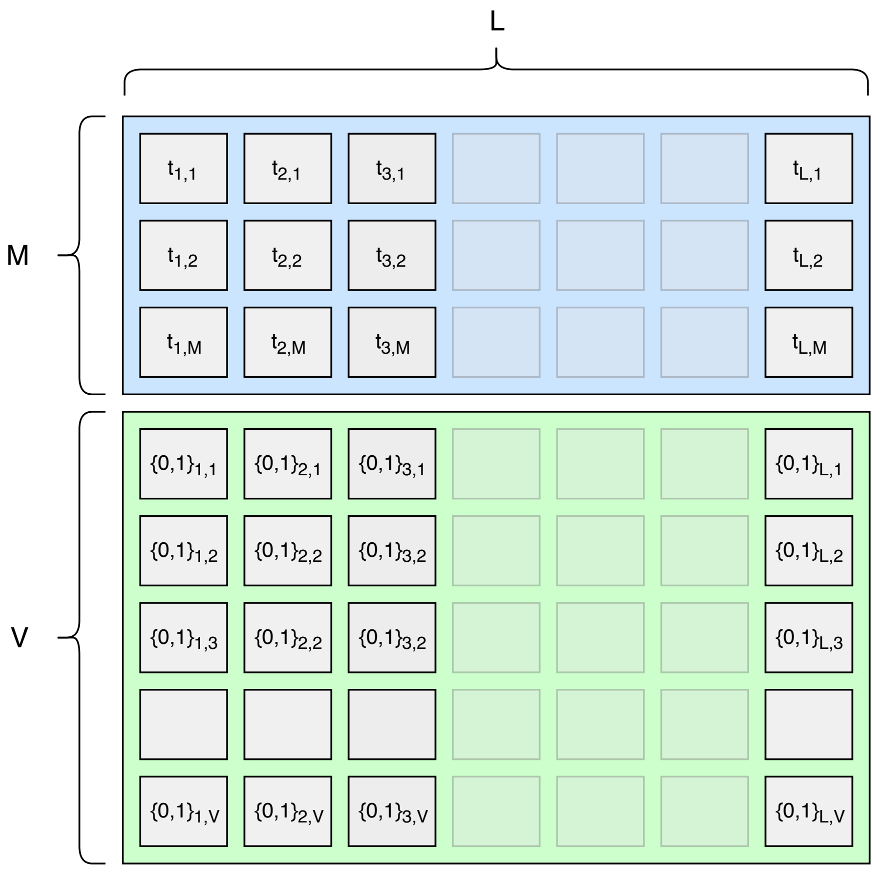


FIGURE 7.10: One hot encoding augmented with lexicon representation of text. M represents the number of classes, V represents the size of the vocabulary, and L represents the length of the instances in words.

7.5 Evaluating deep supervised argument mining

This section focuses on the evaluation of the representation schemes developed during the course of this thesis. As such, this evaluation is more complex than the previous ones mentioned in this thesis, and is divided into three sections, which are evaluating three levels of granularity with respect to the representation of documents for stance classification with a deep neural network: instance-level, sentence-level and word-level representation.

7.5.1 Evaluation tasks

We evaluate our approaches using two different classification tasks: standard classification (DSC) and transfer learning for classification (DTL). The first task consists in the classification of typical stance classes, typically {agreement, disagreement} and {agreement, disagreement, neutral}. The second task aims to classify stance on human-labelled data such as IAC, ABCD and SETC using

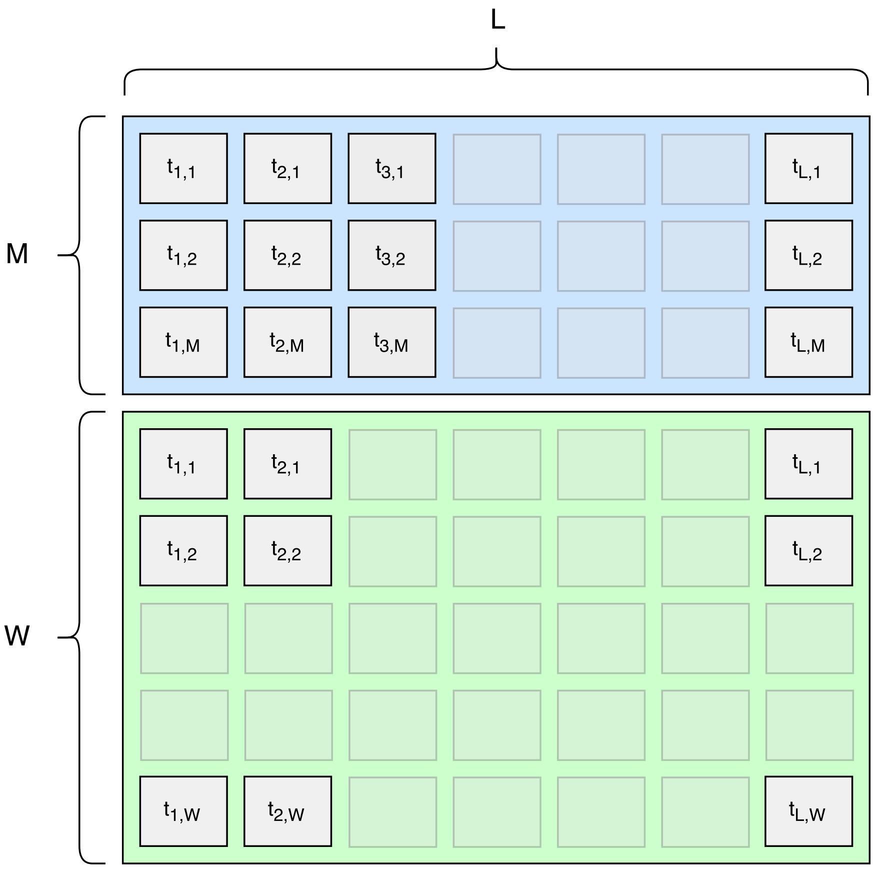


FIGURE 7.11: Embedding augmented with lexicon representation of text. W represents the size of the embedding, M represents the number of classes, and L represents the length of the instance in words.

class knowledge learned from weakly labelled data such as RNLC. Because the instance-level representations are flat and not temporally ordered, we use a traditional classifier to test them. Conversely, because the sentence and word-level representations are temporally ordered, we use a recurrent neural network to test them. The details of the learning algorithms are given below.

7.5.1.1 Learning algorithms

In the context of this evaluation we use two types of neural networks.

- Multilayer Perceptron (MLP), introduced in section 3.4.4, is suitable for representations with a fixed size and no temporal dimension. It takes as input a flat representation such as one generated from document statistics. It is used for instance-level representations, which do not take into account the structure of the instances, as well as their corresponding baselines.

- Simple Recurrent Neural Network (RNN), introduced in section 3.4.4.1, is well-suited for representations with a variable size and a temporal dimension. It parses through a temporally-structured representation by analysing each time-step one at a time, before finally returning a prediction. It is used for the sentence-level and word-level representations, as well as their corresponding baselines.

7.5.2 Results with instance-level feature models

The instance-level feature model uses summary statistics and flat semantic representations to represent instances. The flatness of the representation means that the learning algorithm used for evaluation cannot be focused on sequential data, and as such we use a Multilayer Perceptron (MLP). INSTLEX$_{\text{BOW+LN}}$ and INSTLEX$_{\text{BOW+RLN}}$ refer to representations that contain the bag of words of a given instance while INSTLEX$_{\text{EMB+LN}}$ and INSTLEX$_{\text{EMB+RLN}}$ refer to representations that contain the mean embedding of an instance. Both of those representations are enhanced by the addition of a vector of the following summary statistics of the lexicon values of the terms present in that instance:

- Minimum value ;
- Maximum value ;
- Mean value ;
- Median value ;
- Standard deviation.

These four representations are compared to the following baselines of varying strength:

BOLEX$_{\text{LN}}$ is a bag-of-lexicon values, i. e., a vector of N times the size of the lexicon (300) where each component of the vector is the frequency of the corresponding term in the instance multiplied by the lexicon value for a specific class. The 300 dimensional vector for each class are then aggregated into a 300 $\times$ N vector for a N class classification task. BOLEX$_{\text{LN}}$ uses LEXICNET as a way to learn the lexicon ;

BOLEX$_{\text{RLN}}$ is similar to BOLEX$_{\text{LN}}$ but uses RELEXNET as a lexicon learning method ;

BOW is a simple bag-of-words representation with tf-idf weighting ;

MEANEMB is a mean embedding representation, where a 300-dimensional word embedding is averaged out over all the terms in the instance (after removing stop words) and the resulting average of embeddings is used as a representation.

Results of task DSC-2 The DSC-2 task refers to the binary classification of stance. The results, shown in Table 7.1 and illustrated in Figure 7.12 comparing per dataset and Figure 7.13 averaged over all datasets show that the instance level representations using the mean embedding as a semantic representation tend to outperform other approaches. An ANOVA test in Table 7.2 shows that there is a significantly differing behaviour between the tested algorithms, and we show the results of our post hoc tests in Table 7.3. While in the ABCD dataset there is a clear demarcation between the best-performing representation (+0.22), it is much less obvious in the ABCD and SETC datasets (+0.02 and +0.01 respectively).

Additionally, a similar behaviour can be observed in the representations that make use of the bag-of-words base semantic representation with differences of +0.00, −0.20 and +0.11 between the ones enriched with RELEXNET and the ones enriched with LEXICNET summary statistics. This would tend to show that for some datasets the act of enriching the representation with class-sensitive information such as the lexicon summary statistics mentioned above is enough to improve classification accuracy, with the choice of enrichment strategy being less important. In all cases, the top performing combination of semantic and lexicon features outperforms the lexicon-only and semantic-only features by a large margin. An examination of the Tukey test results in Table 7.3 shows that this difference is significant in all datasets at the $p < 0.05$ threshold.

Results of task DSC-3 In this section we examine the result of a similar experiment but with three classes. We show the results of the DSC-3 task in Table 7.4, and illustrate them in two figures, Figure 7.14 for a dataset breakdown of the results, and Figure 7.15 for results averaged over all datasets. A visual examination of the results shows us that unlike on the DSC-2 task, here the semantic-only representations (BOW and MEANEMB) seem to outperform the enriched

	IAC	ABCD	SETC
INSTLEX$_{BOW+LN}$	0.703	0.729	0.698
INSTLEX$_{BOW+RLN}$	0.703	0.709	0.709
INSTLEX$_{EMB+LN}$	**0.725**	**0.745**	0.714
INSTLEX$_{EMB+RLN}$	0.701	0.743	0.**715**
BOLEX$_{LN}$	0.601	0.590	0.611
BOLEX$_{RLN}$	0.607	0.608	0.597
BOW	0.642	0.676	0.655
MEANEMB	0.667	0.675	0.669

TABLE 7.1: Accuracy results of Task DSC-2 with instance-level feature model

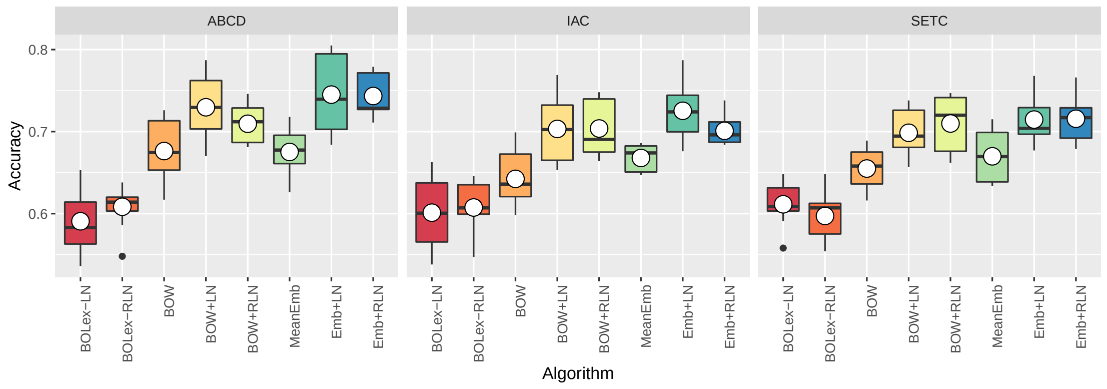


FIGURE 7.12: Box plot of accuracy results of Task DSC-2 with instance-level feature models per dataset

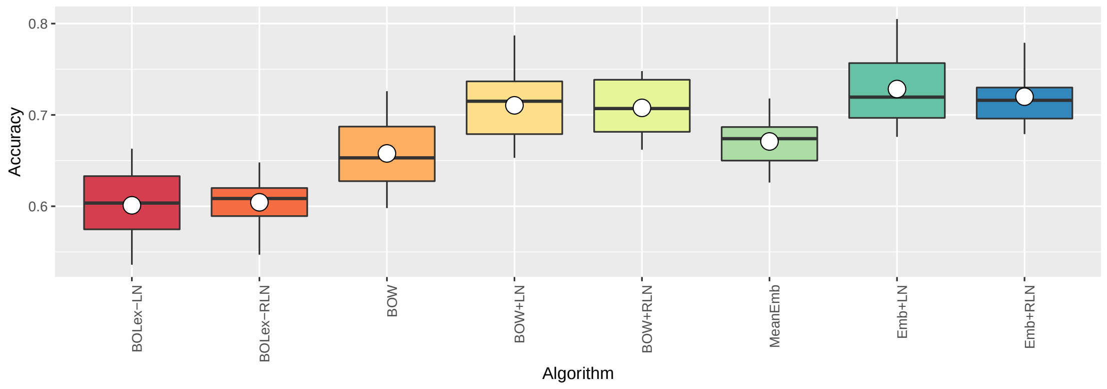


FIGURE 7.13: Box plot of accuracy results of Task DSC-2 with instance-level feature models averaged over all datasets

Dataset	F-Value	P-Value
IAC	19.487	**2.28553852209682e-14**†
ABCD	28.066	**3.28449336468974e-18**†
SETC	25.051	**5.7976380723505e-17**†

TABLE 7.2: Results of ANOVA on task DSC-2. † indicates statistical significance at 0.95 confidence.

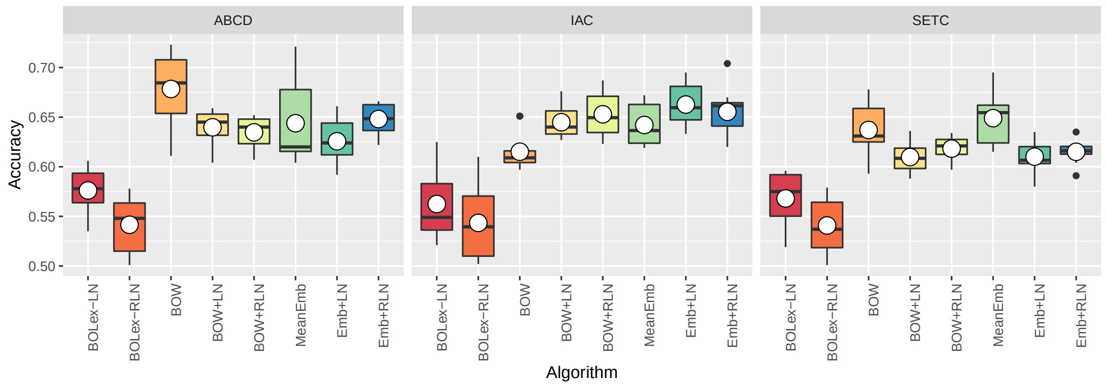


FIGURE 7.14: Box plot of accuracy results of Task DSC-3 with instance-level feature models per dataset

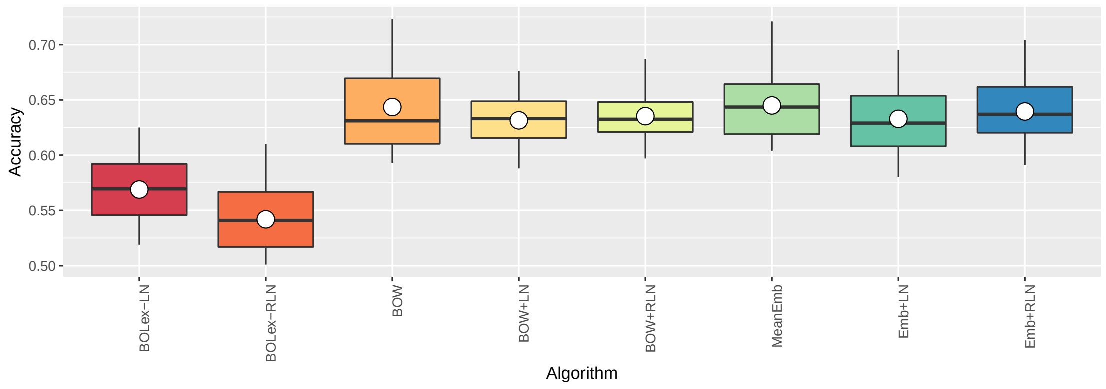


FIGURE 7.15: Box plot of accuracy results of Task DSC-3 with instance-level feature models averaged over all datasets

representations, albeit not by a very large margin. More specifically, we can observe that the MEANEMB representation has the lowest variance between datasets and seems to be more stable while some of the enriched representations go from best to bad depending on the dataset.

