Prague University of Economics and Business
Faculty of Informatics and Statistics

Aspect-based sentiment analysis of conference review forms

MASTER THESIS

Study program: Applied Informatics
Field of study: Knowledge and Web Technologies

Author: Bc. Sára Juranková
Supervisor: prof. Ing. Vojtěch Svátek, Dr.

Prague, December 2020

Acknowledgements

Poděkování.

Abstract

Abstrakt.

Keywords

sentiment analysis, coference reviews, aspect-based sentiment analysis

Abstrakt

Abstract.

Klíčová slova

analýza sentimentu, recenze konferenčních příspěvků, aspektová analýza sentimentu

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

BCC Blind Carbon Copy
CC Carbon Copy
CERT Computer Emergency Response Team
CSS Cascading Styleheets
DOI Digital Object Identifier
HTML Hypertext Markup Language
REST Representational State Transfer
SOAP Simple Object Access Protocol
URI Uniform Resource Identifier
URL Uniform Resource Locator
XML eXtended Markup Language

Introduction

The aim of this thesis is to create a system for extracting opinions and sentiment from conference paper reviews. In other words the system will allow to automatically determine the opinion of the author of the review on different aspects of the reviewed paper.

Because each submitted paper is going to be reviewed by many people, and because each conference has its own structure of the review form as well as different set of criteria it's very difficult to quickly get an idea of the quality of the submitted work. Being able to get a general idea of how good a paper is as well as quickly assessing what are the strong and weak parts of a submission is especially important for meta-reviewers during the discussion periods.

Extracting the opinions of the reviewer on a paper, mapping it onto a unified set of criteria and transforming it into a numerical value could significantly simplify the process of submission acceptance as well as provide a way to compare reviews of the same paper across different conferences and reviewers.

The system created here could also serve as a base for a larger review management system that is able to generate a visual metaphor of each review that reflects different review metrics. Visual images are faster and easier to understand than written text, therefore it should make the meta-analysis more comfortable and effective.

In order to implement such a system, it's necessary to create a set of review metrics to which the fields of different conference review forms will be mapped. Then apply a technique of information extraction to get the opinion of the reviewer on each of these metrics. In order to then get a numeric evaluation of the sentiment of the specific opinion a sentiment analysis technique has to be used.

1. Introduction to sentiment analysis and existing methods

1.1 Sentiment analysis

Sentiment analysis (also known as opinion mining) is a type of text analysis focused on detecting polarity (e.g. positive or negative opinion) within text. This chapter serves as an introduction to the field of sentiment analysis and gives an overview of existing tools and current practices.

1.1.1 Levels of sentiment analysis

Liu [1] describes 3 levels of sentiment analysis:

- **Document level** – determines the sentiment based on the entire text, which is most useful when the document expresses opinion on a single entity
- **Sentence level** – classifies each sentence as positive, negative or neutral, mostly used for subjectivity classification
- **Entity and Aspect level** – recognizes the different entities described in the text and their aspects and extracts opinions linked with these aspects

Given that the review forms express different opinions on different aspects of a paper, an opinion on a single one of these aspects can span across many sentences, sentiment analysis should therefore be done on an aspect level.

1.1.2 Definition of opinion

In order to explain the task of opinion analysis, we first need to have a definition of an opinion. This is how Liu [1] defines an opinion:

> An opinion is a quadruple,
>
> (e, a, s, h, t)
>
> where e is the opinion (or sentiment) target entity, a is the aspect of said entity, s is the sentiment about the target, h is the opinion holder and t is the time when the opinion was expressed.

For the needs of sentiment analysis in the task of extracting opinions from the final versions of paper reviews, a sufficient definition should be just a tuple (e, s). As reviews sometimes may

have more versions than the final one, and because sometimes, a review also may contain an opinion of the author of the reviewed paper (a sentence such as "Although the author believes that his idea is novel, that is not the case" expresses the opinions of two opinion holders – the author and the reviewer), in future the system may be improved to account for this and therefore use the original quadruple definition of an opinion.

1.1.3 Classification of sentiment

1.1.4 Sentiment analysis tasks

Another thing needing definition is the general way sentiment analysis is done. Liu [1] describes the task of sentiment analysis as a process of six steps, working with:

1. entity extraction and categorization
2. opinion holder extraction and categorization
3. aspect extraction and categorization
4. time extraction and standardization
5. aspect sentiment classification
6. opinion quintuple generation

Since conference paper review only describe one entity (the paper) and all opinions should belong to a single person (the reviewer), those parts of the analysis can be left out, as well as time extraction, since that information is not relevant in the generation of visual metaphors. Therefore the steps actually taken in the sentiment analysis of conference paper reviews are these:

1. **aspect extraction and categorization** – Extract aspects expressions of the defined review metrics and cluster them based on the metrics they represent
2. **aspect sentiment classification** – Determine whether an opinion on an aspect is positive, negative or neutral and assign it a numeric value describing the polarity of the opinion as well as the strength of the sentiment
3. **opinion tuple generation** – produce the tuples (aspect, sentiment) based on the previous steps

1.2 Sentiment analysis techniques

1.2.1 Machine learning

Machine learning methods, both supervised and unsupervised, can be used for sentiment analysis. [1]. For example aspect extraction can be done by Conditional Random Field (CRF), a supervised learning method commonly used in natural language processing [2]. Another technique, by Hu and Liu [3], which is further explained in section **??** uses association min-

ing to extract features. However most machine learning methods mostly focus on sentiment analysis on the document level. One such technique, which is fairly simple, but often used for simple sentiment analysis is the naïve Bayes classifier.

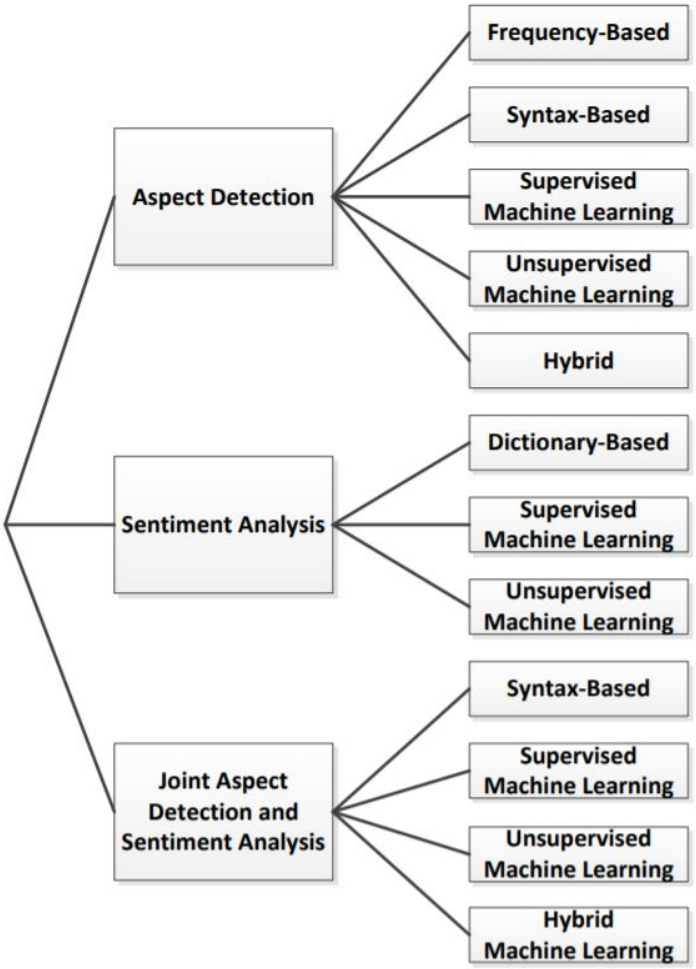

Figure 1.1: Taxonomy for aspect-level sentiment analysis [2]

The naïve Bayes classifier

Naïve Bayes is a simple machine learning algorithm that utilizes Bayes rule together with a strong (or naïve) assumption that the evidence are conditionally independent, given the hypothesis. [4]

Therefore the equation for the probability of a hypothesis H given a set of evidence $E_1, \ldots, E_K$, that the naïve Bayes classifier is based on is:

$$P\left(H|E_1,\ldots,E_K\right) = \frac{P(H)}{P(E_1,\ldots,E_K)} \times \prod_{k=1}^{K} P\left(E_k|H\right)$$

The goal of the classifier is to determine which of the possible hypothesis is the most probable given the evidence.

In the task of sentiment analysis, the different "classes" that serve as hypothesis are the different sentiment polarities we try to detect. So we can for example have three different classes – positive, negative and neutral.

As I already mentioned, when classifying using the naïve Bayes algorithm, we look for the hypothesis for which the probability given the evidence is the highest. Therefore we can simplify the equation for $P(H|E_1,\ldots,E_K)$, leaving the denominator out, because $P(E_1,\ldots,E_K$ will always stay the same for all possible classes/hypothesis:

$$P(H|E_1,\ldots,E_K) = P(H) \times \prod_{k=1}^{K} P(E_k|H)$$

The evidence are the words contained in the text we want to classify. The text is represented as a bag-of-words, meaning instead of considering the position of each word in the text, we represent it as an unordered set.

When training the naïve Bayes classifier we also consider the frequency of each word in each text. We also need a to have each text annotated with its sentiment. The idea behind the bag-of-word approach is depicted on figure 1.2 .

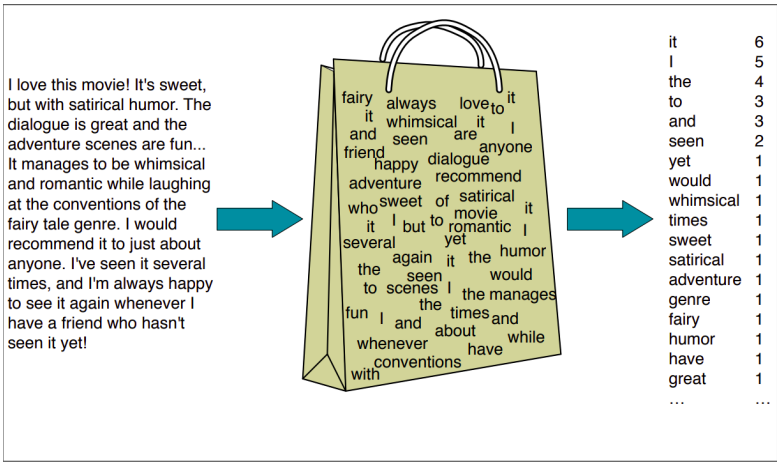

Figure 1.2: Intuition of the multinomial naive Bayes classifier applied to a movie review [5]

Then we calculate all the necessary probabilities ($P(c)$ and $P(w_i|c)$ or $P(H)$ and $P(E_k|H)$ from the original formula) we need for classification of new texts.

The formula for the prior probability of a given class c is:

$$P(c) = \frac{N_c}{N}$$

where N_c is the number of text belonging to the class c and N is the total amount of texts in the training dataset.

The probability of a word w_i occurring in a text with a given class c is:

$$P(w_i|c) = \frac{count(w_i,c)}{\sum_{w \in V} count(w,c)}$$

where V is the vocabulary, consisting of all words found across all texts, $count(w_i,c)$ is number of times the word w_i appears in documents belonging to class c and the sum of $count(w,c)$ over all words in the vocabulary calculates the sum of all words in all documents of class c.

