

Generative and Explainable AI for Market
Surveillance
University of the Aegean
Department of Information and Communication
Systems Engineering

Antonios Lekkas
324202009

Supervisor: Panagiotis Symeonidis

November 2024

Keywords

Chat-GPT, text generation, text summarization, transformer models, similarity measures, path similarity, Wu-Palmer similarity, Leacock-Chodorow similarity, Jiang-Conrath similarity, Python, keyword extraction, sentiment analysis, product analysis, data preprocessing, python libraries

Abstract

One of the areas of interest in sentiment analysis is product analysis. The trend for using sentiment analysis tools using AI is gaining more ground every day, after viewing the favourable results for the companies and the customers so far. The creation and the demonstration of operation of a new application that can help the users view the positive and negative attributes of products according to other customers' opinions can be very helpful.

Contents

1	Introduction	8
2	Related Work	10
2.1	Sentiment Analysis and related work	10
2.2	Levels of sentiment analysis	14
2.3	Feature selection and preprocessing	15
2.3.1	Artificial Intelligence in the preprocessing stage	16
2.4	Sentiment analysis	16
2.4.1	Sentiment analysis and machine learning	18
2.4.2	Reasoning Artificial Intelligence	18
2.4.3	Explainable Artificial Intelligence	19
2.5	Problems of sentiment analysis	20
2.6	BERT	20
2.7	Chat GPT	21
3	Proposed method	26
3.1	Phrase similarity measure example	26
3.2	Software libraries and similarity measures	28
3.2.1	WordNet library	28
3.2.2	Path Similarity	28
3.2.3	Wu-Palmer similarity	28
3.2.4	Leacock Chodorow similarity	29
3.2.5	Li similarity	29
3.2.6	Manjula similarity	29
3.2.7	Jaccard index and similarity	30
3.2.8	Extended Jaccard similarity	30
3.2.9	Weighted Jaccard similarity	30
3.2.10	Weighted Extended Jaccard similarity	31
3.2.11	Information Content	31
3.2.12	Resnik similarity	32
3.2.13	Jiang Conrath similarity	32
3.2.14	Lin similarity	32
3.2.15	word2vec	32
3.2.16	Cosine Measure similarity	33

3.3	The function of the application	33
4	Experimental Evaluation	38
4.1	Performance evaluation metrics	38
4.2	Datasets	41
4.3	Experimental results	46
4.3.1	Artificial dataset	46
4.3.2	crowdfower dataset	47
4.3.3	Yaaser dataset	48
4.4	Analysis of results	56
4.5	Comparison with related work	58
4.6	Problems with the implementation and the experiments	61
5	Conclusions	63
5.1	System assessment	63
5.2	Future research and ideas for improvements	64

List of Figures

2.1	Implementation stages of Generative AI. [51]	17
2.2	The stages of Sentiment Analysis. [1]	18
2.3	The Transformer model architecture. [63]	23
2.4	The Transformers architecture as defined in Chat GPT. [60]	24
3.1	The stages of the operation of the developed application.	33
3.2	The Textbox that a user can see and use with all the buttons that are associated with the actions we are describing.	36
3.3	An example of the bar plot with the results from the sentiment classification that a user can see. Each bar has its own index and colour.	37
4.1	Results from running the test artificial dataset.	39
4.2	Hovering over the bar for the negative results.	40
4.3	Hovering over the bar for the neutral results.	41
4.4	Hovering over the bar for the positive results.	42
4.5	Having chosen a product from the list, the user can press the button Print Comments and the comments concerning this product will appear in the textbox, classified by sentiment.	43
4.6	Having typed in and entered a word , the user can press the button Print Comments and the comments concerning this word, regardless of product name, will appear in the textbox, classified by sentiment.	44
4.7	Results for the entered word hard .	45
4.8	Having chosen a product from the list, the user can press the button Print All and the prevalent negative attributes of the product will appear in the textbox. Below them, the comments concerning this product will appear, classified by sentiment.	46
4.9	Test results from using crowdfower dataset for all threshold values.	50
4.10	Test results from using Yasser dataset for all threshold values.	53
4.11	Precision-recall diagram micro-averaged from running the crowdfower dataset with the threshold at 0.3.	54
4.12	Precision-recall diagram micro-averaged from running the crowdfower dataset with the threshold at 0.5.	54

4.13	Precision-recall diagram micro-averaged from running the crowdflower dataset with the threshold at 0.7.	55
4.14	Precision-recall diagram micro-averaged from running the crowdflower dataset with the threshold at 1.0 for Path, Wu-Palmer, Jiang-Conrath, Leacock-Chodorow similarity.	55
4.15	Precision-recall diagram micro-averaged for the crowdflower dataset, where the accumulated results for all threshold values (0.3,0.5,0.7,1.0) are shown.	56
4.16	Precision-recall diagram micro-averaged from running the yasser dataset with the threshold at 0.3.	57
4.17	Precision-recall diagram micro-averaged from running the yasser dataset with the threshold at 0.5.	58
4.18	Precision-recall diagram micro-averaged from running the yasser dataset with the threshold at 0.7.	59
4.19	Precision-recall diagram micro-averaged from running the yasser dataset with the threshold at 1.0.	59
4.20	Precision-recall diagram micro-averaged for the Yasser dataset, where the accumulated results for all threshold values (0.3,0.5,0.7,1.0) are shown.	60

List of Tables

4.1	Performance metrics results for artificial dataset.	47
4.2	Performance metrics results for crowdflower dataset.	47
4.3	Precision results for crowdflower dataset organized from best to worst.	48
4.4	Recall results for crowdflower dataset organized from best to worst.	49
4.5	F1 results for crowdflower dataset organized from best to worst. .	49
4.6	Performance metrics results for Yasser dataset.	51
4.7	Precision results for Yasser dataset organized from best result to worst.	51
4.8	Recall results for Yasser dataset organized from best result to worst.	52
4.9	F1 results for Yasser dataset organized from best result to worst.	52

Chapter 1

Introduction

The volume of data in the internet right now is so large that it cannot be processed without the aid of programming and machine learning techniques. The volume of twitter data is a significant part of this volume due to the pervasive nature of platforms like twitter. People will write and comment on topics of interest with no limit to the range of topics.

In effect, twitter has become an online platform where any user can comment on any topic concerning oneself or the rest of the world and in any language. A new twitter comment can be posted within a few seconds or milliseconds. It is useful to process this volume of comments in order to understand the opinion of the public on certain issues.

This project is focused on comments posted in the english language and concerning commercial products. We aim to demonstrate how a fairly simple system can be developed to process any number of twitter posts in order to ascertain the quality and characteristics of products, such as an ipad or an iphone.

This is being done with python language programming and the use of certain programs. The program we have developed can do the following:

- It can accept data from twitter posts and process it.
- It can analyze the body of the text of each post and categorize the sentiment of the post as positive, neutral or negative.
- It can export results about the amount of different sentiment towards a product, or even in relation to a search keyword that is provided by the user in a textbox.
- It can visualize the results and project the posts for each category in an interactive way.
- It can project the negative characteristics of a product in a textbox, if there is a significant amount of negative opinions for a specific characteristic of a product.

The tools we use in order to achieve the above are python programming, xml programming and sentiment analysis techniques. Python is a programming language that can be used for software development, that can read and write files and that can handle big data. XML is a markup language that can be used to carry and store data. Sentiment analysis, in a nutshell, is the process of extracting information from a source and identifying the opinion (or the sentiment) of the writer towards a subject. The information is extracted from a text that has been written by a human and so, the use of natural language processing techniques and text analysis is essential for sentiment analysis. For the aims of this project we will engage in a subfield of sentiment analysis that is called product analysis and has to do with the attributes and qualities a product has and the opinion of customers, or potential customers, towards it.

Furthermore, we aim to show that it is possible to choose between different techniques to produce quicker results without sacrificing accuracy. In the end, it is about quick results in a world that competition is one click away. For the aims of this projects we will not delve into machine learning methods, but it is possible to integrate our program in such systems.

Finally, we aim to show how there can be a connection to another useful tool for graph databases, neo4j. We will show how the relationships between the different entities of data in our system are represented and how this creates different options for the management of the categorized data.