An ANOVA test in Table 7.5 lends evidence to there being a difference between algorithms, which we investigate in a post hoc test. The results of our Tukey test in Table 7.6 confirm those

	IAC		ABCD		SETC	
	Diff.	Adj. P-Value	Diff.	Adj. P-Value	Diff.	Adj. P-Value
BOLex$_{RLN}$/BOLex$_{LN}$	0.01	1.00	0.02	0.95	-0.01	0.96
BOW/BOLex$_{LN}$	0.04	0.13	0.09	**0.00**†	0.04	**0.03**†
InstLex$_{BOW+LN}$/BOLex$_{LN}$	0.10	**0.00**†	0.14	**0.00**†	0.09	**0.00**†
InstLex$_{BOW+RLN}$/BOLex$_{LN}$	0.10	**0.00**†	0.12	**0.00**†	0.10	**0.00**†
MeanEmb/BOLex$_{LN}$	0.07	**0.00**†	0.08	**0.00**†	0.06	**0.00**†
InstLex$_{EMB+LN}$/BOLex$_{LN}$	0.12	**0.00**†	0.15	**0.00**†	0.10	**0.00**†
InstLex$_{EMB+RLN}$/BOLex$_{LN}$	0.10	**0.00**†	0.15	**0.00**†	0.10	**0.00**†
BOW/BOLex$_{RLN}$	0.03	0.31	0.07	**0.00**†	0.06	**0.00**†
InstLex$_{BOW+LN}$/BOLex$_{RLN}$	0.10	**0.00**†	0.12	**0.00**†	0.10	**0.00**†
InstLex$_{BOW+RLN}$/BOLex$_{RLN}$	0.10	**0.00**†	0.10	**0.00**†	0.11	**0.00**†
MeanEmb/BOLex$_{RLN}$	0.06	**0.00**†	0.07	**0.00**†	0.07	**0.00**†
InstLex$_{EMB+LN}$/BOLex$_{RLN}$	0.12	**0.00**†	0.14	**0.00**†	0.12	**0.00**†
InstLex$_{EMB+RLN}$/BOLex$_{RLN}$	0.09	**0.00**†	0.14	**0.00**†	0.12	**0.00**†
InstLex$_{BOW+LN}$/BOW	0.06	**0.00**†	0.05	**0.02**†	0.04	**0.03**†
InstLex$_{BOW+RLN}$/BOW	0.06	**0.00**†	0.03	0.42	0.05	**0.00**†
MeanEmb/BOW	0.03	0.70	-0.00	1.00	0.01	0.95
InstLex$_{EMB+LN}$/BOW	0.08	**0.00**†	0.07	**0.00**†	0.06	**0.00**†
InstLex$_{EMB+RLN}$/BOW	0.06	**0.01**†	0.07	**0.00**†	0.06	**0.00**†
InstLex$_{BOW+RLN}$/InstLex$_{BOW+LN}$	0.00	1.00	-0.02	0.91	0.01	0.99
MeanEmb/InstLex$_{BOW+LN}$	-0.04	0.29	-0.05	**0.02**†	-0.03	0.39
InstLex$_{EMB+LN}$/InstLex$_{BOW+LN}$	0.02	0.83	0.02	0.98	0.02	0.93
InstLex$_{EMB+RLN}$/InstLex$_{BOW+LN}$	-0.00	1.00	0.01	0.99	0.02	0.90
MeanEmb/InstLex$_{BOW+RLN}$	-0.04	0.27	-0.03	0.38	-0.04	0.07
InstLex$_{EMB+LN}$/InstLex$_{BOW+RLN}$	0.02	0.84	0.04	0.34	0.00	1.00
InstLex$_{EMB+RLN}$/InstLex$_{BOW+RLN}$	-0.00	1.00	0.03	0.40	0.01	1.00
InstLex$_{EMB+LN}$/MeanEmb	0.06	**0.01**†	0.07	**0.00**†	0.04	**0.02**†
InstLex$_{EMB+RLN}$/MeanEmb	0.03	0.37	0.07	**0.00**†	0.05	**0.02**†
InstLex$_{EMB+RLN}$/InstLex$_{EMB+LN}$	-0.02	0.74	-0.00	1.00	0.00	1.00

TABLE 7.3: Results of Tukey test on task DSC-2. † indicates statistical significance at 0.95 confidence.

	IAC	ABCD	SETC
INSTLEX$_{\text{BOW+LN}}$	0.644	0.639	0.609
INSTLEX$_{\text{BOW+RLN}}$	0.652	0.634	0.618
INSTLEX$_{\text{EMB+LN}}$	**0.662**	0.625	0.610
INSTLEX$_{\text{EMB+RLN}}$	0.655	0.648	0.615
BOLEX$_{\text{LN}}$	0.562	0.576	0.568
BOLEX$_{\text{RLN}}$	0.543	0.541	0.540
BOW	0.615	**0.678**	0.637
MEANEMB	0.642	0.643	**0.648**

TABLE 7.4: Accuracy results of Task DSC-3 with instance-level feature model

Dataset	F-Value	P-Value
IAC	29.829	**6.7687303583772e-19**†
ABCD	25.27	**4.6728008570218e-17**†
SETC	26.628	**1.25565864578366e-17**†

TABLE 7.5: Results of ANOVA on task DSC-3. † indicates statistical significance at 0.95 confidence.

observations, with the only significant differences being the lexicon-only representations being outperformed by all other approaches.

Results of task DTL-2 The DTL-2 task consists in the binary classification of stance using a model learned from a different dataset. In this case, both the LEXICNET/RELEXNET models used to provide the lexicon part of the representation and the final model were learned on the RNLC dataset. We display the results in Table 7.7 and illustrate them per dataset in Figure 7.16 and averaged over all datasets in Figure 7.17. Visual inspection of the results shows a clear dominance of the enriched approaches over the semantics-only and lexicon-only baselines.

The results of the ANOVA are shown in Table 7.8 and of the resulting Tukey HSD test in Table 7.9. The results of the Tukey test confirm that the enriched approaches significantly outperform both the semantic-only and the lexicon-only approaches at the $p < 0.01$ level.

	IAC		ABCD		SETC	
	Diff.	Adj. P-Value	Diff.	Adj. P-Value	Diff.	Adj. P-Value
BOLex$_{\text{RLN}}$/BOLex$_{\text{LN}}$	-0.02	0.74	-0.03	0.11	-0.03	0.11
BOW/BOLex$_{\text{LN}}$	0.05	**0.00**†	0.10	**0.00** †	0.07	**0.00**†
InstLex$_{\text{BOW+LN}}$/BOLex$_{\text{LN}}$	0.08	**0.00**†	0.06	**0.00** †	0.04	**0.00**†
InstLex$_{\text{BOW+RLN}}$/BOLex$_{\text{LN}}$	0.09	**0.00**†	0.06	**0.00** †	0.05	**0.00**†
MeanEmb/BOLex$_{\text{LN}}$	0.08	**0.00**†	0.07	**0.00** †	0.08	**0.00**†
InstLex$_{\text{EMB+LN}}$/BOLex$_{\text{LN}}$	0.10	**0.00**†	0.05	**0.00** †	0.04	**0.00**†
InstLex$_{\text{EMB+RLN}}$/BOLex$_{\text{LN}}$	0.09	**0.00**†	0.07	**0.00** †	0.05	**0.00**†
BOW/BOLex$_{\text{RLN}}$	0.07	**0.00**†	0.14	**0.00** †	0.10	**0.00**†
InstLex$_{\text{BOW+LN}}$/BOLex$_{\text{RLN}}$	0.10	**0.00**†	0.10	**0.00** †	0.07	**0.00**†
InstLex$_{\text{BOW+RLN}}$/BOLex$_{\text{RLN}}$	0.11	**0.00**†	0.09	**0.00** †	0.08	**0.00**†
MeanEmb/BOLex$_{\text{RLN}}$	0.10	**0.00**†	0.10	**0.00**†	0.11	**0.00**†
InstLex$_{\text{EMB+LN}}$/BOLex$_{\text{RLN}}$	0.12	**0.00**†	0.08	**0.00** †	0.07	**0.00**†
InstLex$_{\text{EMB+RLN}}$/BOLex$_{\text{RLN}}$	0.11	**0.00**†	0.11	**0.00**†	0.07	**0.00**†
InstLex$_{\text{BOW+LN}}$/BOW	0.03	0.20	-0.04	0.05	-0.03	0.11
InstLex$_{\text{BOW+RLN}}$/BOW	0.04	**0.04**†	-0.04	**0.01**†	-0.02	0.53
MeanEmb/BOW	0.03	0.31	-0.03	0.11	0.01	0.92
InstLex$_{\text{EMB+LN}}$/BOW	0.05	**0.00**†	-0.05	**0.00**†	-0.03	0.12
InstLex$_{\text{EMB+RLN}}$/BOW	0.04	**0.02**†	-0.03	0.23	-0.02	0.32
InstLex$_{\text{BOW+RLN}}$/InstLex$_{\text{BOW+LN}}$	0.01	1.00	-0.01	1.00	0.01	0.99
MeanEmb/InstLex$_{\text{BOW+LN}}$	-0.00	1.00	0.00	1.00	0.04	**0.00**†
InstLex$_{\text{EMB+LN}}$/InstLex$_{\text{BOW+LN}}$	0.02	0.80	-0.01	0.94	0.00	1.00
InstLex$_{\text{EMB+RLN}}$/InstLex$_{\text{BOW+LN}}$	0.01	0.99	0.01	1.00	0.01	1.00
MeanEmb/InstLex$_{\text{BOW+RLN}}$	-0.01	0.98	0.01	1.00	0.03	**0.04**†
InstLex$_{\text{EMB+LN}}$/InstLex$_{\text{BOW+RLN}}$	0.01	0.99	-0.01	1.00	-0.01	0.99
InstLex$_{\text{EMB+RLN}}$/InstLex$_{\text{BOW+RLN}}$	0.00	1.00	0.01	0.96	-0.00	1.00
InstLex$_{\text{EMB+LN}}$/MeanEmb	0.02	0.65	-0.02	0.82	-0.04	**0.00**†
InstLex$_{\text{EMB+RLN}}$/MeanEmb	0.01	0.95	0.00	1.00	-0.03	**0.02**†
InstLex$_{\text{EMB+RLN}}$/InstLex$_{\text{EMB+LN}}$	-0.01	1.00	0.02	0.61	0.00	1.00

TABLE 7.6: Results of Tukey test on task DSC-3. † indicates statistical significance at 0.95 confidence.

	IAC	ABCD	SETC
INSTLEX$_{BOW+LN}$	**0.726**	0.690	0.689
INSTLEX$_{BOW+RLN}$	0.690	0.692	0.705
INSTLEX$_{EMB+LN}$	0.693	0.702	0.706
INSTLEX$_{EMB+RLN}$	0.701	**0.702**	**0.728**
BOLEX$_{LN}$	0.552	0.597	0.612
BOLEX$_{RLN}$	0.599	0.591	0.595
BOW	0.617	0.614	0.622
MEANEMB	0.609	0.624	0.614

TABLE 7.7: Accuracy results of Task DTL-2 with instance-level feature models

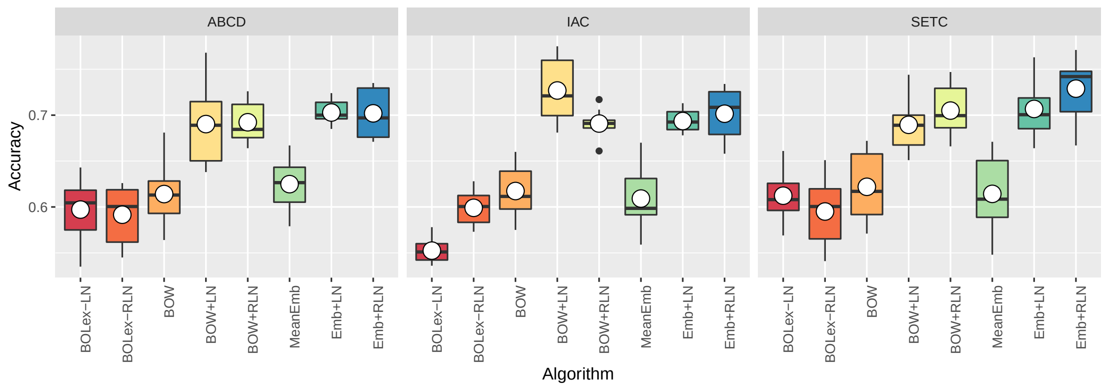


FIGURE 7.16: Box plot of accuracy results of Task DTL-2 with instance-level feature models

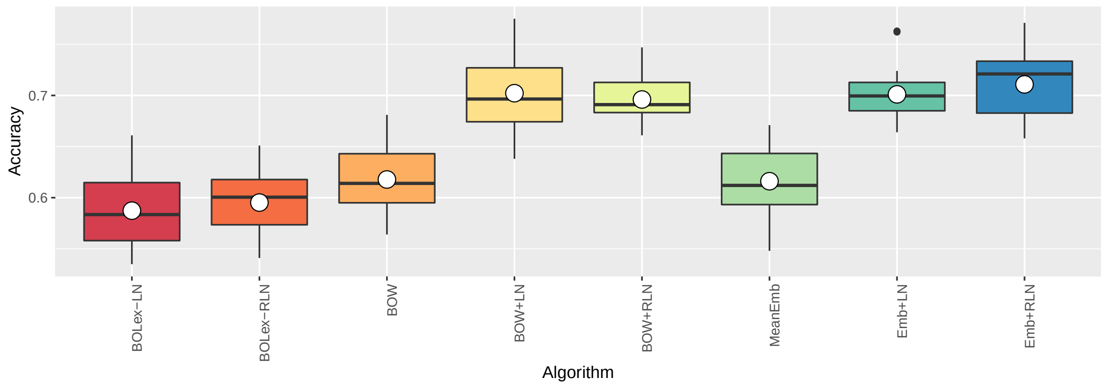


FIGURE 7.17: Box plot of accuracy results of Task DTL-2 with instance-level feature models

Dataset	F-Value	P-Value
IAC	66.317	**7.51944104659384e-29**†
ABCD	25.079	**5.64325437695852e-17**†
SETC	25.374	**4.21737073354391e-17**†

TABLE 7.8: Results of ANOVA on task DTL-2. † indicates statistical significance at 0.95 confidence.

7.5.3 Results with sentence-level feature model

In this section we review and comment on the results of our experiment with the sentence-level feature models developed throughout this chapter. Unlike the instance-level models, the sentence-level models retain some form of temporality in the sense that each sentence is encoded separately before being aggregated together. Because of this particularity, instances can be of variable length and call for a learning algorithm that can accommodate such differences, in our case a recurrent neural network. Four representations are evaluated, where two of them use a sentence bag of words as the semantic part of their representation (SENTLEX$_{\text{BOW+LN}}$ and SENTLEX$_{\text{BOW+RLN}}$) and the two others use an average of the word embeddings present in the sentence (SENTLEX$_{\text{EMB+LN}}$ and SENTLEX$_{\text{EMB+RLN}}$). Those are then augmented by average lexicon values for each class and averaged over all the terms present in the sentence, and the sentences are then aggregated into the final vector representation.

The following baselines were used:

VSBOLEX$_{\text{LN}}$, a vector of LEXICNET values averaged over all words of each sentence (VSLN in the plots) ;

VSBOLEX$_{\text{RLN}}$, a vector of RELEXNET values averaged over all words of each sentence (VRSLN in the plots) ;

VSBOW , a vector of sentence bags of words, is a simple vector of bag of words for each sentence (BOW in the plots);

VSEMB , a vector of sentence embeddings, is a simple vector of word embeddings averaged over all words of each sentence (MeanEmb in the plots).

	IAC		ABCD		SETC	
	Diff.	Adj. P-Value	Diff.	Adj. P-Value	Diff.	Adj. P-Value
BOLex$_{\text{RLN}}$/BOLex$_{\text{LN}}$	0.05	**0.00**†	-0.01	1.00	-0.02	0.94
BOW/BOLex$_{\text{LN}}$	0.06	**0.00**†	0.02	0.93	0.01	1.00
InstLex$_{\text{BOW+LN}}$/BOLex$_{\text{LN}}$	0.17	**0.00**†	0.09	**0.00**†	0.08	**0.00**†
InstLex$_{\text{BOW+RLN}}$/BOLex$_{\text{LN}}$	0.14	**0.00**†	0.10	**0.00**†	0.09	**0.00**†
MeanEmb/BOLex$_{\text{LN}}$	0.06	**0.00**†	0.03	0.50	0.00	1.00
InstLex$_{\text{EMB+LN}}$/BOLex$_{\text{LN}}$	0.14	**0.00**†	0.11	**0.00**†	0.09	**0.00**†
InstLex$_{\text{EMB+RLN}}$/BOLex$_{\text{LN}}$	0.15	**0.00**†	0.10	**0.00**†	0.12	**0.00**†
BOW/BOLex$_{\text{RLN}}$	0.02	0.69	0.02	0.74	0.03	0.62
InstLex$_{\text{BOW+LN}}$/BOLex$_{\text{RLN}}$	0.13	**0.00**†	0.10	**0.00**†	0.09	**0.00**†
InstLex$_{\text{BOW+RLN}}$/BOLex$_{\text{RLN}}$	0.09	**0.00**†	0.10	**0.00**†	0.11	**0.00**†
MeanEmb/BOLex$_{\text{RLN}}$	0.01	0.98	0.03	0.26	0.02	0.90
InstLex$_{\text{EMB+LN}}$/BOLex$_{\text{RLN}}$	0.09	**0.00**†	0.11	**0.00**†	0.11	**0.00**†
InstLex$_{\text{EMB+RLN}}$/BOLex$_{\text{RLN}}$	0.10	**0.00**†	0.11	**0.00**†	0.13	**0.00**†
InstLex$_{\text{BOW+LN}}$/BOW	0.11	**0.00**†	0.08	**0.00**†	0.07	**0.00**†
InstLex$_{\text{BOW+RLN}}$/BOW	0.07	**0.00**†	0.08	**0.00**†	0.08	**0.00**†
MeanEmb/BOW	-0.01	0.99	0.01	0.99	-0.01	1.00
InstLex$_{\text{EMB+LN}}$/BOW	0.08	**0.00**†	0.09	**0.00**†	0.08	**0.00**†
InstLex$_{\text{EMB+RLN}}$/BOW	0.08	**0.00**†	0.09	**0.00**†	0.11	**0.00**†
InstLex$_{\text{BOW+RLN}}$/InstLex$_{\text{BOW+LN}}$	-0.04	**0.03**†	0.00	1.00	0.02	0.97
MeanEmb/InstLex$_{\text{BOW+LN}}$	-0.12	**0.00**†	-0.07	**0.00**†	-0.08	**0.00**†
InstLex$_{\text{EMB+LN}}$/InstLex$_{\text{BOW+LN}}$	-0.03	0.05	0.01	0.98	0.02	0.94
InstLex$_{\text{EMB+RLN}}$/InstLex$_{\text{BOW+LN}}$	-0.03	0.28	0.01	0.99	0.04	0.16
MeanEmb/InstLex$_{\text{BOW+RLN}}$	-0.08	**0.00**†	-0.07	**0.00**†	-0.09	**0.00**†
InstLex$_{\text{EMB+LN}}$/InstLex$_{\text{BOW+RLN}}$	0.00	1.00	0.01	0.99	0.00	1.00
InstLex$_{\text{EMB+RLN}}$/InstLex$_{\text{BOW+RLN}}$	0.01	0.97	0.01	1.00	0.02	0.75
InstLex$_{\text{EMB+LN}}$/MeanEmb	0.08	**0.00**†	0.08	**0.00**†	0.09	**0.00**†
InstLex$_{\text{EMB+RLN}}$/MeanEmb	0.09	**0.00**†	0.08	**0.00**†	0.11	**0.00**†
InstLex$_{\text{EMB+RLN}}$/InstLex$_{\text{EMB+LN}}$	0.01	1.00	-0.00	1.00	0.02	0.82

TABLE 7.9: Results of Tukey test on DTL-2. † indicates statistical significance at 0.95 confidence.

