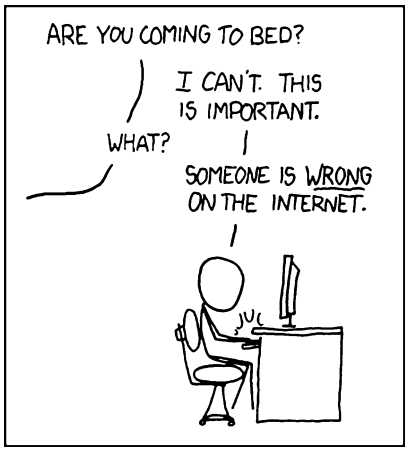
UiO : Department of Informatics
University of Oslo

Syntax for fine-grained opinion analysis

Trond Thorbjørnsen

Master's Thesis Autumn 2016


ARE YOU COMING TO BED?
I CAN'T. THIS IS IMPORTANT.
WHAT?
SOMEONE IS WRONG ON THE INTERNET.

Previous page: xkcd, Duty Calls (`https://xkcd.com/386/`)

Acknowledgements

Lilja Øvrelid and *Stephan Oepen*: I am truly grateful for the dedicated supervision through my masters project. Thank you for your patience and your open doors.

Angelina Ivanova: thank you for the work on comparing dependency representations and the parser models you have provided. I am also thankful to *Richard Johansson* for his work on the topic of opinion analysis, and the extensive replies he has given on my questions.

To all current and former members of the Language Technology Group at the Department of Informatics: It has been a privilege to be a master student in this group because of you all. Some special thanks: *Emanuele Lapponi*: your enthusiasm for language technology and your inclusiveness is exceptional. Thank you for being there in stressful times. *Milen Kouylekov*: thank you for acting as a shadow supervisor and for your feedback on this thesis. I would also like to emphasise the concrete support *Elisabeth Lien*, *Arne Skjærholt*, *Murhaf Fares* and *Pierre Lison* have given me on several topics.

Joakim, John, Sindre, Tomas, and all other fellow students during my studies: Thank you for the learning environment you have build. A special thank you to *Joakim Pedersen Berg* for the excellent proofreading.

I am also very grateful for my supportive family and friends. Thank you all. To *Solve*: thank you for asking me the right questions on the topic of statistics. To my baby daughter *Elise*: thanks for all your laughter and smile.

And finally, to my wonderful wife, *Anke*, my gratitude for your support, encouragement and love is boundless.

Trond Thorbjørnsen
Oslo, Norway
August 2016

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In our daily decision making, we often rely on the opinions of others. This is apparent when it comes to our purchasing habits, for example. Traditionally we get opinions about products from friends, professional reviews and guidance in speciality stores, to name but a few. But the development of user-generated content and social media has added an extra level to this. We find opinions on products in many places: on personal blogs, Facebook and product comparison pages. This applies not only to physical products for sale, but to films, concerts, restaurants, hotels, etc.

The growth of user-generated content on the internet has, according to Liu (2012), for the first time in human history generated a large amount of opinionated data. These sources of data have been one of the reasons for the increase in research on *opinion mining* or *sentiment analysis* over the last decade. The field consists of a multitude of different tasks.

A common task in this context is *document level analysis*, where the sentiment of a document as a whole is predicted. An important data source for this has been the *Movie Review* corpus by Pang, Lee, and Vaithyanathan (2002), which contains film reviews organised in two folder depending on their polarity. This corpus has been used for numerous experiments, improving performance in this task.

Another task in the field, is what we call *fine-grained opinion analysis*. This involves, among other things, detecting opinionated subparts of a text and linking targets of opinion and opinion holders together. Most longer texts involve both positive and negative sentiments. While document level opinion analysis works for analysing product reviews, it is of limited value for general articles and blog posts. For this, we need a deeper analysis of the written text, in order to extract the opinionated information.

THESIS

The state of the art in fine-grained opinion analysis is under continuous development. Previous work, e.g. Johansson and Moschitti (2013), has shown that syntactic analysis constitutes an important source of information for fine-grained opinion analysis. How can we take further advantage of syntactic and semantic information for opinion analysis?

Our goal with this thesis is to investigate how the choice of syntactic representation influences the performance of a system for fine-grained opinion analysis. We will limit our investigation to one subtask which depends largely upon grammatical analysis: opinion holder classification.

We will create an experimental setup for opinion holder analysis, letting us compare the results using different representations for syntactic information, more specifically on the type of dependency representation. The source code for this project, is available on this project's GitHub page.[1]

RESEARCH QUESTIONS

The research questions we want to answer in this work are:

- Which design and architectural choices are suitable to construct a flexible environment for experimentation with the MPQA dataset and a variety of linguistic analysis tools and target representations?
- To what degree can the empirical results of Johansson and Moschitti (2013) be replicated with an abstractly equivalent, but technically slightly different ensemble of syntactic-semantic pre-processors and machine learning techniques?
- What quantitative and qualitative effects can be observed on the sub-task of opinion holder classification when using different types of interface representations to syntactic analysis?

CHAPTER OVERVIEW

Background
In Chapter 2, we will provide the necessary background for the project. We will first give an overview of the field of opinion analysis, then we will describe the tasks involved in fine-grained opinion analysis. Further, we will review two previous works in this field by Choi, Breck, and Cardie (2006) and by Johansson and Moschitti (2013). The last part of the background chapter is about grammatical

[1] The project's GitHub page: `https://github.com/trondth/master`

analysis. We will describe the two major representations for syntactic analysis, constituent and dependency representation, before we go into different dependency representations. Finally, we will discuss how differences in representation may affect a task like opinion analysis.

Prerequisites: Experimental infrastructure
To build up our experimental setup, we rely on several third party tools and data sources. In Chapter 3, we will describe the tools and data sources used in this work. First, we will provide detailed information about the opinion analysis corpus we decided to use, MPQA (Wiebe, Wilson, and Cardie 2005; Wilson 2008, chapter 7).[2] Then, we will describe the lexicon we use for tagging words with prior polarity. Further, we will describe the third-party tools for pre-processing, syntactic parsing, and machine learning. Finally, we will describe our evaluation metrics.

Opinion expression detection
To identify opinions in text, use sequence labelling to detect each opinion expression in a text. In Chapter 4, we give a detailed description of our system. For this task, we perform systematic feature selection experiments. We also give the results from the held-out test set for this part of the system, and compare the results to Johansson and Moschitti (2013).

Opinion Holder Classification
This constitutes the main part of our experimental work, where we perform a comparison of dependency representations. In Chapter 5, we give a detailed description of our construction of this part of the system, including discussions about important choices we made during the setup of this task. We will examine both feature statistics and development results in the light of the choice of syntactic representation, before going into a error analysis, where we will present statistics on errors and discuss some examples. Finally, we will present the results from running our system on held-out data.

Conclusion
In Chapter 6, we will sum up the contributions of this work and consider possibilities for future research.

[2] MPQA Corpus: `http://mpqa.cs.pitt.edu`

Chapter 2

Background

There are different levels of opinion analysis. A document reviewing a single product can be classified as positive or negative to this product. At the sentence level, we can detect opinion expressions and link each expression to targets and holders.

Extracting opinions from a text at the sentence level can be done in several ways, but common approaches rely on finding opinion or sentiment words. In a sentence like 'This hotel is great', the word 'great' indicates that this sentence contains an opinion. There are a lot of words like that, but most do not necessarily function as opinion word in all cases. 'This hotel is great grandfather's' is not an opinionated sentence, for example. Nevertheless, in many cases, a shallow analyser of the opinion word, the distance to the candidate targets and a way to treat negation words, will give fairly good results.

The main approaches for automated sentiment analysis have indeed relied on fairly simple methods, not considering the syntactic structure of the sentence. The most common sources of information used are word distance and token level part-of-speech tags.

In this master's project, we will create a setup that allows one to experimentally determine and contrast the contributions of different types of syntactic analysis to opinion analysis. We will in this chapter explain some of the algorithms and look at some of the features that have been investigated and provide the reader with necessary background. We will start out by giving an overview of opinion analysis subtasks, then we will examine two previous works before we give some background for grammatical analysis and different dependency representations.

2.1 OVERVIEW OF TASKS IN OPINION ANALYSIS

In this section, we will provide a high-level overview of the tasks of opinion analysis, broken down into subtasks. These will be illustrated by two running examples:

(1) a. The bathroom was clean according to my husband.

b. Google isn't universally loved by newspaper execs and other content providers.

Example (1a) is obtained from a store review, of the store Dollar Tree.[1] and example (1b) is taken from a blog[2]. While example (1a) has a straightforward syntactic structure, example (1b) is more complex. Passive, negation and a conjoined noun phrase as the agent make this example interesting.

We will use the term *target* as a common term for targets of expressions of opinion. Other terms referring to opinion targets commonly found in the literature are *entity* (Liu 2012) and *topic* (Li et al. 2010) (Kim and Hovy 2006).

Often, an opinion is not stated for a target as a whole, but for a certain part and a certain attribute. The sentence: 'The bathroom was clean' is not an opinion about the location in itself, but on an attribute, the cleanliness of its *bathroom*.

Liu (2012) defines an opinion as a quintuple: $\langle e, a, s, h, t \rangle$, where e is the target entity, a is an aspect of this target, s is the sentiment on the aspect, h is the opinion holder and t is the time.

What triggers an opinion, is a linguistic element, for which we will use the term *opinion expression*. In example (1a), the phrase 'clean' is such an expression. In example (1b), the trigger will be the word 'loved'.

This definition holds for a *regular opinion*, concerning a single target, not other types of opinions, for example comparative opinions between several targets. We will concentrate on regular opinions.

Now, we need to understand how we can extract opinion quintuples from a text. Liu lists six main tasks involved in this analysis (Liu 2012).

1. target entity extraction and categorisation
2. aspect extraction and categorisation
3. opinion holder extraction and categorisation
4. time extraction and standardisation
5. aspect sentiment classification
6. opinion quintuple generation

Under the discussion of these steps, some basic preprocessing of the text is assumed:

Lemmatisation Tokenising and normalising each word to their stem form.

Part-of-speech (POS) Part-of-speech tagging of the text.

Named entity recognition We will come back to this in Section 2.1.1.

[1] `http://www.yelp.com/biz/dollar-tree-universal-city`

[2] *The writer underground*: `http://writerunderground.com/2009/09/14/google-to-struggling-newspapers-have-you-guys-ever-tried-porn/`

Several of the steps rely on a subjectivity or opinion words lexicon. There are several available resources, some of which we will discuss further in Chapter 3. In example (1a), the opinion word is *clean* and in example (1b), *loved*.

2.1.1 Target extraction and categorisation

This is a core task in all opinion analysis. The target of the opinion has to be identified. Words representing a target are found and grouped together.

This can be done by a lexical approach, especially for smaller domains. Often, we are interested in a small set of targets. A phone company would like to know opinions about phones, and in such a case, a list of targets could be created by hand. Targets could be represented in different forms. A target appearing as a pronoun or with a nickname, should be normalised to a base form.

The task of finding these potential targets is part of a classic problem: named entity recognition. Named targets could be extracted by their part-of-speech tags or by their syntactic functions, for example subject or objects.

Li et al. (2010), who use the term *entity* for opinion targets, which can be confusing, propose two expansion methods in the task of detecting synonyms or nicknames for targets, a dictionary method that uses Wikipedia to find entry pages for a target and a web-based method that searches for a target and top-m terms that correlate with the target from the top-n documents.

In reviews, the target is also often a part of meta information, and in the text, references only occur to aspects of the target. In example (1a), the target is the name of the store, *Dollar Tree*, which is not mentioned at all in this review. For example (1b), *Google* is the target.

2.1.2 Aspect extraction and categorisation

Since opinions are frequently directed at a certain aspect of a target, this step is necessary.

The sentence 'Google has a great mail application for cell phones.' is a positive opinion on one aspect of Google: its mail application. We need to extract and categorise the aspects not only when we are interested in different aspects, but also when we are searching for sentiments as a whole. There have been several approaches to this subtask.

Nouns and noun phrases The aspects are extracted by looking at the frequency of nouns and noun phrases in a large text for the given domain. Highly frequent nouns are likely to be an aspect of opinions (Hu and Liu 2004). In the domain of hotel reviews, for example, words like *room, bed* and *breakfast* are very frequent.

Opinion and target relations If a sentence contains opinion words, but no high-frequency nouns or noun phrases, we can seek to determine which noun phrase these opinion words relate to. Hu and Liu (2004) picked the nearest

neighbour, but dependency parsing has also been used for solving this.

Supervised learning With previously annotated data, we can train a model for machine learning on the basis of different features. These features can, for example represent the context of the word or they can show grammatical properties, like syntactic relation between opinion expression and aspect.

In example (1a) *bathroom* is an aspect of the target, the store on which the review is made. In example (1b), a system can find several possible aspects. We can consider the NPs in the sentence. The three NPs with shortest distance to the opinion word are: *Google*, *newspaper execs and content providers* and *newspaper execs*. For *newspaper execs*, *execs* could be the aspect of the target *newspaper*. For the target *Google*, the aspect is Google in general, the target would then be ⟨Google, GENERAL⟩. In the case of multiple targets or aspects, we can generate multiple opinion tuples.

The aspects are organised in a similar way as the targets. The word *execs*, could represent the term CEO. We can at this point generate an n-best list of potential target-aspect pairs:

- ⟨newspapers, CEO⟩
- ⟨newspaperexecs, GENERAL⟩
- ⟨Google, GENERAL⟩
- ⟨contentproviders, GENERAL⟩

For both aspects and opinions, there is a problem with pronouns referencing outside the sentence. The *coreference resolution* problem is a traditional problem in language technology. Several approaches have been proposed (Soon, Ng, and Lim 2001; McCarthy and Lehnert 1995) and coreference resolution was also the topic for the 2011 CONLL Shared Task (Pradhan et al. 2011). In terms of opinion analysis, coreference resolution has been investigated by Ding and Liu (2010), who builds upon traditional approaches with supervised learning, but proposes additional features for this domain.

2.1.3 Opinion holder and time extraction

Often, the opinion holder is the author of the text and can be extracted directly from metadata. The same goes for the time extraction.

But the opinion holder may equally often be a person or another named entity explicitly mentioned in the text. In such cases, similar methods to aspect extraction can be used. As a named entity, an opinion holder would normally appear as a noun phrase. An extraction of explicit opinion holders could thus be limited to the noun phrases.

Finally, an opinion holder is not necessarily the writer or an entity explicitly mentioned in the text. Such implicit opinion holders often appear for expressions with passive voice.

A system may need to first utilise linguistic features to decide if the text refers

to opinion holders or a time for the opinion, and if no such reference is made in the text, the system can extract the holder and time from the document with extra-linguistic methods.

If we return to our running example, we find that in example (1a), the opinion holder is the husband of the writer and the time can be extracted from meta-information provided as part of the HTML document header. example (1b) is a reported opinion were the writer refers to other opinion holders. The opinion holders are a coordination of two different entities, *newspaper execs* and *content providers*. This could be solved by creating an opinion tuple for each opinion holder. There is no reference to the time of the opinion, and this could either be registered as unknown or with the time when the opinion is reported, i.e the publication date for the article.

In this thesis, the linking of opinion expressions and holders is the main task. We will investigate how the choice of syntactic representation influences this task in Chapter 5.

2.1.4 *Aspect sentiment classification*

The opinion of a certain aspect is commonly classified as positive, negative or neutral. For every sentiment expression, we need to determine which aspects are covered by this expression. According to Liu (2012), there have been two main approaches for this task, *supervised learning* and lexicon-based techniques.

The features used for supervised learning can be extracted with parsing, i.e. syntactic analysis. Since a supervised learning method is based on training data, it performs best on a limited domain. The approach may not perform well across domains (where domain variation was not represented in the training data).

A lexical approach often can perform better across domains, according to Liu (2012). Central to this approach is a sentiment lexicon, a list of sentiment words and phrases. Sentences containing a sentiment word or phrase are marked, and relations from the sentiment word to the target/aspect and the opinion holder are extracted. In addition, a lexicon-based approach can use compositional semantics and rules of opinion. Rules of opinion can be written in a form similar to context-free grammars.

An example of a lexicon-based method is from Ding, Liu, and Yu (2008). First, the sentiment words and phrases are marked with a sentiment score of $+1$ or -1. Second, so-called sentiment shifters are applied. If a negation appears, the sentiment score is inverted. Then, 'but'-clauses are handled. The next step is to aggregate the sentiment score, where each aspect in the sentence should get a sentiment score. There are many different aggregation methods. Liu (2012) gives the aspects a score based on the distance to the sentiment words.

In example (1a), we have the opinion word *clean*. This is marked with $+1$. There is no negation or 'but'-clause in this sentence, and with the aggregate function described by Liu (2012), *bathroom* is given a sentiment score of $1/2$. The sen-

timent word in example (1b) is *loved*. This positive sentiment is inverted because of the negation in *isn't*. The distance to *Google* (the general aspect of the target entity Google) gives a sentiment score of $-1/3$.

2.1.5 *Opinion quintuple generation*

The last step is to generate opinion quintuples for each opinion in the sentence. For each aspect, one tuple is generated for each opinion holder. For example (1a), 'The bathroom was clean according to my husband', this tuple is created:

⟨Dollar Tree, bathroom, positive, my husband, 2013-10-30⟩

The target entity, *Dollar Tree*, and the date is a part of the meta information. The rest of the information is available from the text itself, although the opinion holder is not completely identified. We can also retrieve the name of the author from the metadata, and a more specific source would be 'the husband of Able L.'

For example (1b), also the target is an explicit part of the sentence. The opinion holder 'newspaper execs and other content providers' is a coordination, and one possibility is to create one tuple for each of these opinion holders.

⟨Google, GENERAL, negative, newspaper execs, 2012-09-08⟩

⟨Google, GENERAL, negative, other content providers, 2012-09-08⟩

2.2 PREVIOUS WORK ON OPINION ANALYSIS

Fairly good results in sentiment analysis have been obtained with very few features. In addition to word counting and word distance, part-of-speech tags are a quite common feature. This is already described in Section 2.1.

The research question for this master project is the question of how syntactic (and semantic) information affects opinion analysis. In the following sections, we will describe two important works that address this question.

2.2.1 *Choi, Breck and Cardie*

An early seminal work in this regard is Choi, Breck, and Cardie (2006). In the following section, we will present this work with a focus on their use of syntactic and semantic information.

Choi, Breck, and Cardie (2006) present an approach for simultaneous extraction of opinion expressions and opinion holders[3], and linking the relations between these. Their system, that crucially depends on syntactic and semantic information, shows a substantial improvement of performance compared to prior results.

Compared to the tasks described in Section 2.1, the system is less fine-grained. Different aspects of a target are not considered. For an *opinion expression* O_i and

[3]Choi, Breck, and Cardie (2006) use the term sources of opinion.

an opinion holder (source entity) S_j in a sentence, the system aims to find a relation $L_{i,j}$, where S_j expresses O_i.

If we consider our two example sentences , we would expect the following:

(2) [The bathroom](1) was *clean*(2) according to [my husband](2).

(3) [Google](3) isn't universally *loved*(4,5) by [newspaper execs](4) and [other content providers](5).

The superscripted numbers show relations from an opinion holder (opinion holders and targets in square brackets) to an opinion phrase (emphasised). As the example shows, a single opinion phrase could have more than one relation.

Choi, Breck, and Cardie (2006) developed two sequence-tagging classifiers for the extraction of opinion expressions and holders and a binary classifier to identify pairwise link relations using the MPQA 1.2 corpus, a predecessor to the corpus we will use in this master's project. For finding the optimal set of relations in each sentence, the authors use Integer Linear Programming. Besides both phrase structure and dependency syntax, their two baseline methods further use a semantic role labelling (SRL) system, and it is also described how SRL can be incorporated in the system. The experiments show that their approach improves both relation and opinion holder classification compared to previous results.

2.2.1.1 *Opinion expression extraction*

The classifiers used for opinion expression extraction are developed by linear-chain Conditional Random Fields (CRF). The segments are tagged with an *IOB-scheme*. In this format, there is one token per line, with a tab or space delimited list of observations for each target and, at the end of each line, a label which marks the beginning, the inside and the outside of a sequence. This format is described in details in Section 4.2.1. Choi, Breck, and Cardie (2006) use the same features for both the opinion holder and opinion expression classifiers. The CRF learns the feature weights for each sub-task. In their system, the following features are used for each token x_i:

word window The four words preceding and following x_i.

POS window The part-of-speech tags for the two preceding and following tokens.

grammatical role The grammatical roles obtained by converting the output from the parser of Collins (1997)[4] to a dependency tree.

dictionary Whether x_i is in the opinion expression dictionary. This is also computed for the neighbouring tokens and for x_i's parent 'chunk' in the dependency parse. The dictionary is selected from training data and in addition includes 500 opinion words from the MPQA Final Report. We will return to MPQA in Section 3.

[4]Collins: `ftp://ftp.cis.upenn.edu/pub/mcollins/PARSER.tar.gz`

semantic class The semantic role of x_i, obtained by using the Sundance parser. [5]
WordNet hypernym The superordinate of x_i in the lexical database WordNet.