Because the naïve Bayes classifier multiplies all evidence likelihoods together, this equation is usually adjusted to account for the fact that some words might never appear in the training set or never appear in conjuction with some class. This makes their probability given a class zero and therefore the probability of said class also zero, independently on all the other words that appeared in the text. This behavior is usually corrected by givin these words non-zero probabilities e.g. using the add-one (Laplace) smoothing[5]:

$$P(w_i|c) = \frac{count(w_i,c) + 1}{(\sum_{w \in V} count(w,c)) + |V|}$$

To give a clear example of an application of the naïve Bayes classifier on the task of sentiment analysis, here is how we would calculate the probability of a sentence “This phone is great.” being positive:

$$\begin{aligned} P(positive|\text{“}this\text{”},\text{“}phone\text{”},\text{“}is\text{”},\text{“}great\text{”}) = \\ = P(\text{“}this\text{”}|positive) \\ \times P(\text{“}phone\text{”}|positive) \\ \times P(\text{“}is\text{”}|positive) \\ \times P(\text{“}great\text{”}|positive) \\ \times P(positive) \end{aligned}$$

1.2.2 Dictionary-based Approaches

The biggest indicators of sentiment in a text are sentiment words (also called opinion words). These words, often adjectives or adverbs, help to detect the expression of sentiment as well as its polarity. For example words such as *great,amazing* or *good* indicate a positive sentiment, on the other hand words like *terrible*, *awful* or *bad* express negative feelings. [1]

In order to obtain a sentiment lexicon, Liu [1] mentions 3 main approaches:

- **manual approach** – is usually combined with automated methods because of its labor intensity
- **dictionary-based approach** – usually uses a list of a small number of sentiment words as a seed and then generates the dictionary through tools such as Princeton University [6] by integrating their synonyms or antonyms
- **corpus-based approach** – the aim is to create a sentiment lexicon for a specific domain, for example by using a seed of sentiment words and then including other words in the lexicon by searching the sentences for conjoined adjectives, where one is already known as a sentiment word

There also are already compiled dictionaries of sentiment words, such as SentiWordNet [1], which assigns to each word both a positive score and a negative score (on a scale from 0 to 1) and allows to obtain an objectivity score based on the two. SentiWords [2] or SenticNet [3], assign to each word they contain a sentiment score between -1 (extremely negative) and +1 (extremely positive).

Lexicon-based approaches of sentiment analysis, as the name suggests, utilize lexicons of sentiment words as well as other constructs like sentiment shifters, but-clauses and other words or phrases that affect sentiment.

Sentimentr

One tool that utilizes a lexicon-based method for sentiment analysis is sentimentr.

As explained in the sentimentr documentation "sentimentr attempts to take into account valence shifters (i.e., negators, amplifiers (intensifiers), de-amplifiers (downtoners), and adversative conjunctions) while maintaining speed. Simply put, sentimentr is an augmented dictionary lookup." [7].

The way sentimentr works is that instead of simply comparing the words in sentence with the sentiment lexicon and judging the sentiment of a sentence based on, say, the sum of polarities of sentiment words found within it, it also adjusts the polarity of each sentiment word based on sentiment shifters found in the proximity of that word. The influence of different valence shifters is as follows:

Negators. Negators are words such as "no, not, never". The influence of negators on the polarity of a sentiment word is simple – if the number of negators in the left and right context of a sentiment word is even the polarity stays the same, however if the number is odd, the polarity is negated (so a word with originally negative polarity becomes positive and vice versa)

[1] `https://github.com/aesuli/sentiwordnet`
[2] `https://hlt-nlp.fbk.eu/technologies/sentiwords`
[3] `https://sentic.net/`

Amplifiers. Amplifiers are words like "especially, major, significantly". As the name suggests they increase (amplify) the polarity of a sentiment word. There is however one exception to this rule – if the number of negators in the context of a sentiment words is odd, the influence of amplifiers is negated, aka they start working as the de-amplifiers described bellow.

De-amplifiers De-amplifiers, like "slightly, somewhat sort of" etc. work analogously to amplifiers, except they decrease the polarity of a sentiment word instead of increasing it.

Adversative conjuction Adversative conjuctions are perhaps the most complex of the valence shifters. These are words such as "bur, however, albeit". The relative position of an adversative conjuction to the sentiment word plays an important role when determining its influence. If the conjuction comes before the sentiment word it increases its polarity, however if it comes after it decreases it. According to the sentimentr documentation "This corresponds to the belief that an adversative conjunction makes the next clause of greater values while lowering the value placed on the prior clause.". [7]

The final sentiment score of a sentence is calculated as follows:

$$\text{sentiment}(s) = \frac{\sum_{w_i \in Pol} sentiment(w_i)}{\sqrt{|w_i|}}$$

where s is the sentence, w_i is the i-th word of the sentence, Pol is a set o polar/sentiment words in the sentence, $sentiment(w_i)$ is the calculated sentiment of w_i and $|w_i|$ is the length of the sentence.

The sentiment of a word adjusted by the valence shifters is calculated as:

A Holistic Lexicon-Based Approach

Another lexicon-based approach is one called "a holistic lexicon-based approach".[8] This one, contrary to the *sentimentr* method described above which determines the sentiment on a sentence level, focuses on aspect-based sentiment analysis.

The basic algorithm finds all words or phrases describing features in a sentence as well as opinion (or sentiment) words. Then, for each feature in the sentence, its sentiment score is calculated using the polarity of the opinion words and their distance in the sentence from the feature expression using the following function:

$$\text{score}(f) = \frac{\sum_{w_i : w_i \in s \wedge w_i \in V} w_i.SO}{dis(w_i, f)}$$

where:

- w_i is an opinion word

- V is the set of all opinion words
- s is the sentence that contains the feature f
- $dis(w_i, f)$ is the distance between feature f and opinion word w_i in the sentence s
- $wi.SO$ is the semantic orientation of the word w_i

Then "If the final score is positive, then the opinion on the feature in the sentence s is positive. If the final score is negative, then the opinion on the feature is negative. It is neutral otherwise."[8].

The algorithm is also extended to deal with negation (by negating the polarity of a sentiment word which follows after a negation word), "but" clauses (by first trying to determine the sentiment of an opinion word within the "but" clause using the basic algorithm and if the sentiment score is zero it assigns the negation of the clause before "but"). Then it has these three rules for dealing with context-dependent opinion words:

- **Intra-sentence conjunction rule** – this rule is based on the idea that "a sentence only expresses one opinion orientation unless there is a 'but' word which changes the direction."[8]. Therefore if the orientation of one opinion word depends on the context but there is another opinion word in the sentence for which the orientation is known and the clauses containing the two opinion words are connected by a conjunction such as "and", we can assign that orientation to the context-dependent orientation word as well.
- **Pseudo intra-sentence conjunction rule** – this rule applies to sentences without an explicit conjunction, but otherwise works the similarly to the previous rule
- **Inter-sentence conjunction rule** – if the opinion orientation is still undetermined after the application of previous rules, the inter-sentence conjunction rule helps assign the orientation using the context of the surrounding sentences. According to the authors "The idea is that people usually express the same opinion (positive or negative) across sentences unless there is an indication of opinion change using words such as 'but' and 'however'."[8].

1.3 Aspect extraction

1.3.1 Frequency extraction

In order to extract aspect expressions, Hu and Liu [3] propose using a Part Of Speech (POS) tagger to identify nouns and noun phrases as possible aspect expression candidates. Then they use a simplified Apriori algorithm to calculate frequency of these candidates and finally keep only the ones that are frequent enough.

Their reasoning is that the vocabulary people use when commenting on an entity converges, so the frequently co-occurring sets of terms should represent the important aspects.

1.3.2 Taxonomy based extraction

Another approach, by Carenini[9], uses user's prior knowledge of the domain to build a hierarchy of features. Their technique uses the previously mentioned unsupervised learning technique developed by Hu and Liu and adds user-defined features and similarity matching in order to eliminate redundancy and to generate a set of features in an organized way, reflecting hierarchical relationships between them. The way their method works is that first a set of "crude features" is generated by Hu and Liu's method.

Then the WordNet lexical database is used to perform similarity matching between the user-defined taxonomy of features and the crude features. Only the crude features that are similar enough to the features already included in the taxonomy are then further used.

Finally the newly discovered feature candidates which passed the similarity matching are inspected by a user and if the user considers these candidates valid feature expression, they are added to the final taxonomy.

1.3.3 Patterns for aspect extraction

A very different technique for aspect extraction is described by Asghar et al.[10]. Their opinion mining framework comprises of a set of heuristic patterns for extraction of aspects as well as sentiments or opinions.

In order to get the pairs of aspects and the related sentiments, the reviews first go through a preprocessing phase, part of which is POS tagging. After that, the POS tagged sentences are matched against a set of patterns. Each pattern is a sequence of POS tags, so in order to get a match against one of them, the sentence must contain that sequence of tags as well.

The part of the pattern which is a noun, noun phrase or a verb is then considered to be a candidate term for an aspect while mostly adjectives and adverbs represent the expression of sentiment.

2. Natural language processing

Natural language is any language thas has not been specially constructed but rather has evolved naturally through use and is "aquired by its users without special instructions as a normal part of the process of maturization and socialization".[11]. These languages such as English, Arabic, Vietnamese or Hindi differ from non-natural language, which are artificially constructed for specific uses, like for example programming languages or symbolic languages used for studying logic.

Natural language processing (NLP) is "an interdisciplinary domain which is concerned with understanding natural languages as well as using them to enable human–computer interaction".[12] The field of applications of NLP is far reaching and includes use cases such as:

- Identifying spam e-mails[13]
- Chatbots for customer support and engagement[14]
- Improvement in clinical documentation[15]

and many others. An overview of the usage of NLP can be seen on table 2.1.

Table 2.1: Breakdown of various NLP tasks performed by modern NLP software[14]

Word Tagging	**Sentence Parsing**	**Text Classification**	**Text Pair Matching**	**Text Generation**
Word segmentation	Constituency parsing	Sentiment analysis	Semantic textual similarity	Language modeling
Shallow syntax-chucking	Semantic parsing	Text classification	Natural language inference	Machine translation
Named entity recognition	Dependency parsing	Temporal processing	Relation prediction	Simplification
Part-of-speech tagging		Coreference resolution		Summarization
Semantic role labeling				Dialogue
Word sense disambiguation				Question answering

It is also a very hard task, given the fact that natural languages do not adhere to the strict rules non-natural languages usually do.

> Human language is highly ambiguous... It is also ever changing and evolving. People are great at producing language and understanding language, and are capable of expressing, perceiving, and interpreting very elaborate and nuanced meanings. At the same time, while we humans are great users of language, we are also very poor at formally understanding and describing the rules that govern language. [16, Page 1]

In fact, the "imitation game", a test developped by Alan Turing in the 50's as a test of artificial intelligence, relies on the ability of a computer program to impersonate a human in a written conversation.[17]

Because of its potential, NLP has been studied for decades (in fact the first proposals for machine-translation pre-date the invention of the digital computer[18]).

This chapter serves as an introduction to the field of natural language processing, first describing the common tasks of NLP and then introducing some frequently used tools and methods.

2.1 Common tasks in natural language processing

With many machine learning (ML) and data mining tasks, we work with data in a form of a large table, where each row represents one object and each column represents a property of the objects. Most ML methods are therefore designed to work with these tables.