	IAC	ABCD	SETC
SENTLEX$_{BOW+LN}$	**0.776**	0.769	0.741
SENTLEX$_{BOW+RLN}$	0.757	0.762	0.748
SENTLEX$_{EMB+LN}$	0.772	**0.788**	**0.774**
SENTLEX$_{EMB+RLN}$	0.762	0.775	0.762
VSBOLEX$_{LN}$	0.639	0.622	0.617
VSBOLEX$_{RLN}$	0.587	0.603	0.601
VSBOW	0.575	0.552	0.583
VSEMB	0.743	0.740	0.722

TABLE 7.10: Accuracy results of Task DSC-2 with sentence-level feature model

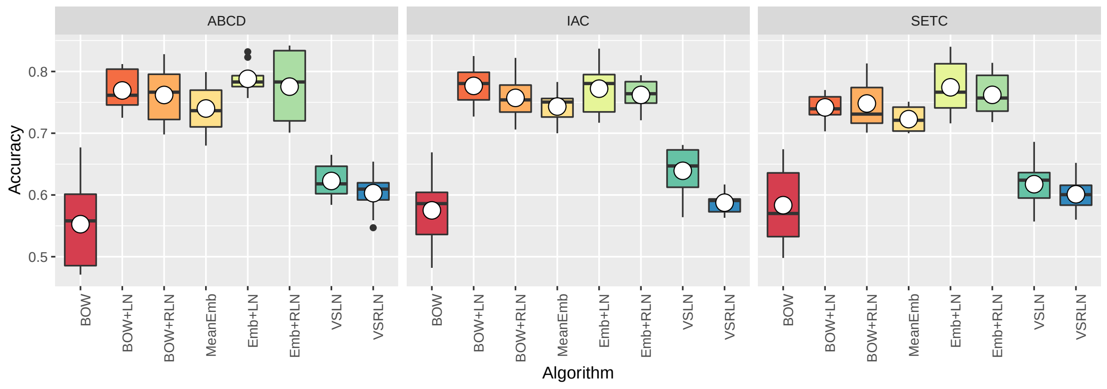


FIGURE 7.18: Box plot of accuracy results of Task DSC-2 with sentence-level feature model per dataset

Results of task DSC-2 This section details the results of the DSC-2 task, which focuses on binary classification of stance. We show the results in Table 7.10 and see that the embedding-based and LEXICNET-enriched representation seems to dominate, with all enriched representations having a large difference over the semantic-only and lexicon-only approaches except for the sentence embedding representation (VSEMB). We illustrate those results in Figure 7.18 to compare per dataset and Figure 7.19 by averaging over all datasets. We can see from Figure 7.19 that all representations perform relatively similarly, and that the MEANEMB baseline performs very close to the proposed approaches.

An examination of our ANOVA table in Table 7.11 and following that of the results of our paired comparisons using the Tukey HSD test in Table 7.12 confirms our observations, showing that

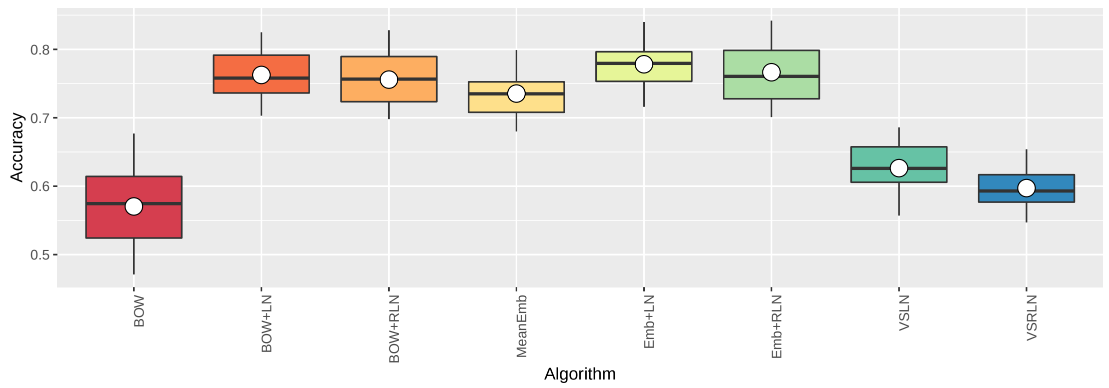


FIGURE 7.19: Box plot of accuracy results of Task DSC-2 with sentence-level feature model over all datasets

Dataset	F-Value	P-Value
IAC	55.105	**2.09981432691958e-26**†
ABCD	43.623	**1.98767852848713e-23**†
SETC	40.715	**1.41664252315884e-22**†

TABLE 7.11: Results of ANOVA on task DSC-2. † indicates statistical significance at 0.95 confidence.

while all approaches outperformed the bag of words and bag of lexicon values baselines, the difference between the embedding baseline MEANEMB and the proposed approaches did not reach statistical significance.

Results of task DSC-3 The results of the DSC-3 evaluation task in Table 7.13 show that in the IAC and ABCD datasets, while the proposed approaches outperform the non-enriched representations, the embedding-based representation VSEMB comes very close to the top-performing accuracy (within 0.05 in IAC and 0.09 in ABCD). The enriched representations however perform relatively similar to each other, with a very low level of variation in IAC (all results ranging from 0.668 to 0.683) and a slightly larger range for the ABCD dataset (ranging from 0.645 to 0.697). On the SETC dataset, the VSEMB baseline outperforms all other approaches by a very large margin (0.032). We illustrate those results in Figures 7.20 and 7.21, firstly per dataset and finally averaging over all datasets. Figure 7.21 shows that overall, the VSEMB baseline seems to perform at least as good if not slightly better than the proposed representations.

	IAC		ABCD		SETC	
	Diff.	Adj. P-Value	Diff.	Adj. P-Value	Diff.	Adj. P-Value
SentLex$_{\text{BOW+LN}}$/VSBOW	0.20	**0.00**†	0.22	**0.00**†	0.16	**0.00**†
SentLex$_{\text{BOW+RLN}}$/VSBOW	0.18	**0.00**†	0.21	**0.00**†	0.16	**0.00**†
VSEmb/VSBOW	0.17	**0.00**†	0.19	**0.00**†	0.14	**0.00**†
SentLex$_{\text{EMB+LN}}$/VSBOW	0.20	**0.00**†	0.24	**0.00**†	0.19	**0.00**†
SentLex$_{\text{EMB+RLN}}$/VSBOW	0.19	**0.00**†	0.22	**0.00**†	0.18	**0.00**†
VSBOLex$_{\text{LN}}$/VSBOW	0.06	**0.00**†	0.07	**0.02**†	0.03	0.53
VSBOLex$_{\text{RLN}}$/VSBOW	0.01	0.99	0.05	0.20	0.02	0.97
SentLex$_{\text{BOW+RLN}}$/SentLex$_{\text{BOW+LN}}$	-0.02	0.93	-0.01	1.00	0.01	1.00
VSEmb/SentLex$_{\text{BOW+LN}}$	-0.03	0.47	-0.03	0.82	-0.02	0.96
SentLex$_{\text{EMB+LN}}$/SentLex$_{\text{BOW+LN}}$	-0.00	1.00	0.02	0.98	0.03	0.59
SentLex$_{\text{EMB+RLN}}$/SentLex$_{\text{BOW+LN}}$	-0.01	0.99	0.01	1.00	0.02	0.94
VSBOLex$_{\text{LN}}$/SentLex$_{\text{BOW+LN}}$	-0.14	**0.00**†	-0.15	**0.00**†	-0.12	**0.00**†
VSBOLex$_{\text{RLN}}$/SentLex$_{\text{BOW+LN}}$	-0.19	**0.00**†	-0.17	**0.00**†	-0.14	**0.00**†
VSEmb/SentLex$_{\text{BOW+RLN}}$	-0.01	0.99	-0.02	0.96	-0.03	0.83
SentLex$_{\text{EMB+LN}}$/SentLex$_{\text{BOW+RLN}}$	0.01	0.98	0.03	0.89	0.03	0.81
SentLex$_{\text{EMB+RLN}}$/SentLex$_{\text{BOW+RLN}}$	0.00	1.00	0.01	1.00	0.01	0.99
VSBOLex$_{\text{LN}}$/SentLex$_{\text{BOW+RLN}}$	-0.12	**0.00**†	-0.14	**0.00**†	-0.13	**0.00**†
VSBOLex$_{\text{RLN}}$/SentLex$_{\text{BOW+RLN}}$	-0.17	**0.00**†	-0.16	**0.00**†	-0.15	**0.00**†
SentLex$_{\text{EMB+LN}}$/VSEmb	0.03	0.65	0.05	0.25	0.05	0.08
SentLex$_{\text{EMB+RLN}}$/VSEmb	0.02	0.94	0.04	0.64	0.04	0.34
VSBOLex$_{\text{LN}}$/VSEmb	-0.10	**0.00**†	-0.12	**0.00**†	-0.11	**0.00**†
VSBOLex$_{\text{RLN}}$/VSEmb	-0.16	**0.00**†	-0.14	**0.00**†	-0.12	**0.00**†
SentLex$_{\text{EMB+RLN}}$/SentLex$_{\text{EMB+LN}}$	-0.01	1.00	-0.01	1.00	-0.01	1.00
VSBOLex$_{\text{LN}}$/SentLex$_{\text{EMB+LN}}$	-0.13	**0.00**†	-0.17	0.00†	-0.16	**0.00**†
VSBOLex$_{\text{RLN}}$/SentLex$_{\text{EMB+LN}}$	-0.18	**0.00**†	-0.19	**0.00**†	-0.17	**0.00**†
VSBOLex$_{\text{LN}}$/SentLex$_{\text{EMB+RLN}}$	-0.12	**0.00**†	-0.15	**0.00**†	-0.14	**0.00**†
VSBOLex$_{\text{RLN}}$/SentLex$_{\text{EMB+RLN}}$	-0.17	**0.00**†	-0.17	**0.00**†	-0.16	**0.00**†
VSBOLex$_{\text{RLN}}$/VSBOLex$_{\text{LN}}$	-0.05	0.05	-0.02	0.97	-0.02	0.98

TABLE 7.12: Results of Tukey test on task DSC-2 with sentence-level representations. † indicates statistical significance at 0.95 confidence.

	IAC	ABCD	SETC
SENTLEX$_{\text{BOW+LN}}$	**0.683**	0.693	0.643
SENTLEX$_{\text{BOW+RLN}}$	0.668	0.645	0.634
SENTLEX$_{\text{EMB+LN}}$	0.668	0.668	0.628
SENTLEX$_{\text{EMB+RLN}}$	0.677	**0.697**	0.631
VSBOLEX$_{\text{LN}}$	0.568	0.574	0.566
VSBOLEX$_{\text{RLN}}$	0.552	0.539	0.552
VSBOW	0.569	0.540	0.531
VSEMB	0.677	0.688	**0.675**

TABLE 7.13: Accuracy results of Task DSC-3 with sentence-level feature model

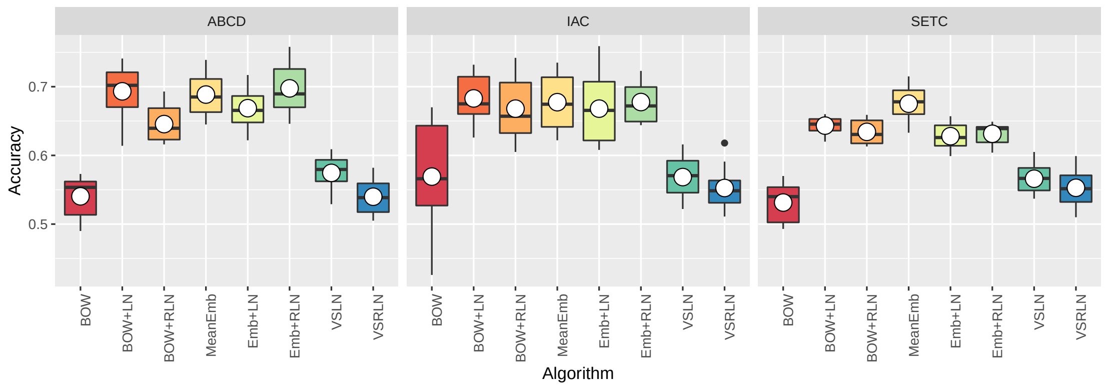


FIGURE 7.20: Box plot of accuracy results of Task DSC-3 with sentence-level feature model per dataset

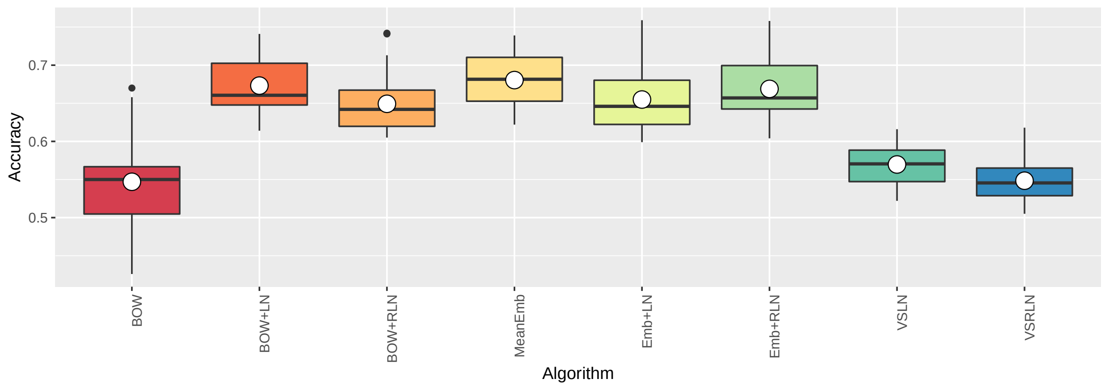


FIGURE 7.21: Box plot of accuracy results of Task DSC-3 with sentence-level feature model over all datasets

Dataset	F-Value	P-Value
IAC	15.132	**5.40848272169679e-12**†
ABCD	46.38	**3.39252319607527e-24**†
SETC	49.286	**5.75305314347075e-25**†

TABLE 7.14: Results of ANOVA on task DSC-3. † indicates statistical significance at 0.95 confidence.

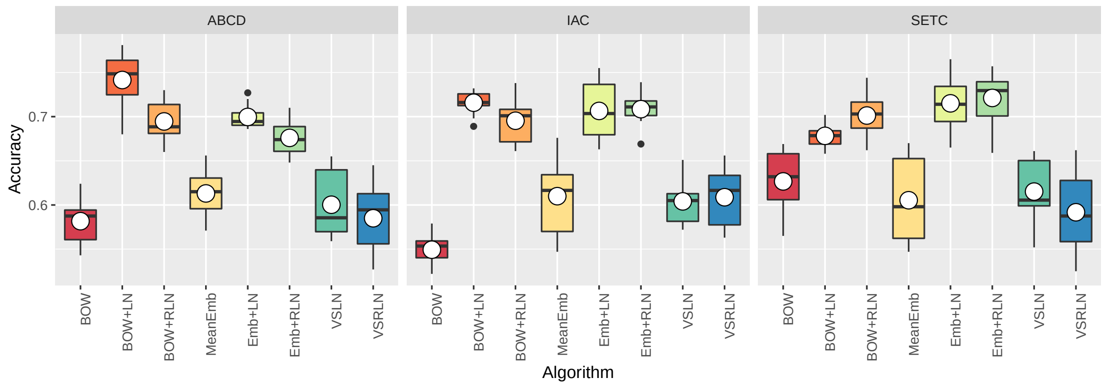


FIGURE 7.22: Box plot of accuracy results of Task DTL-2 with sentence-level feature model per dataset

Further investigation using our ANOVA table in Table 7.14 shows that there exist some significant differences within representations over all datasets, which warrants the Tukey post hoc test shown in Table 7.15. Those result agree with the rest of our analysis, showing that while there is some difference observed with visual inspection of the results, the differences over all do not reach statistical significance except for the poor performance of the lexicon-only baselines.

Results of task DTL-2 The results in Table 7.16 show that SENTLEX$_{\text{BOW+LN}}$ outperforms all other representations except for the SETC dataset, where SENTLEX$_{\text{EMB+RLN}}$ performs best. Unlike previous tasks, the embedding baseline here (VSEMB) does not perform well, and is typically 0.1 to 0.15 lower than the best performing representation. We illustrate those results in two figures. In Figure 7.22 we can observe the breakdown of scores per dataset, and we can see that the baselines tend to perform consistently bad, while the proposed representations tend to perform consistently well and with less overall variance in the results. In Figure 7.23, we can see those same results averaged over all datasets, and which emphasise the same conclusion.

	IAC		ABCD		SETC	
	Diff.	Adj. P-Value	Diff.	Adj. P-Value	Diff.	Adj. P-Value
SentLex$_{BOW+LN}$/VSBOW	0.11	**0.00**†	0.15	**0.00**†	0.11	**0.00**†
SentLex$_{BOW+RLN}$/VSBOW	0.10	**0.00**†	0.11	**0.00**†	0.10	**0.00**†
VSEmb/VSBOW	0.11	**0.00**†	0.15	**0.00**†	0.14	**0.00**†
SentLex$_{EMB+LN}$/VSBOW	0.10	**0.00**†	0.13	**0.00**†	0.10	**0.00**†
SentLex$_{EMB+RLN}$/VSBOW	0.11	**0.00**†	0.16	**0.00**†	0.10	**0.00**†
VSBOLex$_{LN}$/VSBOW	-0.00	1.00	0.03	0.26	0.03	**0.02**†
VSBOLex$_{RLN}$/VSBOW	-0.02	0.99	-0.00	1.00	0.02	0.43
SentLex$_{BOW+RLN}$/SentLex$_{BOW+LN}$	-0.01	1.00	-0.05	**0.03**†	-0.01	0.99
VSEmb/SentLex$_{BOW+LN}$	-0.01	1.00	-0.00	1.00	0.03	0.05
SentLex$_{EMB+LN}$/SentLex$_{BOW+LN}$	-0.01	1.00	-0.02	0.67	-0.02	0.80
SentLex$_{EMB+RLN}$/SentLex$_{BOW+LN}$	-0.01	1.00	0.00	1.00	-0.01	0.94
VSBOLex$_{LN}$/SentLex$_{BOW+LN}$	-0.11	**0.00**†	-0.12	**0.00**†	-0.08	**0.00**†
VSBOLex$_{RLN}$/SentLex$_{BOW+LN}$	-0.13	**0.00**†	-0.15	**0.00**†	-0.09	**0.00**†
VSEmb/SentLex$_{BOW+RLN}$	0.01	1.00	0.04	0.07	0.04	**0.00**†
SentLex$_{EMB+LN}$/SentLex$_{BOW+RLN}$	0.00	1.00	0.02	0.73	-0.01	1.00
SentLex$_{EMB+RLN}$/SentLex$_{BOW+RLN}$	0.01	1.00	0.05	**0.01**†	-0.00	1.00
VSBOLex$_{LN}$/SentLex$_{BOW+RLN}$	-0.10	**0.00**†	-0.07	**0.00**†	-0.07	**0.00**†
VSBOLex$_{RLN}$/SentLex$_{BOW+RLN}$	-0.12	**0.00**†	-0.11	**0.00**†	-0.08	**0.00**†
SentLex$_{EMB+LN}$/VSEmb	-0.01	1.00	-0.02	0.86	-0.05	**0.00**†
SentLex$_{EMB+RLN}$/VSEmb	0.00	1.00	0.01	1.00	-0.04	**0.00**†
VSBOLex$_{LN}$/VSEmb	-0.11	**0.00**†	-0.11	**0.00**†	-0.11	**0.00**†
VSBOLex$_{RLN}$/VSEmb	-0.12	**0.00**†	-0.15	**0.00**†	-0.12	**0.00**†
SentLex$_{EMB+RLN}$/SentLex$_{EMB+LN}$	0.01	1.00	0.03	0.48	0.00	1.00
VSBOLex$_{LN}$/SentLex$_{EMB+LN}$	-0.10	**0.00**†	-0.09	**0.00**†	-0.06	**0.00**†
VSBOLex$_{RLN}$/SentLex$_{EMB+LN}$	-0.12	**0.00**†	-0.13	**0.00**†	-0.08	**0.00**†
VSBOLex$_{LN}$/SentLex$_{EMB+RLN}$	-0.11	**0.00**†	-0.12	**0.00**†	-0.07	**0.00**†
VSBOLex$_{RLN}$/SentLex$_{EMB+RLN}$	-0.13	**0.00**†	-0.16	**0.00**†	-0.08	**0.00**†
VSBOLex$_{RLN}$/VSBOLex$_{LN}$	-0.02	0.99	-0.03	0.24	-0.01	0.90

TABLE 7.15: Results of Tukey test on task DSC-3. † indicates statistical significance at 0.95 confidence.