2.2.1.2 Relation classification

Choi, Breck, and Cardie (2006) developed a maximum entropy binary classifier for classification of the relation between an opinion and a holder. For an opinion-holder pair, this classifier decides whether there is a relation between the pair. The opinion expressions and the opinion holder entities are taken from the n-best sequences of the entity extraction described in Section 2.2.1.1. The classifier does not consider whether the proposed entities are correct.

A training and a test instance are created for each possible opinion-holder pair O_i, S_j, where the spans of O_i and S_j do not overlap.

During training, instances where neither the opinion O_i nor opinion holder S_j overlap with the gold standard are filtered out. The features for each instance of a potential link $L_{i,j}$ are:

opinion entity word The words that O_i consists of.
phrase type The syntactic category. Different features are used for O_i and S_j.
grammatical role As described for the entity extraction.
position If S_j precedes O_i or not.
distance The token distance between O_i and S_j, categorised in four groups.
dependency path The path between O_i and S_j in a dependency tree.
voice A boolean stating if the opinion has active voice or not i.e. if the subject is the agent in the sentence.
syntactic frame Various relations between O_i and S_j. Choi, Breck, and Cardie (2006) use several syntactic frames compounded of various types of information, such as grammatical role, phrase type, distance and lexical information. An example of such a frame is: '`NP_near_NP`', where both O_i and S_j are of the phrase type NP and the distance is classified as 'near'.

2.2.2 Johansson & Moschitti

Choi, Breck, and Cardie (2006)'s discoveries have been used in several later works. Recently, Johansson and Moschitti (2013) presented a summary of their work on this field. We will look into some notable similarities and differences between the two articles.

Johansson and Moschitti (2013) use the *MPQA 2.0 Opinion Corpus*, which we will present in more detail in Chapter 3. In this corpus, the opinion expressions are annotated as so-called *direct subjective expressions* (DSEs), *expressive subjective elements* (ESEs) and *objective speech events* (OSEs). The opinion expressions are extracted with a sequence labeler and assigned one of the labels above.

[5] Sundance: `http://www.cs.utah.edu/~riloff/publications.html`

Feature set	P	R	F
Baseline	63.41.5	46.81.2	53.81.1
All syntactic features	62.51.4	53.21.2	57.51.1
Removed SYNTACTIC PATH	64.41.5	48.71.2	55.51.1
Removed LEXICAL PATH	62.61.4	53.21.2	57.51.1
Removed DOMINANCE	62.31.5	52.91.2	57.21.1

Table 2.1: Investigation of the contribution of syntactic features for opinion expression detection.

For training, Johansson and Moschitti (2013) combine the *Structured Perceptron* and the *Passive-Aggressive* algorithms. We will not go into details about the machine learning algorithms.

The task of extracting the opinion structure is divided into three: A set of opinion expressions, their opinion holders and polarities. Different sets of features are extracted for each of these.

2.2.2.1 Opinion expression detection

For opinion expression detection, Johansson and Moschitti (2013) constructed a strong baseline, using fewer features compared to Choi, Breck, and Cardie (2006). Their baseline does not rely on syntactic or semantic information. The system relies only on four observations, extracted from a sliding window including the preceding and succeeding word:

- Word form
- Lemma
- Part-of-speech tag
- Prior polarity, retrieved from Wilson, Wiebe, and Hoffmann (2005).

They expand their system using reranking, with syntactic and semantic information using the LTH SRL parser, which we will describe in Section 3.4, in their final system. They combine this syntactic and semantic information with the provisional information from the baseline.

Some of the features are also extracted from a pair of opinion expressions. The article shows the effect of different features with several tables. We will quote one of the tables from their feature ablation study.

Table 2.1 shows the precision, recall and F-measure for opinion expression determined after applying the syntactic features. From the results, we can see that *Syntactic path* is the feature of most importance for the gain in F-measure. The table on the contribution of semantic features is similar. There, *Connecting argument label* is the most import feature for the increase in F-measure. Combining syntactic and semantic features gives an F-measure of 58.0, suggesting an overlap in what we can learn from these sources of information.

2.2.2.2 *Opinion holder classification*

For opinion holder classification, the system of Johansson and Moschitti (2013), including the baseline system, is largely dependent on syntactic and semantic features.

We will describe their feature set in short terms here, but we will come back to all these features in details in Section 5.6.

Syntactic path This feature represents the path in the dependency tree between the holder candidate and the expression.

Expression head word, POS and lemma Word form, POS and lemma for the head word of the expression.

Holder candidate head word and POS Word form and POS for the holder candidate

Holder candidate context words and POS Word form and POS from the words preceding and succeeding the holder candidate.

Dominating expression type If the expression is syntactically dominated by another expression, this feature is extracted.

Expression verb voice A feature that tells us whether there is an active or a passive voice in the expression.

Expression dependency relation to parent This feature represents the label of the dependency relation from the expression head word to the parent.

Shallow semantic relation If there is a semantic relation between the holder and candidate, a feature representing this is used.

2.3 GRAMMATICAL ANALYSIS

Previous work (Choi, Breck, and Cardie 2006; Johansson and Moschitti 2013), see Section 2.2, shows that grammatical analysis has a central role in opinion analysis. The aim of this project is to compare different dependency representations, and thus we need to look into grammatical analysis. We give an overview in this section and will come back to our choice of parser in Chapter 3.

2.3.1 *Constituent and Dependency representations*

There are two major approaches to the representation of syntactic structure in statistical parsing. Constituent/Phrase structure representations have been developed for many years and are widely used. More recently, dependency representations have gained popularity.

In dependency representations, sentences are described with direct syntactic or semantic relations between the words in a sentence. With constituent analysis on the other hand, phrases/constituents are fundamental. For this master's project, we will investigate how different dependency representations affect the

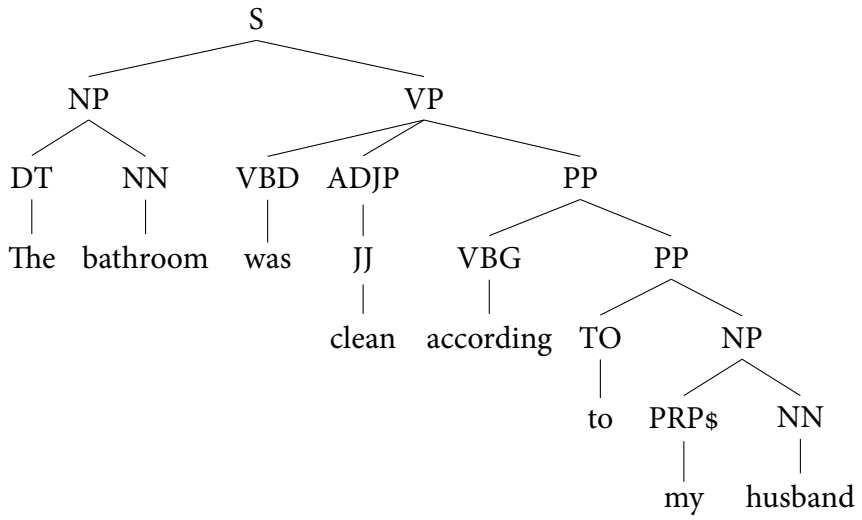


Figure 2.1: Phrase-structure parse tree from the Stanford Parser.

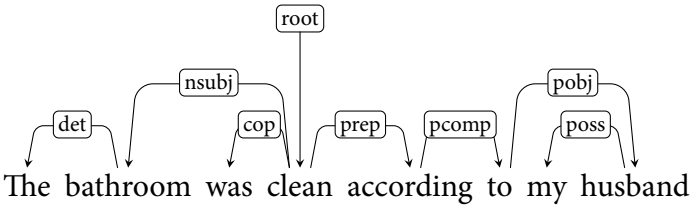


Figure 2.2: Dependency parse from the Stanford Parser.

results. Each of the formalisms have different varieties, and thus the information we can gather varies.

In example (1a), the example sentence "The bathroom was clean according to my husband." has been parsed with the online Stanford Parser[6] to show typical parsing outputs for the two formalisms. Figure 2.1 shows a phrase-structure parse tree. The output gives information of the constituent structure of the sentence with part-of-speech tags of the phrases in the sentence. For other variants of phrase-structure grammars, we can obtain more information. For example, with Head-Driven Phrase Structure Grammar (HPSG), we will get information of the lexical head of the phrases.

Dependency graphs are labeled directed graphs. The definition of a well-formed dependency graph in Nivre et al. (2007), provides five formal requirements of these graphs:

[6] `http://nlp.stanford.edu:8080/parser/`

- Node 0 is a root node
- The graph is connected
- Every node has at most one head
- The graph is acyclic
- The graph is projective

It is questionable for several of these requirements whether they have a foundation in linguistic theory, or if they are applied for purely practical reasons. For the last requirement, projectiveness, Nivre et al. (2007) state themselves 'that virtually all treebanks annotated with dependency graphs contain non-projective structures.' All the dependency representations we will use in this project fulfil the other four of these requirements.

In figure 2.2, we see that there are no non-lexical nodes in the dependency graph. Instead we obtain information on the syntactic function of the relation between words.

According to Ivanova et al. (2012) it is often claimed that dependency representations are more 'semantic' in spirit. Predicate-argument relations are directly expressed in a dependency graph. For the purpose of opinion analysis, this property is interesting. In most cases where the opinion holder for an expression, in the form of a predicate, is explicitly mentioned in a sentence, we will find the holder as one of the arguments of the predicate.

2.3.2 Different dependency representations

A necessary resource for all statistical parsing is annotated data. For constituent analysis, the Penn Treebank (PTB) has been a central resource, widely used. Even though dependency representations have gained popularity, there is no similar resource for dependency annotated data. Therefore, several constituent-to-dependency conversion methods have been constructed, in order to convert from the Penn Treebank constituent trees to dependency representations (Collins 1999; Yamada and Matsumoto 2003; Johansson and Nugues 2007). This has led to the development of different dependency representations, with large variation. Ivanova et al. (2012) show that there are large representational differences between different representations.

As the aim for this master's project is to investigate how syntactic representation affects opinion analysis, we will examine the differences between some of the representations. We will come back to the choice of tools for parsing in Chapter 3.

We will briefly describe the representations we will use in this project before we investigate the practical effect the differences can have on a task like opinion analysis in Chapter 5.

2.3.2.1 CoNLL Syntactic Dependencies (CD)

Dependency parsing has been the topic of several CONLL Shared tasks in the last decade. In 2008, the task was "Joint Parsing of Syntactic and Semantic Dependencies" (Surdeanu et al. 2008). The data source for this task is derived from the Penn Treebank with the conversion procedure described in Johansson and Nugues (2007). Like other systems, they create head-dependent pairs by selecting one word as a head in the constituent phrase and setting the rest as dependents. They exploit the functional information present in PTB: grammatical function labels and long-distance grammatical relations. This representation gives dependency graphs where the head tends to be a functional word.

2.3.2.2 Stanford Basic Dependencies (SB)

Stanford Typed Dependencies (de Marneffe and Manning 2008) are also based on conversion from a PTB constituent tree to a dependency graph. Two versions exist: in the basic form, every token in a sentence must be a part of the dependency graph. In the collapsed form, some words, especially prepositions, are omitted from the graph, and instead turned into a part of a relation. The latter representation will therefore give non-connected dependency graphs. For our purpose, we will use the basic Stanford representation, where the graph is connected.

A set of design choices have been followed when creating this representation. One interesting choice is that relations should be between content words, not via function words. This gives quite different heads compared to CD.

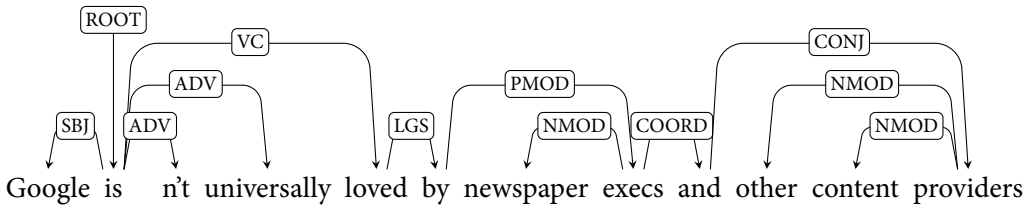
2.3.2.3 DELPH-IN Syntactic Derivation Tree (DT)

This dependency representation contrasts with the two previous representations by not being based the dependencies on conversion from PTB trees, but instead basing on the much richer Head-driven phrase structure grammar (HPSG) that is used in the English Resource Grammar (ERG). (Zhang and Wang 2009; Ivanova et al. 2012)

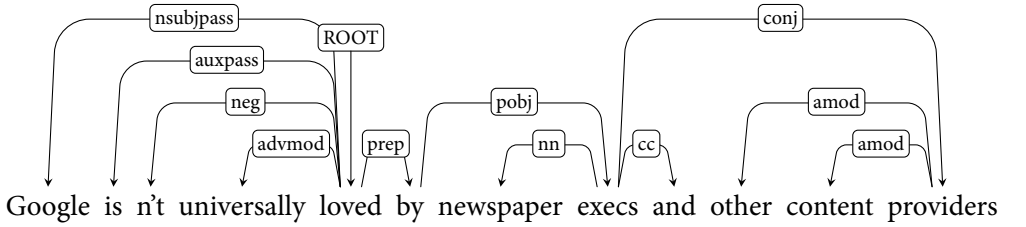
This procedure creates a derivation tree based on the HPSG parse tree, were the nodes are labeled with identifiers of HPSG constructions (Ivanova et al. 2012). This derivation tree is then converted to a dependency graph and the fine-grained labels are simplified to a set of around 50 standard labels.

2.3.2.4 Comparison

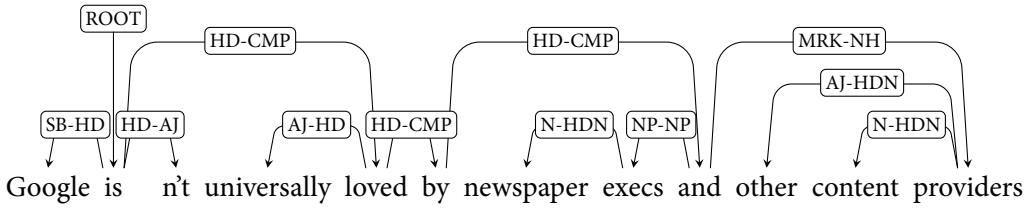
On a syntactic level, there is a major difference in the choice of head between the representations described above, which also affects the sentence root. In a complex verb phrase, either an auxiliary verb or the main verb could be the head. In a noun phrase, the head could be the determiner or the noun. For coordination, the conjunction or either of the conjuncts could be the head. Ivanova et al. (2012) name that the choice is between functional and substantive / content-centered head. Both CD and DT have more functional heads, while SB has more content



(a) CD representation



(b) SB representation



(c) DT representation

Figure 2.3: Dependency graphs for (a), (b) and (c) representations. The period marks and the edge to these are omitted in the figure.

centered heads. This also affects the choice of sentence root. An exception from this tendency is the relation between nouns and their adjectives and determiners, which these three dependency representations treat similarly; the determiner and adjectives are dependents of the noun.

Figure 2.3 shows the dependency graphs for the dependency formats described above for the sentence in example (1b). We will now compare some detailed differences between these formats.

2.3.2.5 Root choice

The root choice is related to the head status, which is discussed above. Both CD and DT tend to have a functional head, according to Ivanova et al. (2012), while SB is inclined to have a substantial head. In figure 2.3, we see that for the part of CD and DT, the root word is the finite auxiliary word 'is', whereas for SB, root is the lexical verb 'loved'.

	Syntactic path to holder	Syntactic path to subject
CD	VC↓LGS↓PMOD↓	SBJ↓
SB	prep↓pobj↓	nsubjpass↓
DT	HD-CMP↓HD-CMP↓HD-CMP↓	SB-HD↓

Table 2.2: The choice of dependency representation can lead to quite different features for machine learning. This example shows the syntactic path between the head of an opinion expression and its holder in DT, SB and CD. The third column shows how this also affect the syntactic path to the other possible holder in the sentence, 'Google'

2.3.2.6 *Conjunction*

Analysis of coordination, according to Ivanova et al. (2012), is a well-known area of differences between various dependency representations. In figure 2.3, we see that all three representations have different strategies for the coordination conjunction 'and'. With CD, 'and' is the daughter of the first conjunct and the head of the consequent conjunct. For SB, the first conjunct is the head of both the consequent conjunct and the coordinating conjunct. A third graph structure is found in DT, where 'and' is the head of both conjuncts.

2.3.2.7 *Infinitive and infinitive marker*

How an infinitive with its infinitive marker is represented is not shown in this example. However, this is also described by Ivanova et al. (2012). They state that the infinitive marker is the daughter of the infinitive in SB, while the infinitive marker is the head in DT and CD.

2.3.3 *How different representations affect a task like opinion analysis*

In our system, dependency parsing is used for features in the task of pairing opinion expressions with their opinion holders. As of today, we are not aware of any systematic comparison made on this subject.

For the task of opinion holder classification, described briefly in Section 2.1.3, and which we will investigate in full details in Chapter 5, we will expect that the differences in the representations have an effect on two parts of the system.

First, as possible opinion holders will be a noun phrase, we need to identify such phrases. In our selection of dependency representations, the relations between the determiner, adjectives and the corresponding noun is the same, so for large parts this will be the same. But for the part of conjunction of noun phrases, there are some differences that will have an impact on this selection. In figure 2.3, we see that the opinion holder is a coordination of the two noun phrases 'newspaper execs' and 'other content providers'. In figure 2.3c, the coordination conjunction is the head of both these noun phrases, and thus, we will not have a

single noun that is the head of the whole phrase. This in contrast to figure 2.3a and figure 2.3b, where 'execs' is the head of the whole opinion holder.

Second, when building a machine learning model, we will extract features that illustrate the relationship between opinion expressions and their holders. These features can be vary quite a bit between different dependency representations. In figure 2.3b, a syntactic path from the head of the opinion expression, 'isn't universally loved', to the opinion holder, will be quite different, which we list in table 2.2. The feature types express the path from the holder to the expression, with the edge labels and arrows indicating the direction in the tree.

A different syntactic path is not a problem in itself, as this will be the same both for training and classification. But the situation that CD and DT often lead to more complex paths, can contribute to a more sparse data material for the classifier.

We will now turn to our implementation of a system for opinion expression detection, opinion holder classification and experimenting with different dependency representations. In the following chapter, we will describe the prerequisites for these tasks: the corpus, the lexicon for prior polarity, pre-processing tools, syntactic and semantic parser, classifier, semantic role labeler and evaluation metrics. Then, in Chapter 4 and Chapter 5, we will describe our implementation of the task of opinion expression detection and opinion holder classification, and discuss our findings.

Chapter 3

Prerequisites: Experimental infrastructure

Our work is experimental in nature. We are using a multitude of third party tools in the experimental environment. In this chapter, the data sources and tools are presented, and we will describe how we have implemented these in the system. The details in the pipeline for each part of the project will be presented in Chapter 4 and in Chapter 5. The setup is largely inspired by the work of Johansson and Moschitti (2013), but we have selected the tools independently of their work.

3.1 MPQA OPINION CORPUS

A widely used resource for opinion analysis is the MPQA Opinion Corpus 2.0 (Wiebe, Wilson, and Cardie 2005; Wilson 2008, chapter 7)[1]. This version contains 692 documents and 15802 sentences, and represents a variety of sources, including news articles, question-answering documents, travel guides, letters and e-mails.

The corpus is divided in five different sets (see table 3.1 for an overview):

MPQA original subset News sources. 187 documents.

OpQA (Opinion Question Answering) subset Questions and answers, half opinion-based, half fact-based. 98 documents.

XBank Wall Street Journal texts from the Penn TreeBank. 85 documents.

ULA (Unified Linguistic Annotation) Subcorpus of the American National Corpus. This selection of 48 documents includes travel guides, transcription of spoken conversation, fundraising letters, excerpts from a report and a linguistics textbook and magazine articles.

[1] MPQA Corpus: `http://mpqa.cs.pitt.edu`

Table 3.1: Overview: The data sets in MPQA.

Source	Content	Count	Expressions[a]
MPQA original	News sources	187	37 061
OpQA	Q&A	98	7895
XBank	WSJ from the Penn Treebank	85	1927
ULA	Varied	48	5348
ULA-LU	Varied	24	1882

[a] Including zero span expressions.

ULA-LU (Language Understanding subcorpus) 24 documents including emails, spoken language transcipts, newswire text, Wall Street Journal texts from the Penn TreeBank and translation of Arabic source texts.

3.1.1 The annotation

The annotation in the MPQA Corpus is briefly described in this section. Example (4), example (5) and example (6) retrieved from the corpus illustrate some of the annotation. The square brackets indicate opinion expressions and curly brackets opinion holders. Opinion expressions are labeled with a reference to the corresponding opinion holder and the type of expression.

(4) $\{\text{SRI}\}_{(\text{H}_1)}$ $[\text{reported}]_{(\text{H}_1/\text{OSE})}$ that $\{\text{Swiss Life}\}_{(\text{H}_2)}$ $[\text{decided}]_{(\text{H}_2/\text{DSE})}$ to pull out of the Chinese market earlier this year.