When it comes to texts in natural language, they are in their raw form just a series of characters. In order to add some structure to this unstructured data, we usually perform a number of preprocessing steps, which allow us to transform the text into a representation better suited for our needs. This section focuses on the most common preprocessing steps of natural language processing.

2.1.1 Tokenization

The goal of tokenization is to separate the text into smaller units and it is "a fundamental step in both traditional NLP methods... and advanced deep learning-based architectures".[19]

The text may be segmented into paragraphs but most commonly tokenization refers to splitting the text into sentences and words.

Tokenization can remove punctuation too, but that may cause issues such as wrongly splitting up abbreviations with periods (e.g. dr.), where the period following that abbreviation should be considered as part of the same token and not be removed.[13]

There are different techniques of tokenization, their usage depends on the language and the purpose of tokenzation, so this section provides examples of some commonly used tokenizers.

White space tokenization

White space tokenization splits the text into tokens based on white spaces, such as tabulation characters, spaces, newline characters etc..

Although this is a fast and easy way to implement tokenisation, this technique only works in languages where meaningful units are separated by spaces e.g English, but even then, it does not work well for open compound words such as *living room* or *full moon.*[20]

Dictionary based tokenization

Dictionary based tokenization uses a dictionary of tokens to segment the text. If the token is not found in the dictionary, special rules are applied to tokenize it.[21]

For languages without spaces between words, there is an additional step of word segmentation where we find sequences of characters that have a certain meaning.[20]

Regular expression tokenization

Regular expression tokenizers are rule based tokenizers, which use regular expressions to control the tokenization of text into tokens.[21]

It is a useful technique when you want more control over the tokenization of the text by creating your own regular expression by which the text is tokenized.

2.1.2 Stop words removal

Some words which often appear in analyzed document have little informational value and can be excluded. This process is called stop words removal. Stop words removal includes getting rid of common language articles, pronouns and prepositions such as “and”, “the” or “to” in English and other words which may be considered insignificant.[13]

Generally the more often a word or a term appears in a collection of documents, the lesser informational value it has. Therefore the strategy for creating a list of stop words is to sort the terms by total number of times ech term appears in the document collection and pick the most frequent terms as stop words.[22]

Stop words removal is an especially important step in the field of information retrieval, where excluding words with little informational value tends to have a huge impact on the speed of these systems as well as the volume of data that has to be stored.

2.1.3 Lemmatization and Stemming

Since documents use different forms of a word, we often need to reduce the inflectional forms and sometimes derivationally related forms of a word to a common base form. This is especially important (and difficult) with synthetic languages which can be defined as ‘a‘ny

language in which syntactic relations within sentences are expressed by inflection (the change in the form of a word that indicates distinctions of tense, person, gender, number, mood, voice, and case) or by agglutination (word formation by means of morpheme, or word unit, clustering). Latin is an example of an inflected language; Hungarian and Finnish are examples of agglutinative languages."[23].

The two techniques of text normalization are stemming and lemmatization.

Stemming

The algorithms knows as stemmers produce "stems" of a word by cutting off the beginning and the end of the word, usually by usining a list of common prefixes and suffixes.[24]

It is a crude heuristic process, which is why the produced stems often do not correspond to the morphological root of the word. If given the token *saw*, stemming might return *saw* (or possibly just *s*, whereas lemmatization (described in the next section) would likely return either *see* or *saw* depending on whether the use of the token was as a verb or a noun.[25]

Lemmatization

Lemmatization is a process of applying morphological analysis to words in order to remove inflectional endings and transform the words into their base or dictionary forms called "lemmas". Lemmas unlike stems are actual language words.[22]

In lemmatization the normalization depends on the part-of-speech of a word so it either has the be automatically determined in the previous step (the process of automatically determining the part-of-speech of a word in a sentence is described in section INSERT) or this context has to be supplied in another way.

Lemmatization is more sophisticated than stemming, producing more accurate results and meaningful tokens by considering the context, however it has is trade-offs. Compared to stemming, the process of lemmatization is slower and significantly harder to implement.

2.1.4 Part-of-speech tagging

Part-of-speech tagging (POS) is the process of assigning part-of-speech tags to words in a sentence. A POS tag is a label assigned to each token in a document to indicate the part of speech and often also other grammatical categories such as tense or number (plural/singular).[26]

Because POS taggers usually tend to take into account different grammatical categories, their tag set is usually larger that the number of part-of-speech categories of the language they

are intended for. In English, there are frequently listed and tought eight parts of speech (noun, pronoun, verb, adjective, adverb, preposition, conjunction, article and interjection), but the commonly used Penn Treebank tagset contains 36 POS tags and 12 other tags (for punctuation and currency symbols).[27]

2.2 Tools for natural language processing

As Python is "the leading coding language for NLP because of its simple syntax, structure, and rich text processing tool" I have decided to program my implementation of chosen aspect-base sentiment analysis method in this language. In this section I want to provide a brief introduction to two of Python's NLP libraries – NLTK and spaCy.

2.2.1 Python's NLTK library

Python's Natural Language Toolkit (NLTK) is an open source library which contains a wide range of tools and algorithms for building programs aimed at natural language processing. It provides a way to perform standard NLP tasks such as part-of-speech tagging, syntactic parsing or text classification. An overview of some NLTK modules with their functionalities is depicted on figure 2.2.

WordNet

NLTK also provides a WordNet interface, which is a semantically oriented dictionary of English comprising of 155289 words and 117659 synonym sets[29].

WordNet allow us to find a list synonyms to a given word, which in the context of WordNet is called a *synset*. In order to get a synset we call the `synsets()` function with the word for which we wan the synsets as an argument along with an optional part-of-speech tag:

```
from nltk.corpus import wordnet
syns = wordnet.synsets("dog")
print(syns)
```

The function outputs a list of synsets:

```
[Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'),
Synset('cad.n.01'), Synset('frank.n.02'), Synset('pawl.n.01'),
Synset('andiron.n.01'), Synset('chase.v.01')]
```

Each synset is identified with a 3-part name in the form <lemma>.<pos>.<number>, where:

- **<lemma>** is the the lemma of the word

Table 2.2: Language processing tasks and corresponding NLTK modules with examples of functionality[28]

Language processing task	**NLTK modules**	**Functionality**
Accessing corpora	nltk.corpus	Standardized interfaces to corpora and lexicons
String processing	nltk.tokenize, nltk.stem	Tokenizers, sentence tokenizers, stemmers
Part-of-speech tagging	nltk.tag	n-gram, backoff, Brill, HMM, TnT
Classification	nltk.classify, nltk.cluster	Decision tree, maximum entropy, naive Bayes, EM, k-means
Chunking	nltk.chunk	Regular expression, n-gram, named entity
Parsing	nltk.parse	Chart, feature-based, unification, probabilistic, dependency
Evaluation metrics	nltk.metrics	Precision, recall, agreement coefficients
Applications	nltk.app, nltk.chat	Graphical concordancer, parsers, WordNet browser, chatbots

- **<pos>** is a part-of-speech tag
- **<number>** is the sense number used to disambiguate word meanings[30]

WordNet also implements a number of ways how to find similarity between two synsets. Here are some examples along with their description taken from the WordNet howto guide[31]

- `path_similarity` returns a score denoting how similar two word senses are, based on the shortest path that connects the senses in the is-a (hypernym/hypnoym) taxonomy.
- `lch_similarity` returns a score denoting how similar two word senses are, based on the shortest path that connects the senses (as above) and the maximum depth of the taxonomy in which the senses occur.
- `wup_similarity` returns a score denoting how similar two word senses are, based on the depth of the two senses in the taxonomy and that of their Least Common Subsumer (most specific ancestor node).

Another function WordNet offers is the is the `derivationally_related_forms()` function which allows us to find terms which are in a different syntactic category but have the same root form and are semantically related to a word we supply as an argument.

2.2.2 spaCy

spaCy is NLP library for Python and Cython (a programming language written mostly in Python with additional C-inspired syntax). It is a newer library than NLTK, using an object-oriented approach as opposed to NLTK's string approach. Unlike NLTK it has support for word vectors (multi-dimensional meaning representations of a word) and its processing is generally faster than that of NLTK (due to its Cython implementation).

Same as NLTK it provides tools for many NLP tasks such as POS tagging, tokenization, measuring similarity between words etc..[32]

3. Description of the domain of analyzed conference paper reviews

The data on which the analysis will be done are all from events and conferences focused on semantic technology (ST). Sentiment analysis is usually applied to product reviews or posts on social media, therefore this work will serve as an exploration of the possibility of creating a model which, even though focused on one domain, is generalized across various events of the same umbrella subject.

Eventually, the model can be extended to be less domain-specific, given that the review metrics probably will not differ significantly between different research areas.

In this chapter I want to give you a brief introduction into the studied domain of conferences with focus on semantic technology INSERT MORE TEXT.

3.1 Studied conferences within the field of semantic technology

3.1.1 European Semantic Web Conference ESWC

The European Semantic Web Conference (ESWC) is an international conference on the topic of ST which first began in 2004. According to the ESWC website, "the mission of the ESWC is to bring together researchers and practitioners in all these areas dealing with different aspects of semantics on the Web."[33].

The topics of ESWC conferences include linked open data, machine learning, natural language processing and information retrieval, ontologies, reasoning, semantic data management, services, processes, and cloud computing, social Web and Web science, in-use and industrial, digital libraries and cultural heritage, and e-government.[34]

3.1.2 European Knowledge Acquisition Workshop EKAW

The European Knowledge Acquisition Workshop (EKAW) started in 1987 as a workshop in the field of knowledge-based systems and became a conference in 2000 changing its full title to the International Conference on Knowledge Engineering and Knowledge Management.[35]

As they state on their website about the 2020 EKAW conference "the 22nd International Conference on Knowledge Engineering and Knowledge Management is concerned with all aspects about eliciting, acquiring, modeling and managing knowledge, and the construction of knowledge-intensive systems and services for the semantic web, knowledge management, e-business, natural language processing, intelligent information integration, and so on"[36].

3.1.3 International Semantic Web Conference ISWC

The International Semantic Web Conference (ISWC) is a conference with focus on research on semantic web topics including linked data. It is a successor of the Semantic Web Working Symposium (SWWS) and is held annually since 2002.[37]

3.2 Reviewing process of conference submissions

The aim of this section is to give an overview of the steps of a reviewing process to give the reader a better understanding on the context in which the reviews analyzed in this work are created. While the reviewing process of different conferences may vary, there is set of general steps which they all follow to a certain degree.

Each paper submitted to a conference is reviewed by multiple reviewer (usually three or more). After the reviewers submit their reviews authors may be able to react during a rebuttal phase, by clearing up any misunderstandings, answering questions posed by the reviewers and by defending their position on things a reviewer may have complained about. This is a common reviewing step in bigger conferences such as ISWC or ESWC however that may not be the case for smaller conferences such as EKAW where the authors only get the final reviews and whether the submission was accepted or not. Sometimes the authors get the numerical scores given by the reviewers as well as the written reviews with reviewer's comments, but occasionally only the comments are supplied so as the authors cannot use the numerical scores to argue (for example when the scores given by one reviewer are significantly higher that the scores of other reviewers).

Certain conferences also have an "offline" discussion amongst the reviewers, which is not accessible by the authors, which is usually moderated by a track chair.

Based on the reaction of the authors of the papers to the initial reviews the reviewers have the opportunity to adjust their reviews (or they should at least make it clear that they have read the author's response).