Chapter 2

Related Work

2.1 Sentiment Analysis and related work

Sentiment analysis is a subfield of natural language processing, that concerns the automatic processing of texts in order to ascertain the emotion or the sentiment of the author or authors.

Sentiment analysis techniques are a useful tool in text processing in this modern reality of more and more structured or unstructured data in the internet. The scope of the analysis is limitless. Any issue concerning any group of peoples or even individuals could be a subject of sentiment analysis, i.e. politics, music, sports, products e.t.c..

There are many experiments with sentiment analysis applications on twitter data. Most of them have to do with machine learning methods, either just in the sentiment classification stage or in various stages of the analysis. Twitter itself, can be viewed as a messaging platform, an online communication platform or a platform for comments about any subject that people care. It can also be called a micro-blogging platform, or at least considered as such. The way that we view twitter, influences the way we can use it as users and the way we can use it as developers. In other words, what we think that is appropriate to write in twitter, what we expect to find or read in twitter and what we expect to extract or learn from twitter.

Regarding opinion analysis or product analysis, the data that is posted in twitter can be very useful, if they are processed as we have discussed above. Indeed, because of the widespread use of twitter from users around the world and because in twitter there is almost no obstacle or objection to subjects discussed, data from twitter are a very good source for analyzing the prevalent sentiment about an issue or a product in the community of users. Twitter trends nowadays are a subject of discussion in every day life or even in news outlets. The directness of communication between members of the community and the openness of the discussions (or comments to original posts) can help companies and governments create a strategy, from publicising and advertising to crisis management

and response. As these parties want to have analysis results of the highest accuracy possible, the huge volume of data from twitter that needs to be processed and analysed demands the use of significant computational resources. These resources can be used efficiently with creative approaches including machine learning methods.

So, as we have seen, the motivation and the interest from various parties is enough to bolster the interest of the scientific community and of professionals such as sociologists, advertisers and psychologists. As for data management experts and programmers their interest is bolstered and the evidence for this is the multitude of experiments and new methods of processing data from twitter and extracting useful information that sees no end.

Go et al. created a binary approach to sentiment classification where posts were classified as positive or negative. To achieve this, they extracted posts from twitter with emoticons that helped in the classification. Machine learning methods such as Naive Bayes and Support Vector Machine were used in the classification. The best results, though, were achieved when using the MaxEnt (Maximum Entropy) classifier with an achieved accuracy of 83% [8].

Malhar and Ram came up with a supervised method to classify data. With the use of Support Vector Machine method they achieved accuracy results of 88%. The use of Support Vector Machine in combination with Principal Component Analysis in order to reduce the number of dimensions yielded accuracy results as high as 92% [9].

Anton and Andrey made experiments with sentiment analysis and twitter data using Naive Bayes and SVM. They had better results with SVM, achieving 81% precision and 74% recall [10].

Taboada et al. used a lexicon-based approach to analyze and categorize whole documents. In order to do this they created collections of positive and negative words. Moreover, they proposed a method called semantic orientation calculator, which takes into account intensifiers, that is adverbs to give emphasis and negations. They achieved 76.37% accuracy [6].

The proposal of Taboada et al. has a special interest because it has to do with a lexicon based approach and in this approach a system with a weighted bag-of-words is used. In their work, it can be hypothesized from the beginning that each word has a sentiment polarity, or sentiment orientation, based on its concept and regardless of its use in a text, i.e. out of context. This orientation can be represented by a numerical value in a system called Semantic Orientation CALCulator, or SO-CAL. In their proposal, separate dictionaries for nouns, verbs, adverbs and adjectives were created. These dictionaries were created with polarity ranking, from -5 for negative to 5 for positive, excluding neutral words of 0 polarity ranking. An effort has been made to organize sentiment analysis separately for different parts of speech and using lemmatization of words, that is the conversion of the word to its dictionary form, the lemma. For example, the lemma of *running* is *run*, the lemma of *lively* is *lively* and the lemma of *positional* is *position*. A few notable words that can weigh on the sentiment orientation of a word in the text are also used, words like *exceptional* or *disgraceful*. Another way of capturing the correct sentiment of words and expressions is using intensifiers.

Intensifiers are words that can modify the polarity ranking of a word by a percentage regardless of its polarity being positive or negative. For example the word *slightly*, decreases polarity ranking while the word *extraordinarily* increases it. Negation words, such as *no* and *not*, can also be used to reverse the polarity ranking of a word but in the work of Taboada et al. a method of shift negation is preferred, where a negation word may influence the polarity ranking of a word by a fixed ranking and not by reversing its own. For example, the phrase *not excellent* with a ranking of 5 for *excellent* could have a reversed ranking due to the word *not*, -5, but with shift negation of 3, its polarity ranking will be $5 - 3 = 2$ [6].

Harb et al. also worked with categorizing whole documents. They divided the extracted data into training set and test set. After getting the results of the analysis on the train set, two collections of positive and negative adjectives are created from the text of the documents in the train set. These collections are used in the classification of the documents that belong in the test set. Harb et al achieved a value of f1 at 0.717 for the identification of positive documents and a value of f1 at 0.622 for the identification of negative documents [7].

Zagibalov et al. worked in product analysis in the chinese language. They created a system of interactive training where with more comments they classified the more the list of polarized words was updated with new additions from the text of the new comments. They achieved an f1 value of 87% after 20 iterations [11].

Pak and Peroubek used the same approach as Go et al to classify twitter posts. In their experiments the posts were classified as positive, neutral or negative and they achieved their best results by using the Multinomial Naive Bayes (MNB) classifier [12].

Kouloumpis et al. experimented with different linguistic features and trained the AdaBoost classifier using linguistic features in different combinations. Using hash tags from twitter data they achieved an f1 value of 0.68 and using hash tags and emoticons they achieved an f1 value of 0.65 [13].

Kouloumpis et al. used emoticons for sentiment analysis, but the messages with both positive and negative emoticons were omitted. In their work, they used certain preprocessing techniques in three steps. These were tokenization, normalization and part-of-speech tagging. In the tokenization step, emoticons and abbreviations were identified and considered as separate tokens. In the normalization step, abbreviations were modified and explained, for example *brb* was turned to *be right back*. Intensifiers were also identified and words with all capital letters had them turned to lower and words with repeated letters had them turned to a single letter, for example *happyyyyyyyy* to *happy*. In handling negations, negation words were removed and the word *not* was added to a word preceding or following the negation term [13].

Saif et al. proposed a method of using different linguistic features with data extracted from twitter and organized into collections of semantic concepts. They used the Naive Bayes classifier as a testbench for their experiments. Their proposed method achieved an f1 value of 68.15 for a twitter dataset and 78.20 for another twitter dataset [14].

Hamdan et al. aimed at improving the accuracy at the stage of sentiment classification. They used WordNet library to identify parts of speech and their synonyms and SentiWordNet to compute the frequency of positive and negative words in the text of tweets. Akba et al also aimed at enhancing sentiment classification with the use of feature selection [15].

Lin and Kolcz proposed training Logistic Regression classifiers with hashed sequences of successive words. Different training datasets were used varying in the number of the size of these datasets and in the number of ensembles of classifiers. With a larger number of classifiers the score of the accuracy increased, but the required time for the completion of the classification also increased. The best result was achieved with 21 classifiers and 100 million examples in the dataset. The accuracy of classification that was achieved was 0.81 [17].

Da Silva et al. also proposed an ensemble approach which comprised of SVM, MNB, random forest and logistic regression classifiers. With different datasets different results were achieved, with the worst accuracy score being 76.81% and the best being 84.89% [18].

Hagen, Mathias et al. created an ensemble model which they called Webis. They used four classifiers for twitter sentiment classification. Their best result was an f1 value of 64.84% [19].

Martinez-Cámara, Eugenio et al. also used an ensemble classifier for twitter sentiment classification. In their model they included Word2Vec. Their best result with using ensemble approach was an f1 value of 62.98% [20].

Chalothorn and Ellman using an ensemble approach in combination with Bag-of-words and lexicon methods achieved a higher score for f1, 86.05% [21].