(5) Mostly, $\{\text{I}\}_{(\text{H}_1)}$ $[\text{liked}]_{(\text{H}_1/\text{DSE})}$ to spend time reading encyclopedias–

(6) Terrorism $[\text{thrives}]_{(\text{w/ESE})}$ in an $[\text{atmosphere of } [\text{hate}]_{(\text{implicit/DSE})}]_{(\text{w/ESE})}$

There are seven types of annotation:

3.1.1.1 Agent

The annotation type *agent* holds the opinion holders, and the phrase is also tagged with *nested source*, a list of who is referring to this entity, starting with the writer (w) and ending with the opinion holder. The first occurrence of each agent in the document is also given an *id* attribute. Since the annotation is made on the document level, that ID will often be outside the scope of the sentence. There are two special agent IDs, 'w' (writer) and 'implicit'.

In example (4), the nested source of *decided* is 'w, sri, swisslife'. This list of identifiers, starting with the writer of the text and ending with the entity of the opinion holder, shows that the writer (w) refers Swiss Life's (swisslife) opinion indirectly, via the radio station SRI (sri). There is no direct link between Swiss Life in the text and the corresponding expression. Several agents with the same nested source can occur in the same sentence.

In order to select a holder-opinion pair, we will have to make three choices.

First, we will only compare the last element in the list. If the list is 'w, sri, swisslife', we consider 'w, swisslife' a match. In other words, we will treat an opinion holder as the same whether the writer refers directly or indirectly to the holder. Second, we will link the expression with all corresponding agents. At last, if there is no suitable opinion holder in the sentence, we will treat it as implicit. We will come back to this in Section 3.1.3.

3.1.1.2 *Expressions*

By the term 'expression', in this context, we mean both opinions and non-subjective statements. There are three types of expressions in MPQA 2.0. These types can hold attributes for the intensity and the polarity of the expression in addition to the attribute *nested-source*. With the information in nested-source, we can find a link between an expression and the corresponding opinion holder (agent).

ESE Expressive-subjectivity elements. Indirect opinions are annotated with this type. In example (6), both *thrives* and *atmosphere of hate* are ESEs.

DSE Direct-subjective expression. These are explicit opinion statements. In example (5), *liked* is a DSE.

OSE Objective speech-event. These are expressions that indicates factual information. In example (4), *reported* is an example of a phrase annotated with this type.

An expression is not necessarily represented by an overt phrase in the text. The MPQA contains zero span expressions for those implicit statements. We will ignore these in our project. Furthermore, there are several cases of overlapping expressions, such as in example (6). We will come back to this in sections 3.1.3 and 3.6.

3.1.1.3 *Other annotations*

In MPQA 2.0, there are three other annotation types; *Attitude*, *target* and *inside*. Inside is identical to the sentences in this version of the corpus, and redundant, as this information is found in a separate file. Attitude and target are types linking an expression to targets of opinion. These annotation types are not necessary for our project.

3.1.2 *Data format*

The data is annotated in an offset format. For each document, there are four files with information; the raw text, meta information about the document, the sentence spans and the annotation spans. In addition, the OPQA (question answering) part of the corpus also contains a file with spans for the answers.

For the sentence in example (4), the annotation file contains the information listed in figure 3.1. The file is tab separated, with at least five columns.

```
51 488,496 string GATE_objective-speech-event
   nested-source="w, sri"
46 484,573 string GATE_inside  nested-source="w"
45 484,484 string GATE_objective-speech-event
   nested-source="w" implicit="true"
35 513,520 string GATE_direct-subjective
   expression-intensity="low"
 attitude-link="a30" nested-source="w, sri, swisslife"
    intensity="low" polarity="neutral"
30 484,487 string GATE_agent  nested-source="w, sri"
21 502,512 string GATE_agent  nested-source="w, sri, swisslife"
5 513,523 string GATE_attitude  target-link="t30"
   attitude-uncertain="somewhat-uncertain"
 attitude-type="intention-pos" id="a30" intensity="low"
4 524,554 string GATE_target  id="t30"
```

Figure 3.1: Example of annotation data (from 20020510/21.50.13-28912). The file is a tab separated table. The first column is a unique identifier. Then follows the offset position, data type (string), annotation type (beginning with 'GATE_') and attributes.

id The first column is an annotation number, giving each annotation in a document a unique number.
span The next column shows the character offset.
data type Column 3 is showing the data type (which should always be string).
annotation type The column starting with 'GATE_' shows the annotation type.
attributes A varying number of attributes.

To link an expression with a corresponding opinion holder, the nested-source attribute gives the necessary information. In figure 3.1, annotations number 35 and 21 have the nested-source 'w, sri, swisslife'. It follows that this represents an opinion holder / expression pair. The IDs in the nested-source list are the first occurence of the agent in the document. In many cases, the phrase marked with ID is not a part of the sentence where the expression occurs. We therefore link the opinion holders and the expressions by comparing the nested sources of each.

In the case above, the writer states that the direct source for information is SRI. In the following sentence in the same document, however, example (7), the nested source is 'w, swisslife', which means that the writer refers directly to Swiss Life.

(7) Last month , $\{\text{it}\}_{(\text{H1})}$ $[\text{announced}]_{(\text{H1/OSE})}$ it was cutting 800 jobs worldwide as part of a recovery plan.

We will treat the nested-source lists 'w, swisslife' and 'w, sri, swisslife' as equal, that is, we will only look at the last member of the nested-source list when linking expression and holder.

There are two annotation lines with empty spans in each document for opinion holders not explicitly occuring in the text, implicit and w (the writer of the document).

```
55 0,0 string GATE_agent  id="implicit"
49 0,0 string GATE_agent  id="w"
```

In many cases, the opinion holder does not occur in the same sentence as the expression. As our analysis is on the sentence level, these cases can be treated as imlicit opinion holders.

Annotation number 45 from figure 3.1 is an example of zero span expressions. An empty expression like this can be found at the beginning of each sentence, stating the type of expression for the writer, either an objective-speech or a direct-subjective. These have the attribute 'implicit', which in these cases shows that there is no phrase for the corresponding expression. The MPQA documentation gives as an example that the writer does not write "I write" for each sentence, but this information is implicit.

In some cases (435 cases in 10926 sentences in the training set for this project), spans from the annotation file do not end at the end position of a word. In most of these cases, single-character span occur instead of zero span. In 39 cases, the mismatch is occurring elsewhere in a sentence. These are likely errors in the annotation. A single-character expression in the beginning of a sentence will be treated as a zero span expression. A longer span that ends in the middle of a word, will be interpreted so that the word will be excluded from the expression.

3.1.3 Extraction from MPQA

We will now describe how we have fitted the annotations from MPQA into our data structure. We will continue to follow example (4).

As described in Section 3.1.2, the annotations in MPQA come in an offset format. To combine these annotations with the syntactic features we will extract, we have created a ordered table, where the key is the starting offset position. The value for each member of this table is a list of annotations with the same starting offset position. At this point, we will include all types of annotations, including zero span expressions. For the starting point of the first token in example (4), 'SRI', the annotations are listed in figure 3.2.

For the task of opinion expression detection, the annotation types we need to find are: expressive-subjectivity, direct-subjective and objective-speech-event. For opinion holder detection, we will need to find pairs of opinion holders (agents) and expressions. In figure 3.2, there are two examples of annotation types we are looking for. The first annotation is 'agent', with the 'nested-source' a zero span expression that represents the implicit event that the writer is writing the sentence. For this project, such events are ignored.

```
484: [{'ann_type': 'GATE_inside',
  'data_type': 'string',
  'line_id': '46',
  'nested-source': 'w',
  'nested_source_split': ['w'],
  'slice': slice(484, 573, None)},
 {'ann_type': 'GATE_objective-speech-event',
  'data_type': 'string',
  'implicit': 'true',
  'line_id': '45',
  'nested-source': 'w',
  'nested_source_split': ['w'],
  'slice': slice(484, 484, None)},
 {'ann_type': 'GATE_agent',
  'data_type': 'string',
  'line_id': '30',
  'nested-source': 'w, sri',
  'nested_source_split': ['w', 'sri'],
  'slice': slice(484, 487, None)}]
```

Figure 3.2: Annotation data for the start position of 'SRI' from example (4).

In some cases there is an overlap between expression types. In the training set, there are 776 cases of overlap between expressions. For training, if there are overlapping expressions, we will ignore one of them. In Chapter 4, we will describe this selection.

The structure of the annotation format in MPQA is complicated and not easily accessible for humans. This might also be related to the minor error in example (5) in Johansson and Moschitti (2013). The example is:

> **(implicit)**$_{(H_1)}$: This **[is viewed]**$_{(DSE/H_1)}$ as the **[main impediment]**$_{(ESE/H_1)}$ to the establishment of political order in the country.

In the corpus, the ESE is annotated with the agent 'w', and should not have an implicit holder according to the annotation in MPQA. Johansson (p.c) confirms that this is the case. The example is based on the presumption that the agent was 'w, analysts', which is the case for the DSE in the sentence, and is also a more correct annotation for the ESE.

3.1.4 Data split

For partitioning in a test and training set, the document lists provided from Johansson and Moschitti (2013) on their project page[2] was used. Their training set consists of 541 documents and their held out test set consists of 150 documents.

[2] http://demo.spraakdata.gu.se/richard/unitn_opinion/details.html

	Split from J&M	Our split
Training set	541	487
Development set		54
Test set	150	150

Table 3.2: Data split.

For development, 90 % of the documents in the train set was randomly chosen as a training set. The rest, 54 documents, was chosen as a development set, as shown in table 3.2.

3.2 SENTIMENT LEXICONS

In much of the work reviewed above, so called sentiment lexicons are used for finding previous polarity. Choi, Breck, and Cardie (2006) use a lexicon extracted from the training data combined with approximately 500 opinion words from MPQA Final Report as a feature for the opinion phrase extraction. In this project, we will use the lexicon created by Wilson, Wiebe, and Hoffmann (2005) for opinion expression detection.Each entry in the lexicon is representing a single word that acts as a clue for subjectivity. An example line from this lexicon is:

```
type=strongsubj len=1 word1=happy pos1=adj stemmed1=n
   priorpolarity=positive
```

All entries have the same six fields.

type intensity, either strongly or weakly
len number of words (in this lexicon is it always 1)
word1 subjectivity word or stemmed form
pos1 part-of-speech to disambiguate words
priorpolarity the polarity the clue word gives, either positive, negative, both or neutral

From 'type', we will get the intensity, either strongly or weakly subjective and 'priorpolarity' states the prior polarity, either positive, negative, both or neutral.

We will check every token, both in its original form and lemmatised, and add a feature containing information about the polarity and intensity.

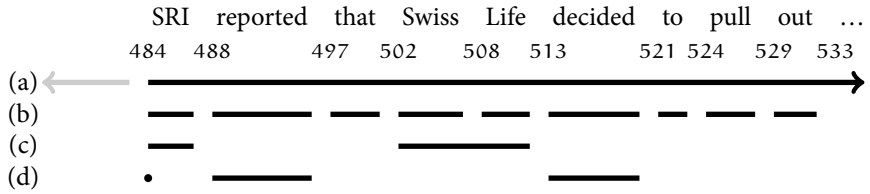


Figure 3.3: Layers of spans from example (4). (a) Sentences. (b) Tokens. (c) Opinion holders. (d) Expressions. There is also a non-span expression at position 4091.

3.3 PRE-PROCESSING

Some common preliminary processing is necessary for most natural language processing. This section includes sentence breaking, tokenisation, part-of-speech tagging and lemmatisation. A specific challenge, in light of the character offset data format described in Section 3.1.2, is to keep the alignment correct. There are several layers of information we need to fit into our data structure. In figure 3.3, we see the different spans, for example (4). This issue has been important for the design choice.

We will build a list of sentences for each document with lists of tokens with references to the offset position. In this data structure, syntactic and other information can be represented. These data can finally be combined with the annotation from MPQA when building training files.

3.3.1 Sentence breaking

MPQA provides sentence break spans for each document. For both training and testing, these will be used. For each document, the raw text is read and saved in a list of tuples, containing each sentence and the offset span. The sentence break in example (4) is defined by the following line in MPQA:

```
2       484,573 string  GATE_sentence
```

The sentence spans from position 484 to position 573 in the text document.

3.3.2 Tokenisation

For tokenisation, the segmentation of words, there is a large variety of tools. For our purpose, we have two requirements to the tokeniser. First, as mentioned in Section 3.3, we have to keep the alignment between the tokens and the offset positions correctly, to fit the annotations correct into our data strucure. Some tokenisation tools will give us information about the offset positions of the tokens.

Second, segments like 'don't' can be treated in several ways when it comes to tokenisation. This can be a problem later in the pipeline, because some tools may

```
[(SRI, slice(484, 487, None)),
(reported, slice(488, 496, None)),
(that, slice(497, 501, None)),
(Swiss, slice(502, 507, None)),
(Life, slice(508, 512, None)),
(decided, slice(513, 520, None)),
```

Figure 3.4: After tokenisation: List of tuples of word form and offset positions.

require a certain standard for tokenisation. A common standard for tokenisation approach is PTB-compliance. POS-taggers are often trained on the Penn Treebank (Marcus, Santorini, and Marcinkiewicz 1993). The word segmentation of the tokeniser needs to be compliant with that in the Penn Treebank. Some examples are:

- Contractions like 'don't' should be split in to 'do' and 'n't'
- Commas followed by whitespace is a separate token.
- Period marks followed by numbers should not be split.

The REPP tokeniser (Dridan and Oepen 2013)[3] is PTB-compliant and has shown premium performance. This tokeniser also returns information about offset positions.

The REPP tokenizer is run for each sentence. The word form and the corresponding offset position for each token in the sentence are kept in a list of tuples. The offset positions are held in the python object 'slice'. This object can be used directly on the document text for retrieving the raw text for that position and the start and stop positions are easy accessible. An example of this structure is found in figure 3.4.

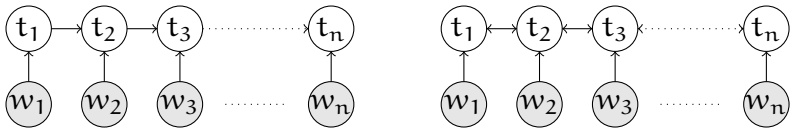
To ensure that the offset positions from tokenisation align correctly with the offset positions in the annotation data, we checked that the start position for each annotation corresponded with a start position for a token.

3.3.3 *Part-of-speech tagging and lemmatisation*

POS-tagging is the assignment of part-of-speech tags to the tokens. There are several different tagsets that can be used, but one of the most common for English is the Penn Treebank tagset(Marcus, Santorini, and Marcinkiewicz 1993).

There are several different approaches for building a POS-tagger, but probabilistic approaches are very common. Markov Models are widely used for this task and both hidden Markov Models (HMM) and conditional Markov Models (CMM) can be used to build to an effective tagger. The Stanford POS-tagger uses

[3] http://moin.delph-in.net/ReppTop



(a) CMM (b) Birectional dep. network

Figure 3.5: Graphical comparison of a regular CMM and a bidirectional dependency network.

Form	Lemma	Offset Position	POS
SRI	srus	484–487	NN
reported	report	488–496	VBD
that	that	497–501	IN
Swiss	Swiss	502–507	NNP
Life	Life	508–512	NNP
decided	decide	513–520	VBD

Table 3.3: Available information after part-of-speech tagging.

a further development of Markov Models, Bidirectional Dependency Networks (Toutanova, Klein, and Manning 2003)[4].

The Stanford POS-tagger was used for both POS-tagging and lemmatisation. The tagger also normalises the word form, as described in Section 3.3.2. The list of tokens is then expanded with the new info. For the first part of the example sentence example (4), the information in table 3.3 are available after tagging.

3.4 GRAMMATICAL STRUCTURE ANALYSIS

In this section we will describe the dependency parsers we will use in our project, and the setup of these tools.

3.4.1 LTH SRL

Johansson and Moschitti (2013) use the system developed by Johansson and Nugues (2008) for the CoNLL-2008 Shared Task. [5] This parser does both a syntactic dependency parsing and semantic role labelling. A model for English is bundled with the parser, the dependency representation provided in this model is CD.

[4] http://nlp.stanford.edu/software/pos-tagger-faq.shtml

[5] `http://nlp.cs.lth.se/software/semantic_parsing:_propbank_nombank_frames`

	LAS	UAS	LACC
SB	90.65	93.05	94.01
CD	90.92	93.97	92.94
DT	90.91	93.18	92.76

Table 3.4: Parsing results of Bohnet & Nivre, w/o punctuation. We got results identical to Ivanova (2015, p. 98)

This representation is described in Section 2.3.

During the development of our system, we used this parser for several reasons. First, we had at the time of development not decided on the parser to use for the investigation of dependency representations. Second it let us compare our results with and without semantic role labelling.

3.4.1.1 Semantic information

3.4.2 Bohnet & Nivre

For the comparison between different dependency representations, described in Section 2.3, we will use the same dependency parser with models trained on different dependency representations. Ivanova (2015) provided models for the Bohnet & Nivre parser (Bohnet and Nivre 2012), which provided a state-of-the-art performance. The parser is distributed as part of the Mate Tools.[6] We use the same version of the tool package as Ivanova (2015).

The parser is sparsely documented[7], but in order to reproduce the results from Ivanova (2015), we only needed to set the *beam* parameter. We controlled that the models gave us the same results as in Ivanova (2015, p. 98), as we see in table 3.4. This was achieved by the following command:

```
java -cp anna-3.3.jar is2.transitionR6j.Parser -model
    /path/to/conll-all-ptb_tok-ptb_pos.mdl -beam 80 -test
    test_gold.conll -out test_gold.conll.conll
```

The parser does a joint dependency parsing and tagging, but we will not use the tag output.

[6] https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/mate-tools/anna-3.3.jar

[7] https://code.google.com/archive/p/mate-tools/wikis/Transition_based_parser.wiki

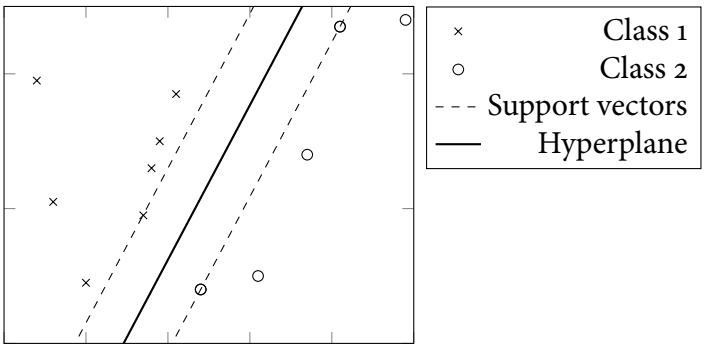


Figure 3.6: Support vectors. This way, we can find the maximum margin hyperplane to divide the two classes.

3.5 CLASSIFICATION AND SEQUENCE LABELLING

Although the focus in this master's project is not to compare different types of machine learners, the experiments heavily depend upon machine learners. In this section, we will present the two machine-learning tasks we will focus on, and the tools we have used for these tasks.

3.5.1 Classification

For the second part of our project, opinion holder extraction, our goal is to match an expression-holder pair. The aim is to locate candidates for opinion holders for each expression, and select the best match. We will build one classifier for each type of expression. For each expression of a certain type, we will list possible holder candidates, and select the best fit.

There is a large number of machine learning algorithms that may be suitable for this task. One approach that has become very popular is Support Vector Machines (SVM), introduced by Cortes and Vapnik (1995).

Regular SVMs are binary classifiers, that is, they guess between two classes. The border between the two classes is linear, and placed in a way that makes the largest possible margin between the classes. This is done by creating two support vectors, see figure 3.6.

The LIBLINEAR library (Fan, Chang, and Hsieh 2008), which provides linear SVM, is available through the toolkit Scikit-learn for python (Pedregosa et al. 2011)[8].

3.5.1.1 Vectorisation

The input format for LIBLINEAR is a table, where each row represents the object to be classified, in our case a holder candidate. The first column represents the

[8] `http://scikit-learn.org/stable/modules/linear_model.html`

```
{'cand_head_pos': 'NNP',
 'cand_head_word': 'Life',
 'context_l_pos': 'VBD',
 'context_l_word': 'decided',
 'context_r_pos': 'NNP',
 'context_r_word': 'Swiss',
 'deprel_to_parent': u'SUB',
 'dom_ex_type': 'dse',
 'ex_head_lemma': 'decide',
 'ex_head_pos': 'VBD',
 'ex_head_word': 'decided',
 'ex_verb_voice': 'Active',
 'synt_path': u'SBJ\u2191'}
```

Figure 3.7: Example of features for a holder candidate before vectorisation.

Class	(...)	dom_ex_type=dse	dom_ex_type=ese	dom_ex_type=ose	(...)
1	(...)	1	0	0	(...)
0	(...)	1	0	0	(...)
0	(...)	0	1	0	(...)
0	(...)	0	1	0	(...)

