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.

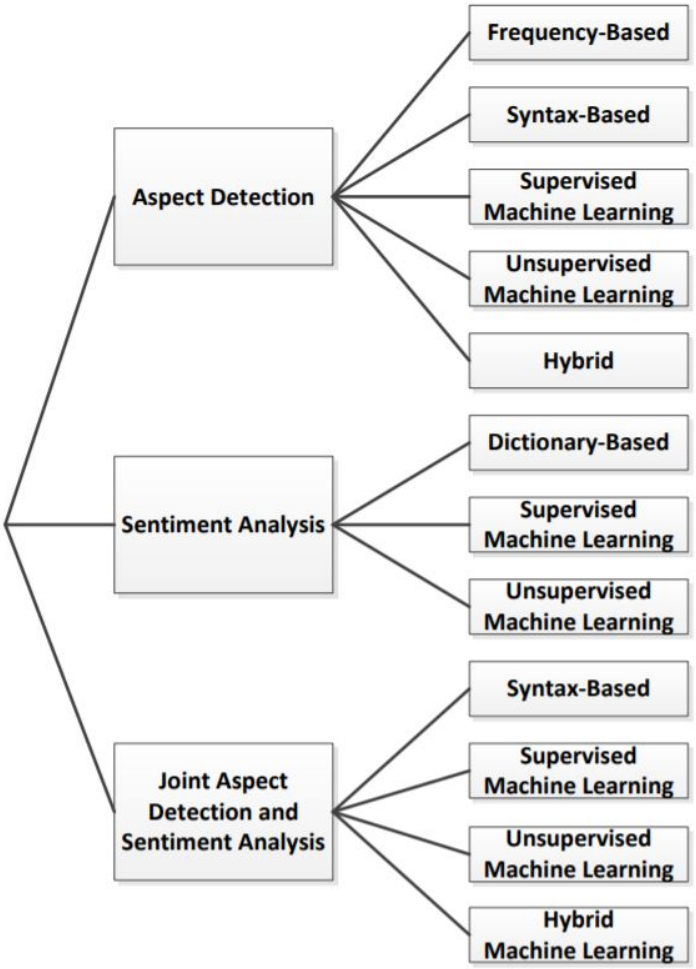

Figure 1.1: Taxonomy for aspect-level sentiment analysis [2]

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 hypothesies is the most probable given the evidence.

In the task of sentiment analysis, the different "classes" that serve as hypothesies 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 anotated with it's sentiment. The idea behind the bag-of-word approach is depicted on figure 1.2 .

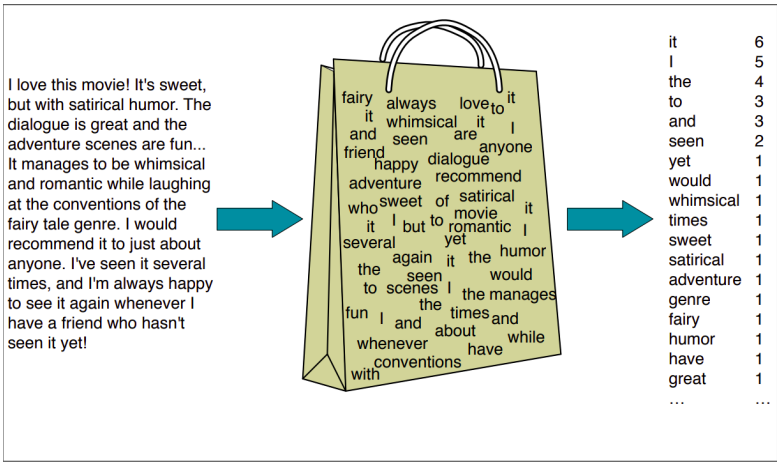

Figure 1.2: Intuition of the multinomial naive Bayes classifier applied to a movie review. [5]

Then we calculate all the neccessary 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 belogning to the class c and N is the total amount of texts in the training dataset.