Fouad et al. achieved even higher accuracy in their results. They used a system of majority voting with three classifiers (SVM, Naive Bayes and logistic regression). The highest score they achieved for accuracy was 93.94% for a certain twitter dataset. For another twitter dataset the value of accuracy that was achieved was 78.70%. It should be noted that for a certain twitter dataset, a better accuracy result was achieved by using only Naive Bayes classifier, at 85.09%, rather than an ensemble method where the accuracy achieved was 84.75% [22].

Xia et al. proposed using characteristics for each text that can indicate emotion, the emotion indicators. With their use they tested models of emotional indicators at word level. They achieved scores of accuracy at 0.692 and 0.726 for certain datasets [23].

Azzouza, Nouredine et al. created a real-time system of opinion analysis of twitter data. The system gets data from twitter directly via Twitter API, preprocesses them and classifies them based on a dictionary method. The score for accuracy that was achieved was 0.559 for a certain collection of data and 0.533 for a certain dataset [24].

Paltoglou and Thelwall by using a lexicon method and focusing on subjective texts achieved high values for f1. For three different twitter datasets they got f1 scores of 76.2%, 80.6% and 86.5% [25].

Masoud et al. chose a different classification in three categories, positive,

negative and unbiased. The system they implemented also took into account and analyzed slang in tweets. They achieved high scores of 92% precision and 87% in grouping results between multiple classes [26].

Asghar et al. also used a lexicon method and by using classifiers sequentially they managed to achieve f1 scores of 0.8, 0.795 and 0.855 for three different datasets [27].

Ghiassi et al. also used Twitter API to get data from twitter and they used two classification methods, dynamic artificial neural network (DAN2) and SVM. They also compiled a short lexicon from twitter data. With DAN2 they achieved accuracy scores of 92.5 on negative classification and 68.2 on positive classification. With SVM they achieved accuracy scores of 91.45 on negative classification and 67.6 on positive classification [28].

Khan et al. used SentiWord, emoticon analysis and an enhanced method of classification. In experiments with six datasets, the f1 score was 83.3% and the accuracy score was 85.7% [29].

Zainuddin et al. used an aspect based approach for extraction and identification of separate entities and their classification. In combination with Principal Component Analysis and other methods for feature selection the accuracy scores that were achieved for three different datasets were 71.62%, 76.55% and 74.24% [30].

Asghar et al. created a system for sentiment analysis in twitter that comprised four classifiers, among them one for slang and one for emoticons. They managed to achieve an f1 score of 0.92 and improved accuracy from 79% to 85% [31].

Neogi, Garg et al. proceeded to analyze sentiment in twitter regarding the farming protest in India. Using Bag-of-Words method and TD-IDF they achieved the best accuracy results with random forest machine learning method. These accuracy scores were 96.62% for Bag-of-Words method and 95.51% for TD-IDF [32].

2.2 Levels of sentiment analysis

Sentiment analysis can be performed at levels in the body of text. These levels are document level, sentence level, phrase level and aspect level.

In document level, the whole text of a document is traversed or investigated in order to measure whether the full text of the document is positive, neutral or negative.

However, in a document there may be found a variety of sentiments or opinions. In that case a more thorough or a more localized analysis will yield results closer to reality. In sentence level, analysis takes place for each sentence. The measurement or the result for each sentence may be aggregated in order to categorize the full text of the document or used independently. Sentence level sentiment analysis can be very useful when dealing with ambiguous statements that need to be analyzed independently.

In phrase level, analysis goes even deeper, in parts of the sentence, phrases or individual words. In this level, analysis could yield even better results when trying to categorise the sentiment of each phrase or word. It is the nature of the text that is analysed that is the guide for choosing analysis level. If a variety of sentiments and descriptions is found in a sentence, this level of analysis should be used.

If it is known that sentences contain multiple aspects, aspect level sentiment analysis should be used. In aspect level, every aspect of the sentence is analysed and the results are then aggregated in order to categorise the sentence.

2.3 Feature selection and preprocessing

When a text is to undergo analysis, it is not enough to choose the analysis level. It must be made sure that the text will be in a format that it will make analysis accurate and easy. Pre-processing of text can help in that goal. Raw text or raw data, especially in today's world where phrases and expressions from everyday life, slang, emojis or abbreviations may find their way in the text could dramatically change the results of the analysis if they are not taken into account or dealt with somehow. Punctuation marks may also affect the procedure of analysis. The nature of data depends on the data source. The text could be a document about philosophy or technical issues or specifications. But it could be a comment from a social media platform such as facebook or twitter. Or a text from a blog concerning a topic of everyday life. It could be short or long and depending on the medium or the author could be highly structured and strictly phrased or a loose combination of words, abbreviations, symbols or phrases or something between them. There could also be pictures or sound in the body of the text, if its source is a multimedia platform and we have to recognize the widespread use of internet memes by now.

The identification and selection of irrelevant features in the body of text is also used to improve analysis results. When considering pre-processing, such as the search for certain features or aspects and their extraction, or the exclusion from analysis, there are a few issues that need attention. The recognition of specific words or terms and the measurement of frequency in the body of text is one option. Another option is word tagging in order to identify which part of speech are they and using them accordingly for analysis ,i.e. article, noun, verb, e.t.c. Negations should also be handled with care because they can alter the meaning of a sentence even by appearing once. Apart from negative and positive sentiments there are also neutral sentiments and they can affect the categorisation of sentiment. One of the simplest approaches is the method Bag of Words (BoW). In the BoW method, a vector of words is created and their occurrence in the body of text is measured by the use of different metrics. The first step is the creation of a vector of words. This vector can be compiled from a collection of documents. The frequency of the occurrence of these words in a text is counted in a vector which is unique for this document. This way the

text can be classified based on the frequency of the occurrence of the words. It should be noted that the BoW method is a simple method that ignores the syntax of words in the documents. [56] [57] [58]

Word embedding is a technique by which the words are represented in a vector space, where a group of synonyms is created. Each word in the group is vectored and represented by a value according to a preordained vocabulary set. This vectorized group can be transformed into a vector of values that can be used for sentiment analysis with the help of techniques such as word2vec.

2.3.1 Artificial Intelligence in the preprocessing stage

In the preprocessing stage, in order to be able to test a similarity algorithm, it might be convenient to create new data just for testing and training purposes. In order for this new data to be of value in testing and training, one option is to be totally new, not copied from anywhere and created in such a way that it is completely original. In this way, a real-life system will not be pre-trained according to already existing data and the results will not be biased towards older data. This method is called Generative Artificial Intelligence and is applied to many different types of data, not just text. Audio and image applications have also been developed. The most famous Generative Artificial Intelligence tool among common users around the world is ChatGPT. The launch of ChatGPT brought forth to the public's attention the progress and the capabilities of Artificial Intelligence so far. It also increased the interest of research community in Artificial Intelligence (AI) applications and expedited the development of more AI tools with a broad range of applications.

Not only translation or coding tools are used nowadays, but also conversational AI tools, which are also known as chatbots. Other AI tools for text have been developed for medical advice generation, itinerary creation or retrieval of information from documents. An example of AI text tools in real life is systems that generate e-mail responses, imitating the writing style of a human, or of the personnel of a specific company.

AI platforms have been developed for the creation of artistic images, or image editing. AI image editing is an especially important field for retrieval of information and security applications. AI platforms also have been created for the generation of video output from text, audio, image or video input. This can be very useful for improving video for specific audiences, such as video personalization applications. AI tools also exist for imitating human speech.