Table 3.5: After vectorisation. Example of how the feature 'dom_ex_type' (see figure 3.7) for four expression-holder candidate pairs is represented after vectorisation. For each feature, all combinations of a single feature and its possible values will be represented by a column. Only one of these combinations will be true (1), the rest are false (0).

class (for binary classification 1 or 0, and the rest of the columns represent every combination of a feature name (table key) and its value. For each feature, only one column has the value 1 (true), the rest have the value 0 (false). An example of this is found in table 3.5. This is called one-hot coding or one-of-K. Vectorisation is the conversion to this format.

Before vectorisation, the features for each token is saved in a table, figure 3.7 gives an example of this table for one token. Scikit-learn has a tool for this conversion, 'DictVectorizer', which will be used in this project.

3.5.2 *Sequence labelling*

The first task is to label the expressions. The expressions will be continuous phrases, or word sequences, in a sentence. We have chosen to solve this task as a sequence labelling task. The task is described in detail in Chapter 4.

We can look more closely at example (4), where 'was happy' is a direct subjective expression. For each word in the sentence, the machine learner should

Word	Lemma	POS	Sent.lex	Label
SRI	srus	NN	-	O
reported	report	VBD	-	B-OSE
that	that	IN	-	O
Swiss	Swiss	NNP	-	O
Life	Life	NNP	-	O
decided	decide	VBD	weak/ne	B-DSE
to	to	TO	-	O
pull	pull	VB	-	O
out	out	RB	-	O
of	of	IN	-	O
the	the	DT	-	O
Chinese	chinese	JJ	-	O
market	market	NN	-	O
earlier	earlier	RBR	-	O
this	this	DT	-	O
year	year	NN	-	O
.	.	.	-	O

Table 3.6: Available information for sequence labelling.

predict whether the word is a part of an expression.

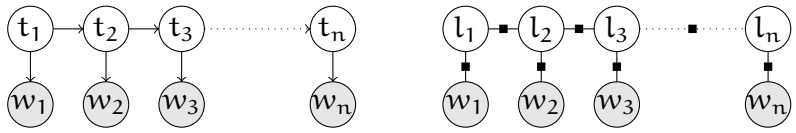
For sequence labelling, we will have the following information:

While a pointwise classifier will look at each word independently, a sequence labeler considers the sequence. A sentence is a linear chain. A word in a sentence is dependent on other words in the sequence.

A typical usage of sequence labelling is part-of-speech tagging.The previous word and its part-of-speech tag is information the model must take into consideration for determining the POS tag for the current word. It is much more likely that a determiner is followed by an adjective or a noun than by a verb.

For determing if a word is part of an opinion expression, the contextual information is essential. We can look at a sentence as a graphical structure. When a sequence labeler considers if 'was' is a part of an expression, the information about the neighbouring words will be of great interest. In contrast to classifiers, a graphical model can model many interdependent variables. (Sutton and McCallum 2010)

A common approach for sequence labelling is Hidden Markov Models (HMMS). Such a model can be viewed as a directed graph, where we find the most probable path through the graph. For each label, we calculate the statistical probabilities of the transitions to the next state and the emission, the probability for certain features given a label.



(a) HMM (b) CRF

Figure 3.8: Graphical comparison of a HMM and a linear chain CRF.

3.5.2.1 *Conditional Random Field*

Linear-chain Conditional Random Fields (CRFs) are a development of HMMs, and have many similarities (Lafferty, McCallum, and Pereira 2001). Both are graph models, based on statistical probabilites. One of the main differences is that while an HMM is generative, a CRF is discriminative, or conditional, see figure 3.8.

Let x be the information we shall use to decide y. A generative model is a model of a joint distribution $p(y, x)$. The model seeks to maximise the joint probability. For classification, with Bayes' rule, this will give a function of the input:

$$f(x) = \arg\max_y p(y)p(x|y)$$

A generative model like HMM will therefore include a model of $p(x)$. Because of this, an HMM can have a wider range of usage, for example for generating text.

A discriminative model on the other hand will model the conditional probability of the output given the input directly, $f(x) = \arg\max p(y|x)$. For sequence labelling, we do not need a model for $p(x)$, and a discriminative model is more suitable.

CRFs are frequently found to give premium performance.There is an open source implementation of linear-chain CRF in the Wapiti toolkit. Details on the setup and tuning of Wapiti is described in Chapter 4.

3.6 EVALUATION METRICS

Intersection-based precision and recall as described by Johansson and Moschitti (2013) has been implemented.

The main idea is that a partially correct span in the system shall not be counted as 100% incorrect. Instead, a value between 0 and 1 is assigned to the span, based on the span coverage c of a span s compared to a span s' (Johansson and Moschitti 2013):

$$c(s, s') = \frac{|s \cap s'|}{|s'|}$$

```
This this DT - B-DSE
is be VBZ - O
our we PRP$ - O
future future NN - O
. . . - O
```

(a)

```
This B-ESE _ _ _
is I-ESE _ _ _
our I-ESE _ _ _
future I-ESE _ _ _
. _ _ _ _
```

(b)

Figure 3.9: Example of overlapping expressions from the training set. Figure (a) is an excerpt from the generated input file for Wapiti. But as there are only one column for the label, we need to collect the information about overlapping expressions in another file. Figure (b) is an example for the corresponding lines in this file.

The length of the span, $|s|$, expresses the number of tokens in the span. From this, we can calculate a span set coverage, C of a set of spans S and a corresponding set S' (Johansson and Moschitti 2013):

$$C(S, S') = \sum_{s_j \in S} \sum_{s'_j \in S'} c(s_j, s'_k)$$

Then, the intersection-based precision P and recall R of a system $\hat{S}$ and a gold S set of spans will be (Johansson and Moschitti 2013):

$$P(S, \hat{S}) = \frac{C(S, \hat{S})}{|\hat{S}|} \qquad R(S, \hat{S}) = \frac{C(\hat{S}, S)}{|S|}$$

In these formulas, the length of the set $|S|$ is the number of spans in the set.

According to Johansson and Moschitti (2013), most work using the MPQA Corpus counts a system span correctly detected if it overlaps with the gold span. Such an evaluation metric will tend to credit longer system spans. The intersection-based precision and recall addresses this issue.

The first implementation of intersection-based precision and recall extracted gold expressions from the generated IOB2-formatted Wapiti test file alone. Because of this, overlapping expressions were not taken into account in the evaluation.

To handle these cases, when generating a test file, we also generate a file containing all the ignored expressions. These expressions are added to the gold expressions for evaluation. An example of this, to show how this will affect the precision and recall, is found in figure 3.9

We see that the phrase 'This is our future.' is annotated with two overlapping expressions. If the system detects 'this' correctly as a B-DSE, the span coverage c(s,s'), the overall recall should be negatively affected for the expression 'This is

	P	R	F-Score
Ignoring overlaps	0.606	0.437	0.508
Including overlaps	0.624	0.429	0.508

Table 3.7: The original and improved evaluation compared on the development set. As expected, we get a higher precision and a lower recall with the improved evaluation script.

our future'. On the other hand, if the system correctly detects 'This is our future', this should not lower the precision. Our improved evaluation should give us a higher precision and a lower recall. This is also indicated when comparing the evaluation for one setup of the system on the development set, as we see in table 3.7.

There is one caveat in this: overlapping expressions of the same type will give unwanted results. In the example above, if both expressions were of the same type, and the system predicted this correctly, the span set coverage would be 2, but for precision, this would be divided by the number of expressions in the proposed set, giving a precision of 2. We have not found any examples of this kind of overlap in the training set, and we will not look further into this question.

Because we take the overlapping expressions into account, our evaluation will slightly differ from the one described by Johansson and Moschitti (2013).

3.6.1 Opinion holders

For opinion holders, an intersection-based system for precision and recall will be more complex. Johansson and Moschitti (2013) explains that this complexity has three reasons. First, only holders for correctly extracted opinion expressions should be credited. Second, MPQA does not have a direct link between an expression and an opinion holder in a sentence. Instead it has coreference chains, as described in Section 3.1.2. We will have to use these chains to determine the pairs. Finally, a holder entity may be the writer or there may be an implicit holder. Because we have a sentence based system, in addition to the holders that MPQA annotates as implicit, holders that are not mentioned explicitly in the sentence will be counted as implicit.

Because of this, the evaluation of opinion holders will follow this method (Johansson and Moschitti 2013): If a system has proposed an opinion expression/holder pair e and h, we first matched the closest corresponding expression e' in the gold, $e' = \arg\max_x c(x, e)$, regardless of their expression type. The most closely corresponding gold opinion holder in the gold expressions coreference chain (agents with the same nested source) is selected, and then the precision and recall scores for this pair are found using the formula for the span coverage, $c(h', h)$ and $c(h, h')$. For the special cases writer and implicit, if the system guesses correctly, it is given a full score for that pair, and a full error otherwise.

Chapter 4

Opinion expression detection

The task of opinion expression detection can be accomplished by sequence labelling. In large part, our system follows the work of Johansson and Moschitti (2013), but when it comes to the concrete tools being used and the retrieval of data from MPQA corpus, we have chosen the tools independently. In doing so, our primary goal was not to improve their system, but to build an independent pipeline for experimenting with different dependency representations for opinion holder classification.

In figure 4.1, we give an overview of the prerequisite task, i.e. detection of opinion expressions. The tools for this task are described in Chapter 3. In the following, we describe our implementation, the tuning of our system, and the results.

4.1 INTERNAL REPRESENTATION

In order to relate the various layers of annotation, from MPQA as well as from morphosyntactic analysis, we adopt an internal representation in terms of character offset positions. In the following, we will describe this representation.

For each document in the document list (see Section 3.1.4), the raw text from MPQA is read, split into sentences with the sentence split offset positions from MPQA and tokenised with REPP, see Section 3.3.2. This is saved in a list structure: a list of sentences in the document, each consisting of lists of token objects with word forms and start and end offset positions. Then, the POS-tagger and lemmatiser is run and the information from these is added to the according token objects.

The annotation data from MPQA is saved in an ordered table, with the starting offset position as key. This includes overlapping and zero span expressions. This list is then sorted by the starting offset position. We then iterate throughout the list, building two hash tables with starting offset position as a key, where one holds

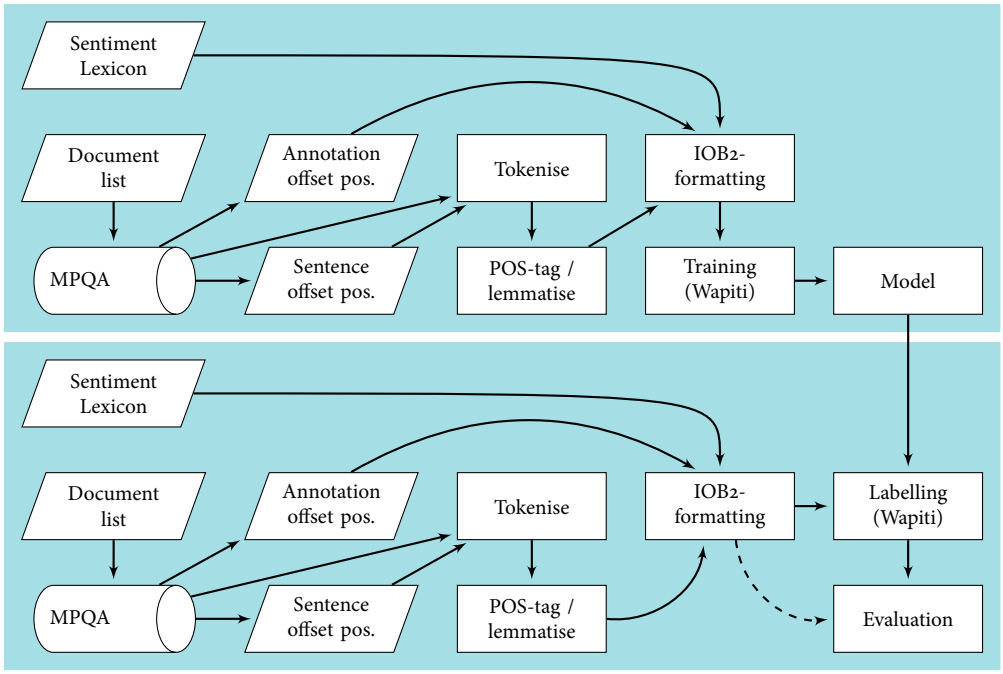


Figure 4.1: Pipeline for opinion expression detection.

the expressions we will use for training, and the other will hold expressions we are ignoring for the training, but include for evaluation purposes, as described in Section 3.6.

The sentiment lexicon, see Section 3.2, is read into a table with the word as a key. The lexicon contains some homonyms from different parts of speech, to handle that, each value is a table, with the part-of-speech tag as key and the values from the sentiment lexicon on the specific word as value.

4.2 EXPERIMENTAL SETUP

In Chapter 3, we elaborated on the background for using a CRF sequence labeler for opinion expression detection and the Wapiti toolkit implementation of CRF (Lavergne, Cappé, and Yvon 2010). In this section we will describe the setup for Wapiti.

Figure 4.1 shows the pipeline for training and testing. In addition to the training file in IOB2-format, we specify the feature selection in a pattern file. In the following, these two file formats are described.

Label	Expression type	Description
B-DSE	Direct-subjective expression	Beginning of expression
I-DSE	Direct-subjective expression	Inside expression
B-ESE	Expressive-subjectivity elements	Beginning of expression
I-ESE	Expressive-subjectivity elements	Inside expression
B-OSE	Objective speech-events	Beginning of expression
I-OSE	Objective speech-events	Inside expression
O	Not a part of an expression	Outside

Table 4.1: List of IOB2-labels used.

4.2.1 Training data in IOB2

A Wapiti training file should have one line for each token, with a tab or space delimited list of observations for this token. The last element on the list should be the label.

A format that follows these requirements and lets us represent expressions is IOB2 (Sang and Veenstra 1999). This format is commonly used for sequence labelling tasks, e.g. chunking and named entity recognition. Each expression type will be prefixed with either B (beginning) or I (inside). The first token in an expressive-subjectivity element, for example, will be assigned the label B-ESE, and the following tokens in the expressions will be given the label I-ESE. In table 4.2, the sentence from example (6), 'Terrorism thrives in an atmosphere of hate', provides an example of this format. There are four observations. Word form, lemma and POS-tag are parts of the token object in our internal structure, and can be retrieved directly. The only verb in the sentence, 'thrives', forms one expression, and the phrase 'atmosphere of hate' forms another. For prior polarity, if the lemma or the word form is a key in the subjectivity lexicon, the prior polarity from the lexicon is retrieved. In table 4.2, there are three tokens marked with a prior polarity, of which 'thrives' and 'hate' are parts of an expression, and 'Terrorism' is in close proximity of an expression.

Table 4.1 gives an overview of the labels employed for expression detection. The first column shows the labels which should be predicted. For each of the three expression types, direct-subjective expressions (DSE), expressive-subjectivity elements (ESE) and objective speech-events (OSE) we have two labels. The prefix B-, declares that the token is the beginning of the expression and the prefix I- states that the token is a part of the same expression as the previous token. Tokens not part of an expression will have the label O for outside. In total, this will give 7 labels.

We see that overlapping expressions annotated in MPQA, as we described in Chapter 3, cannot be represented within this format. When creating the IOB2-

Word form	Lemma	POS	Prior Polarity	Expression (label)
Terrorism	Terrorism	NNP	weak/ne	O
thrives	thrive	VBZ	weak/po	B-ESE
in	in	IN	-	O
an	a	DT	-	O
atmosphere	atmosphere	NN	-	B-ESE
of	of	IN	-	I-ESE
hate	hate	NN	strong/ne	I-ESE

Table 4.2: Example of IOB2-format from the training file.

file, we will ignore any new expression beginning before the end of the previous expression. For evaluation purposes, we will generate a file containing ignored expression. This has been described in detail in Section 3.6.

4.2.2 Pattern file

Wapiti uses a pattern file for combining observations from the data file to features for training. The format is almost identical to CRF++ templates[1]. A pattern, a single line in the pattern file, consist of three parts: type, identifier and commands. See figure 4.2 for an example of a pattern.

The first character in each line defines the type, and can be either `u` for unigram, `b` for bigram or `*` for both. We distinguish these feature template bigrams from word level N-grams, which we address in Section 4.3. In our example pattern, the unigram feature type is used.

The type is followed by an identifier. If the same identifier is used for several lines, the patterns in these lines are collapsed. This way, bag-of-words features can be created. Identifiers could be any string, but in our system, we will give deducible names to identifiers. The identifier in our example, `pos_i_n2o0`, is indicating that this is a N-gram feature of POS-tags, where N is 2 and there is no offset.

The main part of a line contains one or multiple commands. This is the part of the pattern expanded to an observation string. All commands refer to a column and a line offset relative to an active pattern. In figure 4.3, we see how the pattern in figure 4.2 is applied for a unigram. There are two commands, segregated by a static string. The first command, `%x[0,2]`, refers to the third column of the current token (the column count starts at 0). In the example, this command expands to `NN`. This is followed by a slash and the last command, `%[1,2]`, i.e. the value of the third column for the immediately following token, which expands to `IN`. The complete observation string this pattern expands to, for this token, is `NN/IN`.

[1] `https://taku910.github.io/crfpp` accessed 25 March 2016

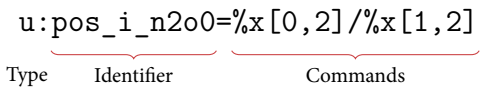


Figure 4.2: Example line from Wapiti pattern file. The type states if the pattern should expand to an unigram, a bigram or both. In this example, the pattern is a unigram pattern. The identifier can be compared to a variable name. In most cases, this is a unique string. For building bag-of-words features, the same identifier could be used. The commands are expanded by the information in the training file. In this example there are two commands, that will expand to respectively the POS-tags of the current and next word.

In addition to `%x[offset,column]`, which retrieves observations directly, there are two commands that support a subset of regular expressions. The command `%t[offset,column,pattern]` returns a boolean if the pattern is matched and `%m[offset,column,pattern]` returns the matched string.

According to the Wapiti manual, these three commands also come in a form that ignores the case of the observation, when the command name is given in uppercase.

4.3 FEATURE SELECTION EXPERIMENTS

To select the patterns for our system, we performed a series of feature selection experiments, where we investigated different combinations of features. Table 4.3 shows an excerpt of the preliminary results from this investigation. These results are from training/testing on the development set. After this experimentation, we made some minor changes on the system, and the current program code will give slightly different results. For the purpose of reproducing these results, the original development training and test IOB2-files are available on our project page.[2] The preliminary results presented here are the original results at the time of experimentation. The number in the first column refers to the pattern file name. Pattern 1 is a basic pattern file that provides unigram features for the current token for each observation in the training file. As we see in the table, with this pattern file, we get an F-score of 0.397.

In the following, we will provide give more details about some of the features.

4.3.1 N-grams of POS-tags

N-grams are commonly used in language technology tasks. They are sequences of words that allow us to estimate the probability of a word given the previous N words. For our feature selection experiment, we constructed patterns that give N-grams up to 5-grams. When creating features for a token i, the context both before

[2] Available along with the other source code at `https://github.com/trondth/master`

	0	1	2	3	4	
-4	Terrorism	Terrorism	NNP	-	O	
-3	thrives	thrive	VBZ	weak/po	B-ESE	
-2	in	in	IN	-	O	
-1	an	a	DT	-	O	
0	atmosphere	atmosphere	NN	-	B-ESE	← Current
1	of	of	IN	-	I-ESE	
2	hate	hate	NN	strong/ne	I-ESE	

Figure 4.3: Example of a Wapiti Pattern. The pattern `u:pos_i_n2o0=%x[0,2]/%x[1,2]` is a a unigram pattern. The pattern will expand to the observation string `NN/IN` for the current token in this example.

and after the token is interesting. Because of this, we created N-grams with offsets that give a window of $\pm$N from i.

In figure 4.3, the trigram of POS-tags for the current token will expand to three observation strings (with a feature template unigram, see Section 4.2.2): `IN/DT/NN`, `DT/NN/IN` and `NN/IN/NN`.

We expect that N at one point will be so large that it will create a sparse data problem. Our investigation will seek to find the best length of N for this task. In table 4.3, experiments number 14–18 (POS N-gram), we see that a 2-gram gives the best F-score on the development set compared to other lengths of N with an F-score of 0.492. This indicates that this is an optimal N-gram length for this task.

4.3.2 POS-N-gram/lemma

We also considered some features based on the lemma. One possibility is to combine a POS-tag N-gram with the current lemma. This will give information of the structural context of the lemma. We tried this out with different length of N, as shown in table 4.3, no. 20–23 (POS N/lemma). The best of these combinations obtained an F-score of 0.495. As we can see, the effect of this feature is very limited.

4.3.3 Lemma/POS-tag bag-of-words

Another widely used representation is bag-of-words, in which word order and grammatical structure are ignored. We created bag-of-words with a combination of lemma and POS-tag from the previous tokens. Experiment 35 in table 4.3 shows that this feature improves the F-score to a value of 0.508.