	IAC	ABCD	SETC
SENTLEX$_{BOW+LN}$	**0.715**	**0.741**	0.678
SENTLEX$_{BOW+RLN}$	0.695	0.694	0.701
SENTLEX$_{EMB+LN}$	0.706	0.699	0.715
SENTLEX$_{EMB+RLN}$	0.708	0.676	**0.721**
VSBOLEX$_{LN}$	0.604	0.600	0.615
VSBOLEX$_{RLN}$	0.608	0.585	0.591
VSBOW	0.549	0.581	0.626
VSEMB	0.610	0.613	0.605

TABLE 7.16: Accuracy results of Task DTL-2 with sentence-level feature model

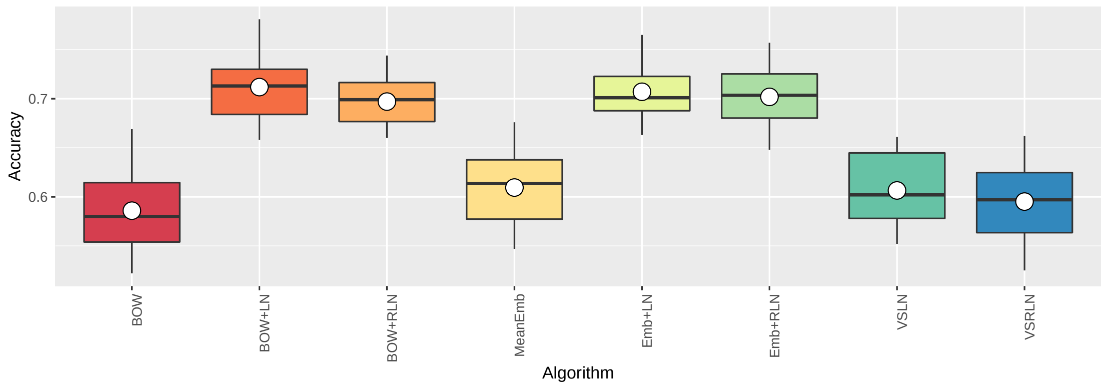


FIGURE 7.23: Box plot of accuracy results of Task DTL-2 with sentence-level feature model over all datasets

We perform an ANOVA (Table 7.17) to test our conclusion and detect a significant change between approaches, which leads us to the results of a Tukey HSD test displayed in Table 7.18). An inspection of those results shows that overall, the proposed representations perform significantly better than both lexicon-only and semantic-only baselines, and that there is no statistically significant difference among baseline approaches as well as among the proposed representations.

Dataset	F-Value	P-Value
IAC	49.611	**4.74298763062588e-25**†
ABCD	43.858	**1.70378503723316e-23**†
SETC	22.186	**1.11453141231639e-15**†

TABLE 7.17: Results of ANOVA on task DTL-2. † indicates statistical significance at 0.95 confidence.

7.5.4 Results with word-level feature model

In this section we review and comment on the results of our experiment with the word-level feature models developed throughout this chapter. Unlike the sentence-level models, the word-level models do not average over words of each sentence but retain all information both at the semantic and the lexicon level. Similarly to sentence-level models, the representations are thus of variable length and are processed by a recurrent neural network. We evaluate four representations, representing the combinations of either one hot encoding (detailed in Section 3.3.1.1) and word embedding for the semantic part of each word, and LEXICNET vector or RELEXNET vector for the lexicon part of each word.

The following baselines were used:

VLN , a vector of LEXICNET values for each term ;

VRLN , a vector of RELEXNET values for each term ;

ONEHOT , a vector of one-hot encoded terms ;

EMBEDDING , a vector of word embeddings.

Results of task DSC-2 We show the results of the DSC-2 task in Table 7.19, and illustrate them in Figures 7.24 (per dataset) and 7.25 (averaged over all datasets). A visual inspection of the plots and the related table shows that the LEXICNET-enriched and word embedding based representation WOLLEX$_{\text{EMB+LN}}$ outperforms all other approaches in the IAC and SETC datasets and comes close to the best-performing method in the ABCD dataset, where the EMBEDDING baseline outperforms it by a small margin (0.032).

	IAC		ABCD		SETC	
	Diff.	Adj. P-Value	Diff.	Adj. P-Value	Diff.	Adj. P-Value
SentLex$_{BOW+LN}$/VSBOW	0.17	**0.00**†	0.16	**0.00**†	0.05	**0.03**†
SentLex$_{BOW+RLN}$/VSBOW	0.15	**0.00**†	0.11	**0.00**†	0.07	**0.00**†
VSEmb/VSBOW	0.06	**0.00**†	0.03	0.26	-0.02	0.88
SentLex$_{EMB+LN}$/VSBOW	0.16	**0.00**†	0.12	**0.00**†	0.09	**0.00**†
SentLex$_{EMB+RLN}$/VSBOW	0.16	**0.00**†	0.09	**0.00**†	0.09	**0.00**†
VSBOLex$_{LN}$/VSBOW	0.05	**0.00**†	0.02	0.84	-0.01	1.00
VSBOLex$_{RLN}$/VSBOW	0.06	**0.00**†	0.00	1.00	-0.03	0.36
SentLex$_{BOW+RLN}$/SentLex$_{BOW+LN}$	-0.02	0.75	-0.05	**0.01**†	0.02	0.83
VSEmb/SentLex$_{BOW+LN}$	-0.11	**0.00**†	-0.13	**0.00**†	-0.07	**0.00**†
SentLex$_{EMB+LN}$/SentLex$_{BOW+LN}$	-0.01	1.00	-0.04	**0.04**†	0.04	0.30
SentLex$_{EMB+RLN}$/SentLex$_{BOW+LN}$	-0.01	1.00	-0.07	**0.00**†	0.04	0.14
VSBOLex$_{LN}$/SentLex$_{BOW+LN}$	-0.11	**0.00**†	-0.14	**0.00**†	-0.06	**0.00**†
VSBOLex$_{RLN}$/SentLex$_{BOW+LN}$	-0.11	**0.00**†	-0.16	**0.00**†	-0.09	**0.00**†
VSEmb/SentLex$_{BOW+RLN}$	-0.09	**0.00**†	-0.08	**0.00**†	-0.10	**0.00**†
SentLex$_{EMB+LN}$/SentLex$_{BOW+RLN}$	0.01	0.99	0.01	1.00	0.01	0.99
SentLex$_{EMB+RLN}$/SentLex$_{BOW+RLN}$	0.01	0.97	-0.02	0.84	0.02	0.91
VSBOLex$_{LN}$/SentLex$_{BOW+RLN}$	-0.09	**0.00**†	-0.09	**0.00**†	-0.09	**0.00**†
VSBOLex$_{RLN}$/SentLex$_{BOW+RLN}$	-0.09	**0.00**†	-0.11	**0.00**†	-0.11	**0.00**†
SentLex$_{EMB+LN}$/VSEmb	0.10	**0.00**†	0.09	**0.00**†	0.11	**0.00**†
SentLex$_{EMB+RLN}$/VSEmb	0.10	**0.00**†	0.06	**0.00**†	0.12	**0.00**†
VSBOLex$_{LN}$/VSEmb	-0.01	1.00	-0.01	0.98	0.01	1.00
VSBOLex$_{RLN}$/VSEmb	-0.00	1.00	-0.03	0.39	-0.01	0.99
SentLex$_{EMB+RLN}$/SentLex$_{EMB+LN}$	0.00	1.00	-0.02	0.61	0.01	1.00
VSBOLex$_{LN}$/SentLex$_{EMB+LN}$	-0.10	**0.00**†	-0.10	**0.00**†	-0.10	**0.00**†
VSBOLex$_{RLN}$/SentLex$_{EMB+LN}$	-0.10	**0.00**†	-0.11	**0.00**†	-0.12	**0.00**†
VSBOLex$_{LN}$/SentLex$_{EMB+RLN}$	-0.10	**0.00**†	-0.08	**0.00**†	-0.11	**0.00**†
VSBOLex$_{RLN}$/SentLex$_{EMB+RLN}$	-0.10	**0.00**†	-0.09	**0.00**†	-0.13	**0.00**†
VSBOLex$_{RLN}$/VSBOLex$_{LN}$	0.00	1.00	-0.02	0.94	-0.02	0.82

TABLE 7.18: Results of Tukey test on task DTL-2. † indicates statistical significance at 0.95 confidence.

	IAC	ABCD	SETC
WoLLEx$_{\text{OH+LN}}$	0.790	0.771	0.792
WoLLEx$_{\text{OH+RLN}}$	0.755	0.762	0.791
WoLLEx$_{\text{EMB+LN}}$	**0.817**	0.745	**0.797**
WoLLEx$_{\text{EMB+RLN}}$	0.775	0.770	0.758
VLN	0.634	0.607	0.610
VRLN	0.579	0.603	0.609
OneHot	0.712	0.748	0.735
Embedding	0.752	**0.777**	0.736

TABLE 7.19: Accuracy results of Task DSC-2 with word-level feature model

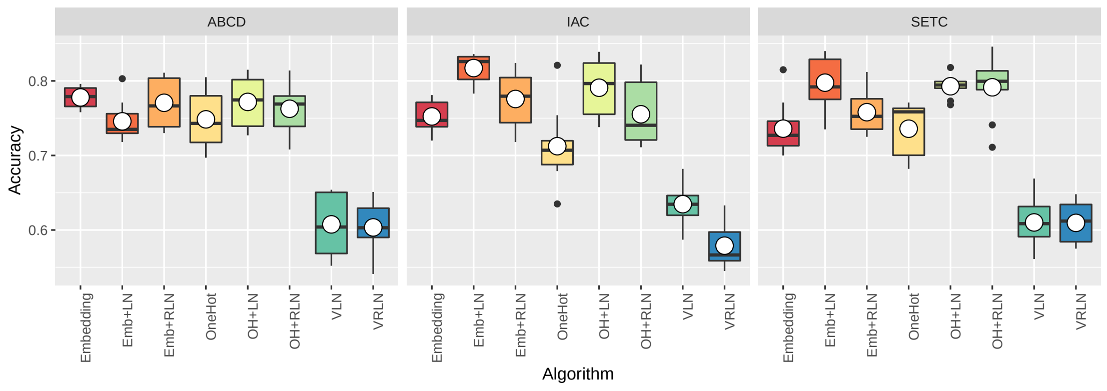


FIGURE 7.24: Box plot of accuracy results of Task DSC-2 with word-level feature model

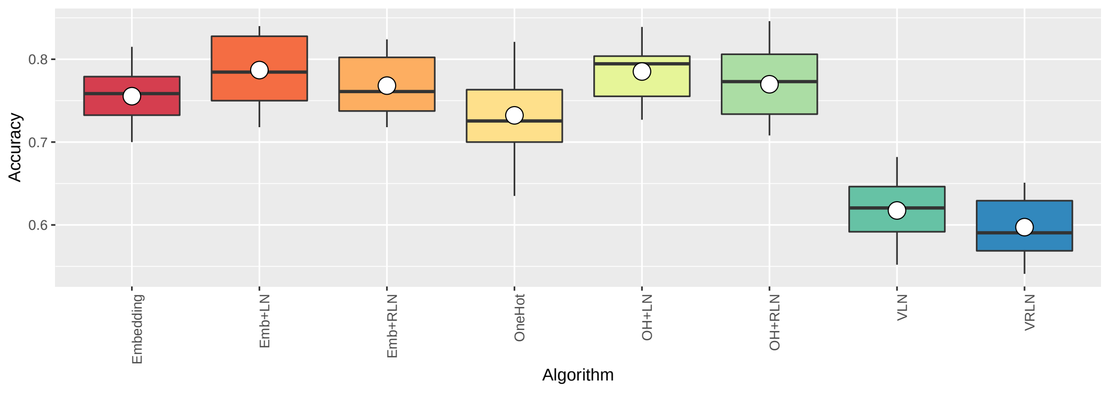


FIGURE 7.25: Box plot of accuracy results of Task DSC-2 with word-level feature model

Dataset	F-Value	P-Value
IAC	53.935	**3.9884104071254e-26**†
ABCD	49.597	**4.78186413583018e-25**†
SETC	58.401	**3.65354147763317e-27**†

TABLE 7.20: Results of ANOVA on task DSC-2. † indicates statistical significance at 0.95 confidence.

The results of the ANOVA in Table 7.20 show that there is a significant difference worth investigating between approaches, which leads us to the results of a Tukey HSD test, in Table 7.21. The results from the Tukey test confirm the observations that we can make from the plot, where the only significant differences are between the very low performing lexicon-only representations and the rest of the approaches, including the embedding and the one hot baselines. The WOLLEX$_{\text{EMB+LN}}$ representation also outperforms the ONEHOT and EMBEDDING on two of the datasets, IAC and SETC, with a statistically significant difference.

Results of task DSC-3 In this section we perform an evaluation of our proposed representations on the DSC-3 task, a three class stance classification problem. The test results in Table 7.22 show that WOLLEX$_{\text{OH+LN}}$ and WOLLEX$_{\text{EMB+LN}}$ tend to outperform other approaches on all datasets. The strongest baseline, EMBEDDING, comes within 0.004 to 0.016 of the top performing representation, which is close enough that we do not expect to see a statistically significant difference in our further testing. We illustrate those results in Figure 7.26, which shows a box and whiskers plot of the accuracy broken down per dataset, and Figure 7.27, which shows those same scores aggregated over all datasets. We can see from those plots that embedding-based methods tend to fare better on IAC and ABCD, with them having a more consistent set of scores than enriched methods based on one-hot encoding.

The p-values from our ANOVA, in Table 7.23, indicate that there is a significant difference in the results of this experiment, which we investigate with a Tukey HSD test shown in Table 7.24. Further testing confirms our previous observation, which is that the EMBEDDING baseline is close enough that statistical significance is not attained on the difference between it and our

	IAC		ABCD		SETC	
	Diff.	Adj. P-Value	Diff.	Adj. P-Value	Diff.	Adj. P-Value
WoLLex$_{\text{EMB+LN}}$/Embedding	0.06	**0.00**†	-0.03	0.38	0.06	**0.00**†
WoLLex$_{\text{EMB+RLN}}$/Embedding	0.02	0.81	-0.01	1.00	0.02	0.78
OneHot/Embedding	-0.04	0.20	-0.03	0.50	0.00	1.00
WoLLex$_{\text{OH+LN}}$/Embedding	0.04	0.23	-0.01	1.00	0.06	**0.00**†
WoLLex$_{\text{OH+RLN}}$/Embedding	0.00	1.00	-0.02	0.97	0.06	**0.01**†
VLN/Embedding	-0.12	**0.00**†	-0.17	**0.00**†	-0.13	**0.00**†
VRLN/Embedding	-0.17	**0.00**†	-0.17	**0.00**†	-0.13	**0.00**†
WoLLex$_{\text{EMB+RLN}}$/WoLLex$_{\text{EMB+LN}}$	-0.04	0.16	0.02	0.70	-0.04	0.12
OneHot/WoLLex$_{\text{EMB+LN}}$	-0.10	**0.00**†	0.00	1.00	-0.06	**0.00**†
WoLLex$_{\text{OH+LN}}$/WoLLex$_{\text{EMB+LN}}$	-0.03	0.70	0.03	0.65	0.00	1.00
WoLLex$_{\text{OH+RLN}}$/WoLLex$_{\text{EMB+LN}}$	-0.06	**0.00**†	0.02	0.95	-0.01	1.00
VLN/WoLLex$_{\text{EMB+LN}}$	-0.18	**0.00**†	-0.14	**0.00**†	-0.19	**0.00**†
VRLN/WoLLex$_{\text{EMB+LN}}$	-0.24	**0.00**†	-0.14	**0.00**†	-0.19	**0.00**†
OneHot/WoLLex$_{\text{EMB+RLN}}$	-0.06	**0.00**†	-0.02	0.80	-0.02	0.78
WoLLex$_{\text{OH+LN}}$/WoLLex$_{\text{EMB+RLN}}$	0.02	0.98	0.00	1.00	0.03	0.24
WoLLex$_{\text{OH+RLN}}$/WoLLex$_{\text{EMB+RLN}}$	-0.02	0.89	-0.01	1.00	0.03	0.29
VLN/WoLLex$_{\text{EMB+RLN}}$	-0.14	**0.00**†	-0.16	**0.00**†	-0.15	**0.00**†
VRLN/WoLLex$_{\text{EMB+RLN}}$	-0.20	**0.00**†	-0.17	**0.00**†	-0.15	**0.00**†
WoLLex$_{\text{OH+LN}}$/OneHot	0.08	**0.00**†	0.02	0.76	0.06	**0.00**†
WoLLex$_{\text{OH+RLN}}$/OneHot	0.04	0.13	0.01	0.98	0.06	**0.01**†
VLN/OneHot	-0.08	**0.00**†	-0.14	**0.00**†	-0.13	**0.00**†
VRLN/OneHot	-0.13	**0.00**†	-0.14	**0.00**†	-0.13	**0.00**†
WoLLex$_{\text{OH+RLN}}$/WoLLex$_{\text{OH+LN}}$	-0.04	0.33	-0.01	1.00	0.00	1.00
VLN/WoLLex$_{\text{OH+LN}}$	-0.16	**0.00**†	-0.16	**0.00**†	-0.18	**0.00**†
VRLN/WoLLex$_{\text{OH+LN}}$	-0.21	**0.00**†	-0.17	**0.00**†	-0.18	**0.00**†
VLN/WoLLex$_{\text{OH+RLN}}$	-0.12	**0.00**†	-0.16	**0.00**†	-0.18	**0.00**†
VRLN/WoLLex$_{\text{OH+RLN}}$	-0.18	**0.00**†	-0.16	**0.00**†	-0.18	**0.00**†
VRLN/VLN	-0.06	**0.02**†	0.00	1.00	0.00	1.00

TABLE 7.21: Tukey test for IAC, ABCD and SETC on DSC-2. † indicates statistical significance at 0.95 confidence.