The probabily of a word w_i occuring 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's how we'd 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 Dictionnary-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, utilise 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 utilises 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 judgin 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. Negator 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 sugests they increase (amplify) the polarity of a sentiment word. There's 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 Deamplifiers, like "sligtly, 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 it's 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, it's 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 assings 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's another opinion word in the sentence for which the orientation is known and the clauses containg the two opinion words are connected by a conjunction such as "and", we can assing that orientation to the context-dependent orientation word as well.
- **Pseudo intra-sentence conjunction rule** – this rule applies to sentences without an explicit conjuction, 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 conjuction rule helps assign the orientation using the context of the surrouding sentences. Acording 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 developped 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.

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

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

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 it's 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's "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]

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.[20]

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.”[21].

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.[22]

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.[23]

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.[20]

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 determin-

ing the part-of-speech of a word in a sentece 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 tradeoffs. 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).[24]

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).[25]

2.2 Tools for natural language processing

2.2.1 Python's NLTK library

Wordnet

3. Analysed data

3.1 Source of data

3.2 Data preprocessing

3.2.1 Chosen aspects for extraction

3.2.2 Data preprocessing for aspect vocabulary extraction

3.2.3 Data preprocessing for sentiment vocabulary extraction

3.2.4 Interesting findings during the preprocessing phase

3.3 Previous research on conference paper reviews

4. 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's already divided by headers into sections for the respective criterion.

4.1 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 hierarchial 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 don't need to seperate 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 4.1.

4.2 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.

4.2.1 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 4.1: 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, descibing the sequence of POS tags we want to asign 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 listining 4.2. 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.. [26]

Listing 4.2: 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 it's 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 contaning 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's discarded. Through various experiment I found that the best value for

minimum support is 2 %, which gives us reasonable candidates for aspect expressions, but doesn't 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 shouldn't 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.

4.2.2 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.

4.3 Extraction by review structure

The data from the 2018 ESWC conference I've obtained had the review text divided into sections where some of the sections represented the different criterias. I have decided to leverage this data to extract frequent words from each one these sections accross the reviews. This is usefull because it allows to extract new possible aspect terms that would not match with any of the terms in the taxonomy, because I originally did not think to include them.

4.4 Similarity matching againts 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've decided to use the wordnet tool and it's 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). [27]

```
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 4.1: Algorithm for similarity measurement between two terms

The main obstacle of using this metric is that it doesn't work well with adjectives. That is because all nouns are part of one big hierachy, but that's 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 theausaur (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've 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 4.1. 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 hiearachy 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 treshold 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 treshold will become an aspect expression candidate under the same aspect as the term it was most similar to.

4.5 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 manuall 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 4.2.

4.6 Resulting taxonomy of aspects

```
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 4.2: The interactive process of user validation of proposed aspect taxonomy.

5. Creation of sentiment lexicon

5.1 Implementation of sentiment lexicon generation using Naive Bayes

5.2 Created sentiment lexicon

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

6.1 Description of the algorithm

6.2 Design of classes

7. Evaluation of results

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. MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich. *Introduction to Information Retrieval*. Cambridge University Press, 2008. Available from DOI: 10.1017/CBO9780511809071.
21. ENCYCLOPAEDIA BRITANNICA, The Editors of. *Synthetic language*. Encyclopædia Britannica. Available also from: https://www.britannica.com/topic/synthetic-language. online, accessed 03-November-2020.
22. 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.
23. BERI, Aditya. *Stemming vs Lemmatization*. 2014. Available also from: https://towardsdatascience.com/stemming-vs-lemmatization-2daddabcb221. online, accessed 03-November-2020.
24. ENGINE, Sketch. *POS tags*. 2018. Available also from: https://www.sketchengine.eu/blog/pos-tags/. online, accessed 03-November-2020.
25. 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.

26. 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.

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

Přílohy

A. Formulář v plném znění

B. Zdrojové kódy výpočetních procedur