4.3.4 Proximity of prior polarity

We expect the prior polarity of a word to imply the appearance of an opinion expression in the proximity of this word. To retrieve this information, we made

No.	Description	P	R	F	Time (s)
1	Unigram from each column	0.608	0.295	**0.397**	12
14	POS 1-gram	0.593	0.392	0.472	77
15	14 + POS 2-gram	0.605	0.415	**0.492**	125
16	15 + POS 3-gram	0.583	0.409	0.481	213
17	16 + POS 4-gram	0.563	0.404	0.470	387
18	17 + POS 5-gram	0.572	0.401	0.472	857
20	14 + POS 1/lemma	0.587	0.391	0.469	112
21	15 + POS 2/lemma	0.590	0.420	0.491	260
22	16 + POS 3/lemma	0.593	0.424	**0.495**	712
23	17 + POS 4/lemma	0.597	0.396	0.476	1143
25	14 + POS 1/polarity_i	0.599	0.386	0.470	100
26	15 + POS 2/polarity_i	0.593	0.420	**0.491**	165
30	20 + POS 1/polarity_i	0.590	0.393	0.472	127
31	21 + POS 2/polarity_i	0.584	0.421	**0.489**	286
32	22 + POS 2/polarity_i	0.589	0.405	0.479	754
33	23 + POS 2/polarity_i	0.578	0.404	0.476	1634
35	15 + Lemma/POS bag	0.588	0.447	**0.508**	264
36	22 + Lemma/POS bag	0.591	0.433	0.499	712
37	26 + Lemma/POS bag	0.583	0.441	0.502	272
38	31 + Lemma/POS bag	0.582	0.436	0.499	399
42	35 + Polarity/POS bag-of-words	0.580	0.439	0.500	359
43	42 + offset	0.598	0.439	0.506	461
45	15 + First letter case	0.601	0.411	0.488	133
46	15 + Polarity, string	0.611	0.402	0.485	138
47	15 + Polarity, boolean	0.623	0.409	0.494	188
48	35 + Polarity, boolean	0.601	0.435	0.504	326

Table 4.3: Investigation for feature selection. The evaluation is done with the development set and retrieved with an early version of the evaluation script. The first column refers to the experiment number and the corresponding pattern file name. For each experiment, except experiment 1, both feature level unigram and bigram are being used. Detailed explanation of features is found in Section 4.3. We have highlighted the most interesting results, which we explain in detail in the text. Experiment 35 gives the best overall result, and we used this in further experiments.

Evaluation metrics	P	R	F
Ignoring overlaps	0.605	0.450	0.516
Including overlaps	0.621	0.441	0.515

Table 4.4: Held-out experiments.

two experiments where the prior polarity of the 4 nearest words is selected as a feature, one where the observation string is a boolean, stating if the word has prior polarity or not and another where the prior polarity is used as an observation string. In table 4.3, experiments 46 and 47, we see that these have a limited positive effect on the precision. Of these, experiment 47, where the observation string is a boolean, has the best performance with an F-score of 0.494. But combing this feature with the best performing experiment at this point, experiment 35, showed an F-score of 0.504, which is lower than experiment 35 alone.

4.3.5 Polarity/POS-tag bag-of-words

For another experiment, we combined the prior polarity of a word with the POS-tag for nearby words in a bag-of-words feature. For prior polarity alone, a bag-of-words feature seems to give a sparse data problem. By combining prior polarity and POS-tag, we avoid this problem. In table 4.3, experiment 42 shows a negative effect of expanding experiment 35 with this feature. This gives an F-score of 0.500 which is lower than experiment 35 alone. We also tried a variant with additional offsets for prior polarity/POS-tag bag-of-words, but this also gave a lower F-score than experiment 35 alone, with an F-score of 0.506.

4.4 SELECTING A SETUP

In our investigation, experiment 35 gave best performance, and we will use the pattern file used in this experiment in our system. We also tried running experiments on different learning methods in Wapiti. The learning methods should converge around the same maximum, and as expected, all the setups gave results very similar to the results in table 4.3. The default learning algorithm, L-BFGS, gave the fastest performance, and we chose to stick with this default algorithm.

4.5 HELD-OUT EXPERIMENTS

Running the system in its best-performing configuration (corresponding to experiment 35) on held-out data gives a slightly better F-score than the experiment on the development set. Table 4.4 shows an F-score of 0.516 with the same evalu-

ation metrics as for the development set. With the alternative evaluation metrics, described in Section 3.6, we get an F-score of 0.515. As expected, the precision is higher and the recall lower when including the overlaps. The results indicate that the system generalises well to new data.,

Our results are somewhat lower than the baseline of Johansson and Moschitti (2013), which we presented in Section 2.2.2. Their baseline F-score of 53.8 is clearly higher. The main difference between our system is the choice of sequence labeler.

We can expect this somewhat lower performance also to effect the overall performance after opinion holder classification.

Chapter 5

Opinion Holder Classification

So far, we have provided the background for opinion analysis and constructed a system for opinion expression detection. We will now turn to the main question addressed in this thesis, namely the comparison of how dependency representations perform in the task of opinion holder classification.

The task of opinion holder detection is, according to Johansson and Moschitti (2013), similar to argument detection in semantic role labelling. For DSEs (direct-subjective expressions), the task is often straightforward. The opinion holder for such expressions will often occur as a direct semantic argument. For the other expression types, more complex relations are frequent.

In this chapter, we will first describe our pipeline in details, before we give some statistics on the features in our system. We will then run the system on the development set and discuss the findings. Then, we will use the data from these experiments for an error analysis. Finally we will apply the classifiers on the held-out test set in order to compare the performance of our system with that of Johansson and Moschitti (2013), and to get an indication of how well our system generalises.

5.1 SYSTEM OVERVIEW

Our system for opinion holder detection also builds on the work of Johansson and Moschitti (2013). This part of the system has a more complex pipeline than opinion expression detection. In the following, we give a overview of the task, before we give a detailed description of each part in this system. The source code is available on the project github page.[1]

In figure 5.1, we give a simplified illustration of the pipeline for opinion holder classification. The necessary information from MPQA was already retrieved in the

[1]Project github page `https://github.com/trondth/master`

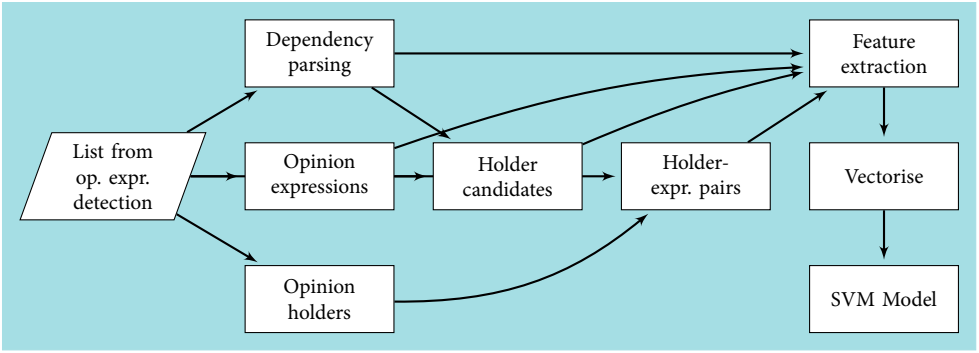


Figure 5.1: Simplified pipeline for opinion holder classification.

task of opinion expression detection. We extend the data structure from this task with syntactic information from dependency parsing, with different dependency representations. We continue to use the partitioning in training, testing and development from Chapter 4 in this task.

From each sentence we extract three lists, as illustrated in figure 5.1: opinion holders (for training and testing), opinion expressions (gold and system detected) and holder candidates. From the lists of opinion holders and opinion expressions, we will make a list of expression-holder pairs.

Then, for each expression, we will create features for each holder candidate. This information is vectorised, and an SVM model is built. This is also illustrated in figure 5.1 In addition, we will create separate classifiers for the special cases of the holder being the writer, or where the holder of an expression is not mentioned explicitly in the sentence.

In the following we will go into detail for each task.

5.2 INTERNAL STRUCTURE

We build further upon the internal structure described in Chapter 4, a list of sentences with a list of tokens where each token is a table with all available information. For the test set, the predicted expressions are added to this data structure.

We will run experiments both with the predicted expressions from Chapter 4 and with gold expressions. As our main goal is to compare different dependency representations, we will base this investigation predominantly on the output from running the system with gold expressions.

5.3 DEPENDENCY PARSING

As explained in Section 3.4, we first developed our system with the LTH SRL dependency parser and semantic role labeler, developed by Johansson and Nugues (2008). This parser is distributed with models for English.

Both the input and the output data for this parser are in the CONLL-2008 format, see table 5.1. We created functions to create CONLL-2008 files from our internal list structure, and read the format into our internal list structure. The parser was run with default settings.

For the comparison between different dependency representations, the Bohnet & Nivre parser (Bohnet and Nivre 2012) with models provided by Ivanova (2015) was used. This parser performs joint POS tagging and dependency parsing. As explained in Section 3.4, where we provide more details about the parser and our setup, we executed the parser with the same parameters used by Ivanova (2015). The parser uses a different CONLL format, based on the CONLL-2009 format. Each line represents one token and has at least 13 columns. Sentences are separated with an empty line. For input to the parser, we only need to provide information in the first two columns: word id and word form. The rest of the columns can be left blank, indicated by an underscore for each field. From the output, we can retrieve the POS tag, the dependency head word id and the dependency label from columns 6, 10 and 12 respectively. As we have POS tags available from the first part of our project, we will not use this information. The head and label, on the other hand, are read into our list structure.

5.3.1 Dependency representations

We will here provide a short recap of the different dependency representations described in Section 2.3.

In our system, we will compare the results when using the three different dependency representations: *CONLL Syntactic Dependencies* (CD), *Stanford Basic Dependencies* (SB) and *DELPH-IN Syntactic Derivation Tree* (DT).

A key difference between them is the head status. While CD and DT tend to have functional words as heads, SB to a large extent assigns head status to content words. This can have a large effect on syntactic features between the opinion expression and the opinion holder. For our purposes, one important exception to this tendency must be taken into consideration: In all three dependency representations, the determiner and the adjective are considered dependents of the noun. A difference here would have skewed the selection of holder candidates, which in large parts depends on the dependency structure of the noun phrases.

The treatment of coordination is another difference between the dependency representations, and this may have an impact on the selection of the opinion holders in noun phrase conjunctions. The conjunction is the head of the conjuncts in DT, while the first conjunct is the head in CD and SB.

	CoNLL-2008		CoNLL-2009	
Col.	Name	Example	Name	Example
1	ID	2	ID	2
2	FORM (unsplit)	reported	FORM	reported
3	LEMMA	report	LEMMA	_
4	GPOS	_	PLEMMA	_
5	PPOS	VBD	POS	_
6	SPLIT FORM	reported	PPOS	VBD
7	SPLIT LEMMA	report	FEAT	_
8	PPOSS	VBD	PFEAT	_
9	HEAD	0	HEAD	-1
10	DEPREL	ROOT	PHEAD	0
11	PRED	report.01	DEPREL	_
12	ARG	_	PDEPREL	ROOT
13	ARG	_	FILLPRED	_
14	ARG	_	PRED	_
15…	ARG…	_	APREDs	

Table 5.1: Comparison of the CoNLL-2008 and CoNLL-2009 data formats. CoNLL-2008 is used by the LTH SRL parser. The dependency head and dependency relation label is retrieved from columns 9 and 10. Column 11 indicates a predicate for the semantic role labelling, and the rest of the columns are arguments for the semantic role labelling. CoNLL-2009 is used by the Bohnet & Nivre parser. Column 10 contains the dependency head, and column 12 the label.

5.4 EXTRACTING EACH EXPRESSION/HOLDER PAIR

As described in Section 3.1.1.1 and 3.1.2, MPQA does not make a direct link between an expression and its holder, but instead between the holder and a nested source list. The first occurrence of a single holder in a document is tagged with an *id*, the following are tagged with a *nested-source*, referring to the id of the first occurrence. MPQA also contains 588 instances of opinion holders with neither a nested source list nor an id.

Our system works at the sentence level, and therefore we will treat a holder id and a holder nested-source equally. In many cases, multiple words in the same sentence will represent the same holder. To extract these pairs, we will use the following procedure.

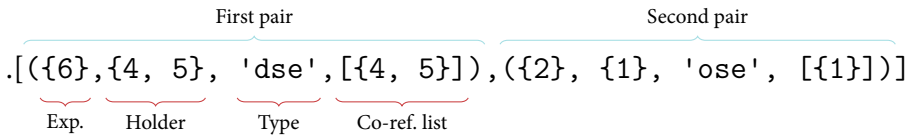


Figure 5.2: In the excerpt from the example sentence, which we see in figure 5.3, there are two expressions, both with holders that explicitly appear in the sentence. This will give a list of two tuples, where each tuple gives information about the expression (Exp.), the opinion holder (Holder), the expression type (Type) and a co-reference list (Co.ref list).

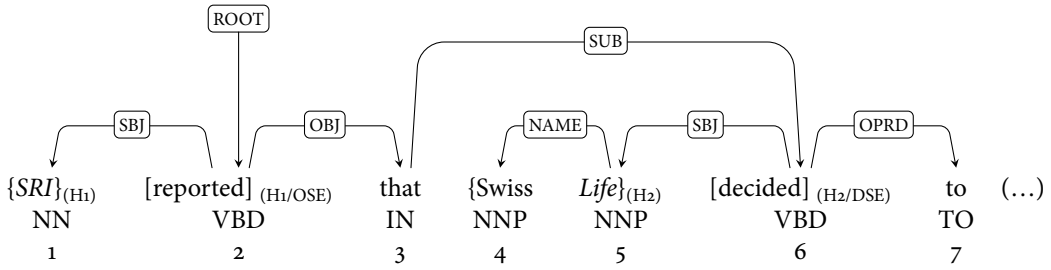


Figure 5.3: Holder candidate selection. An excerpt of the example sentence parsed with the LTH SRL parser and tagged with opinion holder/expression. We select the nouns, words that have a POS tag starting with 'NN', with the exception of words dominated by another 'NN' or words part of the expression type for which we select holder candidates.

- We create a table containing a set of word ids for each opinion holder (in MPQA annotated as GATE_agent) in the sentence. If there is a nested source for the holder, the last element in the nested source list will be the key for the holder. Otherwise, the holder id will be used as a key if a holder id exists; if not, this holder is ignored. If a key already exists, the word id will be added to a set containing all word ids representing the same opinion holder.
- For each opinion expression, if the last element in the nested source list exists we will append a 4-tuple of the expression, its holder, expression type and co-reference list. If the last element is 'w' (writer) or 'implicit', the second element in the tuple will be this string, otherwise this element will be a set of word ids. If the last element of the nested source does not exist in the table of holders for the sentence, the opinion holder for the expression will be treated as 'implicit'.

In figure 5.3, there are two expressions, 'reported' and 'decided', each with one opinion holder, respectively 'SRI' and 'Swiss Life'. That gives the list of tuples shown in figure 5.2, representing the expression/holder pairs in the sentence.

Holder is not a candidate[a]	DT	SB	CD
Not overlapping with expression	14	11	12
Not part of same expression type	66	95	89
Not part of an expression	120	175	151

[a] Exclusive external holders (implict/w).

Table 5.2: Number of holders that are not part of a candidate in the development test set, when the system is run with gold expressions. The total number of expression-holder pairs is 2516.

5.5 SELECTION OF HOLDER CANDIDATES

If an expression has an explicit opinion holder, and this appears in the sentence, the holder will most likely be one of the noun phrases in the sentence. For each expression, all the noun phrases in the sentence are instances for the classifier.

Since noun phrase constituents are not native to dependency structures, we will formulate a heuristic that selects noun phrases in terms of dependency relations. In Johansson and Moschitti (2013), the holder candidates are tokens tagged as nouns, i.e. having a POS-tag starting with NN or PRP, that are not a part of an opinion expression or directly dominated by another candidate token. (Johansson p.c.)

In figure 5.3, we see that there are three tokens starting with NN; *SRI*, *Swiss* and *Life*, where *Swiss* is directly dominated by *Life*. We will therefore have two opinion holder candidates, *SRI* and *Life*.

We will create a list of holder candidates, where each holder candidate is the head of a subtree, representing potential holders. For each expression type, we do this by selecting all words where the POS-tag starts with NN or PRP. We will then restrict this list by removing words that are directly dominated by another noun phrase. In addition, we will remove words that are part of an opinion expression.

It is not apparent from Johansson and Moschitti (2013) how disjointness of opinion expressions and holders was ensured. We will therefore investigate three variants of this restriction:

1. Restrict holder candidates to NP heads that are not part of any expression.
2. Restrict holder candidates for an expression e to NP heads that are not part of an expression of the same expression type as e.
3. Restrict holder candidates for an expression e to NP heads that are not part of e.

In table 5.2, we see that the effect of these restrictions varies between the different dependency representations. Following the algorithm described above, DT covers a larger number of actual holders than the other representations. One

should note, however, that we only count expression-holder pairs for this table where there are no overlap between a holder and a candidate as an error. We will come back to the question of partial overlap in Section 5.11. We also see that a restriction of holder candidates to holders where the heads are not a part of any expression leads to more holders without a candidate. We will return to the question of how selection influences the performance in Section 5.10.2.

After classification, the corresponding noun phrase for the selected holder candidate is found by collecting all tokens in the subtree of the token.

5.6 FEATURE SET

The feature set used in this task is the same as the baseline from Johansson and Moschitti (2013). This section contains a description of the feature templates and methods we have used for extracting them. We will use the DSE in figure 5.3, *decided*, as a running example. We will show the feature types for this expression with respect to each of the two holder candidates in figure 5.3, *SRI* and *Life*.

5.6.1 Syntactic path

With this feature, we want to express the syntactic relationship between the expression and opinion holder.

The feature represents the path in the dependency tree between the holder candidate and the expression. For the running example, we will extract the following features for the opinion holder candidates *SRI* and *Swiss*: `SBJ↑OBJ↓SUB` and `SBJ↓`.

For all features involving graph traversal, we made use of the csgraph module for the python library SciPy.[2] We converted the dependency graph to a two-dimensional array and created a csgraph object of this array, giving us access to code finding the shortest path between two words.

5.6.2 Expression head word, POS and lemma

Similar to what one sees in semantic role labelling, different expressions often have different relations to their opinion holders. In semantic analysis, the agent could both appear as the subject and the object of different predicates. Because of this, we need features that help the classifier separate between different expressions.

We will expect an opinion expression to be connected in a dependency graph, therefore finding the head word of expressions can be done straightforwardly. We traverse over each word in the expression, returning the first word with a head word that is not part of the expression.

[2] `http://docs.scipy.org/doc/scipy/reference/sparse.csgraph.html`

(a) Unambiguous. (b) Ambiguous.

Figure 5.4: The head of an expression can be ambiguous depending on the dependency representation.

A possible problem when selecting the expression head word is that it can be ambiguous, depending on the dependency representation. In figure 5.4, we illustrate this problem. If 'foo bar' is an expression, then both 'foo' and 'bar' will have heads outside the expression in (b). In this case, we will select the first word with its head outside the expression as head word, 'foo'. In (a), 'bar' is the only token in expression with head outside expression, and is therefore head of the expression.

We create features for word form, POS and lemma for the head word of the expression. In figure 5.3, both expressions consist of a single token, and it is thus the head word of the expression.

For *decided*, we will have the features `decided`, `VBD` and `decide`, and since this expression consists of a single word, that word will also be the expression head.

5.6.3 Holder candidate head word and POS

Similar to the above features, we will extract features for word form and POS for the holder candidate. For the first holder candidate in figure 5.3, this is `SRI` and `NN`.

5.6.4 Holder candidate context words and POS

The words or part-of-speech of words in direct proximity to an opinion holder candidate could indicate whether the candidate is an opinion holder.

These features are extracted from the immediately preceding and succeeding words. As the first holder candidate in figure 5.3 is the first word in the sentence, only the features from the following word are extracted, `reported` and `VBD`

5.6.5 Dominating expression type

If the expression is syntactically dominated by another expression, we will extract a feature representing this. We will only extract this feature when a direct head of the expression is part of an expression.In cases where there are several dominating expression types, a compound of these features in alphabetical order is used as feature type, for example `DSEESE`.

In figure 5.3, *decided* is directly dominated by *that*, which is not a part of any expression. Therefore, we will not extract this feature for this example.

5.6.6 Expression verb voice

The dependency relation of the holder for an expression with the passive verb voice will most likely differ from a sentence with active voice.

We extract a feature that tells us whether there is an active or a passive voice in the expression. If the following criteria are fulfilled, this feature will have the type `Passive`:

- One of the tokens in the expression must be a participle (the POS tag must be VBN)
- One of the tokens in the expression must be the lemma *be*
- The head of the VBN has to be the lemma *be*.

This feature returns three possible types, `Active`, `Passive` and `None`, the last is extracted for expressions without verbs.

5.6.7 Expression dependency relation to parent

From the expression head word, we will extract a feature for the label of the dependency relation of the parent. In figure 5.3, the label of the edge from *decided* to its head *that* is `SUB`.