The track chair also can write meta reviews, which serve as summaries of the more in-depth reviews by the other reviewers. That is a standard step in conferences where there is also a conference chair above the track chair who makes the final decision about a paper acceptance. In that case the conference chair mostly decides based on the conclusion given in the meta review, it is quite rare for them to decide otherwise.

In terms of anonymity in the reviewing process, there are many different options and situations. Usually the author is not given the names of reviewers, but reviewers can see the names of the authors. Occasionally conferences follow a double-blind model, where the paper's authorship is anonymized. However that is not always possible, for example resource tracks (tracks aimed at sharing resources including datasets, software frameworks, ontologies,

methodologies or metrics) are just about impossible to anonymize due to the fact that the resources the papers refer to need to be publicly available and in use.

Sometimes conferences however do allow the authors to know the names of the reviewers (albeit the reviewers usually have the option to stay anonymous if the wish).

Certain conferences also keep the reviewers anonymous from one another or from the meta reviewer. The goal here is to give less experienced reviewers a chance to express themselves without worrying about the opinion of senior reviewers and vice versa to keep more experienced reviewers from undermining the opinions given by a less experienced reviewer.

3.3 The Structure of conference paper reviews

Different conferences structure their reviews differently. Some are in the form of a completely unstructured text, while some clearly separate the comments in the reviews by the criteria.

For example the structure of the 2018 EKAW conference reviews is that there is a set of criteria which are assigned numeric scores (relevance, overall evaluation and reviewer's confidence) followed by two yes or no scoring (best paper candidate and poster & demo candidate). It is then followed by a summary of the reviewed paper. The reviewers comments are divided into three parts which are *Reasons to accept*, *Reasons to reject* and *Overall evaluation*. The reviews also sometimes contain confidential remarks for the program committee.

The 2017 ISWC and 2018 ISWC conferences were more detailed with its numerical scoring, assigning these scores to:

- Reviewer's confidence
- Appropriateness
- Clarity and quality of writing
- Related work
- Originality/innovativeness
- Impact of ideas and results
- Implementation and soundness
- Evaluation
- Overall paper evaluation

Following these numerical scores, the rest of the ISWC reviews are however in the form of unstructured text, unless the author of the review specifically made the decision to structure their comments either by the judged aspect or by the positivity or negativity of their comments.

The most structured reviews I have had to my disposal to study were from the 2018 ESWC conference. There, numeric score were assigned to:

- Relevance to ESWC
- Novelty of the proposed solution
- Correctness and completeness of the proposed solution
- Evaluation of the state-of-the-art
- Demonstration and discussion of the properties of the proposed approach
- Reproducibility and generality of the experimental study
- Overall score

while the textual part of the review was clearly divided into the same categories.

3.4 Previous research on conference paper reviews

While there is quite a lot of existing research regarding information extraction from research papers, mostly used for paper summarization and extraction of keywords, I could not find much research focused on information extraction or opinion mining from the reviews of such papers. In terms of on possible generic metrics, previous research on this topic was done by Svátek; Strossa [38] in their paper on the possibility of pictorial representation of review scores. The metrics they propose are these:

- Relevance
- Novelty
- Technical quality
- State of the art
- Evaluation
- Significance
- Presentation

Their overview of different metrics of nine confereces with focus on semantic technology and knowledge engineering and their mapping onto their set of generic criteria can be found on table 3.1.

Table 3.1: Proposed mapping between generic review metrics and form fields of KE conferences[38]

Review metric	ECAI (2016)	EKAW (2020)	ESWC (2018)	FOIS (2016)	IJCAI (2019)	ISWC, SEMANTiCS (2018)	KR (2014)
Relevance	Relevance	NA	Relevance to ESWC	NA	Relevance	Appropriateness	Relevance of the paper to KR
Novelty	Originality	Novelty	Novelty of the proposed solution	Novelty or innovation	Originality	Originality / innovativeness	Novelty of the contribution
Technical quality	Technical quality	Technical soundness and depth	Correctness and completeness of the proposed solution; Demonstration and discussion of the properties of the proposed approach	Scientific or technical quality	Technical quality	Implementation and soundness	Technical quality
State of the art	Scholarship	NA	Evaluation of the state-of-the-art	References	Scholarship	Related work	Discussion of related work
Evaluation	NA	NA	Reproducibility and generality of the experimental study	NA	NA	Evaluation	NA
Significance	Significance	NA	NA	NA	Significance	Impact of ideas and results	NA
Presentation	Presentation quality	Clarity and quality of writing	NA	Presentation	Clarity and quality of writing	Clarity and quality of writing	Quality of the presentation

One research I found that was specifically focused on sentiment analysis of reviews of scientific publications was the research done be Bucur; Kuhn; Ceolin [39]. They used a dataset of eleven reviews, each of which was manually annotated by their respective authors. The aspects of the reviews they focused on were syntax, style and content so each reviewer was asked to which of these aspects a specific comment in their review focused on, whether the comment was positive or negative, whether an action by the author of the paper was required or just suggested, what was the impact for the overall quality of the paper and whether the author addressed the point raised in the comment.[39] In the eleven reviews there was a total of 421 review comment, most of which (around 44 %) targeted a paragraph or an even smaller part of the paper, almost 30 % were about the paper as a whole and 27 % of comments focused on a section of the paper. The first fairly interesting outcome of this study was that the authors of the study also annotated the review comments themselves and they gathered annotations from peer reviewers. The compared the results and the level of disagreement using a variation of the Mean Squared Error metric, calculated as the square root of the mean squared differences between the average responses of the groups for numerical dimensions and as thee the squared differences from the ratio for each category separately for nominal dimensions. From the results it was clear that "the model experts and the peers always agree with each other more than they agree with the ground truth in the form of the original reviewer" which according to the authors seems to indicate that "they misinterpret the review comments in a relatively small but consistent manner"[39].

I also want to include here the result of the annotating phase done by the model experts (the authors), which you can see on figure 3.1. As you can see on the graph most of the review comments are about the content of the paper, with much smaller percentages being comments about the style or the syntax. Also the amount of negative comment far exceeds the number of neutral or positive comments.

3.1.

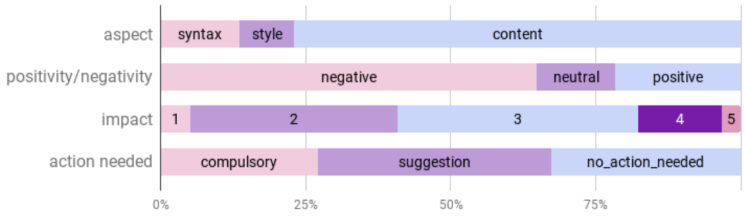

Figure 3.1: The results of the model expert annotations[39]

They have applied 18 different lexicon-based sentiment analysis tools to compare the results and found that the best permorming tool was the SOCAL method[40] with a maximum accuracy of 72.8 %. Most of the methods performed quite poorly according to the authors, however they discovered that the methods with more complex rules performed the best, even if the size of their sentiment lexicon was not large. It is important to note in the context of this work that the sentiment analysis was done separately from the aspects purely focusing on the polarity of the comment and they did not develop or used any tools to automatically

determine the aspect of the comment.

Another research that is focused on the processes of scientific publishing and more importantly peer reviewing of these publications was done by by the same authors as the last paper. This study is focused on creating a unified model for representation of publications and their assessments "as well as the involved processes, actors, and provenance in general"[41] in the format of linked data. Their vision is that to give more context to reviews, by linking them to other data, such as information about the reviewer and about the author of a paper, as well as provide a way to link specific parts of a review to the part of the paper they comment on.

In their study they performed a user study based on a set of 7 competency questions to determine the practicality of the ontology they propose. The respondents were ask to judge the importance of each of these questions on a scale form 1 to 5 (1 meaning *not important at all* and 5 *very important*). One of these questions, was "What is the distribution of the review comments with respect to whether they address the content or the presentation (syntax and style) of the article?" which was given the average importance score of 3.64, which was the second highest importance score in their set of question. This tells me that the focus of my paper might indeed be valuable for the community, as it focuses on an even more fine-grained aspect-based sentiment analysis of the reviews.

4. Analyzed data

4.1 Source of data

As was previously mentioned, the data analyzed in this work are reviews of submissions to conferences focused on semantic technology. Namely from five different conferences – EKAW 2018, ESWC 2018, ESWC 2019, ISWC 2017 and ISWC 2018.

The data for each conference was sourced differently, mainly because with the exception of the ESWC 2019 conference, these reviews are not publicly available.

Data from EKAW 2018 ware gathered with the help of the programme committee co-chair Chiara Ghidini and my supervisor. The reviewers were asked to give consent to the use of their reviews, being able to choose between two levels of consent:

- **Level 1**: I am giving a consent to using my EKAW 2018 review text/score in an anonymized form for the purposes of a sentiment analysis study YES / NO
- **Level 2**: I am giving a consent to using my EKAW 2018 text in a snippet published for illustrative purposes, after the elimination of potentially identity-revealing named entities YES / NO

As a result I was able to obtain 247 review, where only 17 reviewers gave permission on level 1, the rest gave permission on both levels.

The data from the ISWC 2017 and ISWC 2018 were gathered in a similar manner, resulting in 11 and 20 reviews respectively.

I also was provided with a small dataset of the ESWC 2018 reviews by Vojtěch Svátek, consisting of a total of 6 reviews of two different papers.

The ESWC 2019 data was publicly available through a SPARQL endpoint at `https://metadata.2019.eswc-conferences.org/sparql`. The server hosting the SPARQL endpoint recently went down and apparently will not be made available anytime soon, but I have managed to gather some data when the server was still functional.

4.2 Data preprocessing

4.2.1 Data preprocessing for aspect vocabulary extraction

Fist I tokenize each review into sentences. Then, each sentence goes through word tokenzation using the `word_tokenize` function from NLTK and each token is assigned a POS tag with the `pos_tag` function.

Because the reviews are sometimes not formatted correctly the tokenization might lead to a wrong output. For example there might not be a space after a punctuation symbol (like a period) and as a result the tokenizer does not separate the punctuation symbol from the next word. For that reason all characters which are not alphanumerical or are not a hyphen are replaced by an empty string in each token.

All tokens are then lemmatized using the `WordNetLemmatizer`. No stopwords are removed in the preprocessing, as only nouns, noun phrases and adjectives are extracted, so there is no significant overlap with any traditional stopwords and some stopwords, such as prepositions are necessary for noun phrase identification.

4.2.2 Data preprocessing for sentiment vocabulary extraction

Because I am using the Naïve Bayes classifier for sentiment vocabulary extraction, a dataset consisting of 1000 review sentences (taken from the ESWC 2019 dataset) was created. Because reviewers often express two different sentiments in a single sentence, such as "Overall it is a very good paper, but there are some limitations.", sentences like these had to be split in two – the positive and the negative part. As was described in section INSERT, the Naïve Bayes classifier treats a text as a bag of words with an assigned label, there is no syntactic or semantic analysis applied and so a dual polarity in a single example would not be appropriately handled.

Each sentence was labeled with either positive or negative sentiment by me as well as another annotator (independently as to not influence each other). The results of the two sets of annotations were then compared and it was found that the label did not match for 8 sentences. This was firstly because some sentences described both positive and negative sentiment and were not split correctly which led to each annotator choosing a different sentiment for the entire sentence. This was corrected by only keeping a part of the sentence expressing a single opinion polarity. Secondly, the opposing annotations were in some cases a result of a lack of context for the sentence when each sentence is annotated separately. This was fixed by looking at the original review and choosing the appropriate polarity based on the context. Finally some sentences were labeled incorrectly due to a simple mistake of the annotator.