	IAC	ABCD	SETC
WoLLEx$_{\text{OH+LN}}$	**0.685**	0.671	**0.637**
WoLLEx$_{\text{OH+RLN}}$	0.647	0.650	0.635
WoLLEx$_{\text{EMB+LN}}$	0.681	**0.685**	0.627
WoLLEx$_{\text{EMB+RLN}}$	0.642	0.665	0.605
VLN	0.579	0.554	0.551
VRLN	0.554	0.536	0.548
OneHot	0.520	0.626	0.578
Embedding	0.676	0.681	0.621

TABLE 7.22: Accuracy results of Task DSC-3 with word-level feature model

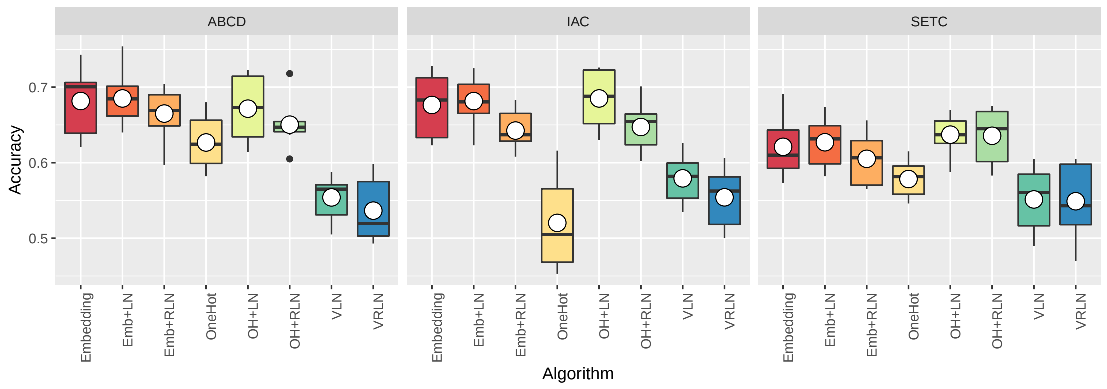


FIGURE 7.26: Box plot of accuracy results of Task DSC-3 with word-level feature model

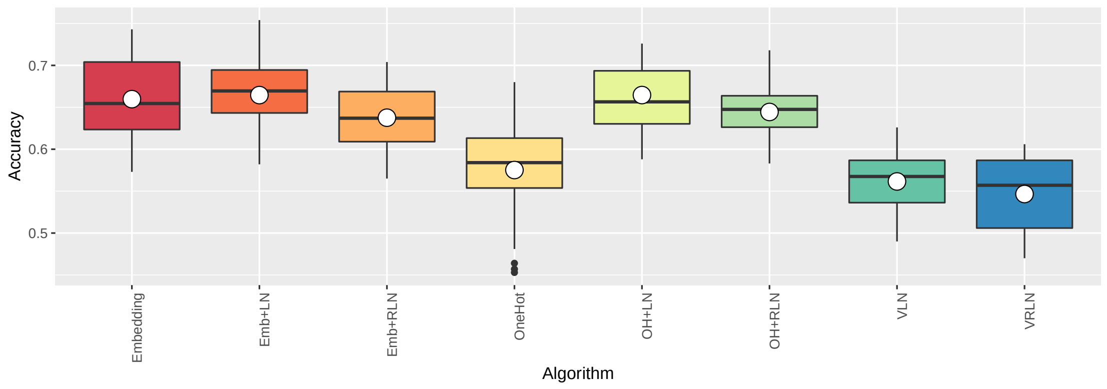


FIGURE 7.27: Box plot of accuracy results of Task DSC-3 with word-level feature model

Dataset	F-Value	P-Value
IAC	27.104	**8.01565094976069e-18**[†]
ABCD	25.16	**5.20748057425818e-17**[†]
SETC	9.952	**1.25650747438696e-08**[†]

TABLE 7.23: Results of ANOVA on task DSC-3. [†] indicates statistical significance at 0.95 confidence.

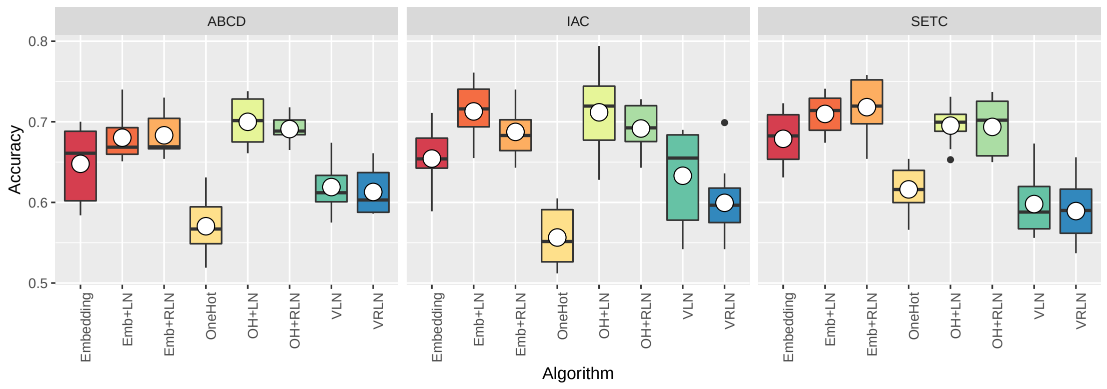


FIGURE 7.28: Box plot of accuracy results of Task DTL-2 with word-level feature model

approaches. However, the other baselines such as the lexicon-only representations as well as one-hot encoding are outperformed with a significant margin in most datasets.

Results of task DTL-2 In this section we comment on the results of task DTL-2, displayed in Table 7.25, Figures 7.28 and 7.29. A visual inspection of those results shows that proposed approaches outperform the baseline approaches by a reasonable margin, beating the best baseline by a range of 0.03 to 0.057. The LEXICNET-enriched representations tend to perform better on the IAC and ABCD datasets, and are extremely close to the results of the RELEXNET-enriched representations on the SETC dataset. There is no consistent difference between representations based on one-hot encoding and based on word embeddings, except that embedding-based techniques tend to yield a better accuracy on the SETC dataset.

The ANOVA test displayed in Figure 7.26 shows evidence of there being a significant difference within the results previously discussed, and therefore we perform a post hoc test using the Tukey HSD testing procedure, and show the results in Table 7.27. The results of the Tukey HSD test coincide with our previous observations and do not reveal any counter-intuitive result.

	IAC		ABCD		SETC	
	Diff.	Adj. P-Value	Diff.	Adj. P-Value	Diff.	Adj. P-Value
WoLLex$_{EMB+LN}$/Embedding	0.01	1.00	0.00	1.00	0.01	1.00
WoLLex$_{EMB+RLN}$/Embedding	-0.03	0.51	-0.02	0.97	-0.02	0.98
OneHot/Embedding	-0.16	**0.00**†	-0.05	**0.03**†	-0.04	0.17
WoLLex$_{OH+LN}$/Embedding	0.01	1.00	-0.01	1.00	0.02	0.98
WoLLex$_{OH+RLN}$/Embedding	-0.03	0.69	-0.03	0.54	0.01	0.99
VLN/Embedding	-0.10	**0.00**†	-0.13	**0.00**†	-0.07	**0.00**†
VRLN/Embedding	-0.12	**0.00**†	-0.15	**0.00**†	-0.07	**0.00**†
WoLLex$_{EMB+RLN}$/WoLLex$_{EMB+LN}$	-0.04	0.33	-0.02	0.92	-0.02	0.88
OneHot/WoLLex$_{EMB+LN}$	-0.16	**0.00**†	-0.06	**0.01**†	-0.05	0.07
WoLLex$_{OH+LN}$/WoLLex$_{EMB+LN}$	0.00	1.00	-0.01	0.99	0.01	1.00
WoLLex$_{OH+RLN}$/WoLLex$_{EMB+LN}$	-0.03	0.50	-0.03	0.41	0.01	1.00
VLN/WoLLex$_{EMB+LN}$	-0.10	**0.00**†	-0.13	**0.00**†	-0.08	**0.00**†
VRLN/WoLLex$_{EMB+LN}$	-0.13	**0.00**†	-0.15	**0.00**†	-0.08	**0.00**†
OneHot/WoLLex$_{EMB+RLN}$	-0.12	**0.00**†	-0.04	0.28	-0.03	0.72
WoLLex$_{OH+LN}$/WoLLex$_{EMB+RLN}$	0.04	0.23	0.01	1.00	0.03	0.52
WoLLex$_{OH+RLN}$/WoLLex$_{EMB+RLN}$	0.00	1.00	-0.01	0.99	0.03	0.58
VLN/WoLLex$_{EMB+RLN}$	-0.06	**0.01**†	-0.11	**0.00**†	-0.05	0.03
VRLN/WoLLex$_{EMB+RLN}$	-0.09	**0.00**†	-0.13	**0.00**†	-0.06	**0.02**†
WoLLex$_{OH+LN}$/OneHot	0.16	**0.00**†	0.04	0.13	0.06	**0.01**†
WoLLex$_{OH+RLN}$/OneHot	0.13	**0.00**†	0.02	0.83	0.06	**0.02**†
VLN/OneHot	0.06	**0.02**†	-0.07	**0.00**†	-0.03	0.72
VRLN/OneHot	0.03	0.52	-0.09	**0.00**†	-0.03	0.62
WoLLex$_{OH+RLN}$/WoLLex$_{OH+LN}$	-0.04	0.37	-0.02	0.90	0.00	1.00
VLN/WoLLex$_{OH+LN}$	-0.11	**0.00**†	-0.12	**0.00**†	-0.09	**0.00**†
VRLN/WoLLex$_{OH+LN}$	-0.13	**0.00**†	-0.13	**0.00**†	-0.09	**0.00**†
VLN/WoLLex$_{OH+RLN}$	-0.07	**0.00**†	-0.10	**0.00**†	-0.08	**0.00**†
VRLN/WoLLex$_{OH+RLN}$	-0.09	**0.00**†	-0.11	**0.00**†	-0.09	**0.00**†
VRLN/VLN	-0.03	0.83	-0.02	0.96	0.00	1.00

Table 7.24: Tukey test for IAC, ABCD and SETC on DSC-3. † indicates statistical significance at 0.95 confidence.

	IAC	ABCD	SETC
WoLLEX$_{\text{OH+LN}}$	0.711	**0.700**	0.695
WoLLEX$_{\text{OH+RLN}}$	0.691	0.691	0.693
WoLLEX$_{\text{EMB+LN}}$	**0.712**	0.680	0.709
WoLLEX$_{\text{EMB+RLN}}$	0.687	0.683	**0.718**
VLN	0.633	0.619	0.598
VRLN	0.599	0.613	0.589
ONEHOT	0.556	0.570	0.616
EMBEDDING	0.654	0.647	0.678

TABLE 7.25: Accuracy results of Task DTL-2 with word-level feature model

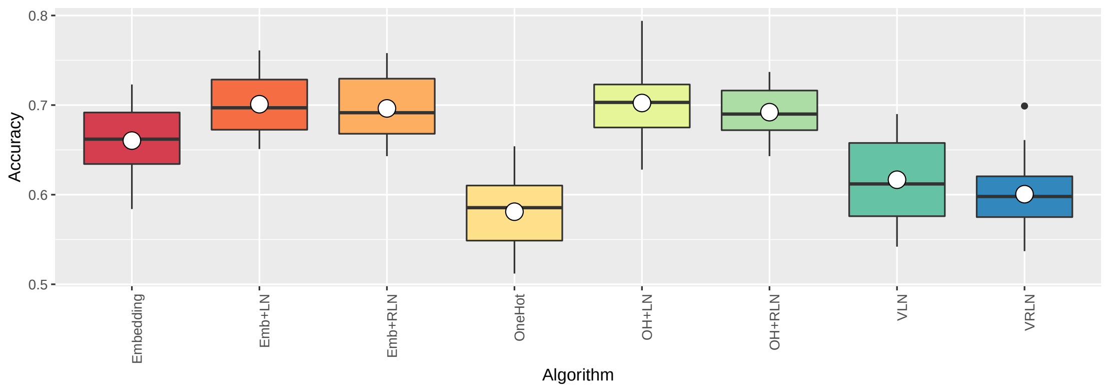


FIGURE 7.29: Box plot of accuracy results of Task DTL-2 with word-level feature model

Dataset	F-Value	P-Value
IAC	18.185	**1.07737810299774e-13**[†]
ABCD	20.656	**5.98930389381991e-15**[†]
SETC	25.37	**4.23477489899523e-17**[†]

TABLE 7.26: Results of ANOVA on task DTL-2. [†] indicates statistical significance at 0.95 confidence.

	IAC		ABCD		SETC	
	Diff.	Adj. P-Value	Diff.	Adj. P-Value	Diff.	Adj. P-Value
WoLLex$_{\text{EMB+LN}}$/Embedding	0.06	0.05	0.03	0.33	0.03	0.43
WoLLex$_{\text{EMB+RLN}}$/Embedding	0.03	0.64	0.04	0.23	0.04	0.15
OneHot/Embedding	-0.10	**0.00**†	-0.08	**0.00**†	-0.06	**0.00**†
WoLLex$_{\text{OH+LN}}$/Embedding	0.06	0.06	0.05	**0.01**†	0.02	0.95
WoLLex$_{\text{OH+RLN}}$/Embedding	0.04	0.49	0.04	0.07	0.01	0.97
VLN/Embedding	-0.02	0.95	-0.03	0.48	-0.08	**0.00**†
VRLN/Embedding	-0.05	0.08	-0.03	0.24	-0.09	**0.00**†
WoLLex$_{\text{EMB+RLN}}$/WoLLex$_{\text{EMB+LN}}$	-0.03	0.87	0.00	1.00	0.01	1.00
OneHot/WoLLex$_{\text{EMB+LN}}$	-0.16	**0.00**†	-0.11	**0.00**†	-0.09	**0.00**†
WoLLex$_{\text{OH+LN}}$/WoLLex$_{\text{EMB+LN}}$	0.00	1.00	0.02	0.87	-0.01	0.98
WoLLex$_{\text{OH+RLN}}$/WoLLex$_{\text{EMB+LN}}$	-0.02	0.95	0.01	1.00	-0.02	0.96
VLN/WoLLex$_{\text{EMB+LN}}$	-0.08	**0.00**†	-0.06	**0.00**†	-0.11	**0.00**†
VRLN/WoLLex$_{\text{EMB+LN}}$	-0.11	**0.00**†	-0.07	**0.00**†	-0.12	**0.00**†
OneHot/WoLLex$_{\text{EMB+RLN}}$	-0.13	**0.00**†	-0.11	**0.00**†	-0.10	**0.00**†
WoLLex$_{\text{OH+LN}}$/WoLLex$_{\text{EMB+RLN}}$	0.02	0.90	0.02	0.94	-0.02	0.78
WoLLex$_{\text{OH+RLN}}$/WoLLex$_{\text{EMB+RLN}}$	0.00	1.00	0.01	1.00	-0.02	0.71
VLN/WoLLex$_{\text{EMB+RLN}}$	-0.05	0.09	-0.06	**0.00**†	-0.12	**0.00**†
VRLN/WoLLex$_{\text{EMB+RLN}}$	-0.09	**0.00**†	-0.07	**0.00**†	-0.13	**0.00**†
WoLLex$_{\text{OH+LN}}$/OneHot	0.16	**0.00**†	0.13	**0.00**†	0.08	**0.00**†
WoLLex$_{\text{OH+RLN}}$/OneHot	0.14	**0.00**†	0.12	**0.00**†	0.08	**0.00**†
VLN/OneHot	0.08	**0.00**†	0.05	**0.02**†	-0.02	0.92
VRLN/OneHot	0.04	0.31	0.04	0.07	-0.03	0.61
WoLLex$_{\text{OH+RLN}}$/WoLLex$_{\text{OH+LN}}$	-0.02	0.96	-0.01	1.00	0.00	1.00
VLN/WoLLex$_{\text{OH+LN}}$	-0.08	**0.00**†	-0.08	**0.00**†	-0.10	**0.00**†
VRLN/WoLLex$_{\text{OH+LN}}$	-0.11	**0.00**†	-0.09	**0.00**†	-0.11	**0.00**†
VLN/WoLLex$_{\text{OH+RLN}}$	-0.06	0.05	-0.07	**0.00**†	-0.10	**0.00**†
VRLN/WoLLex$_{\text{OH+RLN}}$	-0.09	**0.00**†	-0.08	**0.00**†	-0.10	**0.00**†
VRLN/VLN	-0.03	0.62	-0.01	1.00	-0.01	1.00

TABLE 7.27: Tukey test for IAC, ABCD and SETC on DTL-2. † indicates statistical significance at 0.95 confidence.

7.5.5 Discussion

In this section we demonstrated and commented on the results from multiple experiments, testing the usefulness of augmenting semantic representations of text with additional knowledge in the form of a lexicon representation of words, at multiple levels of granularity starting from an instance-level representation, where a flat vector of unchanging dimension represents the text to be classified, to a word-level representation where each term of an instance is represented on a semantic level as well as on a lexicon level. We now seek to answer the two research questions that were asked in the beginning of this chapter with the evidence collected through our experiments: (1) can a lexicon enhance a traditional text representation scheme for stance classification? and (2) at which granularity, if any, does this enhancement perform best?

7.5.5.1 Lexicon-based enhancement of text representation

While a visual inspection of the results often shows that the enriched representations outperform their non-enriched counterparts, further statistical testing more often than not shows that such difference is not statistically significant. One probable cause for this is the fact that we are testing several hypotheses by pairing each representation together in order to detect any statistical significance in their difference. The high number of possible pairings forces us to use an extremely conservative testing procedure in order to reduce the chance of increasing our type 1 error rate. However, the existing results show a lot of promise in the field of text representation: it is possible and effective to use resources such as lexicons in order to enhance classification accuracy in an artificial manner.

In order to overcome this problem we perform a further analysis by grouping algorithms together according to the following groups: proposed approaches, containing all enriched representations with a semantic and a lexicon part, and the top ranking baseline based on the use of word embeddings alone. We aggregate the results of all those algorithms on the DSC-2, DSC-3, and DTL-2 tasks and display those results in Table 7.28.

On the DSC-2 tasks, we find the mean average accuracy to be 0.753 for the enriched representations and 0.720 for the best baselines. A two-tailed Welch T-test for unequal variances reveals a p-value of $1.316e-08$, which passes the $p < 0.05$ as well as the $p < 0.01$ threshold

Task	DSC-2	DSC-3	DTL-2
Baseline approaches	0.720	**0.661**†	0.628
Enriched representations	**0.753**†	0.649	**0.701**†

TABLE 7.28: Mean accuracies when averaging over types of approaches

and shows that enriched representations overall outperform significantly very strong baselines. Similarly, with respect to the DSC-3 tasks we find mean average accuracies of 0.649 and 0.661 for enriched and baseline representations (respectively), yielding a p-value of 0.01239. While it does not pass the $p < 0.01$ threshold, it does pass the $p < 0.05$ threshold and achieves statistical significance, lending evidence to the fact that lexicon enrichment of semantic representations has a detrimental effect on classification in a 3 class stance classification task. A possible reason could be that the lexicons computed for 3 classes are not reliable for text representation, which is in agreement with the poor performance of shallow lexicon-based classification in evaluation task SSC-3, Section 6.4.3.2. Finally, with respect to the DTL-2 task enriched representations achieve an average accuracy of 0.701, while baseline approaches achieve an average of 0.628. Further testing yields a p-value of $2.2e - 16$, which easily reaches statistical significance at the $p < 0.05$ and $p < 0.01$ level.

Those further experiments lend more evidence to the fact that enriching semantic representations with class-sensitive information such as lexicon knowledge can help improve classification accuracy as long as the lexicon representation is good itself. Indeed, it appears that the performance of the lexicon-enhanced representations are highly correlated to the performance of lexicon classifiers with the corresponding number of classes as seen in the previous chapter (DSC2 and DTL-2 with SSC-2, DSC-3 with SSC-3).