5.6.8 Shallow semantic relation

In the comparison between the different dependency representations, we will not use semantic information. But to be able to compare our results with Johansson and Moschitti (2013), and to show the effect of this feature, we will use this feature for the purpose of illustration.

For the two holder candidates in figure 5.3, LTH SRL only detects any semantic relation between *decided* and *SRI*, `A0`. Between *decided* and *Life*, the LTH SRL parser did not recognise a semantic relation.

5.7 SCIKIT-LEARN

As explained in Section 3.5.1, we will use linear SVM for the opinion holder classification. We will make use of the implementation in the python toolkit Scikit-learn. In addition to the LIBLINEAR library, Scikit-learn also includes a vectoriser, DICTVECTORIZER, that can create a one-hot coded matrix (described in Section 3.5.1.1) based on a list of dictionaries.

5.7.1 Building the classifier

When constructing the classifiers for each expression type, we construct two lists representing the pairs between expressions and their holder candidates. One list

will contain the labels for classification, true or false. The other list contains a dictionary with features for each possible expression-holder pair. This list of dictionaries is then converted (fitted) to a matrix with the vectoriser in Scikit-learn, and the label list is converted to an array. Then, the SVM model is created.

Similarly, for the prediction, a list of dictionaries for the pairs to predict is built. The vectoriser in Scikit-learn can transform this list to a one-hot encoded matrix with the same columns as in the matrix from the training. Then, we can use this matrix as input for prediction with the SVM model.

For each expression type, we will have three classifiers, one for *explicit* holder candidates, one for *implicit* holders and one for *writer* as holders. We want exactly one opinion holder per possible expression-holder pair, and will select the one with the highest SVM score. This was also the strategy employed in the work of Johansson and Moschitti (2013).

Specifically, we will employ the following:

1. First, for each expression, we will choose the explicit holder candidate with the highest SVM score.
2. Then, if the classifier for implicit holders classifies the holder for the current expression as *implicit*, we will compare the SVM scores of the two, and select the one with the highest score.
3. Finally, if the classifier for writer as holder classifies the holder as *writer*, we will compare the SVM score for writer with the SVM score from 2.
4. A special case obtains, when the opinion holder is neither *writer* or *implicit*, and there are no holder candidates for the expression. These cases, the system will treat as implicit.

5.8 EVALUATION

The evaluation metrics for opinion holder classification is described in Section 3.6.

- First we find the gold expression e' that has the largest overlap with the proposed expression e with $e' = \arg\max_x c(x, e)$ regardless of expression type.
- Second, we select the gold holder for this expression that has the largest overlap with the proposed holder and determine the span coverage with $c(h', h)$ and $c(h, h')$, respectively for precision and recall, where h is the proposed holder and h' is the gold holder, as explained in Section 3.6.
- From the sum of span coverages and number of pairs, the precision and recall are determined.

One consideration we have had is the treatment of a situation where a system expression does not have any overlap with a gold expression. In the development corpus, there was 270 cases where there were no overlap between the predicted expression and a gold expression. These cases where counted as false positives.

We will return to this in Section 5.8.1.2.

Another consideration is whether we count gold expressions that have overlap with each other, as explained in Section 3.6. For opinion expression detection, we chose to report a number both with and without the overlapping expressions. This had an effect on the precision and recall, but the F-score was almost the same in both cases. For the holder classification task, we may not see the same effect, because the matching of a single pair is independent of the matching of other pairs. Still, we will take the number of overlapping expressions into consideration, although it is not possible for our opinion expression system to detect them.

5.8.1 *Lists for comparison*

To evaluate, a list of tables is created, with one table for each detected expression. Each table holds an expression, the proposed holder for this expression and the gold holder with the largest overlap to the proposed holder, in addition to the SVM confidence score. We use this list to calculate the span coverage for each expression-holder pair.

To find the precision and recall, we will divide the sum of the span coverage for all pairs by the number of possible pairs. Although this seems like a trivial task, there are several considerations to be made.

In the following, we will describe our method for performing those calculations.

5.8.1.1 *Number of gold pairs*

For recall, we need the number of gold expression-holder pairs. As explained in Section 3.6.1, the pairs are not found directly in MPQA, but derived through the available information. We cannot simply use the list of gold expressions, because this will include expressions that are not annotated with a holder. Also, we cannot use the full list of gold holder-expression pairs, as our task is not to find every holder in the co-reference chain, but only one holder per expression. We will therefore count gold expression-holder pairs, but only count the first occurrence for each expression. As explained above, overlapping expressions will also be counted, although our system of opinion expression detection is unable to handle overlaps. There were 96 occurrences of overlap of holder-expression pairs in the development set.

We also need to ensure that recall was calculated based on the same number of expression-holder pairs, independent of the dependency representation, in order to have a correct comparison of the performance of the dependency representations.

In table 5.3, we see that for our development set, the number of gold pairs is 2516 in all dependency representations.

	DT	SB	CD
Gold pairs	2516	2516	2516
System pairs (gold ex)	2516	2516	2516
Correctly detected ex.	1357	1357	1357
Falsely detected ex.	270	270	270
System pairs (sys. ex)	1627	1627	1627

Table 5.3: Number of gold/system pairs in the development test set for each dependency representation. The number of system pairs depends on the number of expressions. There is no variation between the dependency representations.

5.8.1.2 Number of system pairs

For precision, we need to have the number of proposed system pairs. We cannot use the length of the list created for evaluation directly, because this list will not include detected expressions with no overlap to a gold expression. We will name these falsely detected expressions. Instead, we will base the number on detected expressions.

If we want to run the system for only internal holders (explicitly mentioned in the sentence), we will not be able to classify holders for expressions without holder candidates. In this case, the system length should be adjusted accordingly.

When running the full system on gold expressions, the number of system expressions-holder pairs will be equal to the number of expressions. In table 5.3, we see that for the full system, for all dependency representations, we get the same number of system pairs.

5.8.2 Statistical significance

To get an indication on whether or not a difference in the evaluation results is a result of a coincidence due to the particular test set, we should perform statistical hypothesis testing or significance testing.

In statistics, there are some traditional approaches to significance testing, for example Student's t-test. Most of these approaches are based on the assumption that the data are normally distributed, which for natural language processing often is not the case (Berg-Kirkpatrick, Burkett, and Klein 2012; Søgaard et al. 2014). Other measures, for example Wilcoxon signed-rank test (Wilcoxon 1945) or the bootstrapping methods (Efron and Tibshirani 1994), do not presume normally distributed data.

Bootstrapping refers to a multitude of tests that rely on resampling of data. For our purpose, we can implement bootstrapping by generating a large number

of pseudo test sets with Monte Carlo resampling of the original test set.

Let us consider a tiny example with an excerpt from our development results, and perform a pairwise significance test on this. In table 5.4, the first two rows shows the span coverage (for precision) from a set x with 5 pairs using respectively CD and DT. We see that the precision for CD is higher than for DT. The difference in precision gives an $\delta(x)$-value, which in the case of the example is 0.18.

We first formulate a *null hypothesis*, that CD is not really performing better than DT, that the higher precision was a coincidence. If we had a large number of test sets, and the null hypothesis was true, we could expect that occasionally, the precision with CD was $\delta(x)$ better than the precision with DT. But because we do not have a large number of test sets, we will estimate this by the means of bootstrapping.

We will make a number of resamplings of the test set, by randomly selecting pairs (with replacement) from the main sample. We will then find a $\delta(x^{*(i)})$ for each of these samples. Because these resamplings are created out of the x, the expected $\delta(x^{*(i)}$ is close to $\delta(x)$ (not to 0. Therefore, we will have to shift the means of these samples by $\delta(x)$, so we will count the occurrences where $\delta(x^{*(i)})$ is higher than $2 \times \delta(x)$. Finally we will find the p-value by dividing this count on the number of resamples. In the example in table 5.4, $-0.02 \not> 2 \times 0.18$ and $0.38 > 2 \times 0.18$. This will give a p-value of 0.5. When running the bootstrapping significance test with 10 000 resamples (which we will be using in later testing), we got a p-value of 0.2315, which indicates that there is 23.15 % chance that the null hypothesis is correct, and we should not consider the difference in precision from these five pairs significant.

Before we leave significance, we should stress that even though a difference in performance between the otherwise same system run with DT and with CD is statistic significant, a change in the system, for example in the feature set, may benefit one over the other. Different results, no matter if they are significant or not, is not enough for us to draw general conclusions for all systems.

5.9 FEATURE STATISTICS

In this section, we provide some statistics on features extracted from the training and development set combined. We will focus on the features that vary across dependency representations, i.e. the syntactic features. First we will give an overview of the features.

Table 5.5 presents the number of feature types for internal expression-holder pairs in the training and development sets. There is a large variety in the number of types between the feature templates. Only syntactic path and expression dependency relation to the parent have a different number of types using the different dependency representations. We see that while some features have a small (some-

	Span coverages						
	1	2	3	4	5	Precision	$\delta(\cdot)$
x_{CD}	1.00	0.00	1.00	1.00	0.09	0.62	0.18
x_{DT}	1.00	0.00	0.00	1.00	0.21	0.44	
$x^{*(1)}_{CD}$	0.09	0.00	0.00	0.00	1.00	0.22	−0.02
$x^{*(1)}_{DT}$	0.21	0.00	0.00	1.00	0.00	0.24	
$x^{*(2)}_{CD}$	1.00	1.00	1.00	1.00	0.09	0.82	0.38
$x^{*(2)}_{DT}$	1.00	0.00	0.00	1.00	0.21	0.44	
...							
$x^{*(b)}$							

Table 5.4: Pairwise significance testing with bootstrapping. In this small example, we want to test the significance of the results that CD has higher precision than DT. From the original test set x, a number of resamples is created.

	DT	SB	CD
Syntactic path	3724	3052	3929
Exp. deprel. to parent	38	41	27
Expression head word	3033	3463	3077
Expression head lemma	2144	2424	2182
Expression head POS	35	37	35
Candidate head pos	32	30	28
Candidate head word	1686	1737	1731
	DT/SB/CD		
Dominating expression type			3
Expression verb voice			6
Context right POS			37
Context right word			1048
Context left POS			38
Context left word			2262

Table 5.5: Number of feature types for internal expression-holder pairs in the training and development sets. There is a total number of 16267 pairs with internal holders in these two sets.

	Most common	DT	SB	CD
Syntactic path	SB-HD↑	5047		
Exp. deprel. to parent	HD-CMP	6697		
	Most common	DT	SB	CD
Syntactic path	nsubj↑		5804	
Exp. deprel. to parent	root		4814	
	Most common	DT	SB	CD
Syntactic path	SBJ↑			5518
Exp. deprel. to parent	ROOT			4535
	Most common	DT	SB	CD
Expression verb voice	Active	10 976	11 485	10 958
Dominating expression type	dse	1337	3068	1706
Expression head word	said	1644	1694	1644
Expression head lemma	say	2025	2104	2024
Expression head POS	VBD	4591	4138	4620
Candidate head POS	NNP	5450	5765	5732
Candidate head word	he	1280	1282	1282
Context right POS	NNP	3322	3142	3216
Context right word	,	1462	1448	1428
Context left POS	VBD	5512	5537	5565
Context left word	said	2128	2145	2156

Table 5.6: The most common feature types for each feature template using different dependency representations. The total number of expression-holder pairs is 16 267.

times fixed) number of possible types, others vary a lot. Some of the features have the same number of types independent of the dependency representation, but many of the features rely on syntactic information. This includes *syntactic path* and *expression dependency relation to parent*, in addition to the features extracted from the expression and the candidate head. There are a total number of 16 267 expression-holder pairs where the holder explicitly appears in the sentence.

The most common feature type, and its count, will give us a further indication about differences between dependency representations. In table 5.6, we give such an overview. In table 5.6, we list the most common feature type for each feature and their count. For expression head word, lemma, and OSE, candidate head word and OSE, and context words and OSE, there are small differences between the dependency representations. The most common expression head word is *said*, the

DT		SB		CD	
Synt. path	Count	Synt. path	Count	Synt. path	Count
SB-HD↑	5047	nsubj↑	5804	SBJ↑	5518
SB-HD↑HD-CMP↓	679	nsubj↑ccomp↓	889	NMOD↑	756
SB-HD↑HD-CMP↓HD-CMP↓	472	poss↑	548	SBJ↑OBJ↓	582
HD-CMP↑	395	pobj↑prep↑	488	SBJ↑ADV↓	268
HDN-AJ↓	269	nsubj↑xcomp↓	253	PMOD↑NMOD↑	266
SP-HD↑	263	nsubj↑conj↓	231	SBJ↑VC↓	256
HD-CMP↑HDN-AJ↑	250	nsubj↑ccomp↑	221	NMOD↓	233
SB-HD↑HD-CMP↓HD-CMP↓HD-CMP↓	247	rcmod↓	200	SBJ↑COORD↓CONJ↓	183
SB-HD↑HD-AJ↓	195	nsubj↑prep↓pobj↓	173	SBJ↑OBJ↑	181
SB-HD↑HD-AJ↓HD-CMP↓	151	nsubj↑dobj↓	155	SBJ↑OBJ↓SUB↓	175

Table 5.7: The most common types for syntactic path, all expression types.

		Average length		
	Int. holders	DT	SB	CD
DSE	8624	2.40	1.86	2.30
ESE	4472	5.43	3.87	5.15
OSE	3171	2.07	1.73	1.97
Total	16 267	3.17	2.39	3.02

Table 5.8: Syntactic path: Average path length and number of feature types.

DSE		ESE		OSE	
Synt. path	Count	Synt. path	Count	Synt. path	Count
SB-HD↑	3574	SB-HD↑HD-CMP↓HD-CMP↓	302	SB-HD↑	1457
SB-HD↑HD-CMP↓	394	SB-HD↑HD-CMP↓	217	HD-CMP↑	188
HDN-AJ↓	222	SB-HD↑HD-CMP↓HD-CMP↓HD-CMP↓	188	SB-HD↑HD-CMP↑	106
SP-HD↑	212	SB-HD↑HD-CMP↓HD-CMP↓HD-CMP↓HD-CMP↓	82	SB-HD↑HD-CMP↓	68
HD-CMP↑	205	SB-HD↑HD-AJ↓HD-CMP↓	71	SP-HD↑	49

Table 5.9: The most common types for syntactic path, DT.

DSE		ESE		OSE	
Synt. path	Count	Synt. path	Count	Synt. path	Count
nsubj↑	4149	nsubj↑ccomp↓	679	nsubj↑	1643
poss↑	418	nsubj↑prep↓pobj↓	100	nsubj↑ccomp↑	167
pobj↑prep↑	386	nsubj↑ccomp↓aux↓	79	poss↑	126
nsubj↑ccomp↓	191	nsubj↑ccomp↓advmod↓	78	pobj↑pcomp↑	109
rcmod↓	172	nsubj↑ccomp↓conj↓	77	pobj↑prep↑	93

Table 5.10: The most common types for syntactic path, SB.

DSE		ESE		OSE	
Synt. path	Count	Synt. path	Count	Synt. path	Count
SBJ↑	3896	SBJ↑OBJ↓	402	SBJ↑	1606
NMOD↑	587	SBJ↑OBJ↓SUB↓	144	NMOD↑	164
PMOD↑NMOD↑	222	SBJ↑OBJ↓PRD↓	112	SBJ↑OBJ↑	147
SBJ↑VC↓	217	SBJ↑ADV↓PMOD↓	93	PMOD↑PMOD↑	108
NMOD↓	196	SBJ↑OBJ↓ADV↓	90	SBJ↑ADV↓	88

Table 5.11: The most common types for syntactic path, CD.

most common candidate head OSE is *he*.

There are three features that stand out in this context: *syntactic path*, *expression dependency relation to parent*, and *dominating expression type*. For syntactic path, the most common feature type in the different dependency representations is not the same string, but all these strings represent the same path: a direct link from the subject. For DT, this is SB-HD↑, for SB: nsubj↑, and for CD: SBJ↑. The count varies, but compared to the total number of pairs, 16 267, only roughly 1/3 of the expression-holder pairs have this feature type.

For the expression dependency relation to parent, HD-CMP (combining a head with a complement, used for the syntactic arguments of a range of head categories, including verbs, prepositions, relational nouns and adjectives, and others) is most common for DT, while the most common feature type for the other dependency representations is representing the root label. In Section 2.3, we discussed the head status and root choice, which we here see in plain numbers. With 6697 feature tags of this type, this is also much more common than the most common feature type for this feature for SB and CD.

Further, the most common dominating expression type is dse, but the variation is large, ranging from 1337 to 3068, as we see in table 5.6. With SB this feature

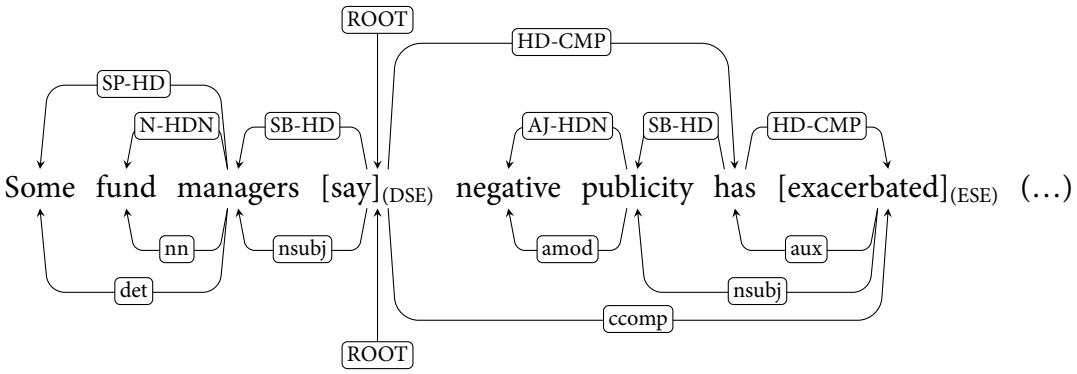


Figure 5.5: Dominating expression type. Expressions have square brackets. From 'exacerbated', with DT (top), there is no direct edge, while with SB, there is a direct edge.

type is found in 19 % of the pairs, whereas with DT, it is found in only 8 % of the pairs. In figure 5.5, we see that SB has a direct edge from the direct expression 'say' to the expressive-subjectivity element 'exacerbated'. This will give `dse` as dominating expression type for 'exacerbated'. For DT, there is no such direct link, thus, this feature is not extracted for the ESE.

The most common expression verb voice is unsurprisingly *active*. The numbers vary, but the active voice is found in approximately 2/3 of the pairs for all dependency representations.

5.9.1 Syntactic path

The syntactic path feature illustrates well some of the differences between the dependency representations. In table 5.7, we list the 10 most common feature types for each of the representations. Again, we see that a direct syntactic edge from the holder to opinion expression is the most common for both DT, SB, and CD, respectively `SB-HD↑`, `nsubj↑`, and CD: `SBJ↑`. For all representations, this feature type is over 6 times more common than the second most common type.

In table 5.8, we see that for ESE, the syntactic paths are much more complex. The average steps in the path from holder to expression in a dependency graph is twice as large for ESE as for the other expression types. A more complex path, will be challenging for the classifier, mainly because it means that the data will be more sparse. We can also see this phenomenon in table 5.9, 5.10, and table 5.11, where the most common feature types for ESE are complex compared to DSE and OSE. In addition, we see from the counts that the data material for ESE is more sparse.

Further, these tables show some differences between SB on one side, and DT and CD on the other. We see that the average path length for SB is distinctly shorter than that of DT and CD. We also see that the most common paths for SB are more frequent than the most common types for the other two representations.

	Parser	P	R	F
With semantic features	LTH SRL	0.558	0.396	0.463
W/o semantic features	LTH SRL	0.555	0.391	0.459
W/o semantic features	Bohnet & Nivre (CD)	0.565	0.389	0.461

[a] Run with the setting (1) described in Section 5.10.2

Table 5.12: The effect of the semantic feature for system expression with holder candidate selection by restricting candidates overlapping with expressions of the same expression type. The third row compares the results to Bohnet & Nivre, run with the same dependency representation as LTH SRL.

A more widespread distribution of the feature types can have two opposite effects on the performance of the classifier. It can lead to a more fine-grained feature that models the relations better, but it can also lead to more sparsity.

5.10 DEVELOPMENT RESULTS

In this section, we will present results from the development test, and do some error analysis based on these results. We will then gather our findings in a comparison between the dependency representations.

5.10.1 Comparison of LTH SRL and Bohnet & Nivre

In most of our experiments, we will not make use of semantic role labelling. As our goal is to compare different dependency representations, semantic features may skew the comparison. We would still like to see the effect the feature described in Section 5.6.8 has on the performance.

In table 5.12, we present the intersection-based precision, recall and F-score for our system with semantic and syntactic information from LTH SRL, with and without the semantic feature. In addition, we present the results with the Bohnet & Nivre parser, using CD dependency representation, the same as with LTH SRL.

We see that we get a marginally better precision and recall with the shallow semantic relation feature. We also see that the Bohnet & Nivre parser is performing slightly better than LTH SRL, but the results are comparable.