When creating the lexicon, first all contractions are expanded, then the review is tokenized into words and assigned a POS tag. Because only words with a high enough frequency are kept, I have also decided to remove stopwords, based on my own stopword list. The NLTK corpus also includes a dictionary of stopwords, however it includes words that I expected to have a noticeable influence on the polarity of a sentence such as "should" which rarely points to a positive sentiment in reviews. The list of stopwords that were used is:

```
["the", "be", "of", "a", "to", "and", "in", "it", "i", "this",
 "that", "do", "for", "on", "have"]
```

The tokens were again lemmatized, with the exception of adjectives. In this task especially

```
def get_wordnet_pos(pos_tag):

    if pos_tag.startswith('J'):
        return wn.ADJ
    elif pos_tag.startswith('V'):
        return wn.VERB
    elif pos_tag.startswith('N'):
        return wn.NOUN
    elif pos_tag.startswith('R'):
        return wn.ADV
    else:
        return wn.NOUN
```

Figure 4.1: Transformation between Treebank and WordNet POS tag sets

adjectives needed to be kept in the same form as they were originally written in the review, as the distinction between for example “good” and “better” might be important for the polarity of the adjective.

It is important to note that the `WordNetLemmatizer` that is used for lemmatization is based on the WordNet POS tag set, which is significantly smaller that the Treebank tag set that is used for POS tagging. Because the POS tagger uses the Treebank tag set, all POS tags are transformed into WordNet tags using the function on figure 4.1. For example all Treebank tags that refer to verbs begin with “V” and they all get transformed into the single tag that WordNet has for verbs.

5. Implementation of aspect extraction

In order to create a lexicon of terms that represent the chosen set of criteria, to be used for identifying aspect expressions in the reviews, I have decided to use two main approaches. One is the taxonomy extraction mentioned in 1.3.2 and the second one is extraction of frequent words used by the reviewers for different criteria in a text that is already divided by headers into sections for the respective criterion.

5.1 Manually created taxonomy

As described in section 1.3.2, in order to perform taxonomy extraction, we need to manually create a user defined taxonomy.

The original idea is that the taxonomy is a hierarchical representation where the top level of the hierarchy represents the feature of an object and the following levels represent the aspects of that feature. Because in my case, I do not need to separate the chosen criteria any further I have used this representation to have the main criteria in the top level and only have one level underneath the top level, to specify possible aspect expressions for each criterion, which serves as a seed for future expansion of the taxonomy by including extracted crude features with enough similarity to the these expressions.

You can see this taxonomy here 5.1.

5.2 Crude Features extraction

The next step of taxonomy based extraction is to obtain a set of crude features. The next two subsections describe the extraction algorithms based on Bings method of aspect extraction.

5.2.1 Extraction of frequent nouns and noun phrases

The first method of extracting terms that are likely to represent an aspect is to extract frequent nouns and noun phrases. For that the content of the file is tokenized and each token is assigned a Part-of-Speech (POS) tag. Then, to get the nouns and noun phrases, the extracted tuples (token, pos_tag) are parsed to determine the multi-token sequences which represent nouns and NNPs.

Listing 5.1: Manually created taxonomy for aspect extraction

```
{
    'relevance': {
        'appropriateness', relevance'
    },
    'novelty': {
        'originality', 'innovativeness', 'innovation',
        'novelty of contribution', 'novelty', 'impact', '
           significance'
    },
    'technical quality': {
        'scientific quality', 'implementation', 'soundness',
        'technical quality'
    },
    'state of the art': {
        'scholarship', 'references', 'related work',
        'state of the art'
    },
    'evaluation': {
        'reproducibility', 'evaluation'
    },
    'presentation': {
        'clarity', 'quality of writing', 'presentation'
    },
}
```

For the parsing I've used the RegexpParser from the nltk library, which utilizes a user defined grammar, consisting of labeled regular expression rules, describing the sequence of POS tags we want to assign the label to. The result of the parsing is a tree structure, where each sequence corresponding to the regular expression is labeled accordingly, so this allows us to pick the subtrees labeled by the parser as noun phrases.

The code snippet which shows the defined grammar for NP extraction can be seen on listing 5.2. The grammar uses POS tags, so NN are nouns, IN are prepositions and JJ are adjectives. The first regular expression, labeled NBAR, searches for nouns and adjectives, terminated with nouns, allowing us to discover phrases like "black box", where the meaning changes when we consider both of these words separately. The second regular expression, labeled NP, looks for NBAR expressions connected with prepositions such as "of", "in" etc.. [42]

Listing 5.2: Grammar for the extraction of noun phrases.

```
RegexpParser( """
        NBAR:
            {<NN.*|JJ>*<NN.*>}

        NP:
            {<NBAR>}
            {<NBAR><IN><NBAR>}
        """)
```

In order to then extract only the wanted noun phrase sequences, I traverse the tree, and for each subtree, labeled as NP, I lemmatize each token of the sequence, check if its length is at least two characters, but less than 20 characters, and if so, I join the lemmatized tokens into a single string and append it to the list of NPs in the file.

By performing this extraction with each file, I get a list of lists, where each list represents the extracted nouns and noun phrases from one file. To then obtain the ones that are frequent enough across all the reviews, and may therefore represent aspects, I calculate the support of a noun phrase across the reviews. The following equation represents the calculation of the support metric for a word w_i where N_{w_i} is the number of reviews containing the word w_i and N is the total number of reviews:

$$support(w_i) = \frac{N_{w_i}}{N}$$

In order to perform this calculation I transform the data into a matrix, where each row represents one review, each column represents an aspect candidate and the values of an element in row-i and column-j is 1 if the aspect candidate j is present in review i or 0 if it is not. To get the support of an aspect candidate j, I than divide the sum of column-j by the total number of rows. If the support is greater than the minimum support, the aspect candidate is kept, if not, it is discarded. Through various experiment I found that the best value for

minimum support is 2 %, which gives us reasonable candidates for aspect expressions, but does not include too many candidates to make the process of manually confirming them too tedious. It may seem like a very "minimal" minimal support, however the number of reviews I had to my disposal to extract frequent noun phrases was fairly small, so a small support was necessary to account for that fact. It should not be too big of an issue though, considering that the aspect candidates are further tested to determine their similarity to the manually created taxonomy, which reduces the number of aspect candidates the user has to go through.

5.2.2 Extraction of frequent adjectives

Although Bing only focuses on nouns and noun phrases, by going through the training data, I have discovered that fairly often, the aspect expression found in the reviews take the form of adjectives.

Consider the phrase "The topic addressed by the paper relevant to the conference.". Here, the adjective *relevant* evidently corresponds to the aspect *relevance*. Because of that, I have decided to extract frequent adjectives from the reviews as well and again calculate the support of each adjective across the reviews with the same technique as I have used with the noun phrase extraction.

5.3 Extraction by review structure

The data from the 2018 ESWC conference I have obtained had the review text divided into sections where the different sections represented the different criteria. The structure of these reviews can be seen on figure 5.1, where in bold are the structural part of the review format which stays the same across all reviews.

I have decided to leverage this data to extract frequent words from each one these sections across the reviews.

The frequent word of each section are included in the new aspect expression taxonomy directly, they do not go through the same process of similarity matching against the manually created taxonomy as the candidates that were chosen purely on their frequency. This is useful because it allows to extract new possible aspect terms that would not match with any of the terms in the taxonomy, which I originally did not think to include.

Each term frequent enough in an aspect section across all the reviews is included in the taxonomy based on a match between the ESWC set of aspect and my set of metrics. The mapping between the two sets of metrics was done according to table 3.1.

The aspect expression candidates created by this method are still evaluated by the user as explained in section 5.5.

5.4 Similarity matching against the manually created taxonomy

After we obtain a set of crude features, the next step is to map these features to the user defined taxonomy. As previously described, we need to calculate the similarity of a crude feature to the aspects in the taxonomy. If the feature is similar enough, it is passed to the final step of the taxonomy extraction, which is an interactive revision process. This process is described in more detail in the next section.

For measuring the similarity between two terms, I have decided to use the wordnet tool and its path_similarity metric which returns a score denoting how similar two word senses are, based on the shortest path that connects the senses in the hypernym hierarchy. The score is in the range 0 to 1 where 1 denotes identity (when word is compared to itself). [28]

The main obstacle of using this metric is that it does not work well with adjectives. That is because all nouns are part of one big hierarchy, but that is not the case for other parts of speech such as adjective, adverb etc.. So for example the similarity for the words "relevance" and "relevant" is zero, even though the words are closely related. Because the extracted crude features may contain adjectives (in the case of noun phrases) or be adjectives themselves (in the case of frequent adjectives extraction), I have decided to implement a workaround by transforming the adjectives to their closest related noun, and perform the similarity measurement on these nouns. If the similarity of these nouns is greater than the similarity threshold, the original (albeit lemmatized) word is passed on.

Another issue is measuring similarity with terms consisting of multiple words. Certain multi-token terms are already present in the wordnet thesaurus (such as *state of the art*), and getting their synsets to perform similarity matching is as simple as replacing the spaces between words with underscores. However some multi-token words included both in the user defined taxonomy and in the crude features cannot be found in wordnet directly.

To solve this issue I have decided to calculate the maximum similarity between each token of one term to all the tokens of the other term. Of these similarities I then choose the maximal one.

The final pseudocode for similarity measuring between two terms is on figure 5.2. When comparing frequent adjectives to the taxonomy, the part-of-speech argument is set accordingly.

Another issue of wordnet's path_similarity is that it is asymmetrical. So sometimes path_similarity(x,y) returns None or 0 while path_similarity(y,x) returns a non-zero value. This is because for some words, a fake root in the hierarchy may be added to find a path between to two words, but this depends on the order in which the two words are supplied to the path_similarity function.

Therefore the final calculation of similarity between two terms term1 and term2 is defined as the maximum between similarity(term1,term2) and similarity(term2,term1).

The threshold for similarity of a term to the taxonomy was set to 0.3. Therefore every term with a similarity to any term in the manually created taxonomy equal or greater than the threshold will become an aspect expression candidate under the same aspect as the term it was most similar to.

5.5 User validation of final taxonomy

When aspect expression candidates are generated and sorted by the aspect they most likely represent they have to pass the final validation. This validation is a manual process, where a user goes through every aspect candidate and decides between three options:

- The aspect expression candidate is added under the aspect which was algorithmically determined as the most probable.
- The user disagrees with the most probable aspect aspect and sorts the aspect expression candidate under a different aspect.
- The user decides not to include the aspect expression candidate in the taxonomy at all.

The interactive process of the user validation of the final taxonomy can be seen on figure 5.3.