2.4 Sentiment analysis

When preprocessing is done, we can move on to the main task of sentiment analysis. The first stage is to identify emotionally charged or emotionally polarized words or phrases that can give a clear indication about the sentiment of the text. This is called subjectivity classification. The next step is sentiment classification where the main task of the analysis takes place. Text can

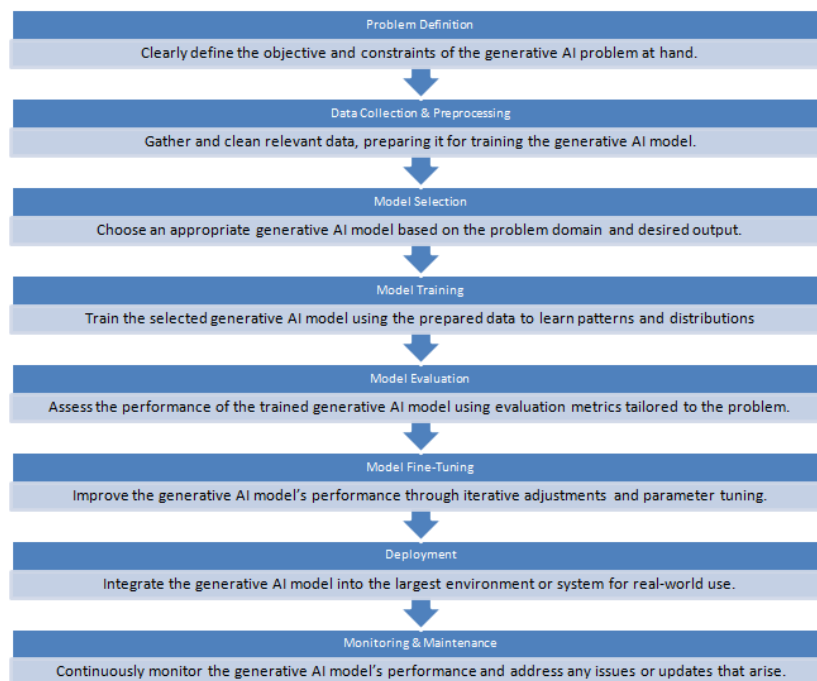


Figure 2.1: Implementation stages of Generative AI. [51]

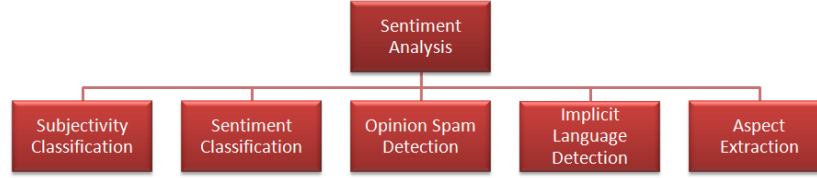


Figure 2.2: The stages of Sentiment Analysis. [1]

be classified to positive or negative, but a category of neutral can be included in order to represent better borderline cases. Machine learning techniques are used in this stage, as well as in the next stage which is called opinion spam detection. In this third stage, fake or false data is identified and removed from consideration in order to get more accurate results. The fourth stage implicit language detection. That is the detection of ironic or sarcastic expressions that can change the sentiment of the text, or even negate it completely. The last stage is aspect extraction. That is the extraction of aspects based on a pre-compiled set of words and phrases and analyzing them separately. The analysis does not take into account only the sentiment classification level of each aspect but also its frequency, or its position in the text.

2.4.1 Sentiment analysis and machine learning

There is extensive use of machine learning methods in sentiment analysis, as in many other fields. The stage in which machine learning is used very much is sentiment classification. It is noteworthy, though, that machine learning can be used in other stages as well. Machine learning is useful as well for training the various models, and thus improve their performance. It can also be used for selecting the segments of data that are the most influential in sentiment classification. There are two main approaches, the supervised and the unsupervised. The unsupervised methods include a combination with lexicon based approaches in sentiment analysis, which include databases and lexicons specifically designed for sentiment analysis. The supervised methods generally produce more accurate results, though.

2.4.2 Reasoning Artificial Intelligence

In the previous section, we discussed Generative AI and its applications. The use of AI is not just confined to the preprocessing stage. In order to extract useful results and an in-depth analysis other methods that belong to the sub-field of Reasoning AI can be used.

In effect Reasoning AI is used to produce new conclusions from already known information using what we know about logical principles and relations.

It used to be that research in Reasoning AI was focused on formal language but this has shifted to natural language, mainly because the AI tools that would analyze formal language would reach a point where a human would be needed to explain certain expressions and terms.

There are three kinds of reasoning, deductive, inductive and abductive reasoning. In deductive reasoning every conclusion has to be proven and supported by existing data. In inductive reasoning, known data is used to make a general conclusion about a subject. In abductive reasoning, a probable explanation is formed about a situation based on the known data. [2]

There are various methods of reasoning that belong into one or more of the categories we described above. Symbolic reasoning systems are used for deductive and inductive reasoning applications. Symbolic reasoning applications use propositional logic and first-order logic. Probabilistic reasoning systems use probability and symbolic logic. Probabilistic reasoning can be used to tackle problems that symbolic reasoning can't, such as managing data with a degree of uncertainty. Probabilistic reasoning can be used to solve diagnosis and prediction problems. For problems that symbolic and probabilistic methods can't give results, neural reasoning methods have been proposed. Neural-Symbolic reasoning methods use vectors to represent data. Neural-evidence reasoning methods use data from the environment for delivering results, and can work with both structured and unstructured data. [52]

Another method of AI reasoning has been developed, in order to tackle the issues that are presented when dealing with natural language. It is called Logical Reasoning over Natural Language (LRNL). This is a process of continuous iterative reasoning that begins from a knowledge base and builds successively from it to new conclusions until a final conclusion is reached. [2]

2.4.3 Explainable Artificial Intelligence

After getting the results from the analysis of data, sometimes it is better to have also an explanation of how the results were reached. This happens in cases where the user would like to make certain that the results indicate to a correct course of action, i.e. solutions to mechanical failures or treatments to illnesses. This explanation will be available to the user alongside the results. Explainable AI can be used for any type of data that has been analyzed, that is text, audio, images or video. It is fair to mention that as AI solutions are developed for most sectors of everyday life, the need for AI explainability increases. For AI systems, there is the need to be lawful, to be accepted by the public and to not create unnecessary fear. There is also the need for accountability, even for AI systems.

There are different techniques for AI explainability. One technique is the assignment of weights to certain language features, depending on their influence to the analysis results. Another technique is example driven explainability, when we need to demonstrate the reasons for the outcome of the analysis by citing examples that justify this outcome. Another is called provenance, and it has to do with showing the justification of the analysis results by following logical

reasoning steps to the final conclusion. [4] The scope of explainability may also vary. It may affect a portion of the data or the whole dataset. The depth of analysis of the results could be more detailed in addition to information about the operation of the system. [39]

There are also different ways of depicting the explanation in order to be easier for a user to understand. Saliency maps can be used to that effect. These are a category of representations that are created in such a way that the most important part of the representation or the data is noticeable first. This characteristic makes it a popular choice in many cases. Raw declarative representations in a form of some kind of representation such as trees is another type of depiction used. Lastly, natural language explanation can be used. This is another technique that uses natural language to aid in the depiction of the explanation.[4] Knowledge graphs can be also used for explainable AI visualization in a way that does not just visualize but also helps us understand more about the relations in data and create new subgraphs that can point to these relations. [39] [42]

2.5 Problems of sentiment analysis

As it has already been stated above, there are problems with sentiment analysis which have to do with expressions of every day life and the use of sarcasm. We also have to be very careful with the sources of data. We have to evaluate the prestige and even handed approach that specific platforms use. For an overall approach, well balanced results ought to be used. We also have to take into account that people, seldom or frequently, make errors and that a system that performs sentiment analysis will have issues recognizing the errors in written word. We also have to remember that for really global sentiment analysis concerning public sentiment for certain issues, problems or products other languages must be used apart from english and there are not developed resources such as libraries and lexicons to assist the task of sentiment analysis for every language. Video or sound segments in multimedia platforms can also convey sentiments to the users that view them, but their analysis must take into account collections and datasets of sounds and images and different applications of machine learning methods. Lastly, in order to get more accurate results, bigger volumes of data have to be analyzed, and if using machine learning methods, a larger volume of training data has to be used for those models. This increases the need for computer resources for an accurate analysis, but in systems where analysis time should take a few seconds at most, they put a big strain in resources, i.e. an application in a smartphone.

2.6 BERT

Bidirectional Encoder Representations for Transformers, or BERT, is a deep learning model created by Google that has been designed for Natural Language

Processing (NLP). The BERT model has been used for sentiment analysis, question answering and text summarization.