5.10.2 Holder candidates

In Section 5.5, we discussed the heuristics for holder candidate selection. A starting point for this selection is selecting the head of all noun phrases. We will then restrict this list of holder candidates for an expression when the candidate is (1) part of any expression, (2) part of an expression of the same type or (3) overlapping with the expression. Due to the difference in number of holders that are not

	System expressions				Gold expressions				
	n/c[a]	P	R	F	n/c[a]	P	R	F	p[b]
DT	9	0.556	0.397	0.464	120	0.631	0.681	0.655	0.07
SB	16	0.573	0.392	**0.465**	175	0.642	0.679	0.660	0.24
CD	10	0.565	0.389	0.461	151	0.646	0.684	**0.665**	

[a] Holder is not a candidate.
[b] Pairwise test between CD and respectively DT and SB. See Section 5.8.2.

Table 5.13: Holder candidate selection, variant (1). Candidates are not part of any expressions. For the system run with gold expressions, this variant gives lower precision and recall than the other two, and has a much larger number of holders that are not candidates. When running the system with predicted expressions, this variant is almost equal to (2), performing better than (3).

covered by a holder candidate, we wanted to investigate this choice further.

We ran the system on both system and gold expressions with these three setups. In tables 5.13, 5.14, and 5.15 we look into the performances with these setups, run on the development set with syntactic information from the Bohnet & Nivre parser.

For the system expressions, we see that the restriction in variant (2) gives a better F-score than the least restricted variant for both DT, SB, and CD. We also see that the number of missing holders candidates varies less than when we run the system on gold expressions. With the restriction in variant (2), we have the same number of holders that are not candidates and in variant (3). Unsurprisingly, variant (2) delivers better results than variant (3), as the restriction only helps narrow down the problem, without any loss. However, there is a difference between variant (1) and (2) when it comes to the holder candidates coverage. In this case, the potential gain from narrowing down the problem is countered by this loss. While CD gives a higher F-score in (1) than in (2), DT and SB give marginally better F-scores in variant (2). If we recall the number of correctly detected system pairs, 1357, the difference in F-score will rely on differences in span coverage of a handful pairs. We should not generalise, as these are quite low numbers we speak of, but for the purpose of comparing the system with that of Johansson and Moschitti (2013), we will use variant (1), where we restrict the selection of the noun phrases used as holder candidates to noun phrases where the head does not overlap with expressions of the same type.

For the system run with gold expressions, we see that the two best performing variants are (2) and (3). We also see that SB gives somewhat better performance in (2), with an F-score of 0.669, than in (3), where the F-score is 0.667. Using the metrics described in Section 5.8.2, we got a p-value of 0.190, which does not indicate significance. DT and CD, with F-scores of respectively 0.665 and 0.674 in

	System expressions				Gold expressions				
	n/c[a]	P	R	F	n/c[a]	P	R	F	p[b]
DT	7	0.560	0.398	0.465	66	0.640	0.692	0.665	0.08
SB	5	0.573	0.393	**0.466**	95	0.648	0.691	0.669	0.24
CD	6	0.564	0.386	0.458	89	0.654	0.696	**0.674**	

[a] Holder is not a candidate.
[b] Pairwise test between CD and respectively DT and SB. See Section 5.8.2.

Table 5.14: Holder candidate selection, variant (2). Candidates are not part of same type of expression. For system expression, this variant performs almost equally to (1), while for gold expressions, this variant performs almost equally to (3). The number of gold expressions where the holder does not have a candidate varies from 66 to 95, between 2 and 4 % of the possible pairs.

	System expressions				Gold expressions				
	n/c[a]	P	R	F	n/c[a]	P	R	F	p[b]
DT	7	0.551	0.391	0.458	14	0.641	0.695	0.667	0.04
SB	5	0.565	0.388	**0.460**	11	0.647	0.689	0.667	0.06
CD	6	0.560	0.381	0.454	12	0.658	0.700	**0.678**	

[a] Holder is not a candidate
[b] Pairwise test between CD and respectively DT and SB. See Section 5.8.2.

Table 5.15: Holder candidate selection, variant (3). Candidates are not part of the expression for which they are potential holders. For system expressions, this gives lower precision and recall than the other variants. For the gold expressions, this gives almost the same results as variant (2).

(2) performs better in (3) with F-scores of 0.667 and 0.678. For DT, this difference is not significant (p-value of 0.205). For CD, however, we got a p-value of 0.046, indicating significance with 5 % significance level. Once again, we will stress that this change in fact represents very few pairs, and we should not make a generalised conclusion out of this.

A more distinct change in the system run on gold expressions, is the large difference in number of holders that our system has the potential to guess correctly. Even though we lose betweeen 56 and 88 holders by not selecting them as holder candidates, our system performs almost equally well. Out of 2516 pairs, 88 is around 3.5 %, not a negligible fraction of the pairs. This indicates that by narrowing down our classification problem, our system will perform better for the pairs that it is classifying.

5.10.3 *Predicted versus gold expressions*

If we now turn to the difference between running the system with gold and with system expressions, we can compare these setups in table 5.14.

As we showed in table 4.4 in Section 4, the intersection-based recall for this was 44.1%. Immediately, it thus appears that the opinion holder classification does a much better job with the expressions detected for the system, than with gold expressions. For the question of opinion holder classification, however, we will treat predicted expressions with any length in overlap to a gold expression equally. Out of 2516 expressions, our opinion expression detection found 1357 expressions. If any overlapping expression was counted as a true positive, these numbers would have given a recall of 53.9%. (This is a simplification that does not consider two predicted expressions that overlap with one gold expression, which should lower the true recall.)

So, simplified, since we depend on these results for the opinion holder classification with predicted expressions, this should give half as good results as running the classification with gold expressions. If we study the results in table 4.4 and compare the recalls for system expressions with the recalls for gold expressions, the system recall is about 55% of the gold recall, which fits well with the difference from opinion expression detection.

For the precision, a main factor for the difference is the different system and gold expression spans for many expressions. Especially if the expression head word is different, the features will be quite different. Both precision and recall can be effected by these cases.

For a thorough analysis of the dependency representations, we will therefore base our investigation on the system run with gold expressions.

5.10.4 *Different challenges for different expression types*

We have also broken down the results in expression types. In table 5.16, we see that there is a large variety between expression types. The direct-subjective expressions (DSE) and objective speech-events (OSE) give much better results than expressive-subjectivity elements (ESE) for all dependency representations. According to Johansson and Moschitti (2013), extracting the opinion holder for ESEs is more complex because the expression and the holder usually are not directly connected syntactically or semantically. From the feature statistics in Section 5.9, especially table 5.8, we can confirm that the syntactic path between expression and holder is more complex with ESEs than with the other dependency representations. We will look further into this in the error analysis.

Table 5.16 also shows that for OSE, there is almost no variation in performance between the dependency representations. F-scores for all three dependency representations were between 0.626. For both DSE and OSE, however, CD gives the best performance, with F-scores of respectively 0.743 and 0.727.

These findings indicate that there are different challenges for different expres-

	Len.	P	R	F
DSE	952	0.691	0.767	0.727
ESE	1107	0.579	0.601	0.590
OSE	457	0.687	0.770	0.726
DSE	952	0.692	0.762	0.725
ESE	1107	0.586	0.600	0.593
OSE	457	0.701	0.755	0.727
DSE	952	0.714	0.776	**0.743**
ESE	1107	0.596	0.607	0.601
OSE	457	0.690	0.767	0.727

Table 5.16: Development results broken down on expression types. The system is run with gold expressions with candidate selection variant (3). The ESE gives considerately worse performance than DSE and OSE for all dependency representations. CD provides the best results for ESE, while SB gives the best results for DSE and OSE.

sion types. The choice of the best dependency representation may thus rely on which type of expression-holder pairs we want to classify.

5.11 ERROR ANALYSIS

To get a better understanding of how each experiment with different dependency representations guesses incorrectly, we will look more closely into the errors in the results for opinion holder classification. There are several possible ways to define an error in this context.

A very strict criterion is to categorise all pairs where the gold and system holder are not identical (in terms of their exact token spans) as errors. We could also select a certain threshold in span coverage, and treat the cases where the span coverage (for either $c(h, \hat{h})$, $c(\hat{h}, h)$, or both) is lower than the threshold as errors. A third variant is to only treat pairs with no overlap between the holder and candidate as errors. In addition, an error analysis for holder candidates can be separated into an error analysis on the pairs with internal holders (expressions with explicit holders in the same sentence) and external (writer and implicit), because they have separate sets of features.

	Criteria for error	
	Not identical	Not overlapping
Not found with any experiment	818	542
Found only with DT	92	101
Found only with SB	99	99
Found only with CD	70	72
Found only with DT and SB	52	59
Found only with DT and CD	86	111
Found only with SB and CD	150	92
Found only with CD, DT, and SB	1149	1440
Total number of pairs	2516	2516

Table 5.17: Overview of errors. We compare two variants of error definitions, and how they impact the performance of the different dependency representations. The second column shows the number of found expression-holder pairs, if a gold holder $\mathfrak{h}$ is identical to the proposed holder $\hat{\mathfrak{h}}$. The last column shows the number of overlaps where there is any overlap at all between the gold and proposed holder

In table 5.17, we give some statistics on the occurrences of errors, and compare the first and third definition of an error, described above. Each row shows how many pairs that are found only with the dependency representations mentioned. The fifth row, for example, gives the counts for pairs found with DT and SB, but not CD. As expected, there are several more cases of errors for the first than for the third definition. An interesting difference is that while there are 150 holders that are found with SB and CD, but not with DT, if we only count the cases where the proposed and gold holder are identical, there are only 92 cases of this when any overlap is counted as correctly guessed. This indicates that DT more often gives a partial overlap. We will look more into this.

More generally, the numbers indicate that there are advantages and disadvantages to every dependency representation. A problem with trying to draw general conclusions out of this, is that we do not know if the problem is the dependency representation itself or the performance of the parser with this dependency representation (for this data set). Because the data set is not based on a treebank that has been manually annotated for each of these dependency representations, we can not be certain of this.

In Section 2.3, we presented some of the findings from Ivanova (2015). In the context of error analysis, it is also of interest to look at the error analysis from that work. There are some findings we can follow up in our further error analysis. First, Ivanova (2015) states that for DT, the coordinating conjunction is especially

Proposed holder to exp.		Gold holder to exp.	
Syntactic path	Count	Syntactic path	Count
DT			
n/a (implicit)	212	n/a (w)	173
n/a (w)	94	n/a (implicit)	57
SB-HD↑	63	AJ-HDN↑	13
SB-HD↑HD-CMP↓	14	SB-HD↑	9
HD-CMP↑HDN-AJ↑	7	SP-HD↑	8
HDN-AJ↓	5	NP-HDN↑	7
AJ-HDN↓	4	HDN-AJ↓	7
HD-CMP↑HD-CMP↓	4	SB-HD↑CL-CL↑MRK-NH↓	6
SB			
n/a (implicit)	207	n/a (w)	173
n/a (w)	98	n/a (implicit)	57
nsubj↑	66	poss↑	15
pobj↑prep↑	13	amod↑	11
nsubj↑prep↓pobj↓	7	nsubj↑conj↓	8
nsubj↑dobj↓	6	nn↑	8
nsubj↑ccomp↓	5	nsubj↑	8
nsubjpass↑	5	nsubj↑ccomp↓	7
CD			
n/a (implicit)	214	n/a (w)	173
n/a (w)	92	n/a (implicit)	57
SBJ↑	60	NMOD↑	40
SBJ↑VC↓	12	SBJ↑	11
PMOD↑NMOD↑	9	SBJ↑OBJ↑	7
NMOD↓	9	SBJ↑COORD↓CONJ↓	7
SBJ↑VC↓OBJ↓	5	NMOD↓	7
SBJ↑OBJ↓	3	SBJ↑OBJ↓	4

Table 5.18: The most common syntactic path from gold and proposed holder to expression, for pairs where all dependency representations have errors. In this table, we only count pairs where there is no overlap between the gold and proposed holder as errors. This will give a total number of 542 pairs with errors.

hard to parse. That DT uses the conjunction as head of the coordinated phrase, has been shown to make it harder to parse with statistical parsers, compared to dependency representations that use the first conjunct as head (Schwartz, Abend, and Rappoport 2012).

We will now go into some more detail. First, we will briefly discuss cases where every dependency representation gave an error. Then we will look into some cases where CD gave the highest performance, and finally we will investigate cases where DT resulted in a partial overlap.

5.11.1 When all dependency representations guess incorrectly

As we see in table 5.17, a majority of the errors are common for all dependency representations. We will briefly discuss some of these. In table 5.18, we show the most frequent syntactic paths from respectively the gold and system holder, and their expression, in pairs where the system guessed incorrectly. We recall from table 5.17 that there are 542 cases of errors for all pairs for the error definition that a pair shall be counted as an error if there is no overlap between gold and proposed holder.

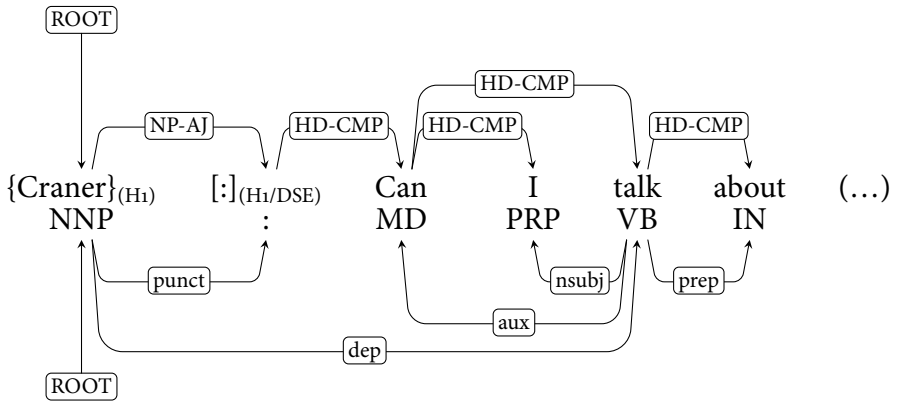
A common error is that the system often guesses incorrectly on the external holders, writer and implicit. table 5.17 shows that there are over 300 occurrences for each of the dependency representations where the proposed holder is wrongly given as either writer or implicit. This suggests that we probably could improve the performance by tuning the system for merging the three classifiers. This is outside the scope of this master's thesis, but nevertheless an interesting question for further research.

We also see that the system in too many cases guesses the subject, if it is syntactically directly connected to the expression, as the opinion holder. Since about 1/3 of the holders are subjects that are directly linked from the expression, this is not surprising.

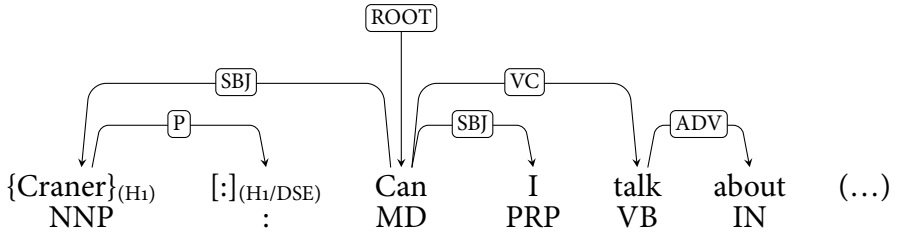
5.11.2 When only CD classifies correctly

In Section 5.10, we saw that CD gave the best performance for the system run with gold expressions. Although the differences are limited. we will investigate some of the cases were DT classifies correctly, and SB and DT wrongly. First, we will discuss the overview in table 5.20. These lists contains the most frequent syntactic paths from the predicted holder and the gold holder to the holder opinion expression for each dependency representation.

The difference in classification of implicit and writer is again interesting, and from the background data, we also see that for all 22 gold holders that are writer in this list, both DT and SB classify them as implicit. The exact same thing is also the situation for implicit holder in the gold data. The differences between the dependency representations when it comes to features for the classifiers for implicit and writer, are small.



(a) Dependency representation for the sentence in example (8) in the DT (top) and SB (bottom) formats.



(b) Dependency representation for the sentence in example (8) in the CD format.

Figure 5.6: Reported speech constructions: DT and SB choose the holder as root of the sentence and CD chooses the auxiliary verb.

Further, as we see the third line for CD in table 5.20, there are two cases where the syntactic path from opinion holder to opinion expression is P↓. One of these cases is the sentence in example (8).

(8) {Craner}$_{(H1)}$[:]$_{(H1/DSE)}$ Can I talk about tangible advances in particular countries?

As described in Section 3.1, some of the data in MPQA is question-answering. The sentence in example (8) is retrieved from that part of the corpus, and consists of a reported speech construction. We see that the colon is the opinion expression. This is somewhat unusual, but not unique in MPQA. Out of 197 colons in the development set, 31 are annotated as an opinion expression. In table 5.19, we see that performance levels for these occurrences are quite different for the three dependency representations, CD finds the most, 27 of the 31, while DT only finds 19.

In figure 5.6, we see dependency graphs for the sentence in example (8) in the three representations. We see that this sentence is difficult to parse. CD chooses 'Can' as root, while both DT and SB choose 'Craner' as root. The best choice of

	DT	SB	CD
Holder found	19	25	27
Holder not found	12	6	4

Table 5.19: Holders found for opinion expressions that consist of a single colon.

root, at least from our perspective, would be the colon. The choices of root probably lead to somewhat strange constructions. With CD, 'Can' has two subjects. For DT, there is very little information in the dependency graph, most of the edges are labelled `HD-CMP` (i.e. as complements, which is the DT analysis of inverted subjects), and no edge is labelled as subject. For DT and SB, that the root is the opinion holder, is probably having an impact on the classification task.

As we see in table 5.17, there are 72 occurrences where DT is the only dependency representation that classifies the holder correctly. Out of these pairs, 30 have external holder and therefore no syntactic path between the holder and the opinion expression. For 28 pairs, there is only one occurrence of the syntactic path in the development set. That means, for most of the errors in the cases with internal errors, the syntactic paths between the gold holder and opinion expression are uncommon. In example (9), we see an example of this. (The parenthesis denotes for which dependency representation the phrases enclosed in curly brackets is proposed.)

(9) {The report on India}$_{\text{(Gold/CD)}}$, for instance, is more than 100 pages long and [painstakingly lists]$_{\text{(DSE)}}$ incidents of rights violation across the country both by {{the government}$_{\text{(DT)}}$ and terrorist, militant, and subversive groups}$_{\text{(SB)}}$.

In figure 5.7a, we can see an excerpt of the grammatical analysis with each of the three dependency representations. This will give the following syntactic paths:

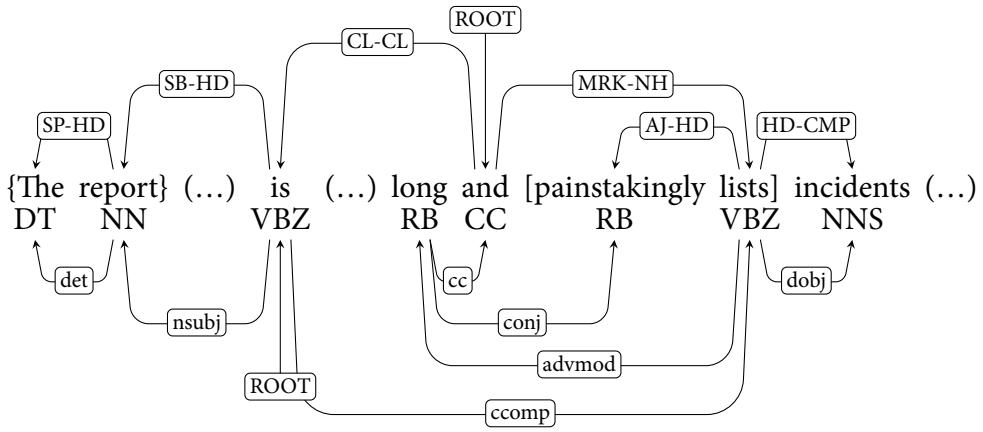
- `SB-HD↑CL-CL↑MRK-NH↓` (DT)
- `nsubj↑ccomp↓` (SB)
- `SBJ↑COORD↓CONJ↓` (CD)

We see that even though 'The report on India' is recognised as a subject in all dependency representations, only CD predicts this as the opinion holder.

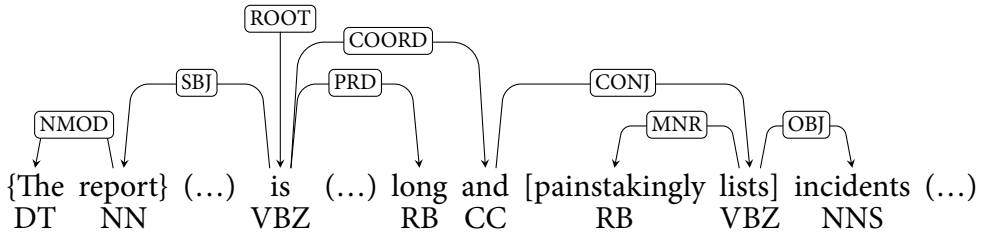
What makes this an interesting case, is the difference in the treatment of the conjunction, which we discussed in Section 2.3.2. We see that there are three different strategies here: DT chooses the conjunction as root, SB lets the coordinating conjunction be a leaf node, and CD has the first conjunct as head and second conjunct as daughter.