7.5.5.2 Granularity of representation

The second research question concerns itself with the optimal granularity of representation in order to make use of enriched representations. The aggregated results are presented in Figure 7.30 from which we observe that while there is little to no difference in the DTL-2 task, the DSC-2

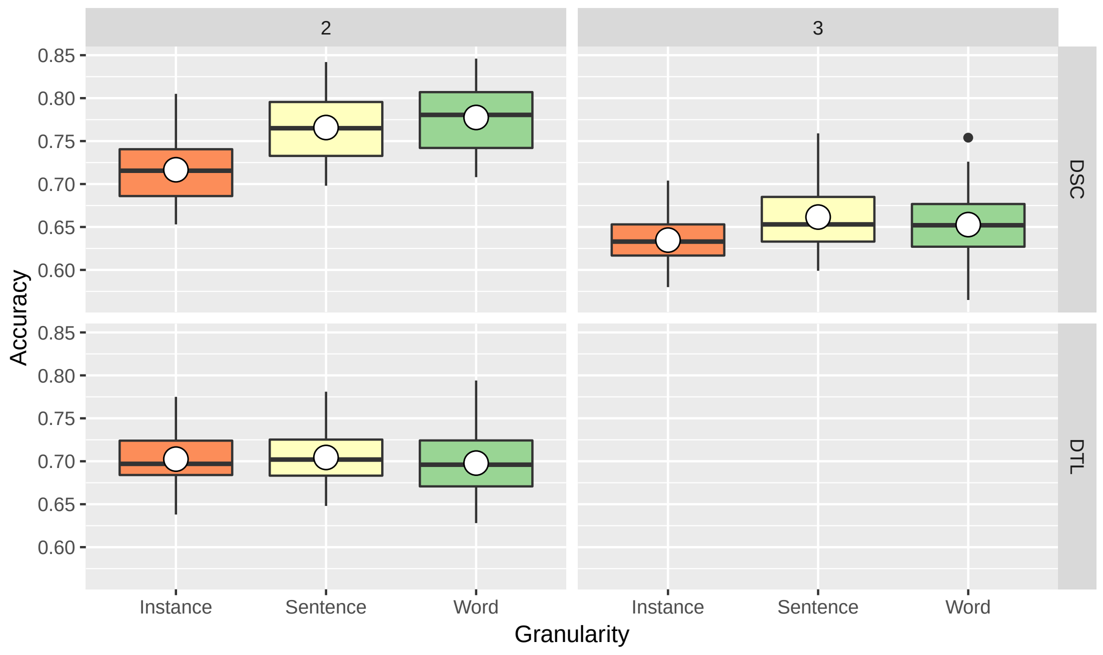


FIGURE 7.30: Box plot of aggregated accuracy results of enriched approaches averaged per granularity level

and DSC-3 tasks seem to be slightly in favour of finer granularity in text representation, with higher accuracy scores for word-level than sentence or instance-level features.

A Tukey HSD test shown in Table 7.29 confirms our observations, showing that there is no significant difference in the DTL-2 task, while there is a clear ordering in DSC-2 and DSC-3. In DSC-2 all comparisons achieve statistical significance at the $p < 0.05$ level, indicating that word-level representations are superior. In DSC-3, there is a clear preference of word and sentence level representations over instance level representations, but no clear difference between word level and sentence level representation. Finally in DTL-2 there is virtually no difference between the approaches, which emphasises the difficulty of learning transferable knowledge from outside corpora.

7.6 Chapter summary

In this chapter we explored a set of representations at different levels of granularity that are designed to help us answer the following research questions: can class-sensitive features learned on a separate corpus (lexicon knowledge) enrich an existing term representation and allow it to better represent argumentative content for the purpose of stance classification? And which

	DSC-2		DSC-3		DTL-2	
	Diff.	P-Value	Diff.	P-Value	Diff.	P-Value
Sentence/Instance	0.05	**0.00**†	0.03	**0.00**†	0.00	0.88
Word/Instance	0.06	**0.00**†	0.02	**0.00**†	-0.00	0.48
Word/Sentence	0.01	**0.04**†	-0.01	0.14	-0.01	0.23

TABLE 7.29: Differences and p-values of a Tukey HSD test comparing granularity level for different tasks

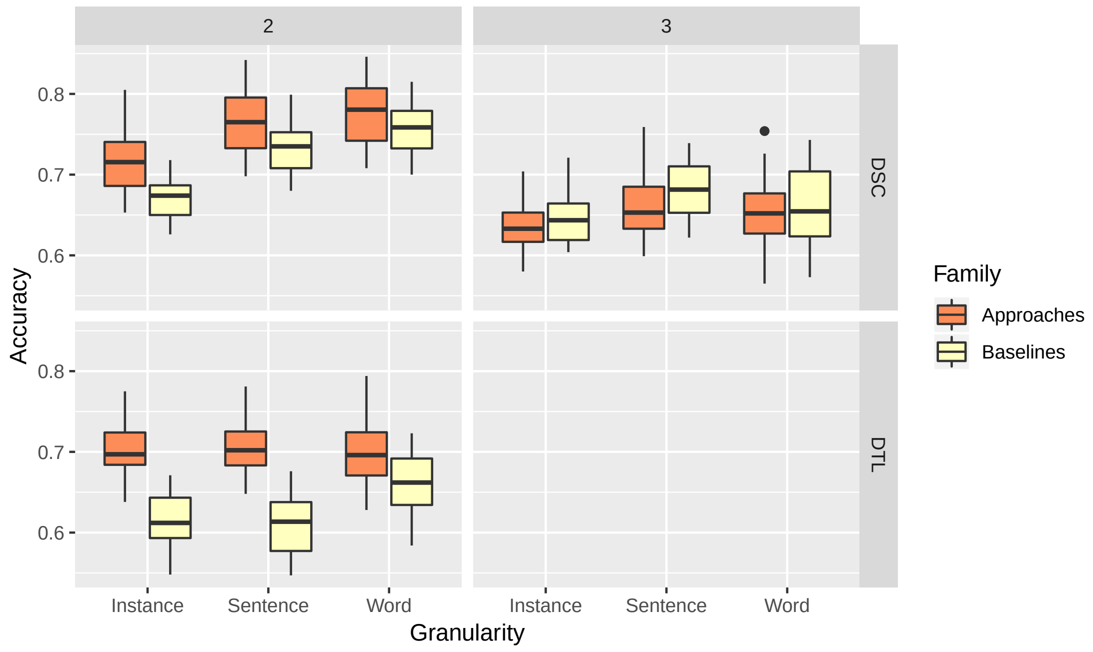


FIGURE 7.31: Box plot of aggregated accuracy results of enriched approaches averaged per granularity level, separated by family of approaches

level of granularity when representing instances works better? We introduced three level of granularity: word, sentence and instance level, and different forms of class-sensitive features at each of those levels. We evaluated our approaches along two axes, firstly by measuring the impact of merging semantic and lexicon-based representations for supervised classification, and secondly by reproducing that experiment at different levels of granularity (instance, sentences, words). We concluded that while it is not easily perceivable by statistical testing, there is a strong positive effect of lexicon-based features for text representation at different levels of granularity. We investigated such effect using the plot in figure 7.31, which shows the interaction between granularity and lexicon-based enrichment, and confirms our previous observation that lexicon-based enrichment seems to benefit more approaches at the instance and sentence level, possibly because the word-level semantic representations do not lose any meaning by averaging out their representation over sentences or entire instances. Secondly, we found that word-level representation is the best fit in most cases, being at worst as good and occasionally superior in classification accuracy to the other levels of representation.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In the past decade, stance detection has slowly become a more important challenge in the area of opinion mining, and has numerous applications in the fields of information retrieval, text summarisation, and argumentation mining. In this thesis we addressed the task of detecting stance in informal text using a diverse set of methods, ranging from unsupervised, heuristic-based methods inspired from the sentiment analysis community to the generation of automated lexicons, which are well-known tools of computational linguistics, and finally by using such lexicons to compute feature models for deep neural networks, in order to bypass the data-hungry feature learning step that is often thought to be paramount to their success.

This thesis started by explaining the underlying motivation of wanting to compute the stance of user-generated, informal texts as well as the potential scope of such research. We put a particular emphasis on building platform-agnostic models and as such avoided large features models from the literature that used hundreds of platform-specific features to represent text. We explained the relevance of social media mining to numerous sub-fields of artificial intelligence such as debate analysis, and positioned stance detection with respect to neighbouring sub-fields such as sentiment analysis.

In Chapter 2, we then reviewed the literature on argument stance classification, as well as the different corpora that were generated for that purpose, before considering the background knowledge necessary for this thesis in Chapter 3. In Chapter 3, we reviewed the basics of supervised machine learning applied to text classification in order to further contextualise the content of this work. In Chapter 4, we explained and detailed the evaluation methodology that we adopted throughout the thesis, starting with the datasets that were used and how they were processed, and finishing with the statistical significance testing procedures that we used to ensure some rigour in our investigation.

The rest of the thesis was organised in three main chapters, describing a different aspect of our work: unsupervised methods for stance classification, lexicon generation for stance classification, and finally the use of stance lexicons to build feature models.

In Chapter 5, we investigated unsupervised classification techniques for stance classification. Unsupervised classification is still a vibrant research area because of the small amounts of labelled data that researchers have access to. While machine learning algorithms require significant amounts of domain expertise in the process of labelling hundreds of thousands of data points, a single expert can encode their pre-existing knowledge using heuristics and rules and bypass that need completely. Because of this, several techniques such as sentiment lexicons are still to this day built manually using dictionary entries and expert domain knowledge. Four techniques were investigated in the context of this chapter. The first two techniques, SP-ADAPT and SP-COSTS, use sentiment as a proxy for stance and an adaptive threshold to improve results. In particular, SP-COSTS uses context knowledge in order to apply the sentiment lexicon on a subset of the terms present in the text to be classified, and shows significant improvement over using the raw text alone. The other techniques developed in this chapter, USS, USS+ and MUSC, are complex heuristics building on top of each other, where MUSC uses USS+ as a fallback method, which uses USS as a fallback method, where the text to be classified falls outside of the range of expertise of the current heuristic. This leads to modest but not significant improvement over naive baselines such as the PMI-LEX method, although we strongly suspect that it is mainly due to statistical reasons rather than lack of intrinsic superiority.

In Chapter 6, we turned our attention to another vibrant area of computational linguistics: computational lexicons, and how to build them from data. We proposed to use well-known techniques for empirical risk minimisation in order to learn lexicons similarly to how the weights of a single perceptron units would be learned, and introduced the LEXICNET approach. The advantages of such a technique over existing lexicon learning methods such as methods based on pointwise mutual information or expectation maximisation is that the optimisation process by gradient descent can be arbitrarily regularised, which helps us avoid learning lexicons which overfit over specific language patterns. We then extended this lexicon learning architecture so that it could learn not only valence terms, which carry an inherent weight towards a class, but also valence shifters, which are not stance-bearing in themselves but can increase or decrease the valence of neighbouring stance-bearing terms. This approach, named RELEXNET[1], shows some modest improvement on free-text datasets, and less so on size-constrained datasets (such as SETC, which was built on Twitter microblogging posts).

In Chapter 7, we investigated the use of previously developed lexicon learning architectures to build feature models for deep learning methods. The intuition behind such models is that features extracted from a stance lexicon provide a form of background knowledge to the classification algorithm which helps the algorithm learn faster by biasing them towards a class.

8.2 Research questions

In the beginning of this thesis, we set out to answer a set of research questions that would guide our investigation in argument stance classification. In this section, we leverage the work presented in the rest of the thesis in order to formulate a concise answer to those questions.

RQ1. *Can we use the direct conversational context to improve classification accuracy by helping a naive classifier focus on relevant terms?* We answer this research question by drawing on the work presented in Chapter 5. We can see from the results of the evaluation in chapter 5 that the addition of some form of context provides an improvement in classification accuracy when trying to predict stance. However, that improvement does not seem to reach the statistical significance threshold we set out for our work.

RQ2. *Can we use empirical risk minimisation techniques to learn a regularised stance classification lexicon?* We answer this research question by drawing on the results presented in

[1]For reasons that have been long forgotten by the author.

Chapter 6. We can see from those results that the lexicon that we proposed, LEXICNET, tends to outperform other approaches. However once again, the difference does not seem to reach statistical significance.

RQ3. *Can we leverage the neighbourhood of words to learn a context-sensitive classification lexicon?* We also answer this research question by drawing on the results presented in Chapter 6. We can see from those results that the extension of the lexicon presented, RELEXNET, outperforms most other approaches. Similarly to LEXICNET, the difference does not consistently reach statistical significance.

RQ4. *Can class-sensitive features learned on a corpus enrich an existing term representation and allow it to better represent argumentative content?* This question is answered using the results presented in Chapter 7. While the effect is not consistently statistically significant, there is a strong positive effect when enhancing representations with lexicon-based features.

RQ5. *At what level of granularity (words, sentences, or entire instance) does lexicon enhancement performs better when applied to standard representation schemes?* The further analysis in Chapter 7 shows us that instance-level and sentence-level representations tend to benefit more from lexicon enhancement. However, while they do benefit more, word-level representations tend to perform better overall, which might be due to being more expressive (while sentence-level and instance-level representations are forced to average their representations over sentences and instances respectively).

Overall, while there are promising results throughout the work currently presented, we find that the lack of data in the field of argument stance mining makes it exceedingly difficult to obtain proper and consistent statistical significance in our experimental results.

8.3 Objectives revisited

In this section we revisit the objectives defined in the introduction and consider the way they were changed throughout this work.

◇ **Develop an unsupervised classification algorithm that leverages conversational context and existing resources from the field of opinion mining.**

  Unsupervised classification algorithms are still popular classifiers in field where annotated data are scarce and often noisy. Domain experts overcome the limitations of such lack of data by designing expert rules that reflect their expertise. Towards this objective in Chapter 5 we introduced a novel set of heuristics based on a hierarchical organisation of simpler heuristics starting with high precision low recall (detection of stance by finding specific language patterns) to high recall low precision (approximation of stance by sentiment). We find that our novel set of heuristics outperform all our baselines based on the sentiment analysis community.

◇ **Develop a novel approach to learn an argument lexicon from weakly labelled data.**

  Another way to deal with a lack of properly labelled data is to leverage weak or distant labelling, where a set of distant labels, often taking the form of language patterns, are used to approximate a labelling over a dataset. In Chapter 6 we introduce LEXICNET, a lexicon learning technique that works using the empirical risk minimisation principle and the backpropagation algorithm in order to compute optimal lexicon scores in a way that can be heavily regularised. The regularisation allows us to not overfit our lexicon to data with low confidence in the labelling, and thus to obtain better classification scores when using lexicons learned on this noisy data to other datasets.

◇ **Develop a novel learning algorithm to jointly learn an argument lexicon and its modifiers.**

  Lexicons work by using an overly simplistic classification model for the sake of interpretability: it is possible for a human user to backtrack from a prediction and understand which terms made the balance go towards one class rather than another. One way that lexicons have been extended in the literature without any negative impact on this interpretability is by the use of lexical valence modifiers, i. e., terms which have no class valence by themselves but can modify the class valence of terms in their neighbourhood, such as for example adverbs like "extremely" or "barely". However this relation makes it impossible for standard algorithms to model during the learning phase, which is why valence modifiers are typically specified by hand. In Chapter 6, we develop RELEXNET, an extension of the LEXICNET architecture that is able to account for modifiers and learn their modifying score during the learning phase.

◇ **Develop a representation scheme using the argument lexicon as a feature extractor that can leverage the temporal nature of text.** While lexicons are powerful classification tools for text, they often lack the representational power to achieve state-of-the-art accuracy when compared to machine learning algorithms that do not account for providing an interpretable model to the user. In Chapter 7, we investigate the use of lexicon scores as a way to enhance existing text representation schemes and observe the potential increase in accuracy on a set of classification tasks.

8.4 Future work

In order to overcome the limitations highlighted in the previous section, we submit the following plan for our future research on interpretable methods for stance detection.

Bayesian evidence combination for large-scale lexicons. One way to overcome the lack of data when learning lexicons would be to combine multiple data sources in the training set, and as such increase its size at a lower cost. This can be extended to combining multiple data sources of different level of confidence, for example by combining the RNLC and IAC datasets in order to classify instances from the ABCD dataset. This can be done by modelling the confidence that we have in each of our labelings inside the cost function that is minimised during the training process. Effectively, it equates to teaching the learning algorithm that making a mistake on a noisy-labelled RNLC example is not as grave an error as making a mistake on an example from the IAC dataset. Furthermore, this confidence estimation can be extended to distinguishing instances within the same dataset. For example, some instances from the RNLC dataset might display more evidence of belonging in one class than others. Rather than removing those low-confidence instances and only keeping the ones with high confidence, as in the case of this thesis, a natural extension would be to keep all data and give a weight to each instance depending on the confidence of its noisy labelling. This relates to the field of cost-sensitive learning, which was surveyed in Frénay and Verleysen (2014). In future work, we will investigate the use of cost-sensitive learning in the context of lexicons and how it can be applied to leverage large collections of noisy-labelled data without losing accuracy.

Context-sensitive attention-driven lexicons. Attention is a recent mechanism which has taken over neural network-based language processing and made recurrent neural networks close to redundant in machine translation Vaswani et al. (2017). It differs from recurrent neural networks in that it gives the neural network the power of deciding where it should focus its attention next, by manipulating a vector of weights over regions of the input space. Our work in 5 has attempted to manipulate a similar notion by using the content of contextual comments to decide where (on which terms) the naive classifier should focus its "attention", but in a naive way that proved to have very little effectiveness in a lot of cases. A better way of modelling context-sensitivity would be to train a black-box algorithm such as a neural network to use contextual information in order to generate a vector of attention weights over the instance to be classified. Those weights could then be used instead of simple term frequencies and combined to the values of the lexicon in order to provide a classification. The resulting model would then retain the interpretability of a lexicon model while striking a compromise with the modelling power of black-box methods such as neural networks. In future work, we will investigate the combination of attention-based methods with lexicon-based classification for high-accuracy and high-interpretability stance detection.

Appendix A

Publications

- J. Clos, N. Wiratunga, J. Jose, S. Massie, and G. Cabanac: Towards Argumentative Opinion Mining in Online Discussions. In: Proceedings of the SICSA Workshop on Argument Mining (2014)
- J. Clos, N. Wiratunga, S. Massie, and G. Cabanac: Shallow techniques for argument mining. In: ECA'15: Proceedings of the European Conference on Argumentation (2016)
- J. Clos and N. Wiratunga: Neural Induction of a Lexicon for Fast and Interpretable Stance Classification. In: International Conference on Language, Data and Knowledge (2017)
- J. Clos and N. Wiratunga: Lexicon Induction from Text for Interpretable Classification. In: 21 International Conference on Theory and Practice of Digital Libraries (2017)
- J. Clos, N. Wiratunga, and S. Massie: Towards Explainable Text Classification by Jointly Learning Lexicon and Modifier Terms. In: IJCAI Workshop on Explainable Artificial Intelligence (2017)

Appendix B

Background in Argumentation Theory

Argumentation theory is a highly interdisciplinary domain, spanning a period of time starting with the study of rhetoric in ancient Greece to, more recently, the use of abstract argumentation for collaborative decision-making systems. This multidisciplinary status allows us to examine it under multiple lenses. Philosophers, linguists, and computer scientists built their own theories of argumentation with the goal of modelling specific social phenomena (e. g., multi-agent negotiation) and therefore in this appendix we will concisely examine selected theories of argumentation and the extent of their uses from the perspective of information systems.