5.6 Resulting taxonomy of aspects

Relevance to ESWC:
numerical score (score interpretation)
Novelty of the Proposed Solution:
numerical score (score interpretation)
Correctness and Completeness of the Proposed Solution:
numerical score (score interpretation)
Evaluation of the State-of-the-Art:
numerical score (score interpretation)
Demonstration and Discussion of the Properties of the Proposed Approach:
numerical score (score interpretation)
Reproducibility and Generality of the Experimental Study:
numerical score (score interpretation)
Overall score:
numerical score (score interpretation)
Reviewer's confidence:
numerical score (score interpretation)
Open Reviewing Opting Out:
numerical score (score interpretation)
Overall evaluation (*Resources and In-Use tracks only*, Research reviewers please only put "n/a"):
numerical score (score interpretation)
———— **Relevance to ESWC** ————
reviewer's comment
———— **Novelty of the Proposed Solution** ————
reviewer's comment
———— **Correctness and Completeness of the Proposed Solution** ————
–
reviewer's comment
———— **Evaluation of the State-of-the-Art** ————
reviewer's comment
———— **Demonstration and Discussion of the Properties of the Proposed Approach** ————
reviewer's comment
———— **Reproducibility and Generality of the Experimental Study** ————
reviewer's comment
———— **Overall score** ————
reviewer's comment

Figure 5.1: ESWC 2018 review structure

```
Data: term 1, term 2, part-of-speech
Result: the similarity of the term as a numeric score between 0 and 1
for both terms do
    if the term is just one word then
        term_synsets = find all synsets of the term ;
        if the part-of-speech is adjectives then
            term_synsets += find all sysets of the closest related noun
    else
        term_underscored = substitute all spaces in the term with underscores ;
        term_synsets = find all synsets of term_underscored ;
    if no synsets are found for a multi-word term then
        term_synsets = synsets of all words in the term (including transformed
         adjectives) ;
score = maximum similarity between all term_synsets of term1 and term_synsets of
 term2 ;
return score
```

Figure 5.2: Algorithm for similarity measurement between two terms

```
Does the term "topic" belong under aspect "relevance" ? [y/n]
y
Does the term "relation" belong under aspect "relevance" ? [y/n]
n
Does it belong under any of these aspects ?:
relevance                [a]
novelty                  [b]
technical quality        [c]
state of the art         [d]
evaluation               [e]
presentation             [f]

none of the above        [n]
n
Does the term "introduction" belong under aspect "novelty" ? [y/n
   ]
n
Does it belong under any of these aspects ?:
relevance                [a]
novelty                  [b]
technical quality        [c]
state of the art         [d]
evaluation               [e]
presentation             [f]

none of the above        [n]
p
```

Figure 5.3: The interactive process of user validation of proposed aspect taxonomy.

6. Creation of sentiment lexicon

Through some initial experimentation with various existing sentiment lexicons such as the SenticNet sentiment lexicon and the NLTK's SentiWordNet sentiment lexicon I discovered that these universal sentiment lexicons are not well suited for application on the domain of conference paper reviews.

One issue is that both of these sentiment lexicons assign sentiment polarity on a scale. Therefore most dictionary words have a certain sentiment polarity which I considered inappropriate in this task, as I did not want words which would generally be considered neutral to somehow skew the polarity of words that might actually be important. These words often have a polarity around the center of the polarity interval (for example if the polarity is assigned on a scale from -1 to 1 they usually have polarity somewhere around 0) and could be removed by using some polarity threshold, but that might also lead to losing some words that are actually important for sentiment classification in this domain. For example in SenticNet, the word "clarification" has polarity of -0.09, but it is often used in sentences such as "The section about experimental results needs some clarification.." where the sentiment is clearly negative.

Another possible issue with pre-made sentiment lexicons is that they mostly do not include punctuation. I believe that punctuation might have great semantic significance in conference paper reviews, as they are often written in plaintext and punctuation is used to compensate for the lack of usual formatting styles such as bullet points.

For that reason, I have decided to create my own sentiment lexicon. This chapter describes the process of compiling a sentiment lexicon from a set of annotated sentences taken from reviews using the Naïve Bayes classifier.

6.1 Implementation of sentiment lexicon generation using Naive Bayes

The Naïve Bayes classifier, as described in section 1.2.1 is a probabilistic classifier which needs a training dataset of labeled data in order to determine the influence of different evidences on the class. The obtainment of this dataset is described in section INSERT.

I have used the NLTK's implementation of the classifier. Of all the lemmatized tokens in the reviews I limited the number of tokens the classifier needs to process (the tokens are considered features by the algorithm, each token being transformed into a column where the value of the column for each row representing a review is true or false depending on the presence of the token in the review) to 450. This was achieved thanks to the `FreqDist` class from the NLTK's `probability` module.

To dataset of labeled review sentences was split to have a testing dataset 50 example sentences to determine the accuracy of the classifier and the rest of sentences was used for training. The test dataset is fairly small, but I considered it far more preferable to leave most examples to the training dataset to produce a hopefully more accurate sentiment lexicon, even though the trade-off is not being able to judge the created lexicon in great detail at this phase.

That being said, the accuracy of the classifier on the testing dataset was 0.8, meaning 80 % of sentences were classified correctly.

The classifier allows us to get a list of features which have the highest contribution to classification through its `show_most_informative_features` method which based on the number we specify as its argument outputs a list of features with their ratio of occurrences in negative and positive sentences. The output when applied to the training data can be seen on table 6.1. We can see that it is more that 21 times more likely that the word "easy" occurs in a sentence labeled as *positive* while a question mark occurs far more often in negative sentences, as was expected in section INSERT.

Table 6.1: Output of the Naïve Bayes classifier

Most Informative Features		
easy = True	positi : negati =	21.3 : 1.0
interesting = True	positi : negati =	15.2 : 1.0
topic = True	positi : negati =	13.6 : 1.0
? = True	negati : positi =	11.5 : 1.0
what = True	negati : positi =	10.8 : 1.0
sound = True	positi : negati =	10.6 : 1.0
community = True	positi : negati =	9.8 : 1.0
but = True	negati : positi =	9.0 : 1.0
not = True	negati : positi =	8.0 : 1.0
idea = True	positi : negati =	8.0 : 1.0
interest = True	positi : negati =	7.8 : 1.0
clearly = True	positi : negati =	7.4 : 1.0
well = True	positi : negati =	7.3 : 1.0
me = True	negati : positi =	7.3 : 1.0
bring = True	positi : negati =	6.7 : 1.0

I have then adjusted the `show_most_informative_features` method to create a function which transforms these ratios of occurrences in sentences with positive or negative sentiments into a sentiment lexicon.

Each of the most informative tokens is given a value of -1 or +1 depending on if they occur more often in negative or positive sentences (where -1 corresponds to negative sentiment and +1 corresponds to positive sentiment).

6.2 Created sentiment lexicon

From the list of most informative features I chose the top 88 words to include to the sentiment lexicon (setting the ratio threshold at 2.6 : 1.0).

I decided to compare the results with the SenticNet sentiment lexicon, to see what is the level of agreement between the two lexicons and found that in 13 cases, the polarity of the sentiment of words found in both lexicons differed and in 31 cases a word from my lexicon was not found in SenticNet. Surprisingly not all the words that were not found in SenticNet were not found due to the aforementioned lack of punctuation in SenticNet or because these words could truly be considered neutral, SenticNet was missing some words which I would consider fairly meaningful in sentiment analysis such as *rather* or *should*.

I also included a list of 38 positive and 62 negative words compiled manually during the process of labeling the training dataset. The resulting sentiment lexicon contains 171 sentiment words out of which 79 have positive polarity of +1 and 92 have a negative polarity of -1.

7. Implementation of aspect-based sentiment analysis for conference paper reviews

7.1 Design of classes

7.1.1 Review

Review
file_name : string sentences : list criteria : dictionary
get_scores() : dictionary add_score(criterion : string, value: int) print_results()

Figure 7.1: Review class diagram

Each review is represented by the `Review` class. When the class is initiated with its constructor the attribute `file_name` is initialized with the file_name of the review (for the purpose of printing the results). The review text is the tokenized into sentence using NLTK's `sent_tokenize` function. To keep track of the numerical scores of the review, which are assigned through the process of sentiment analysis, there is a dictionary `criteria`, in which the keys of the dictionary are the set of generic criteria as chosen in section INSERT. The values are represented by the `CriterionScore` class described in section INSERT.

The class has also three methods. The `add_score` method accepts the name of a criterion and a sentiment value (-1 or +1) that should be added to the criterion and updates the `criteria` dictionary accordingly. The `get_scores` function returns the final scores for a review in the form of a dictionary where the different criteria are the keys and the values are the numerical scores normalized between 1 and 5. Finally the `print_results` function outputs the scores of the review onto the command line in a format shown on figure 7.2.

```
row233.txt
        relevance                       3
        novelty                         5
        technical quality               1
        state of the art                5
        evaluation                      2
        presentation                    1
```

Figure 7.2: Example output of the `print_results` method of the `Review` class

7.1.2 Sentence

Sentence
tokens : list sentence : string sentence_original : string aspects : list opinion_words : list criterion_orientation : dictionary unoriented_aspects : list
print_results()

Figure 7.3: Sentence class diagram

The `Sentence` class represents one sentence from a review. When initialized the original sentence is kept in the `sentence_original` attribute for printing purposes, however for the needs of the sentiment analysis the sentence is also tokenized and lemmatized. I have used the `MWEtokenizer` here as well as NLTK's `word_tokenize` function. The `MWEtokenizer` takes a string in the form of a list of tokens and retokenizes it, "merging multi-word expressions into single tokens, using a lexicon of MWEs" INSERT. The reason is that because some aspect expression are multi-words expressions, such as "state of the art" I need to keep them as single tokens, in order for them to be recognized when matching the sentence against the aspect taxonomy. Therefore each aspect in the taxonomy is added as a multi-word expression to the `MWEtokenizer` lexicon.

The tokens are also lemmatized, with the exception of adjective to keep words such as "better" to be lemmatized to "good" (the reasoning behind that was explained in section INSERT). The tokenized and lemmatized sentence is then kept in the `tokens` attribute of the class and the lemmatized tokens are stringed back together in the `sentence` attribute.

For the sentiment analysis algorithm it is also key to know which aspect expressions and which sentiment words the sentence contains. To get a list of aspect expressions I call the `Taxonomy`'s class method `get_aspects`, which compares the tokens it gets as an argument

with the taxonomy of aspects and returns the matches as a list of the `Aspect` class objects. The list of aspect expression is kept in the `aspects` attribute of the class.

To get a list of sentiment words the `find_opinion_words_sentiment` function is called, which belongs to the sentimentr module. This function gets a list of tokens as an argument and then for each token that is found in the sentiment lexicon it calculates the polarity value using the sentimentr method (described in section INSERT). It returns a list of (sentiment word, polarity) tuples, which are then used to initialize the `OpinionWord` class. However if there is an overlap between a sentiment word and an aspect expression, the word is at this point discarded (it may be used in the future, if the orientation of an aspect expression is not found any other way as explained in more detail in section INSERT). The list of sentiment expression is kept in the `opinion_words` attribute.

If the main part of the sentiment analysis algorithm fails for an aspect expression (it evaluates the polarity at zero) there is a set of rules which try to determine the sentiment in some other ways. The list of aspect expressions which need the further evaluation is kept in the `unoriented_aspects` attribute to which these aspects are continuously added as the review is analyzed.

To keep a track of which criteria a sentence is focused on as well as the polarity of the opinion that is expressed, there is the `criterion_orientation` attribute, which is a dictionary where the keys are the criteria and the values are numerical scores.

The `Sentence` class contains a single method, `print_results`, which allows the user to see the results of the analysis on a more fine-grained level than the `print_results` method of the `Review` class. It prints the original sentence and for each criterion in the sentence shows if the polarity of the opinion on the criterion is positive, negative or neutral.