The operation of BERT is based on transformer technology and fine-tuning the parameters for pre-training. The parameters are fine-tuned after an initial pre-training, by implementing small tasks. It should be noted that a small amount of resources are used for fine-tuning BERT in comparison to pre-training.

A key feature of BERT is bidirectional operation. Traditional models can only process words in one direction, from left to right or from right to left. BERT can look words bidirectionally, that is both before and after a word and thus is an even more powerful tool for understanding text because it can understand the meaning of a word taking into account using its surrounding words.

However, bidirectional operation comes with a few disadvantages. It creates a problem during pre-training because standard conditional language models can only be trained in one direction and not bidirectionally. If this wasn't the case, the model would then predict the target word because it would have already identified it. The solution to this problem is using Masked Language Modeling (MLM) during pre-training. In MLM, we mask a percentage of the input tokens at random and then have the model make predictions for those masked tokens based on the words surrounding them, which are unmasked.

During pre-training another task that BERT is trained in, is Next Sentence Prediction (NSP). NSP is used in order to enable BERT to understand the relationship between two sentences. In effect, two sentences are chosen, where one follows another at 50% of the cases and for the rest, another random sentence follows the other and then the training takes place. BERT also undergoes fine-tuning, but this is an inexpensive process as BERT already relies heavily on the self attention module in the transformer architecture for modeling. While fine-tuning BERT we start the operation of BERT with specific inputs and outputs and fine-tune the parameters from start to finish. [61] [62]

2.7 Chat GPT

The innovations and the development in AI systems have included new inroads in conversational AI. The application of new technologies and the combination of techniques to create new conversational AI systems is one of the most widely known applications of AI in everyday life to public.

Conversational AI started with simple chat-bots that were operating with simple rules. These chat-bots were capable of answering simple questions but they couldn't give complex answers. There were gradual developments in order to make more complex systems that were capable of giving more complex answers that wouldn't necessarily be all predefined by rules. Nowadays, the most famous conversational AI model is Chat GPT.

Chat GPT (Generative Pre-trained Transformer) is a Large Language Model (LLM) that can create large original texts for users, based on their queries or questions. Large Language Models (LLMs) are deep learning models that are

trained in two levels and fall into the wider category of pre-trained language models. In the first level LLMs are trained on large volumes of texts and in the second they are trained on and adapted to the user's preferences. The first level is called pretraining and the second level is called meta-training. The good results of LLMs in natural language processing is noteworthy.

Chat GPT has become a widely popular platform that can answer questions about most aspects of everyday life and more. The large scope of topics it can describe or give information about, makes it even more popular. Another reason for the popularity of Chat GPT is the style of the responses it gives. These responses can constitute of large parts of text written in such a way that resembles the way a human would respond. The unique way of the operation of Chat GPT that has led to such heights of popularity among the general public are based on the application of new technologies. The fields in which Chat GPT has been used so far, are customer service, education, research, healthcare, content generation and gaming. Chat GPT can generate original text and answer questions the users in each service may have. It can also deal with multiple requests at very little time. While Chat GPT is dependant on the already available material for each topic it can query to return an answer, it is still a very valuable tool that can assist professionals and simple users. The originality and the human-like nature of its responses make it an even better assistant.

Chat GPT is pre-trained. When a user uses Chat GPT, the model is already trained with data from the internet. In this way, the model has already recognized patterns and information that can help it generate a response that will comprise of coherent text. The generated text will look like it has been written by a human and not by a machine. In general, the response of Chat GPT is written in the style of the input. Exactly because Chat GPT is pre-trained the output can be formatted or styled in a user's requested style. In effect, the output of Chat GPT can be a text formatted as a formal document or a text written in an informal manner, or in another manner depending on the requests made by the users.

Transformers are a deep learning architecture that is encoder-decoder based. This technology is used for the processing of data in a new way from Chat GPT. Especially in the case of Chat GPT, Chat GPT is dependant on a decoder architecture and not on encoder-decoder architecture. The reason for this is that Chat GPT is essentially a tool for creating new text and not a tool for crating corresponding relations or mapping input to output. Thus, the encoder part is abandoned and the input is used to create output based on the input and other generated output.

The encoder-decoder architecture is not the only feature used for the creation and operation of transformers in Chat GPT. Other features are used in combination with encoder-decoder architecture. These features are a self attention mechanism and positional embedding. Self attention mechanism is used to determine the particular weight and influence of one token over the the amount of information from all the tokens. The basis of the calculation for self attention mechanisms is the calculation of the dot product of vectors of inputs. The

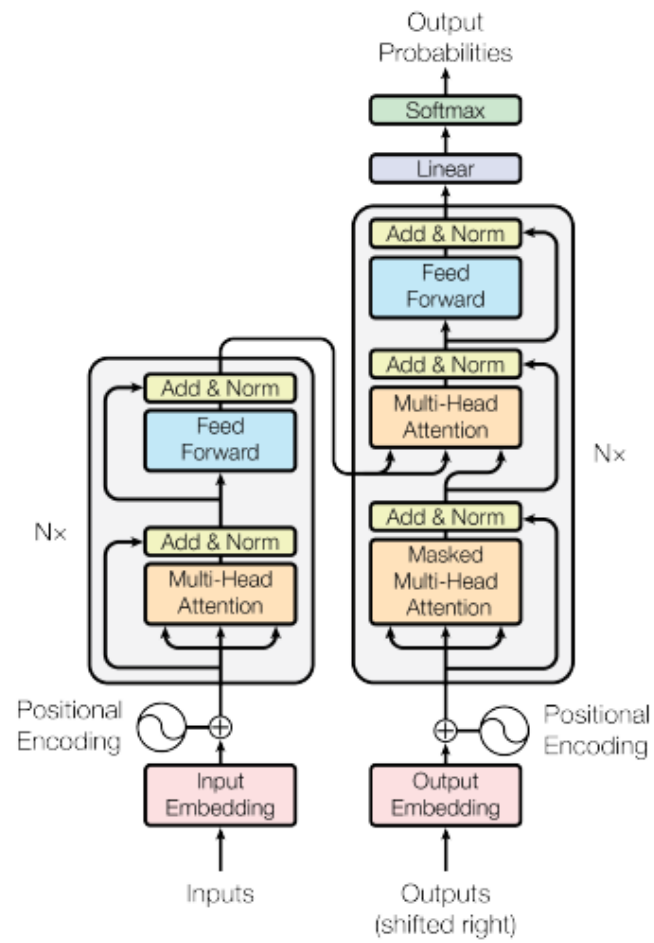


Figure 2.3: The Transformer model architecture. [63]

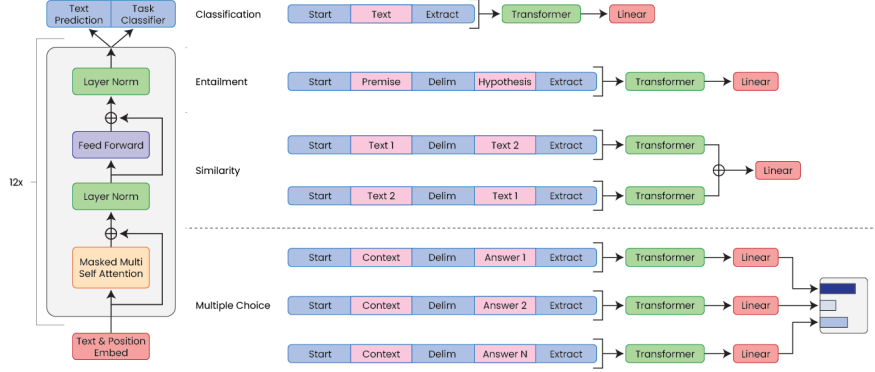


Figure 2.4: The Transformers architecture as defined in Chat GPT. [60]

higher the value of the dot product, the higher the semantic relation between input vectors and their effect on the output of the self attention mechanism. Positional embedding is also important because it helps taking into account the position of each token in a sequence and the importance of its influence in relation to a same token in another position. Positional encoding is added to the input tokens and is represented by sinusoidal functions.