B.1 An overview of argumentation theories

In this section we review the main theories of argumentation from the perspective of logic, rhetoric and computer science. We examine the core elements of arguments under those points of view and draw conclusions on the requirements of a theory of argumentation for argument mining in the social Web.

B.1.1 Monological versus dialogical argumentation

The process of argumentation can be divided in two broad classes, depending on the goal and actors involved. In the case of monological argumentation, only one agent is involved and argumentation is used as a mean of reasoning by laying out arguments interacting with each other and observing the final outcome of the inference process. The agent is in a state of complete knowledge beforehand, and merely articulates arguments in order to come up with a conclusion. Monological argumentation is used for non-monotonic reasoning and decision explanation.

We focus on the case of dialogical argumentation, where the argumentation process is used by multiple agents in order to arrive at a decision by laying out arguments regarding a topic. No agent possesses all the facts and the only prerequisite is that a common ground regarding the topic of discussion is agreed upon. Dialogical argumentation is a better fit to model the interaction between users in social applications.

B.1.2 The structure of an argument: the Toulmin model of argumentation

The best known model of the structure of arguments is attributed to British philosopher Stephen Toulmin. The Toulmin model of practical argumentation as described in Toulmin (2003) states that an argument should contain six parts, the last three of which are optional to a well-formed argument:

1. the **claim** is the conclusion of the argument, that the other parts try to achieve ;
2. the **ground** is the set of facts, or evidence, that are used as a starting point ;
3. the **warrant** is a statement that allows a reader to logically link the ground to the claim ;
4. the **backing** is a set of evidence to justify the warrant ;
5. the **rebuttal** is a possible restriction that can be applied to the claim ;
6. the **qualifier** applies a modality on the claim by establishing the certainty with which the agent makes the statement.

This model is used as a basis for structuring arguments in debate mapping software. It relies on an optimistic, perfect argumentation process between well-intentioned rational agents, which makes it brittle on some platforms such as oral dialogue or web forums, where a lot of information is implied and it is not rare . Another problem is that in the case of a continuing argumentation, the agents can also implicitly refer to previous grounds. An example of this would be:

Ideal argument : [I am a British citizen]$_{claim}$ because [I was born in Bermuda]$_{ground}$, and [a man born in Bermuda is a British citizen]$_{warrant}$.

Realistic argument : [I am a British citizen]$_{claim}$ because [I was born in Bermuda]$_{ground}$.

Continuing argument : [I am a British citizen]$_{claim}$ because [of where I previously said I was born]$_{reference}$.

In addition to the problem of the semantic gap in natural language processing, these limitations underline the limitations of the Toulmin framework in the case of argument mining on the Social Web. The realistic situations in which such a framework would be used offer no assurance except for the presence of a claim, which means that a "pessimistic" framework would require either a greater flexibility in structure, taking into account:

- the temporal nature of the process, due to the necessity of being able to recall previously used grounds, or previously made claims, and test their validity at the time of argument evaluation ;
- the identity of the agents involved, so that we can consider the epistemic state of each agent, their affinities and dislikes of other agents, and other characteristics of the sort.

B.1.3 The argumentation process viewed as a graphical structure

Taking a step backwards and a birds-eye view of the process of argumentation, it is possible to describe it using a graphical structure. This is the domain of abstract argumentation frameworks, whose development was motivated by wanting to leverage the computational properties of graphs for discourse analysis. The two most popular frameworks are Dung's abstract argumentation framework and the bipolar argumentation framework.

Dung's abstract argumentation framework (Dung, 1995) describes the process of argumentation by using only a defeating relationship between arguments. Therefore the resulting graph possesses only one type of edge representing that an argument a_1 defeats an argument a_2, meaning that if the truth value of a_1 is true, then the truth value of a_2 has to be false ;

Bipolar argumentation framework (Amgoud et al., 2008) extends Dung's framework by adding an explicit supporting relationship between arguments. This relationship means that the graph possesses two types of edges, with one type representing that an argument a_1 defeats an argument a_2, and the other type representing that an argument a_1 supports an argument a_2. This new relationship means that if the truth value of a_1 is true, then it lends more weight to the truth value of a_2 in case of an uncertain graph topology.

The following running example, of a debate between multiple agents about the legal nationality of Agent1, will be used to illustrate the differences between the two frameworks

Arg1 : **Agent1**: I am of French origin

Arg2 : **Agent2**: You were born in Algeria, so you are not French

Arg3 : **Agent3**: Algeria was occupied by the French until 1962, so someone born in Algeria before 1962 would be French, unless they chose to stay in Algeria

Arg4 : **Agent2**: Agent1 chose to stay in Algeria

Arg5 : **Agent4**: Agent1 was forced to stay in Algeria

Arg6 : **Agent5**: French people speak French, you don't, so you are not French

Arg7 : **Agent3**: People in French colonies didn't always speak French

Arg8 : **Agent3**: Your birth certificate says Republic of France, so you are French

B.1.4 Dung's unipolar abstract argumentation framework

The abstract argumentation framework as defined by Dung in Dung (1995) structures an argumentation framework as the pair $\langle A, \Re \rangle$ with A being a set of arguments and $\Re$ being a binary relationship on A called a defeating relationship. Considering two arguments A_i and $A_j \in A$, $A_i \Re A_j$ means that A_j is defeated by A_i or that A_i defeats A_j. An argumentation framework is said to be well-grounded if there is at least one argument that is not defeated.

Dung's framework does not provide an explicit supporting relationship between arguments but defines the set of supporting arguments for an argument A_i as the set of arguments that do not

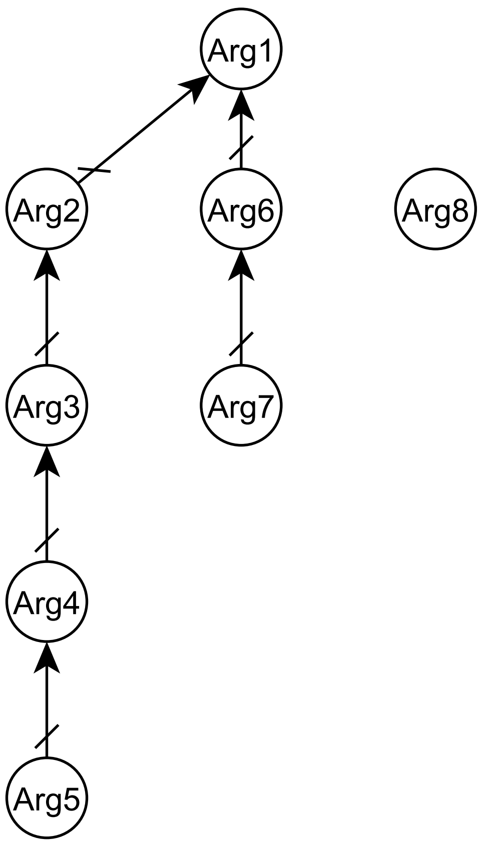


FIGURE B.1: Graphical example of an abstract argumentation framework

defeat A_i itself but defeat any argument of the set of arguments that defeat A_i. An argument which defeats both another argument and its defeater, is called a controversial argument.

Figure B.1 shows an example of a unipolar abstract argumentation framework. In this case arguments A_2 and A_5 are not defeated, which make it well grounded. The state of matter obviously revolves around the claim of A_1. The set of defeaters of A_1 comprises of A_2 and A_3. The set of supporters of A_1 comprises of A_4 and A_5 (because they are the defeaters of A_2 and A_3, which are A_1's defeaters). A_5 defeats both A_3 and A_4, which makes it a controversial argument.

From these basic definitions Dung builds the computational properties of his abstract argumentation framework. These permit the computation of different sets of arguments with special properties, such as for instance admissibility, or the fact that they mutually defend each other (and thus are undefeated when considered as a set).

B.1.5 Bipolar argumentation

Cayrol and Lagasquie-Schiex detail in Cayrol and Lagasquie-Schiex (2005) the extension of Dung's framework to include an explicit supporting relationship between arguments. Two previous attempts at including a notion of bipolarity are cited: the HERMES system Karacapilidis and Papadias (2001) and the DEFLOG system Verheij (2002).

The rationale behind the need for an explicit support relationship is the need for a richer way of describing real-life argumentation. Staying on an abstract level such as Dung's framework forces us to model the support relationship implicitly and thus possesses a low expressive power when modelling real argumentation.

Bipolar argumentation requires us indeed to leave the abstract, almost graph theoretical nature of Dung's framework to have a more detailed view of what does and what does not constitutes an argument. The authors thus give a logical description of arguments as a pair (S, C) attached to a logic L so that S is a set of consistant formulas of L called the **support** and C is a consistant formula of L called the **claim**, so that $S \vdash C$ and S is minimal to obtain C.

The authors then define different types of defeating relationships, related to which part of the argument are defeated and establish the basic properties of a bipolar argumentation system. A bipolar argumentation system is defined as a triple $\langle A, \Re_{def}, \Re_{sup}\rangle$ with:

- A as the set of arguments ;
- $\Re_{def}$ as the defeating relationship between arguments of A ;
- $\Re_{sup}$ as the supporting relationship between arguments of A ;

To illustrate the gain in expressivity, let us go back to the initial dialog between the five agents, and analyse the logic behind their reasoning in the following form: $(support, claim)$.

Arg1 : **Agent1**: I am of French origin : $(\{\}, f)$

Arg2 : **Agent2**: You were born in Algeria, so you are not French : $(\{a, a \rightarrow \neg f\}, \neg f))$

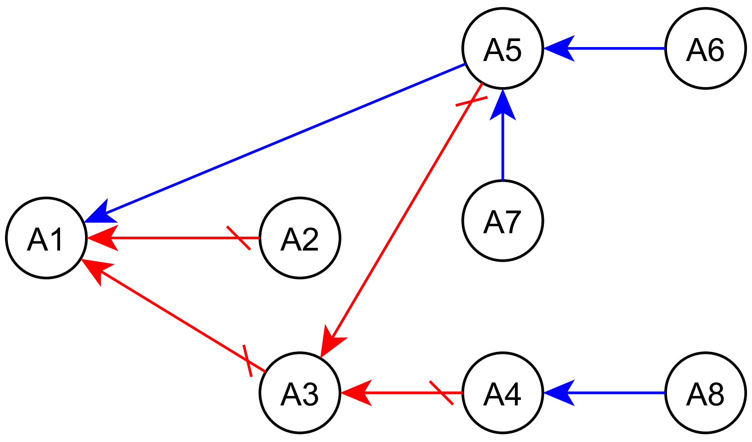


FIGURE B.2: Graphical example of a bipolar argumentation system

Arg3 : **Agent3**: Algeria was occupied by the French until 1962, so someone born in Algeria before 1962 would be French, unless they chose to stay in Algeria : $(\{a, d, d \to o, a \wedge o \wedge \neg c \to f\}, f)$

Arg4 : **Agent2**: Agent1 chose to stay in Algeria : $(\{\}, c)$

Arg5 : **Agent4**: Agent1 was forced to stay in Algeria : $(\{y, y \to \neg c\}, \neg c)$

Arg6 : **Agent5**: French people speak French, you don't, so you are not French : $(\{f \to s, \neg s\}, \neg f)$

Arg7 : **Agent3**: People in French colonies didn't always speak French : $(\{\}, \neg(f \to s))$

Arg8 : **Agent3**: Your birth certificate says Republic of France, so you are French : $(\{b, b \to f\}, f)$

Figure B.2 (page 191) is an example of a graphical representation of the bipolar argumentation system described previously. The crossed arrow represents the defeating relationship, while the normal arrow stands for the supporting relationship. We can see that the relationship between arguments 8 and 1 can now be modeled.

B.1.6 Comparison in context

While Dung's framework aims to stay at an abstract point of view and thus abstains to describe arguments themselves, merely representing the argumentation process as a graphical structure, the bipolar extension by Cayrol and Lagasquie-Schiex offers a richer language to describe the different cases in real life argumentation, almost bridging the gap to the purely logical framework described by Besnard and Hunter (Besnard and Hunter, 2008, Chapter 3). Dung's abstract argumentation framework is advantageous from a purely computational point of view because it defines the acceptability of an argument using only one binary relation instead of two, but lacks the expressivity that can be attained when explicitly modelling the support binary relation. Another extension of Dung's framework worth mentioning is the fuzzy argumentation framework detailed in Janssen et al. (2008) in order to account for the relative strength of the defeating relationship between arguments.

B.2 Logical analysis of the interactions between arguments

A third way to look at argumentation, and in a way a middle ground between the detailed discourse analysis of Toulmin and the graphical structure of Dung, relies on the analysis of the interactions between arguments. Detailed in (Besnard and Hunter, 2008, Chapter 3) as logical argumentation, it describes arguments as logical formulas (in a language L, usually either propositional or first-order predicate logic) of the form $\langle S, C \rangle$ with C being the claim, and S being the support so that S is the minimal set of propositions which hold both $S \vdash C$ and $S \nvdash \bot$.

The rationale behind this point of view is that it allows for a greater flexibility in describing the relationship between arguments, and more precisely to provide a typology of defeat relationships. The two principal ones are:

- the **undercut**: given two arguments $\langle \Phi, \alpha \rangle$ and $\langle \Psi, \beta \rangle$, $\langle \Psi, \beta \rangle$ undercuts $\langle \Phi, \alpha \rangle$ if β negates one or more propositions of Φ which, due to the necessity of minimality of Φ, defeats the whole argument.
- the **rebuttal**: given two arguments $\langle \Phi, \alpha \rangle$ and $\langle \Psi, \beta \rangle$, $\langle \Psi, \beta \rangle$ is a rebuttal for $\langle \Phi, \alpha \rangle$ if $\beta \leftrightarrow \neg\alpha$ is a tautology.

An undercut for an argument need not be a rebuttal and a rebuttal for an argument need not be an undercut. Both types of defeaters can be split up further according to special circumstances, such as a self-defeating argument, an argument defeating its defeaters or even depending on which part of the support an argument undercuts.

B.3 Pragmatic argumentation theory

Because we need to scale classification to large amounts of data a simple conceptual representation of arguments, such as Pragmatic Argumentation Theory (PAT) van Eemeren (1993), Hutchby (2013), is necessary. Indeed, fitting a complex computational model of argument would be computationally expensive and not fit the colloquial nature of social media content and it is thus deemed preferable to use a less complex model with a higher accuracy.

PAT defines an argument as an opinionated piece of text which can arise in the presence of two elements: (1) a **target**, being some other action by another actor, has been called out as being problematic; (2) a **stance**, i.e. whether it is supporting or attacking the target, has been defined. This simplification of the argumentation representation helps with applying those mappings to social media content, by relaxing the definition of an argument. Indeed, users of social media tend to prefer a highly informal language and as such are less likely to worry about formulating a well-formed argument.

B.4 Impact on social media research

As we have observed in this section, there are several ways to model the process of dialogical argumentation at multiple levels of granularity, and we ought to select our model based on our end goal. Because our goal is to model debates in social media, approaches based on a fine grained analysis of linguistic forms such as Toulmin's model or Besnard's logical argumentation are bound to fail in their coverage of the several ways arguments take place in the wild. We therefore adopt a coarser-grained view of argumentation by adopting a bipolar abstract model for the rest of this thesis. The bipolar augmentation of Dung's abstract argumentation framework allows us to model debates in terms of two essential relationships: the attacking stance and the supporting stance. In order to fit within the larger context of social media analysis, we loosen our bipolar relationship to a more general local agreement stance, meaning that an argument is in general agreement with another argument, and disagreement stance, meaning that an argument is in general disagreement with another argument.

Appendix C

Example of lexicon computed by LEXICNET

This appendix presents a simple, toy lexicon computed on the IAC dataset using LEXICNET and a list of seed words from Hu and Liu (2004) (originally intended for sentiment analysis). We can see from a simple inspection that the regularisation stops the scores from inflating a lot over 1, due to the quadratic cost incurred by L2-norm regularisation. The lexicon was kept to a size of a 100 words in order to illustrate the result in a concise manner.

TABLE C.1: Example toy lexicon computed on the IAC dataset, with seed dictionary from Hu and Liu (2004) (part 1 of 4)

Word	Agreement score	Disagreement score
advocates	1.083	0.284
affordable	1.121	0.293
ambivalence	0.081	1.315
annoyed	0.211	1.126
astoundingly	1.269	0.111
avaricious	0.154	1.246
beautifullly	1.265	0.133
beckoning	1.306	0.095
beleaguer	0.136	1.305
bogus	0.492	0.91
bombardment	0.151	1.258
bountiful	1.476	-0.08
bumpping	0.199	1.204
butcher	0.05	1.333
calumny	0.243	1.164
commend	1.303	0.125
courageous	1.142	0.26
craze	0.131	1.307
crowded	0.089	1.293
cumbersome	0.197	1.203
cuss	0.188	1.24
daringly	1.134	0.281
dark	0.304	1.093
defamation	0.18	1.2
desperation	0.173	1.247
difficulty	0.195	1.233

TABLE C.2: Example toy lexicon computed on the IAC dataset, with seed dictionary from Hu and Liu (2004) (part 2 of 4)

Word	Agreement score	Disagreement score
disappoints	0.354	1.02
dissatisfied	0.124	1.239
distinction	1.106	0.287
doomsday	0.081	1.32
doubtless	1.143	0.275
enhanced	1.291	0.078
enlighten	1.297	0.103
excellant	0.892	0.506
failed	-0.099	1.464
faster	1.207	0.19
filth	0.182	1.225
fresher	1.048	0.328
frugal	1.543	-0.095
glitz	1.24	0.169
gloomy	0.272	1.093
guile	0.34	1.052
hallucination	0.291	1.111
harmonious	0.995	0.413
helpful	1.067	0.342
hostility	0.355	1.045
hysterics	0.209	1.18
imperfections	0.093	1.286
indiscretion	0.15	1.265
inequalities	-0.001	1.411
inescapable	0.146	1.291
inflexible	0.178	1.216

Table C.3: Example toy lexicon computed on the IAC dataset, with seed dictionary from Hu and Liu (2004) (part 3 of 4)

Word	Agreement score	Disagreement score
inflexible	0.178	1.216
infuriating	0.337	1.038
inordinate	0.186	1.218
interrupt	0.261	1.137
invaluable	1.001	0.41
irk	0.318	1.069
irrelevant	0.212	1.178
joyous	1.056	0.356
lambast	0.156	1.283
laudably	1.413	-0.007
leaking	0.326	1.018
lethargy	0.115	1.257
majesty	1.226	0.197
maltreatment	0.316	1.06
meticulous	1.26	0.108
misconception	0.195	1.218
mock	0.055	1.369
muscle-flexing	0.183	1.23
neat	1.319	0.101
nettlesome	0.297	1.118
overthrows	0.231	1.18
pigs	0.259	1.096
pollute	0.294	1.108
reasonable	1.126	0.3
reconciliation	1.233	0.126
refreshing	1.354	0.04
regrets	-0.017	1.406

TABLE C.4: Example toy lexicon computed on the IAC dataset, with seed dictionary from Hu and Liu (2004) (part 4 of 4)

Word	Agreement score	Disagreement score
sagged	0.27	1.133
scratches	0.279	1.138
second-class	0.226	1.199
splatter	0.231	1.146
spoiled	0.279	1.067
steadfast	1.303	0.113
struggle	0.233	1.169
succeed	1.076	0.322
sugarcoated	0.308	1.072
sumptuous	1.178	0.198
superficially	0.111	1.268
tarnish	0.152	1.266
tenaciously	1.231	0.168
treason	0.361	1.053
troublesome	0.398	1.016
unconstitutional	0.217	1.206
undermined	0.023	1.37
uneasiness	0.159	1.233
unlamentable	0.397	1.014
whiny	0.244	1.144
worrisome	0.029	1.37
wowing	1.248	0.109

Bibliography

Raquel Mochales Palau and Marie-Francine Moens. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law*, pages 98–107. ACM, 2009.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

Timothy J Berners-Lee. Information management: A proposal. Technical report, European Organization for Nuclear Research (CERN), 1989.

Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, 2010.

Yoke Yie Chen, Xavier Ferrer, Nirmalie Wiratunga, and Enric Plaza. Sentiment and preference guided social recommendation. In *International Conference on Case-Based Reasoning*, pages 79–94. Springer, 2014.

Jérémie Clos, Anil Bandhakavi, Nirmalie Wiratunga, and Guillaume Cabanac. Predicting emotional reaction in social networks. In *European Conference on Information Retrieval*, pages 527–533. Springer, 2017.

Aminu Muhammad, Nirmalie Wiratunga, Robert Lothian, and Richard Glassey. Domain-based lexicon enhancement for sentiment analysis. In *SMA BCS-SGAI*, pages 7–18, 2013.

Chris Reed and Glenn Rowe. Araucaria: Software for argument analysis, diagramming and representation. *International Journal on Artificial Intelligence Tools*, 13(04):961–979, 2004.

Chris Reed, Mathilde Janier, and John Lawrence. Ova+: An argument analysis interface. In *Computational Models of Argument: Proceedings of COMMA*, volume 266, page 463, 2014.

Floris Bex, John Lawrence, Mark Snaith, and Chris Reed. Implementing the argument web. *Communications of the ACM*, 56(10):66–73, 2013.

Tommi Jaakkola, Marina Meila, and Tony Jebara. Maximum entropy discrimination. In *Advances in neural information processing systems*, pages 470–476, 2000.

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

F.H. van Eemeren. *Reconstructing Argumentative Discourse*. Studies in rhetoric and communication. University of Alabama Press, 1993. ISBN 9780817306977. URL https://books.google.co.uk/books?id=Jg5tAAAAIAAJ.

Ian Hutchby. *Confrontation talk: Arguments, asymmetries, and power on talk radio*. Routledge, 2013.

Lu Wang and Claire Cardie. Improving agreement and disagreement identification in online discussions with a socially-tuned sentiment lexicon. *ACL 2014*, page 97, 2014.

Ahmed Hassan, Vahed Qazvinian, and Dragomir Radev. What's with the attitude?: identifying sentences with attitude in online discussions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1245–1255. Association for Computational Linguistics, 2010.

Subrata Ghosh, Konjengbam Anand, Sailaja Rajanala, A Bharath Reddy, and Manish Singh. Unsupervised stance classification in online debates. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, pages 30–36. ACM, 2018.

Swapna Somasundaran and Janyce Wiebe. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10, pages 116–124, Stroudsburg, PA, USA, 2010a. Association for Computational Linguistics.

Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

Philip J Stone, Dexter C Dunphy, and Marshall S Smith. *The general inquirer: A computer approach to content analysis*. MIT press, 1966.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics, 2005.

Vasileios Hatzivassiloglou and Kathleen R McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*, pages 174–181. Association for Computational Linguistics, 1997.

Ellen Riloff and Janyce Wiebe. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 105–112. Association for Computational Linguistics, 2003.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204, 2010.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

Aminu Muhammad, Nirmalie Wiratunga, and Robert Lothian. Contextual sentiment analysis for social media genres. *Knowledge-Based Systems*, 108:92–101, 2016.

Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, et al. The icsi meeting corpus. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–I. IEEE, 2003.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. The ami meeting corpus: A pre-announcement. In *International Workshop on Machine Learning for Multimodal Interaction*, pages 28–39. Springer, 2005.

Iain McCowan, Jean Carletta, W Kraaij, S Ashby, S Bourban, M Flynn, M Guillemot, T Hain, J Kadlec, V Karaiskos, et al. The ami meeting corpus. In *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*, volume 88, 2005.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiao-Dan Zhu, and Colin Cherry. Semeval-2016 task 6: Detecting stance in tweets. In *SemEval NAACL-HLT*, pages 31–41, 2016.

Ruifeng Xu, Yu Zhou, Dongyin Wu, Lin Gui, Jiachen Du, and Yun Xue. Overview of nlpcc shared task 4: Stance detection in chinese microblogs. In *International Conference on Computer Processing of Oriental Languages*, pages 907–916. Springer, 2016.

Marilyn A Walker, Jean E Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. A corpus for research on deliberation and debate. In *LREC*, pages 812–817, 2012a.

William Ferreira and Andreas Vlachos. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL, 2016.

Sara Rosenthal and Kathy McKeown. I couldn't agree more: The role of conversational structure in agreement and disagreement detection in online discussions. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 168–177, 2015.

Pranav Anand, Marilyn Walker, Rob Abbott, Jean E Fox Tree, Robeson Bowmani, and Michael Minor. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the 2nd workshop on computational approaches to subjectivity and sentiment analysis*, pages 1–9. Association for Computational Linguistics, 2011.

Marilyn A. Walker, Pranav Anand, Robert Abbott, and Ricky Grant. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 592–596, Stroudsburg, PA, USA, 2012b. Association for Computational Linguistics. ISBN 978-1-937284-20-6.

Adam Faulkner. Automated classification of stance in student essays: An approach using stance target information and the wikipedia link-based measure. *Science*, 376(12):86, 2014.

Amr Ahmed and Eric P Xing. Staying informed: supervised and semi-supervised multi-view topical analysis of ideological perspective. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1140–1150. Association for Computational Linguistics, 2010.

Kazi Saidul Hasan and Vincent Ng. Stance classification of ideological debates: Data, models, features, and constraints. In *IJCNLP*, pages 1348–1356, 2013a.

Zhihua Zhang and Man Lan. Ecnu at semeval 2016 task 6: relevant or not? supportive or not? a two-step learning system for automatic detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 451–457, 2016.

Martin Tutek, Ivan Sekulic, Paula Gombar, Ivan Paljak, Filip Culinovic, Filip Boltuzic, Mladen Karan, Domagoj Alagić, and Jan Šnajder. Takelab at semeval-2016 task 6: stance classification in tweets using a genetic algorithm based ensemble. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 464–468, 2016.

Peter Krejzl and Josef Steinberger. Uwb at semeval-2016 task 6: stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 408–412, 2016.

Heba Elfardy and Mona Diab. Cu-gwu perspective at semeval-2016 task 6: Ideological stance detection in informal text. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 434–439, 2016.

Michael Wojatzki and Torsten Zesch. ltl. uni-due at semeval-2016 task 6: Stance detection in social media using stacked classifiers. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 428–433, 2016a.

Yuki Igarashi, Hiroya Komatsu, Sosuke Kobayashi, Naoaki Okazaki, and Kentaro Inui. Tohoku at semeval-2016 task 6: Feature-based model versus convolutional neural network for stance detection. In *SemEval@NAACL-HLT*, 2016.

Henrik Bøhler, Petter Asla, Erwin Marsi, and Rune Sætre. Idi @ ntnu at semeval-2016 task 6: Detecting stance in tweets using shallow features and glove vectors for word representation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 445–450, 2016.

Dustin Hillard, Mari Ostendorf, and Elizabeth Shriberg. Detection of agreement vs. disagreement in meetings: Training with unlabeled data. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pages 34–36. Association for Computational Linguistics, 2003.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. Automatic detection of arguments in legal texts. In *Proceedings of the 11th international conference on Artificial intelligence and law*, pages 225–230. ACM, 2007.

Sebastian Germesin and Theresa Wilson. Agreement detection in multiparty conversation. In *Proceedings of the 2009 international conference on Multimodal interfaces*, pages 7–14. ACM, 2009.

Wen Wang, Sibel Yaman, Kristin Precoda, Colleen Richey, and Geoffrey Raymond. Detection of agreement and disagreement in broadcast conversations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 374–378, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-88-6.

Rob Abbott, Marilyn Walker, Pranav Anand, Jean E. Fox Tree, Robeson Bowmani, and Joseph King. How can you say such things?!?: Recognizing disagreement in informal political argument. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 2–11, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-96-1.

Ahmed Hassan, Amjad Abu-Jbara, and Dragomir Radev. Detecting subgroups in online discussions by modeling positive and negative relations among participants. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 59–70. Association for Computational Linguistics, 2012.

Jie Yin, Paul Thomas, Nalin Narang, and Cecile Paris. Unifying local and global agreement and disagreement classification in online debates. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA '12, pages 61–69, Stroudsburg, PA, USA, 2012a. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. Extra-linguistic constraints on stance recognition in ideological debates. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 816–821, 2013b.

Amita Misra and Marilyn Walker. Topic independent identification of agreement and disagreement in social media dialogue. *arXiv preprint arXiv:1709.00661*, 2017.

Marilyn A. Walker, Pranav Anand, Rob Abbott, Jean E. Fox Tree, Craig H. Martell, and Joseph King. That is your evidence?: Classifying stance in online political debate. *Decision Support Systems*, 53:719–729, 2012c.

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. Joint models of disagreement and stance in online debate. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 116–125, 2015.

Qingying Sun, Zhongqing Wang, Qiaoming Zhu, and Guodong Zhou. Exploring various linguistic features for stance detection. In *Natural Language Understanding and Intelligent Applications*, pages 840–847. Springer, 2016.

Can Liu, Wen Li, Bradford Demarest, Yue Chen, Sara Couture, Daniel Dakota, Nikita Haduong, Noah Kaufman, Andrew Lamont, Manan Pancholi, et al. Iucl at semeval-2016 task 6: An ensemble model for stance detection in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 394–400, 2016.

Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay. Ju_nlp at semeval-2016 task 6: detecting stance in tweets using support vector machines. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 440–444, 2016.

Amita Misra, Brian Ecker, Theodore Handleman, Nicolas Hahn, and Marilyn A. Walker. Nlds-ucsc at semeval-2016 task 6: A semi-supervised approach to detecting stance in tweets. In *SemEval@NAACL-HLT*, 2016.

Linchuan Li, Zhiyong Wu, Mingxing Xu, Helen M. Meng, and Lianhong Cai. Recognizing stances in mandarin social ideological debates with text and acoustic features. *2016 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 1–6, 2016.

Saif M Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26, 2017.

Sangyun Hahn, Richard Ladner, and Mari Ostendorf. Agreement/disagreement classification: Exploiting unlabeled data using contrast classifiers. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 53–56. Association for Computational Linguistics, 2006.

Michel Galley, Kathleen McKeown, Julia Hirschberg, and Elizabeth Shriberg. Identifying agreement and disagreement in conversational speech: Use of bayesian networks to model pragmatic dependencies. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 669. Association for Computational Linguistics, 2004.

Danqi Chen and Christopher D. Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, 2014.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013a.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

Dhanya Sridhar, Lise Getoor, and Marilyn Walker. Collective stance classification of posts in online debate forums. In *Proceedings of the Joint Workshop on Social Dynamics and Personal Attributes in Social Media*, pages 109–117, 2014.

Filip Boltuzic and Jan Šnajder. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58, 2014.

Parinaz Sobhani, Diana Inkpen, and Stan Matwin. From argumentation mining to stance classification. *NAACL HLT 2015*, page 67, 2015.

Marcelo Dias and Karin Becker. Inf-ufrgs-opinion-mining at semeval-2016 task 6: Automatic generation of a training corpus for unsupervised identification of stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 378–383, 2016.

Miha Grčar, Darko Cherepnalkoski, Igor Mozetič, and Petra Kralj Novak. Stance and influence of twitter users regarding the brexit referendum. *Computational social networks*, 4(1):6, 2017.

Lu Wang and Claire Cardie. Improving agreement and disagreement identification in online discussions with a socially-tuned sentiment lexicon. *arXiv preprint arXiv:1606.05706*, 2016.

Jie Yin, Paul Thomas, Nalin Narang, and Cecile Paris. Unifying local and global agreement and disagreement classification in online debates. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, pages 61–69. ACL, 2012b.

Isabelle Augenstein, Andreas Vlachos, and Kalina Bontcheva. Usfd at semeval-2016 task 6: Any-target stance detection on twitter with autoencoders. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 389–393, 2016a.

Michael Wojatzki and Torsten Zesch. Stance-based argument mining–modeling implicit argumentation using stance. *Proceedings of the KONVENS, Bochum, Germany*, pages 313–322, 2016b.

Sara S Mourad, Doaa M Shawky, Hatem A Fayed, and Ashraf H Badawi. Stance detection in tweets using a majority vote classifier. In *International Conference on Advanced Machine Learning Technologies and Applications*, pages 375–384. Springer, 2018.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, Michal Lukasik, Kalina Bontcheva, Trevor Cohn, and Isabelle Augenstein. Discourse-aware rumour stance classification in social media using sequential classifiers. *Information Processing & Management*, 54(2):273–290, 2018.

Shaodian Zhang, Lin Qiu, Frank Chen, Weinan Zhang, Yong Yu, and Noémie Elhadad. We make choices we think are going to save us: Debate and stance identification for online breast cancer cam discussions. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 1073–1081. International World Wide Web Conferences Steering Committee, 2017.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. Stance detection with bidirectional conditional encoding. *arXiv preprint arXiv:1606.05464*, 2016b.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. Stance classification with target-specific neural attention networks. In *Proceedings of the International Joint Conferences on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence, 2017.

Sushant Hiray and Venkatesh Duppada. Agree to disagree: Improving disagreement detection with dual grus. In *Proceedings of the Seventh International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*. Seventh International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW), 2017.

Nan Yu, Da Pan, Meishan Zhang, and Guohong Fu. Stance detection in chinese microblogs with neural networks. In *Natural Language Understanding and Intelligent Applications*, pages 893–900. Springer, 2016.

Prashanth Vijayaraghavan, Ivan Sysoev, Soroush Vosoughi, and Deb Roy. Deepstance at semeval-2016 task 6: Detecting stance in tweets using character and word-level cnns. *CoRR*, abs/1606.05694, 2016. URL http://arxiv.org/abs/1606.05694.

Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. pkudblab at semeval-2016 task 6 : A specific convolutional neural network system for effective stance detection. In *SemEval@NAACL-HLT*, 2016.

Guido Zarrella and Amy Marsh. Mitre at semeval-2016 task 6: Transfer learning for stance detection. *arXiv preprint arXiv:1606.03784*, 2016.

Jacob Andreas, Sara Rosenthal, and Kathleen McKeown. Annotating agreement and disagreement in threaded discussion. In *LREC*, pages 818–822, 2012.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research*, 8(Oct):2265–2295, 2007.

Rémi Lebret and Ronan Collobert. Word emdeddings through hellinger pca. *arXiv preprint arXiv:1312.5542*, 2013.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

Yitan Li, Linli Xu, Fei Tian, Liang Jiang, Xiaowei Zhong, and Enhong Chen. Word embedding revisited: A new representation learning and explicit matrix factorization perspective. In *IJCAI*, pages 3650–3656, 2015.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters*, 80:150–156, 2016.

Anil Bandhakavi, Nirmalie Wiratunga, P Deepak, and Stewart Massie. Lexicon based feature extraction for emotion text classification. *Pattern Recognition Letters*, 2016.

James W Pennebaker, Martha E Francis, and Roger J Booth. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71:2001, 2001.

Aminu Muhammad, Nirmalie Wiratunga, and Robert Lothian. A hybrid sentiment lexicon for social media mining. In *Tools with AI (ICTAI), IEEE 26th International Conf. on*, pages 461–468, 2014.

Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, pages 417–422. Citeseer, 2006.

Anil Bandhakavi, Nirmalie Wiratunga, P Deepak, and Stewart Massie. Generating a word-emotion lexicon from# emotional tweets. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (* SEM 2014)*, pages 12–21, 2014.

Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on ACL*. ACL, 2002.

Jérémie Clos, Nirmalie Wiratunga, Stewart Massie, and Guillaume Cabanac. Shallow techniques for argument mining. In *ECA'15: Proceedings of the ECA*, volume 63, page 2, 2016.

Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

Harold W Kuhn and Albert W Tucker. Nonlinear programming, in (j. neyman, ed.) proceedings of the second berkeley symposium on mathematical statistics and probability, 1951.

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Michael C Mozer. A focused backpropagation algorithm for temporal. *Backpropagation: Theory, architectures, and applications*, page 137, 1995.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier, 1997.

William Yang Wang, Dan Bohus, Ece Kamar, and Eric Horvitz. Crowdsourcing the acquisition of natural language corpora: Methods and observations. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 73–78. IEEE, 2012.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics, 2008.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. URL https://www.R-project.org.

Livia Polanyi and Annie Zaenen. Contextual valence shifters. In *Computing attitude and affect in text: Theory and applications*, pages 1–10. Springer, 2006.

Tim Miller. Explanation in artificial intelligence: insights from the social sciences. *arXiv preprint arXiv:1706.07269*, 2017.

Zachary C Lipton. The mythos of model interpretability. *Queue*, 16(3):30, 2018.

David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

Yurii Nesterov. Gradient methods for minimizing composite objective function, 2007.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

J. Ross Quinlan. Learning logical definitions from relations. *Machine learning*, 5(3):239–266, 1990.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct): 2825–2830, 2011.

Swapna Somasundaran and Janyce Wiebe. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124. Association for Computational Linguistics, 2010b.

Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Stephen E. Toulmin. *The Uses of Argument*. Cambridge University Press, July 2003. ISBN 0521534836. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike-20&path=ASIN/0521534836.

Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

Leila Amgoud, Claudette Cayrol, Marie Christine Lagasquie-Schiex, and Pierre Livet. On bipolarity in argumentation frameworks. *Int. J. Intell. Syst.*, 23(10):1062–1093, October 2008. ISSN 0884-8173. doi: 10.1002/int.v23:10.

C. Cayrol and M. C. Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In *Proceedings of the 8th European conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, ECSQARU'05, pages 378–389, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-27326-3, 978-3-540-27326-4. doi: 10.1007/11518655_33. URL http://dx.doi.org/10.1007/11518655_33.

Nikos Karacapilidis and Dimitris Papadias. Computer supported argumentation and collaborative decision making: The hermes system. *Information Systems*, 26:259–277, 2001.

Bart Verheij. On the existence and multiplicity of extensions in dialectical argumentation. *CoRR*, cs.AI/0207067, 2002.

Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. The MIT Press, 2008. ISBN 0262026430, 9780262026437.

J. Janssen, M. De Cock, and D. Vermeir. Fuzzy argumentation frameworks. In *Information Processing and Management of Uncertainty in Knowledge-based Systems*, pages 513–520, 2008.