7.1.3 OpinionWord

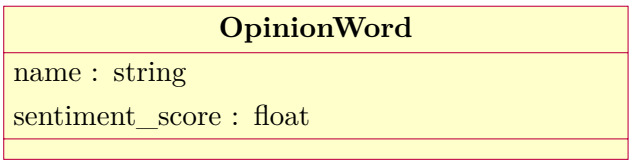

Figure 7.4: OpinionWord class diagram

The `OpinionWord` class represent an opinion/sentiment word with the name of the word in the `name` string attribute and the sentiment polarity determined by the sentimentr algorithm kept in the `sentiment_score` attribute as a decimal value.

7.1.4 CriterionScore

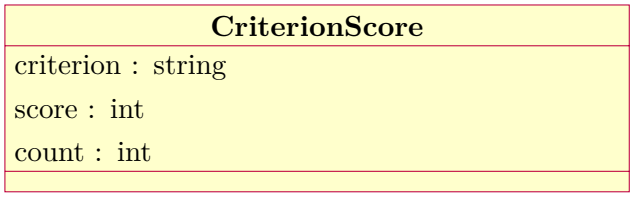

Figure 7.5: CriterionScore class diagram

The `CriterionScore` class is to keep track of the six main criteria which the reviews are evaluated by. The `criterion` string attribute is assigned the name of the criterion, the `score` attribute keeps track of the numerical score of the criterion and the `count` attribute counts the number of times new value is added to the `score`. Any time an aspect expression belonging to the criterion is discovered in a sentence and its orientation is evaluated by the sentiment analysis algorithm it is added to the `score` attribute so the `count` attribute allows as to count the average value so all scores are normalized.

7.1.5 Taxonomy

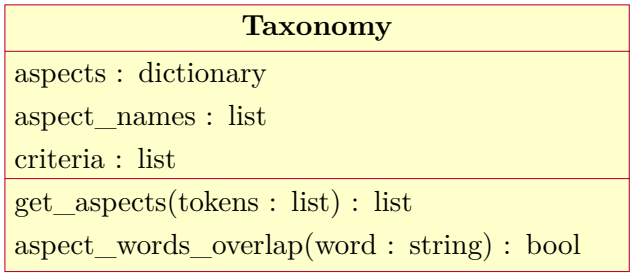

Figure 7.6: Taxonomy class diagram

The `Taxonomy` class represents the taxonomy created in INSERT. The class is initiated only once, when the taxonomy is loaded from the JSON file in which it was saved during the taxonomy generation phase. The `aspects` attribute is a python dictionary which follows the same structure as the JSON (see figure INSERT). For simplicity of other operation the `criteria` attributes keeps a list of all the main criteria/metrics, while the `aspect_names` attribute keeps track of all aspect expressions in the taxonomy.

There are two method in the class, the method `get_aspects` accepts a list of tokens in a sentence, finds if any of the tokens are aspect expressions and if so, returns a list of the `Aspect` class objects that correspond to them. The second method `aspect_words_overlap` accepts a string as an argument and returns *True* if the string overlaps with a string of an aspect expression and *False* otherwise.

7.1.6 Aspect

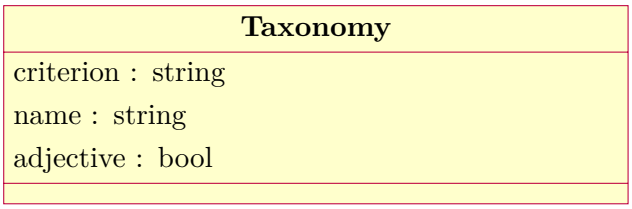

Figure 7.7: Aspect class diagram

The `Aspect` class simply represents an aspect expression, the name of which is saved in the `name` attribute. It also keeps the name of the criterion/metric under which the aspect expression belongs in the `criterion` attribute, in the form of a string.

If the aspect is an adjective the `adjective` attribute is set to *True* to enable some special handling of these expressions. The value of `adjective` is determined by finding the wordnet synsets of the expression and if the POS tag of any of the synsets is that of an adjective (in NLTK's wordnet implementation that means the tag is either *a* for an adjective or *s* for satellite adjectives).

7.2 Description of the algorithm

The aspect-based sentiment algorithm that I have created for the task of extracting criteria orientations from a text is loosely based on the algorithm mentioned in section INSERT, but I have made some changes. The main part of the algorithm is shown on figure 7.8.

7.2.1 Determining opinion orientation using aspect expressions and sentiment words in a sentence

For each review we iterate through its sentences (which are here objects of class `Sentence`). If the number of aspect expressions in the sentence is more than 5 the sentence is not evaluated. That is because if the number of aspects is that high in a single sentence it would be hard to evaluate which opinion words belong to which aspect. It is especially an issue with the numerical evaluations at the beginning of reviews, which are sometimes not structured by newlines or any other separator.

Given a sentence s_i that contains a set of aspect expression we compute a sentiment polarity score for the expression. Given a set of sentiment words in the sentence, we calculate their influence based on the sentiment score given by the sentimentr algorithm which is divided by the distance of the sentiment word from the aspect expression. These scores are then aggregated by a sum function for each aspect expression.

```
Function opinion_orientation:
    for each sentence s_i that contains a set of aspect expressions do
        if the number of aspect expressions in s_i > 5 then
            continue;
        for each aspect a_j in s_i do
            orienation = 0;
            for each opinion word ow_k in s_i do
                distance = distance_betweeen_words(ow_k, a_j, s_i);
                orientation += ow_k.sentiment_score / distance;
            if orientation > 0 then
                orientation = 1;
            else
                if orientation < 0 then
                    orientation = -1;
                else
                    orientation = 0;
            if orientation = 0 and a_j is an adjective then
                orientation = 1;
            else
            add a_j to s_i's unoriented_aspects;
            a_j's criterion orientation in s_i += orientation;
            review.add_score(a_j's criterion, orientation)
        for aspect a_l in s_i's unoriented aspects do
            orientation = apply_intra_sentence_rules(s_i, a_l);
            if orientation = 0 then
                sentiment = find_opinion_words_sentiment(a_l);
            if orientation > 0 then
                orientation = 1;
            else
                if orientation < 0 then
                    orientation = -1;
                else
                    orientation = 0;
            a_l's criterion orientation in s_i += orientation;
            review.add_score(a_l's criterion, orientation)
```

Figure 7.8: opinion_orientation algorithm

```
Function apply_intra_sentence_rules:
    for each opinion word ow_i in the sentence do
        words_between = words between aspect and ow_i;
        if adversative in words_between then
            return −1 × ow_i.sentiment_score
```

Figure 7.9: apply_intra_sentence_rules algorithm

An expression is assigned the sentiment polarity of +1 if it is positive and -1 if it is negative, 0 if the polarity was not determined. If the polarity is 0, it is added to the list of unoriented aspects of a sentence for further processing, but if we succeeded in determining the polarity, the score for an criterion under which the aspect expression belongs in the taxonomy is adjusted in the review as well as the sentence.

7.2.2 Adjectives as aspect expressions

When for some aspect expressions the orientation is unknown after aggregating polarities of nearby opinion words they are added to s_i's list of unoriented aspects. These aspect expressions are then first evaluated using the adjective rule, where if the aspect expression is an adjective, it may be used as an opinion word itself. That is the case in sentences such as "This paper is highly relevant to the conference", where the adjective *relevant* points to the relevance criterion, but also expresses a positive polarity on said criterion.

In cases when the aspect expression is an adjective, its polarity is determined using the sentimentr algorithm as if it was an opinion word, to cover cases such as negation.

7.2.3 Intra sentence rules

If the aspect expression orientation is still unknown after the use of the adjective rule it is evaluated using the intra sentence rules, which rely on the fact that a sentence only expresses one polarity unless it includes an adversative conjunction.

For each so far unoriented aspect expression we find the closest opinion word and all the words between the aspect and the opinion word. If there is a adversative conjunction in between the opinion word and the aspect expression, that likely means the sentiment polarity was inverted by the conjunction and therefore the aspect expression should be given the opposite polarity of the opinion word. This should help in sentences such as "The evaluation shows great results but the dataset was small." in which *dataset* might be an aspect expression pointing to the evaluation criterion but *small* might not be in the sentiment lexicon (for a human, it is easy to point to small as a negative word here based on the context, but that is not always the case, like in a phrase *small error* it would be positive). The closest identified

opinion word in this sentence would be *great*, with a positive polarity, but given the fact that there is *but* in between *great* and *dataset*, the polarity assigned would be negated.

If no adversative conjunction is found, the aspect expression is given the polarity of the closest opinion word without any changes, following the *one sentence – one polarity* idea.

7.2.4 Sentences with neutral sentiment

If the orientation of an aspect expression is still 0 after the application of all rules, it is finally evaluated as neutral. Sometimes, this might mean that the sentiment was present but expressed in a way that the algorithm did not recognize. It might also mean a false positive for an aspect identified within a sentence. That is because there might be an overlap between aspect expressions and expression that are often used within the field for describing an idea presented in a paper . Take the sentence "In this paper authors investigate how state-of-the-art language technologies can be ported to the historical ecology domain." in which the algorithm would think that the *state-of-the-art* expression points to the state of the art criterion but here it is simply a statement about the objective of the paper.

These aspect expression do not influence the numerical scores of a review, the aspects with neutral orientation are however still shown in the algorithm's output at a sentence level.

8. Evaluation of results

8.1 Evaluation using reviews with numerical scores

The reviews from ISWC 2018 contain numerical scores for a wide range of criteria as was mentioned in section 3.3. I have decided to compare the numerical scores outputted by the sentiment analysis algorithm with the ground-truth scores taken from the reviews. Because the ISWC set of criteria is more detailed that the set of criteria the algorithm works with, it was necessary to create a mapping between them which you can see on table 8.1.

Table 8.1: Mapping between the chosen set of criteria and ISWC 2018 criteria

Algorithm's criteria	**ISWC 2018 criteria**
relevance	appropriateness
novelty	originality/innovativeness
	impact of ideas and results
technical quality	implementation and soundness
state of the art	related work
evaluation	evaluation
presentation	clarity and quality of writing

The numerical output was evaluated using the mean absolute error function (MSE), which measures the absolute average distance between the real data Y and the predicted data $\bar{Y}$:

$$MSE = \frac{1}{n} \times \sum_{i=1}^{n} |Y_i - \bar{Y_i}|$$

The MAE was calculated separately for each criterion to see if the algorithm perform better or worse for some them. The default range of [1;5] for scores outputted by the algorithm was matched to the [-2;2] range of the ISWC reviews. Because the algorithm outputs "n/a" instead of a number for criteria for which no sentiment value was found, I have also calculated the number of times the "n/a" value occurs for each criterion.

The results of the numerical evaluation carried over the 20 ISWC 2018 reviews can be seen on table 8.2. It is clear that the algorithm often struggles with finding any criterion score, especially when it comes to relevance, novelty and technical quality. Even when it does give a numerical score it is often fairly off. This could have several explanation. Firstly it is possible that when reviewers have the option of expressing their opinion numerically, they sometimes do not feel to also give a more elaborate explanation. The second explanation is that the algorithm simply does not perform well when it comes to discovering aspect expressions and/or sentiment words. This is studied more closely in the next section, where the results are evaluated on the sentence level.