The self attention module is described by the following mathematical type:

$$Attention(K, Q, V) = softmax(\frac{QK^T}{\sqrt{D_k}})V = AV$$

[60]

where:

Q is a vector that represents a single token in the word sequence,

K is a vector that represents all other tokens in the sequence apart from query token,

V is a vector that represents all token in the input sequence

D_k denotes the dimensions of K

A is called the attention matrix

If this dot product of self attention is high, then there is high semantic similarity between the query token and the other tokens in the input sequence.

Another important feature is multi-head attention, i.e. the parallel processing of input in contrast to a single self attention module. Multi-head attention for Chat GPT enables the faster training and the quicker identification of relationships in the input sequence.

Multihead attention can be described by the following mathematical type:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O$$

[60]

where:

$head_i = Attention(QW_i^Q, KW_i^K, WV_i^V),$
 and $W_i^Q \in R^{d_{model} \times d_k}, W_i^K \in R^{d_{model} \times d_k}, WV_i^V \in R^{d_{model} \times d_v}, W^O \in R^{hd_v \times d_{model}},$
 and W^O is the output projection matrix.

[59] [60]

Chapter 3

Proposed method

3.1 Phrase similarity measure example

At this point, we will demonstrate how the comparison and the measuring of similarity takes place. Supposing we have the following words:

dog
cat
life

For a similarity method, the calculation of similarity happens between synsets. Because we don't know the size of a synset for each word, we take the first synset for our calculations, which is the primary concept of the word.

In order to obtain the value of similarity between two synsets, the mathematical type we use is:

similarity_value = *similarity_measure*(*synsets*(*word1*)[0], *synsets*(*word2*)[0]))

For the four similarity measures we can run, the calculations for the synsets of the words above, are:

- Path similarity: (dog,cat) = 0.2 (dog,life) = 0.05 (cat,life) = 0.07
- Wu-Palmer similarity: (dog,cat) = 0.85 (dog,life) = 0.1 (cat,life) = 0.13
- Leacock-Chodorow similarity: (dog,cat) = 2.03 (dog,life) = 0.69 (cat,life) = 1.0
- Jiang-Conrath similarity: (dog,cat) = 0.45 (dog,life) = 0.06 (cat,life) = 0.06

Afterwards we can choose the greatest value among the similarity results for each similarity measure. For the words we have made the measurements, the greatest value is 2.03 according to Leacock-Chodorow similarity measure results. We have to remember, though, that the range of values for Leacock-Chodorow similarity measure ranges from 0 to 3.689, placing this similarity measure in a

unique position over the other similarity measures that have values that range from 0 to 1. In order to be more just we can compare values between the results of each similarity measure. In this example and in every similarity measure, the greatest value is the similarity result between dog and cat synsets. In effect, they are both animals and pets.

In order to measure the similarity of each word in a phrase compared to a word, we can measure the similarity for each word of the phrase. The mathematical type we use is:

for word_i belonging to phrase :

similarity_value = similarity_measure(synsets(word_i)[0], synsets(word)[0])

Supposing we have the following phrase:

pets are a good companionship

If we measure the similarity for each word of this phrase and the word *cat* we get the following results:

- Path similarity: 0.11 0.05 0.047 0.05 0.05
- Wu-Palmer similarity: 0.64 0.095 0.091 0.095 0.095
- Leacock-Chodorow similarity: 1.44 0.64 0.59 0.64 0.64
- Jiang-Conrath similarity: 0.09 0.06 1e-300 0.059 0.058

1e-300 is the value of Jiang-Conrath similarity when a synset is measured for distance from itself. If we pick a limit or a threshold for the similarity values that we want to take into account, and this is at 0.6, then the values will be:

- Path similarity:
- Wu-Palmer similarity: 0.64
- Leacock-Chodorow similarity: 1.44 0.64 0.64 0.64
- Jiang-Conrath similarity:

If these values take each from one vote (or point) we can see:

- Path similarity: 0
- Wu-Palmer similarity: 1
- Leacock-Chodorow similarity: 4
- Jiang-Conrath similarity: 0

As we can see, depending on the limit we use for each similarity measure, the number of votes can go higher or lower and this has direct effect on the classification of the phrase.

3.2 Software libraries and similarity measures

3.2.1 WordNet library

WordNet is a large dictionary collection of synonym sets in english and in other languages. The words in synonym sets (synsets) are grouped with regard to one concept. Based on the concept that a synset corresponds to, a set of relations such as synonyms, antonyms, or parts of a meaning. For example, synonyms to the word *freedom* can be found, or antonyms to the word *slavery*, or relations such as *orange (is-a) fruit*.

3.2.2 Path Similarity

Path similarity is the simplest measure of similarity between synsets, that takes into account the length of the shortest path between synsets. If p is the length of the shortest path between two synsets, s_i and s_j , then path similarity is given by the mathematical type:

$$path_sim(s_i, s_j) = \frac{1}{1+p}$$

[48] [49]

where p is the length of the shortest path between the two synsets.

The values that path similarity can take vary from 0.0 to 1.0, where 0.0 is the lowest one and 1.0 is the highest one (from the synset to itself).

3.2.3 Wu-Palmer similarity

From the lowest common subsumer (lowest common hypernym of two synsets) the length of the path to the root node is denoted as l . The length from the first synset, s_i , to the lowest common subsumer is denoted as i and from the second synset, s_j , is denoted as j . Wu-Palmer similarity is given by the mathematical type:

$$wu_palmer_sim(s_i, s_j) = \frac{2l}{i+j+2l}$$

[47]

where l is the length of the path from the lowest common subsumer to the root, i is the length of the path from the first synset to the lowest common subsumer and j is the length of the path from the second synset to the lowest common subsumer.

3.2.4 Leacock Chodorow similarity

The shortest path between two synsets, s_i and s_j , is denoted as p , the maximum taxonomy depth is denoted as d . The Leacock-Chodorow similarity is given by the mathematical type:

$$leacock_chodorow_sim(s_i, s_j) = -\log\left(\frac{p+1}{2d}\right) \quad [50]$$

where p is the length of the shortest path between the two synsets and d is the maximum taxonomy depth.

3.2.5 Li similarity

Li's measure is a strange addition to this list. It is actually empirically defined and its main concept is measuring similarity in a non-linear way, in a space of infinite information sources while word similarity is a finite quantity. The shortest path between two synsets, s_i and s_j , is denoted as p , the length of the path from the lowest common subsumer of the two synsets to the root is denoted as l and α and β are two parameters which are adapted manually. The Li similarity is given by the mathematical type:

$$li_sim(s_i, s_j) = e^{-\alpha * p} * \left(\frac{e^{\beta * l} - e^{-\beta * l}}{e^{\beta * l} + e^{-\beta * l}} \right) \quad [46]$$

where p is the length of the shortest path between the two synsets, l is the length of the path from the lowest common subsumer to the root, α and β are two parameters which are adapted manually.

3.2.6 Manjula similarity

Manjula's measure, or MSA, is a similarity measure that takes into account Wu-Palmer similarity measure. Manjula's measure is given by the mathematical type:

$$manjula_sim(s_i, s_j) = \frac{2Ne^{-\frac{L\lambda}{D}}}{R_i + R_j} \quad [53]$$

where L is the shortest path between two synsets, s_i and s_j , D is the depth of the ontology tree, R_i and R_j are the distances of the paths from s_i and s_j to the root, λ is 1 for synsets from a different hierarchy and 0 for synsets in the same hierarchy and N is the distance of the lowest common subsumer of the two synsets to the root.

3.2.7 Jaccard index and similarity

Jaccard index was proposed by Jaccard in 1901.[54] It is used as a measure of similarity or diversity between two given finite sets. It can also be used for measuring similarity between concepts. Jaccard index, or Jaccard similarity, is given by the following mathematical type:

$$jaccard_index(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad [55]$$

where A and B are the given finite sets, $|A \cap B|$ is the cardinality of the intersection of the two sets, i.e. the common elements of the two sets and $|A \cup B|$ is the cardinality of the union of the two sets, i.e. all the elements of the two sets.