Proposed holder to exp.		Gold holder to exp.	
Syntactic path	Count	Syntactic path	Count
DT			
n/a (implicit)	25	n/a (w)	22
n/a (w)	13	n/a (implicit)	8
SB-HD↑HD-CMP↓	4	HD-CMP↑HDN-AJ↑	2
SB-HD↑	2	SB-HD↑HD-CMP↓	2
HD-CMP↑HD-CMP↓HD-CMP↓	1	HD-CMP↑HDN-AJ↑N-HDN↓	1
HD-CMP↑HD-CMP↑HD-CMP↑HD-CMP↑	1	HDN-AJ↓	1
HD-CMP↑HDN-AJ↑(…)HD-CMP↑HD-CMP↑	1	HD-CMP↑HD-AJ↑(…)HD-CMP↑HD-CMP↑	1
SB			
n/a (implicit)	23	n/a (w)	22
n/a (w)	12	n/a (implicit)	8
nsubj↑	4	pobj↑prep↑	7
dobj↑	2	punct↑punct↓	2
nsubj↑punct↓	2	nsubj↑xcomp↓	2
nsubj↑xcomp↓	2	punct↓	2
pobj↑prep↑dobj↑	1	pobj↑prep↑amod↓	1
CD			
n/a (w)	22	n/a (w)	22
n/a (implicit)	8	n/a (implicit)	8
P↓	4	P↓	4
PMOD↑LGS↑	2	PMOD↑LGS↑	2
SBJ↑	2	SBJ↑OBJ↓	2
PMOD↑NMOD↑	2	SBJ↑	2
SBJ↑OBJ↓	2	PMOD↑NMOD↑	2

Table 5.20: The most common syntactic paths when the holder is only found by CD. There are a total number of



(a) Dependency representation for the sentence in example (9) in the DT (top) and SB (bottom) formats.



(b) Dependency representation for the sentence in example (9) in the CD format.

Figure 5.7: Conjunctions: Three different strategies. In DT, the conjunction is the root of the sentence, and the first and second conjunct daughters. With SB, both the conjunction and the second conjunct are daughters of the first conjunct. CD has a third strategy: the conjunction is the daughter of the first conjunct and head of the second.

5.11.3 Partial overlap with DT

Last in this error analysis, we will show an example of partial overlap. In the development set, there are 18 occurrences where DT only finds a part of the gold holder because the coordinating conjunction is head of the phrase. In figure 5.8 we see one example of this. This example also shows another problem occurring with DT. If the period mark at the end of the sentence is a part of the subtree of a holder candidate, this punctuation mark will (with our chosen holder candidate heuristic) become a part of the proposed holder. This shows that the holder candidate selection is not suitable sensitive to the specific analysis of punctuation in DT.

5.11.4 Summarised

During the error analysis, we have seen that a large number of errors occur independent of dependency representation. This holds in particular for the exter-

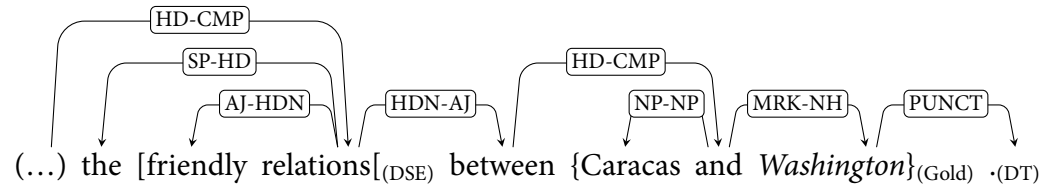


Figure 5.8: Partial overlap with DT. The gold holder, 'Caracas and Washington'is in curly brackets and the proposed holder, 'Washington .' in italic.

nal holders (implicit/writer). For the system in general, there are several possible ways to increase the performance. First, an improvement (tuning) of the algorithm for selection of holders from the classifiers for implicit, writer and internal holder, is something that could have large impact on the performance.

Second, we have seen some examples of disadvantages. Some of these could probably be avoided with improved heuristics for selection of holder candidates, and expansion of holders from a holder candidate. An example of a rule for the expansion, is to omit the period mark at the end of a sentence when expanding a holder candidate.

We have also seen an example that constitutes a more difficult problem to overcome. CD seems to have an advantage when it comes to treatment of colon in reported speech constructions and coordination structures.

Nevertheless, we can not from our experiments definitely recommend a single dependency representation. As we clearly have seen in table 5.17, there are obviously advantages and disadvantages of every dependency representation when it comes to opinion holder classification.

5.12 HELD-OUT RESULTS

Finally, we run the system with the held-out test set. First, we will compare our results with those of Johansson and Moschitti (2013), before we review the performance of our system run with gold expressions.

5.12.1 Comparison to Johansson & Moscitti

In table 5.21, we see that both precision and recall are distinctly higher in Johansson and Moschitti (2013) than in our system. This is somewhat surprising, because the systems are very similar. The dependency parser, the feature set and the classifier are equal. There are some possible differences. First, the difference between the systems in opinion expression detection is going to have an impact also on the performance for opinion holder classification. Second, in our system for evaluation, we have made several choices, especially in terms of counting the number of system/gold pairs which may be different. Third, the heuristics for selecting

	Parser	P	R	F
Johansson and Moscitti	LTH SRL	0.577	0.453	0.508
With semantic features	LTH SRL	0.551	0.405	0.467
W/o semantic features	LTH SRL	0.550	0.405	0.467
W/o semantic features	Bohnet & Nivre (CD)	0.550	0.405	0.467

[a] Run with the setting (2) described in Section 5.10.2

Table 5.21: Heldout: Comparison to Johansson & Moscitti

	Held-out set				Development set		
	P	R	F	p[a]	P	R	F
DT	0.647	0.682	0.664	< 0.01	0.641	0.695	0.667
SB	0.668	0.704	**0.686**		0.647	0.689	0.667
CD	0.657	0.687	0.672	< 0.01	0.658	0.700	**0.678**

[a] Pairwise test between SB and respectively DT and CD. See Section 5.8.2.

Table 5.22: Held-out test. Holder candidate selection, variant (3).

between the three classifiers (internal, 'w' and 'implicit') may also give different results. We also see that the performance with Bohnet & Nivre is equal to the performance of LTH SRL.

5.12.2 *Heldout run with gold expressions.*

Running the system on the held-out test set, gives some different results. First, we see that the SB format now gives the best performance. In contrast to earlier significance tests, the difference in results with the held-out test set is also statistically significant, with a p-value below 0.01.

This supports our earlier findings. Because there are different challenges for different dependency representations, the performance of each dependency representation is sensitive to the choice of data set.

Even though the difference is significant, the difference is not very large. A change in system, for example a slightly different method for holder candidate selection, could affect the performance differently.

With these final results, we will in the next chapter draw the general conclusion for our work. We will summarise each part of our thesis and outline the most important insights we have made, before we points out directions for future work for fine-grained opinion analysis.

Chapter 6

Conclusion

In this work, we have investigated the effect of different syntactic dependency representations through experimental evaluation in the task of fine-grained opinion analysis. In order to do so we have implemented a system for opinion analysis which performs the sub-tasks of opinion expression detection and opinion holder classification.

We wanted to create a flexible system, that lets us experiment with the MPQA dataset and a variety of linguistic analysis tools and target representations. Further, we wanted to see to what degree the empirical results of Johansson and Moschitti (2013) could be replicated with an abstractly equivalent, but technically slightly different ensemble of syntactic-semantic pre-processors and machine learning techniques. Finally, we wanted to use our system to investigate the effects, if any, that can be observed on the sub-task of opinion holder classification when using different types of representations for syntactic analysis.

We gave an overview of the topic of opinion analysis in Chapter 2, and discussed the previous works on fine-grained opinion analysis from Choi, Breck, and Cardie (2006) and Johansson and Moschitti (2013). Further, we provided the necessary background for grammatical analysis. In Chapter 3, we described a number of third party tools and data sources needed in order to build the system, in addition to the evaluation metrics we used in the system. In Chapter 4, we detailed the first sub-task in our system: opinion expression detection. We chose to solve this as a sequence labelling task. Finally, in Chapter 5, we described the main sub-task, where we were going to perform the comparison between the dependency representations.

During our work, we have gathered several insights, which we in the following will present.

BUILDING A FLEXIBLE ENVIRONMENT FOR EXPERIMENTATION

During the implementation of our system, we have come across several issues which were not clearly described in previous work and where we had to make non-trivial choices and subsequently implement these. On this regard, our project may serve as an in-depth clarification and validation of the overall design choices made by Johansson and Moschitti (2013).

How to treat overlapping expressions with intersection-based precision and recall
Because the system for opinion expression detection was solved as a sequence labelling task, it was not possible to handle overlapping expressions. That meant, we needed to omit overlapping expressions from the training and testing files. In order for our system to include these ignored expressions, we presented in Section 3.6 a method for including expressions in the evaluation.

Heuristics for holder candidate selection
In Section 5.5, we discussed the importance of holder candidate selection. We saw that the heuristics we chose in holder candidate selection had a large impact on the number of opinion holders we could possibly classify. We therefore investigated this topic further, by performing a set of experiments using three slightly different heuristics. We learned from these experiments that the best choice of heuristics was dependent on the syntactic representations.

Significance testing for intersection-based precision and recall
For hypothesis testing of the intersection-based precision and recall, we needed to select a measure that does not assume normal distribution. We adapted a bootstrapping test, which is a group of flexible significance tests, that estimates the normal distribution by the means of resampling.

System for error analysis
In order to create an overview over errors, retrieve examples that could be of interest, and compare the system run with different pre-processing tools and other settings, we created a system for error analysis. This system compares each expression-holder pair across different system outputs, and lets us interactively compare different features, retrieve lists of common errors, and print out lists of interesting sentences.

COMPARISON OF DEPENDENCY REPRESENTATIONS

The main part of our project was aimed at comparison of different dependency representations in the task of opinion holder classification.

There are advantages and disadvantages for every dependency representations
The difference in performance between the dependency representations, when running the system on the development set, was not significant. In the error analysis overview in Section 5.10.4, we saw that in most cases when a pair was not found, that this was equal for both DT, SB or CD. There were also, for all dependency representations, a notable number of pairs where each dependency representation performed better than one or two of the other dependency representations.

The heuristics for selection of holder candidates affects dependency representations differently
In Section 5.10.2, we saw that the choice of heuristics for holder candidate selection affected the performance differently, dependending on the choice of dependency representation. This indicates that the heuristics for candidate selection can be tuned to the chosen dependency representation.

There are specific problems for DT and SB when it comes to the treatment of reported speech constructions
A grammatical structure, where CD outperformed the other dependency representations was presented in Section 5.11.2. We saw that the reported speech constructions with a colon was treated differently across the dependency representations, and in the context of opinion holder classification, the analysis made with CD gave better results.

The usage of conjunction as head for a coordination, may be a disadvantage for DT
In Section 5.11.3, we gave an example of how the choice of the conjunction as head of a coordination in DT leads to a choice of holder candidate that made it impossible to cover the whole opinion holder. This, again, suggests that the heuristics for holder candidate selection should be adjusted, when representing the syntactic structure with DT.

6.1 IMPROVING THE SYSTEM

For opinion expression detection, we did not aim to improve the state of the art. Our system here, was mainly constructed to have a complete pipeline for opinion holder classification. We chose to build a system for opinion expression detection similar to the baseline in the work of Johansson and Moschitti (2013). Their proposed system, using reranking and syntactic and semantic information, which we reviewed in Section 2.2.2, we will still consider the state of the art. This work showed that the use of syntactic and semantic information also improved this sub-task. We believe that an attempt to improve opinion expression detection,

should build further upon this.

For opinion holder classification, we summarised some possible improvements at the end of Chapter 5. One challenge in our system was how to merge the output from three different classifiers. Because we had separate classifiers for the external holders (*writer* and *implicit*) in addition to the classifier for internal holders, we needed to use a heuristic in order to select the output from each of these three classifiers. We saw in the error analysis in Section 5.11, that an improvement here could have a large positive impact on the performance.

As discussed earlier, we have also seen that there is a possible gain from adapting the holder candidate selection heuristics to the dependency representation. For DT, two examples are to include coordinating conjunctions as holder candidate, and to omit the period mark at the end of a sentence when expanding the holder candidate.

We have also seen indications in the error statistics in Chapter 5 that there is some degree of complementarity between the dependency representations. There may be a possible gain to combine the information from multiple representations. For example by using features from two or three representations, one could hope to combine the strong points of multiple representations, i.e. via increased recall.

6.2 FUTURE WORK

The latest version of MPQA, was released in December 2015 (Deng and Wiebe 2015). This version of the corpus is expanded with information about the target of opinion. This makes it possible to expand a system for fine-grained opinion analysis to also include target of opinion. As discussed in Section 2.1.1, the methods of classifying targets are similar to the methods we have used for opinion holder classification. It will be an interesting task to extend our system and perform a similar comparison for target classification.

In future comparisons, Universal Dependencies (UD) (De Marneffe et al. 2014; Nivre 2015), the new emerging standard for dependency representations, should be included.

In addition, investigating how to create more refined heuristics for candidate selection and for merging of the output from the different classifiers are tasks that could improve the performance of both opinion holder and opinion target classification. Another issue to investigate further the advantages of each dependency representation. The held-out results shows that we should consider cases where SB performs better than the other.

Finally, we could decide on a dependency representation, and tune both holder candidate selection and heuristics for the merging of external/internal holders, in order to try to improve the state of the art in this field.

Bibliography

Berg-Kirkpatrick, Taylor, David Burkett, and Dan Klein. 2012. "An empirical investigation of statistical significance in nlp." In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning,* 995–1005. Association for Computational Linguistics.

Bohnet, Bernd, and Joakim Nivre. 2012. "A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing." In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning,* 1455 – 1465. Jeju Island, Korea.

Choi, Yejin, Eric Breck, and Claire Cardie. 2006. "Joint extraction of entities and relations for opinion recognition." In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing,* 431–439. Sydney, Australia: Association for Computational Linguistics.

Collins, Michael. 1999. "Head-driven statistical models for natural language parsing." PHD thesis, University of Pennsylvania, Philadelphia.

Cortes, Corinna, and Vladimir Vapnik. 1995. "Support-vector networks." *Machine learning* 20 (3): 273–297.

De Marneffe, Marie-Catherine, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. "Universal Stanford dependencies: A cross-linguistic typology." In *LREC,* 14:4585–92.

de Marneffe, Marie-Catherine, and Christopher D. Manning. 2008. "The Stanford Typed Dependencies Representation." In *Proceedings of the COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation,* 1 – 8. Manchester, UK.

Deng, Lingjia, and Janyce Wiebe. 2015. "Mpqa 3.0: An entity/event-level sentiment corpus." In *Conference of the North American Chapter of the Association of Computational Linguistics: Human Language Technologies.*

Ding, Xiaowen, and Bing Liu. 2010. "Resolving Object and Attribute Coreference in Opinion Mining." In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010),* 268–276. August. Beijing, China: Coling 2010 Organizing Committee.

Ding, Xiaowen, Bing Liu, and Philip S. Yu. 2008. "A holistic lexicon-based approach to opinion mining." *Proceedings of the international conference on Web search and web data mining - WSDM '08* (New York, New York, USA): 231.

Dridan, Rebecca, and Stephan Oepen. 2013. "Document Parsing. Towards Realistic Syntactic Analysis." In *Proceedings of the 13th International Conference on Parsing Technologies.* Nara, Japan, November.

Efron, Bradley, and Robert J Tibshirani. 1994. *An introduction to the bootstrap.* CRC press.

Fan, Re, Kw Chang, and Cj Hsieh. 2008. "LIBLINEAR: A library for large linear classification." *The Journal of Machine Learning* 9:1871–1874.

Hu, Minqing, and Bing Liu. 2004. "Mining and summarizing customer reviews." In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04,* 168–177. Seattle, WA, USA: ACM.

Ivanova, Angelina. 2015. "Bilexical Dependencies as an Intermedium for Data-Driven and HPSG-Based Parsing." PhD diss., University of Oslo.

Ivanova, Angelina, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. "Who Did What to Whom? A Contrastive Study of Syntacto-Semantic Dependencies." In *Proceedings of the Sixth Linguistic Annotation Workshop,* 2 – 11. Jeju, Republic of Korea.

Johansson, Richard, and Alessandro Moschitti. 2013. "Relational Features in Fine-Grained Opinion Analysis." *Computational linguistics* 39 (3): 473–509.

Johansson, Richard, and Pierre Nugues. 2007. "Extended Constituent-to-dependency Conversion for English." In *Proceedings of the 16th Nordic Conference of Computational Linguistics,* 105 – 112. Tartu, Estonia.

———. 2008. "Dependency-based syntactic-semantic analysis with PropBank and NomBank." In *Proceedings of the Twelfth Conference on Computational Natural Language Learning,* 183–187. Association for Computational Linguistics.

Kim, Soo-Min, and Eduard Hovy. 2006. "Extracting Opinions, Opinion Holders, and Topics Expressed in Online News Media Text." In *Proceedings of the Workshop on Sentiment and Subjectivity in Text - SST '06,* 1–8. July. Sydney, Australia.

Lafferty, John, Andrew McCallum, and Fernando CN Pereira. 2001. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data."

Lavergne, Thomas, Olivier Cappé, and François Yvon. 2010. "Practical Very Large Scale CRFs." In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL),* 504–513. Uppsala, Sweden: Association for Computational Linguistics, July.

Li, B, L Zhou, S Feng, and KF Wong. 2010. "A unified graph model for sentence-based opinion retrieval." In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics,* 1367–1375. Uppsala, Sweden: Association for Computational Linguistics.

Liu, Bing. 2012. "Sentiment Analysis and Opinion Mining." *Synthesis Lectures on Human Language Technologies* 5 (May): 1–167.

Marcus, Mitchell, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. "Building a Large Annotated Corpora of English. The Penn Treebank." *Computational Linguistics* 19 (2): 313 – 330.

McCarthy, Joseph F, and Wendy G Lehnert. 1995. "Using decision trees for coreference resolution." *arXiv preprint cmp-lg/9505043.*

Nivre, Joakim. 2015. "Towards a universal grammar for natural language processing." In *International Conference on Intelligent Text Processing and Computational Linguistics,* 3–16. Springer.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryığıt, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. "MaltParser: A Language-Independent System for Data-Driven Dependency Parsing." *Natural Language Engineering* 13 (2).

Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. 2002. "Thumbs up? Sentiment classification using machine learning techniques." In *Proceedings of EMNLP,* 79–86.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12:2825–2830.

Pradhan, Sameer, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. "CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes." In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL 2011).* Portland, Oregon, June.

Sang, Erik F. Tjong Kim, and Jorn Veenstra. 1999. "Representing Text Chunks." In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics,* 173–179. EACL '99. Bergen, Norway: Association for Computational Linguistics.

Schwartz, Roy, Omri Abend, and Ari Rappoport. 2012. "Learnability-Based Syntactic Annotation Design." In *COLING,* 24:2405–2422.

Søgaard, Anders, Anders Johannsen, Barbara Plank, Dirk Hovy, and Héctor Martínez Alonso. 2014. "What's in a p-value in NLP?" In *CoNLL,* 1–10. Citeseer.

Soon, Wee Meng, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. "A machine learning approach to coreference resolution of noun phrases." *Computational linguistics* 27 (4): 521–544.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. "The CoNLL 2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies." In *Proceedings of the 12th Conference on Natural Language Learning,* 159 – 177. Manchester, UK.

Sutton, Charles, and Andrew McCallum. 2010. "An introduction to conditional random fields." *arXiv preprint arXiv:1011.4088.*

Toutanova, Kristina, Dan Klein, and Christopher D Manning. 2003. "Feature-rich part-of-speech tagging with a cyclic dependency network." *In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1 (NAACL '03),* 252–259.

Wiebe, Janyce, Theresa Wilson, and Claire Cardie. 2005. "Annotating expressions of opinions and emotions in language." *Language Resources and Evaluation* 39 (2-3): 165–210.

Wilcoxon, Frank. 1945. "Individual comparisons by ranking methods." *Biometrics bulletin* 1 (6): 80–83.

Wilson, Theresa Ann. 2008. "Fine-grained Subjectivity and Sentiment Analysis: Recognizing the Intensity, Polarity, and Attitudes of Private States." PhD diss., University of Pittsburgh.

Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. 2005. "Recognizing Contextual Polarity in Phrase-level Sentiment Analysis." In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing,* 347–354. HLT '05. Vancouver, British Columbia, Canada: Association for Computational Linguistics.

Yamada, Hiroyasu, and Yuji Matsumoto. 2003. "Statistical Dependency Analysis with Support Vector Machines." In *Proceedings of the 8th International Conference on Parsing Technologies,* 195 – 206. Nancy, France.

Zhang, Yi, and Rui Wang. 2009. "Cross-Domain Dependency Parsing Using a Deep Linguistic Grammar." In *Proceedings of the 47th Meeting of the Association for Computational Linguistics,* 378 – 386. Suntec, Singapore.