Another issue might be the way in which the numerical scores are estimated – each time an aspect expression is discovered and assigned a polarity of +1 or -1 the value is added to the respective criterion score. Finally the scores are averaged by the number of times a value was added and normalized to a given score range. Therefore if for a criterion only one aspect expression is found with a given polarity the final score will always be an extreme in the score range, but that is a much rarer occurrence in the scores given by human reviewers. To check if this might be the issue I have tried to change the normalization of polarity to four different values, where a criterion is added a score of +1 if the orientation of an aspect expression is higher than 0.5, a score of +0.5 if the orientation is higher that 0 and analogously the -0.5 and -1 scores for negative orientations. The results after that change can be seen on table 8.3. It is apparent that this leads to better results and so a more fine-grained approach to polarity is necessary.

Table 8.2: Results of the numerical evaluation of ISWC 2018 reviews

Criterion	**MAE**	**Number of missing values**
relevance	1.25	12
novelty	2.25	14
technical quality	2.375	12
state of the art	1.0	6
evaluation	1.7	3
presentation	1.09	9

Table 8.3: Results of the numerical evaluation of ISWC 2018 reviews using a more granular approach to polarity

Criterion	**MAE**	**Number of missing values**
relevance	0.87	12
novelty	1.58	14
technical quality	1.5	12
state of the art	0.74	6
evaluation	1.05	3
presentation	0.81	9

Conclusion

Závěr je povinnou částí bakalářské/diplomové práce. Obsahuje shrnutí práce a vyjadřuje se k míře splnění cíle, který byl v práci stanoven, případně shrnuje odpovědi na otázky, které byly položeny v úvodu práce.

Závěr k diplomové práci musí být propracovanější – podrobněji to je uvedeno v Náležitostech diplomové práce v rámci Intranetu pro studenty FIS.

Závěr je vnímán jako kapitola (chapter), která začíná na samostatné stránce a která má název Závěr. Název Závěr se nečísluje. Samotný text závěru je členěn do odstavců.

References

1. LIU, Bing. *Sentiment analysis: mining opinions, sentiments, and emotions*. Cambridge University Press, 2015. ISBN 978-1-107-01789-4.
2. SCHOUTEN, K.; FRASINCAR, F. Survey on Aspect-Level Sentiment Analysis. *IEEE Transactions on Knowledge and Data Engineering*. 2016, vol. 28, no. 3, pp. 813–830. ISSN 2326-3865. Available from DOI: 10.1109/TKDE.2015.2485209.
3. HU, Minqing; LIU, Bing. Mining Opinion Features in Customer Reviews. In: *Proceedings of the 19th National Conference on Artifical Intelligence*. San Jose, California: AAAI Press, 2004, pp. 755–760. AAAI'04. ISBN 0-262-51183-5. Available also from: http://dl.acm.org/citation.cfm?id=1597148.1597269.
4. WEBB, Geoffrey I. Naïve Bayes. In: *Encyclopedia of Machine Learning*. Ed. by SAMMUT, Claude; WEBB, Geoffrey I. Boston, MA: Springer US, 2010, pp. 713–714. ISBN 978-0-387-30164-8. Available from DOI: 10.1007/978-0-387-30164-8_576.
5. JURAFSKY, Daniel; MARTIN, James. *Speech and Language Processing*. 3rd ed. draft. 2018. Available also from: https://web.stanford.edu/~jurafsky/slp3/.
6. PRINCETON UNIVERSITY. About WordNet. *WordNet A Lexical Database for English*. 2010. Available also from: https://wordnet.princeton.edu/.
7. RINKER, Tyler; SPINU, Vitalie. *sentimentr*. Zenodo, 2016. Version 0.4.0. Available from DOI: 10.5281/zenodo.222103.
8. DING, Xiaowen; LIU, Bing; YU, Philip S. A Holistic Lexicon-based Approach to Opinion Mining. In: *Proceedings of the 2008 International Conference on Web Search and Data Mining*. Palo Alto, California, USA: ACM, 2008, pp. 231–240. WSDM '08. ISBN 978-1-59593-927-2. Available from DOI: 10.1145/1341531.1341561.
9. CARENINI, Giuseppe; NG, Raymond T.; ZWART, Ed. Extracting Knowledge from Evaluative Text. In: *Proceedings of the 3rd International Conference on Knowledge Capture*. Banff, Alberta, Canada: ACM, 2005, pp. 11–18. K-CAP '05. ISBN 1-59593-163-5. Available from DOI: 10.1145/1088622.1088626.
10. ASGHAR, Dr. Muhammad; KHAN, Aurangzeb; ZAHRA, Rabail; AHMAD, Shakeel; KUNDI, Fazal. Aspect-based opinion mining framework using heuristic patterns. *Cluster Computing*. 2019, vol. 22. Available from DOI: 10.1007/s10586-017-1096-9.
11. LYONS, John. Language, speech and writing. In: *Natural Language and Universal Grammar: Essays in Linguistic Theory*. Cambridge University Press, 1991, vol. 1, pp. 1–11. Available from DOI: 10.1017/CBO9781139165877.003.
12. GUDIVADA, Venkat N.; ARBABIFARD, Kamyar. Chapter 3 - Open-Source Libraries, Application Frameworks, and Workflow Systems for NLP. In: GUDIVADA, Venkat N.; RAO, C.R. (eds.). *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*. Elsevier, 2018, vol. 38, pp. 31–50. Handbook of

Statistics. ISSN 0169-7161. Available from DOI: https://doi.org/10.1016/bs.host.2018.07.007.

13. YSE, Diego Lopez. *Your Guide to Natural Language Processing (NLP)*. 2019. Available also from: https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1. online, accessed 02-November-2020.
14. IORIN, Ilia. *Natural Language Processing (NLP) Use Cases for Business Optimization*. Available also from: https://mobidev.biz/blog/natural-language-processing-nlp-use-cases-business. online, accessed 02-November-2020.
15. TECHLABS, Maruti. *Top 12 Use Cases of Natural Language Processing in Healthcare*. Available also from: https://marutitech.com/use-cases-of-natural-language-processing-in-healthcare/. online, accessed 02-November-2020.
16. GOLDBERG, Yoav. Neural Network Methods for Natural Language Processing. *Synthesis Lectures on Human Language Technologies*. 2017, vol. 10, no. 1, pp. 1–309. Available from DOI: 10.2200/S00762ED1V01Y201703HLT037.
17. TURING, Alan M. Computing Machinery and Intelligence. *Mind*. 1950, vol. 59, no. October, pp. 433–460. Available from DOI: 10.1093/mind/LIX.236.433.
18. HUTCHINS, W. J. *Machine Translation: Past, Present, Future*. USA: John Wiley & Sons, Inc., 1986. ISBN 0470203137.
19. PAI, Aravind. *What is Tokenization in NLP? Here's All You Need To Know*. Available also from: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/. online, accessed 02-November-2020.
20. MUJTABA, Hussain. *Tokenising into Words and Sentences | What is Tokenization and it's Definition?* 2020. Available also from: https://www.mygreatlearning.com/blog/tokenization/. online, accessed 03-November-2020.
21. CHAKRAVARTHY, Srinivas. *Tokenization for Natural Language Processing*. 2019. Available also from: https://towardsdatascience.com/tokenization-for-natural-language-processing-a179a891bad4. online, accessed 03-November-2020.
22. MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich. *Introduction to Information Retrieval*. Cambridge University Press, 2008. Available from DOI: 10.1017/CBO9780511809071.
23. ENCYCLOPAEDIA BRITANNICA, The Editors of. *Synthetic language*. Encyclopædia Britannica. Available also from: https://www.britannica.com/topic/synthetic-language. online, accessed 03-November-2020.
24. GARCÍA Clara, Cabanilles; PEDRO, Juan; RAMIREZ, Benjamín. *What is the difference between stemming and lemmatization?* Available also from: https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/. online, accessed 03-November-2020.
25. BERI, Aditya. *Stemming vs Lemmatization*. 2014. Available also from: https://towardsdatascience.com/stemming-vs-lemmatization-2daddabcb221. online, accessed 03-November-2020.

26. ENGINE, Sketch. *POS tags*. 2018. Available also from: https://www.sketchengine.eu/blog/pos-tags/. online, accessed 03-November-2020.

27. MARCUS, Mitchell P.; MARCINKIEWICZ, Mary Ann; SANTORINI, Beatrice. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.* 1993, vol. 19, no. 2. ISSN 0891-2017.

28. BIRD, Steven; KLEIN, Ewan; LOPER, Edward. *Natural Language Processing with Python*. 1st. O'Reilly Media, Inc., 2009. ISBN 0596516495.

29. *WordNet with NLTK: Finding Synonyms for words in Python*. Guru99. Available also from: https://www.guru99.com/wordnet-nltk.html. online, accessed 09-November-2020.

30. PROJECT, NLTK. *nltk.corpus.reader package*. 2020. Available also from: http://www.nltk.org/api/nltk.corpus.reader.html?highlight=wordnet#nltk.corpus.reader.wordnet.Lemma.synset. online, accessed 09-November-2020.

31. *WordNet Interface*. Available also from: https://www.nltk.org/howto/wordnet.html. online, accessed 09-November-2020.

32. KAKARLA, Swaathi. *Natural Language Processing: NLTK Vs SpaCy*. ActiveState, 2019. Available also from: https://www.activestate.com/blog/natural-language-processing-nltk-vs-spacy/. online, accessed 09-November-2020.

33. *Mission*. Semantic Technology Institute International. Available also from: https://www.eswc-conferences.org/. online, accessed 03-November-2020.

34. *The Semantic Web: Research and Applications 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012, Proceedings*. 1st ed. 2012. 2012. Information Systems and Applications, incl. Internet/Web, and HCI ; 7295. ISBN 3-642-30284-X.

35. *About*. Knowledge Media Institute. Available also from: http://ekaw.org/. online, accessed 04-November-2020.

36. *22nd International Conference on Knowledge Engineering and Knowledge Management*. EKAW 2020. Available also from: https://ekaw2020.inf.unibz.it/. online, accessed 04-November-2020.

37. *International Semantic Web Conference (ISWC)*. SWSA (Semantic Web Science Association). Available also from: http://swsa.semanticweb.org/content/international-semantic-web-conference-iswc. online, accessed 04-November-2020.

38. SVÁTEK, Vojtěch; STROSSA, Petr. Let's Get the Best Papers to the Finish Line: Ontological and Pictorial Representation of Review Scores. 2019.

39. BUCUR, Cristina-Iulia; KUHN, Tobias; CEOLIN, Davide. *Peer Reviewing Revisited: Assessing Research with Interlinked Semantic Comments*. 2019. Available from arXiv: 1910.03218 [cs.DL].

40. TABOADA, Maite; BROOKE, Julian; TOFILOSKI, Milan; VOLL, Kimberly; STEDE, Manfred. Lexicon-Based Methods for Sentiment Analysis. *Computational Linguistics*. 2011, vol. 37, no. 2, pp. 267–307. Available from DOI: 10.1162/COLI_a_00049.

41. BUCUR, Cristina-Iulia; KUHN, Tobias; CEOLIN, Davide. *A Unified Nanopublication Model for Effective and User-Friendly Access to the Elements of Scientific Publishing*. 2020. Available from arXiv: 2006.06348 [cs.DL].

42. KARIMKHAN. *Extracting the noun phrases using nltk*. 2016. Available also from: https://gist.github.com/karimkhanp/4b7626a933759d0113d54b09acef24bf#file-noun_phrase_extractor-py-L34. online, accessed 30-October-2020.

Attachements

A.

B. Zdrojové kódy výpočetních procedur