3.2.8 Extended Jaccard similarity

A variation of Jaccard index is Extended Jaccard similarity, which uses only the common elements of two given sets for comparison of similarity and from them, only the ones with equal values. The reason for using this variation is that Jaccard index can be influenced by the number of elements in each set. The Extended Jaccard similarity is given by the following mathematical type:

$$extended_jaccard_sim(A, B) = \frac{\sum_{i \in A \cap B} x_i}{|A \cup B|} \quad [55]$$

$$\text{where } x_i = \begin{cases} 1, & \text{if } v_i^A = v_i^B \\ 0, & \text{otherwise} \end{cases} \quad \text{with } i \in A \cap B$$

where A and B are the given finite sets, v_i^A and v_i^B are the values of a member i of the two sets.

3.2.9 Weighted Jaccard similarity

Another variation of Jaccard index is Weighted Jaccard similarity, which takes into account the preferences of the user by using weights for the members of the sets. The Weighted Jaccard similarity is given by the following mathematical type:

$$weighted_jaccard_sim(A, B) = \frac{\sum_{i \in A \cap B} w_i}{\sum_{i \in A \cap B} w_j} \quad [55]$$

where A and B are the given finite sets, w_i is the weight of the i -th member of the intersection of the two sets and w_j is the weight of the j -th member of the union of the two sets.

3.2.10 Weighted Extended Jaccard similarity

Another variation of Jaccard index is Weighted Extended Jaccard similarity, which combines the function of Extended Jaccard similarity and Weighted Jaccard similarity. The Extended Weighted Jaccard similarity is given by the following mathematical type:

$$weighted_ext_jacc_sim(A, B) = \frac{\sum_{i \in A \cap B} \alpha \cdot w_i}{\sum_{i \in A \cap B} w_j} \quad [55]$$

$$\text{where } \alpha = \begin{cases} 1, & \text{if } v_i^A = v_i^B \\ 0, & \text{otherwise} \end{cases} \quad \text{with } i \in A \cap B$$

where α is a variable for checking the values of the members of the sets A and B.

3.2.11 Information Content

Information content is the amount of information that is included in a synset. It can be computed, either by taking into account a group of given iterable words or by loading a precomputed weights file. The basic idea here is that if two words have similar meanings, their common information content will be higher. The mathematical type that gives us the definition of information content, denoted as ic , in relation to the probability of a concept, denoted as $P(c)$, is:

$$ic = -\log(P(c)) \quad [35]$$

where ic is the information content and $P(c)$ is the probability of a concept.

The probability of a concept is the number of its frequency over a corpus. In the mathematical type that gives us the probability of concept c , N is the total number of words in the corpus.

$$P(c) = \frac{frequency(c)}{N} \quad [35]$$

where $P(c)$ is the probability of a concept, $frequency(c)$ its frequency over a corpus and N the total number of words in the corpus.

There are different similarity measures based on computations with the information content.

3.2.12 Resnik similarity

The calculation of Resnik similarity for two synsets is based on finding their common subsumer (or hypernym). The result of the calculation depends on the information content of the lowest common subsumer and more specifically, on the highest value of the information content. So, Resnik similarity value, for two synsets s_1 and s_2 , is given by:

$$resnik_sim(s_1, s_2) = ic(lcs(s_1, s_2)) \quad [43]$$

where ic is the information content, lcs is the least common subsumer, s_1 is the first synset and s_2 is the second synset.

3.2.13 Jiang Conrath similarity

The calculation of Jiang Conrath similarity, for two synsets s_1 and s_2 , uses the information content of each synset and the information content of their least common subsumer. Jiang Conrath similarity is given by the following mathematical type:

$$jiang_conrath_sim(s_1, s_2) = \frac{1}{ic(s_1) + ic(s_2) - 2 * ic(lcs(s_1, s_2))} \quad [44]$$

where ic is the information content, lcs is the least common subsumer, s_1 is the first synset and s_2 is the second synset.

3.2.14 Lin similarity

The calculation of Lin similarity, for two synsets s_1 and s_2 , uses the information content of each synset and the information content of their least common subsumer. Lin similarity is given by the following mathematical type:

$$lin_sim(s_1, s_2) = \frac{2 * ic(lcs(s_1, s_2))}{ic(s_1) + ic(s_2)} \quad [45]$$

where ic is the information content, lcs is the least common subsumer, s_1 is the first synset and s_2 is the second synset.

3.2.15 word2vec

Working with synsets to measure similarity is a popular approach to measuring similarity, but there are other approaches as well. One of these approaches is vectorizing words and calculating their relation in the vector space. One method of representing words as vectors is word2vec.



Figure 3.1: The stages of the operation of the developed application.

3.2.16 Cosine Measure similarity

For two vectors, v and u , cosine similarity is defined as the dot product of these vectors over the product of their norms. Cosine similarity is given by the following mathematical type:

$$\text{cosine_sim}(u, v) = \frac{v \cdot u}{||v|| * ||u||}$$

[35]

where v is a vector and u is another vector.

3.3 The function of the application

As we have stated in the introduction, one of the goals of this project is the programming and creation of an application prototype that can get posts from twitter, that can process the posts and that can give results useful to a potential customer, regarding the overall sentiment of the community towards one product and, in particular, towards the negative attributes of the product.

The stages of the operation of the application are:

- Input of data.
- Preprocessing of data.
- Analysis of posts with similarity measures.
- Storing and exporting the results of the analysis.

In the first stage, the input of data, the source of data is, of course, twitter. Posts can be downloaded directly from twitter, via a twitter API, using credentials of an approved developer account in twitter. In real time, that is the goal, in order for the user to get the most recent twitter posts and get an opinion based on these recent posts. For the implementation of this project, two different datasets were used. One was a dataset that we ourselves created in order to test the code. It consists of 45 mock comments about products from twitter posts, with a similar layout to twitter data after they have been extracted from twitter. In order to test how well (or how bad) our system would

do with real data, we looked for datasets of real tweets in the internet and we also run experiments with them.

In the second stage, the data is preprocessed. Each segment of the twitter post we want to use for the algorithm is extracted separately. The text (or comment, for our purposes) of the post is extracted and any punctuation marks are removed. The capital letters of the text are transformed into lower. Lastly, stop words such as no and parts of speech such as articles are removed. When the text of the post is ready, a search is performed in order to find the name of the product from a list of product names we have already compiled. This way we can find group the comments by product, i.e. group them based on the product they refer to. Moreover, the name of the twitter account that is the author of the post is extracted from the data. The name is not used in the application, but it is used to create a graph database to organize and visualize the relations in the dataset.

In the third stage, sentiment analysis takes place. In every comment, the words are taken separately and a set of synonyms, or synset in the wordnet library, is found for every word. Separate lists of words with positive or negative meaning have been compiled and their synsets are being used in the mathematical types in order to calculate the amount of cognitive similarity between the words of the comment and the words in the compiled lists. If no synsets exists, the algorithm simply moves on to the next word. If the amount of cognitive similarity is above a threshold we have set, a mark is stored for either positive or negative. The comment can be categorized as positive, neutral or negative based on the number of positive or negative marks. If positive marks are more than negative marks, the comment is classified as positive, i.e. of a positive sentiment. If negative marks are more than positive marks, the comment is classified as negative and if the number of positive marks is equal to the number of negative marks, the comment is classified as neutral. The value of the threshold ranges from 0 to 1, as similarity score does too. The measures for computing similarity have been discussed in the previous section and will be shown in the results section. It should be noted that for reasons of time constraints and in order to be able to handle the size of the lists of similarities that are produced, we take similarity values of the first member of the synset for every word. This affects the results, but not by much because this is the most frequently used word. It greatly reduces the complexity of handling and analyzing the results.

In the fourth stage, the result of the algorithm is compared to an already created list where we ourselves have categorized the comments as positive, neutral or negative. That is for the artificial dataset we created ourselves. The dataset we used has this categorization already compiled. So we can measure the accuracy of our results and see if they are reliable and can be used in a real-time application. In our implementation we opted for showing results only for path similarity.

After the code has produced results, a textbox runs that is the interface of the application and can accept input from the user. The operations of the textbox are:

- Enter. By clicking Enter, we can enter a search word, on we have typed or one from the list above.
- When the word is entered, a plot is produced with the results of the analysis for the word.
- Print Comments. By clicking the button Print Comments, all the comments are printed grouped by the category that they belong (positive, neutral negative).
- Print All. By clicking Print All, the prevalent negative characteristics for a product (or a search word we have typed in) are printed and all the comments are printed grouped by the category that they belong.
- Clear. By clicking Clear, we can delete everything in the textbox.
- Show Instructions. By clicking Show Instructions, a message with instructions about the operation of each of the buttons is shown in the textbox.

By typing in a search word, we can have the application produce results that are specific to our search word. The button Enter has to be pressed anyway though, even if we choose a search word from the list of products.

By getting the prevalent negative attributes for a product, inductive reasoning is used to make a hypothesis based on the comments of users. If there are enough negative comments for a product aimed at a specific attribute, such as *unsafe* or *breakable*, this will appear as a negative attribute of the product. The algorithm for getting the negative attributes of a product is the same with finding similarities and classifying comments according to the prevalent sentiment of the text of each comment. The difference is in the list of words that is used, that is aimed specifically at different attributes, i.e. for the attribute *weight*, there are specific words such as *heavy*. For every attribute a different iteration of the algorithm runs with a different list of words.

Also, a graph showing the categorization is shown in a browser window. In the graph the user can choose the bar that represents positive, neutral or negative results and view the comments that belong to that category. The user can click with the pointer of the mouse on a bar or hover over it. By creating a representation of the results in a bar plot and by giving the ability to the user of accessing the source data for the results that are presented to the user, we add the attribute of explainability to our system. The results that are shown and the implied suggestions to the user are explainable by returning to the original dataset and giving the user the ability to check if one or more comments have been misclassified and how the categorization results represent accurately the general sentiment in the dataset. The user can use the pointer to hover over a bar. A summarization of comments will appear depending on the category of the bar.

For the testing of the application we constructed a small dataset ourselves with comments not extracted from twitter, but that we ourselves wrote. The reason was speed and efficiency for the testing of the operation of the application.

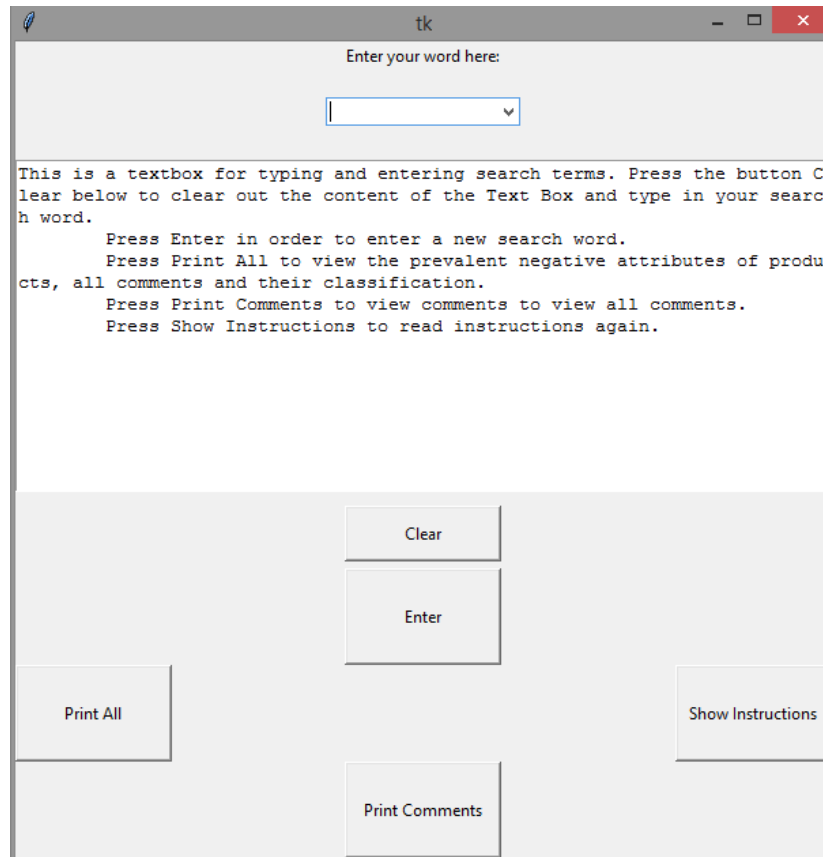


Figure 3.2: The Textbox that a user can see and use with all the buttons that are associated with the actions we are describing.

The accuracy of the similarity classification system was tested with a different dataset, extracted from twitter. For the testing of the application we have chosen one similarity measure, path similarity measure and have set similarity threshold at 0.7 because after a few experiments with different threshold values it was found that it was a value that gave the best results with the dataset we ourselves constructed. Below this value there was diminished accuracy in the classification of comments and above this value there wasn't much or any improvement in the accuracy and sometimes positive comments were classified as negative.

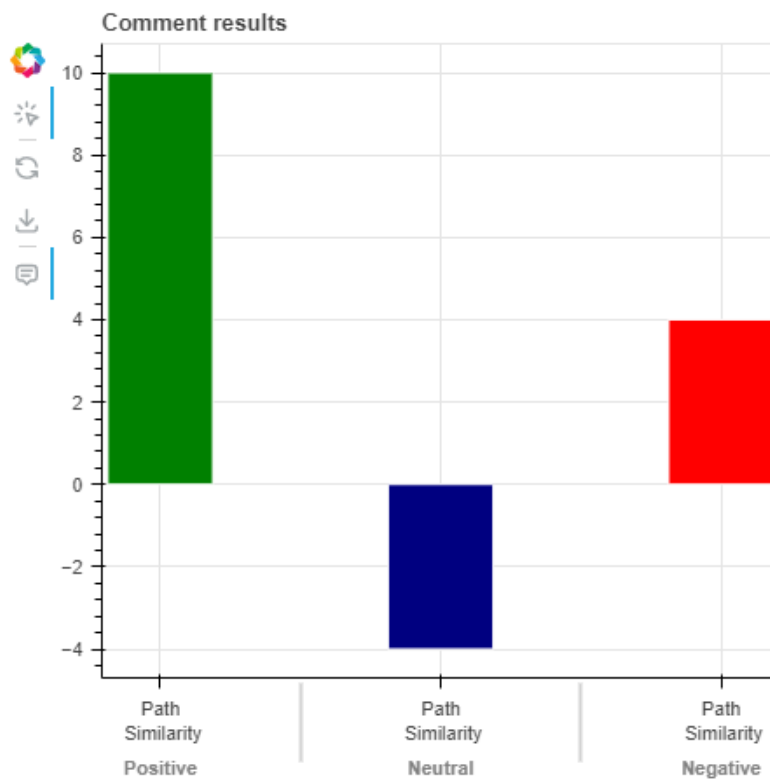


Figure 3.3: An example of the bar plot with the results from the sentiment classification that a user can see. Each bar has its own index and colour.

Chapter 4

Experimental Evaluation

The measure of success for an application can be thought not only its accuracy but also how easy it is to use and how appealing it can look to the user. Overall, the operation of this application is straightforward and at any point the user can press the button **Show Instructions**, in order to be reminded of the functions of the buttons. Other than that, it can be said that this simple textbox does not only show the function of the system but it also shows what can be achieved and in future, new implementations with more user-friendly layouts can be used.

4.1 Performance evaluation metrics

Every application and algorithm is useful if it produces accurate results, or at least close to accurate depending on the purpose of its use in real life. To measure the performance of a method, the following measures will be used:

- Precision.
- Recall.
- F1 (or NDCG).

Precision is the ratio of correctly positive classified results (true positive) to the overall sum of positive classified results, correctly and incorrectly (false positive).

$$precision = \frac{TP}{TP+FP}$$

[38]

where TP is the number of true positives and FP is the number of false positives.

Recall is also called sensitivity. It is the ratio of correctly positive classified results (true positive) to the sum of correctly positive classified results and falsely negative results (false negative).

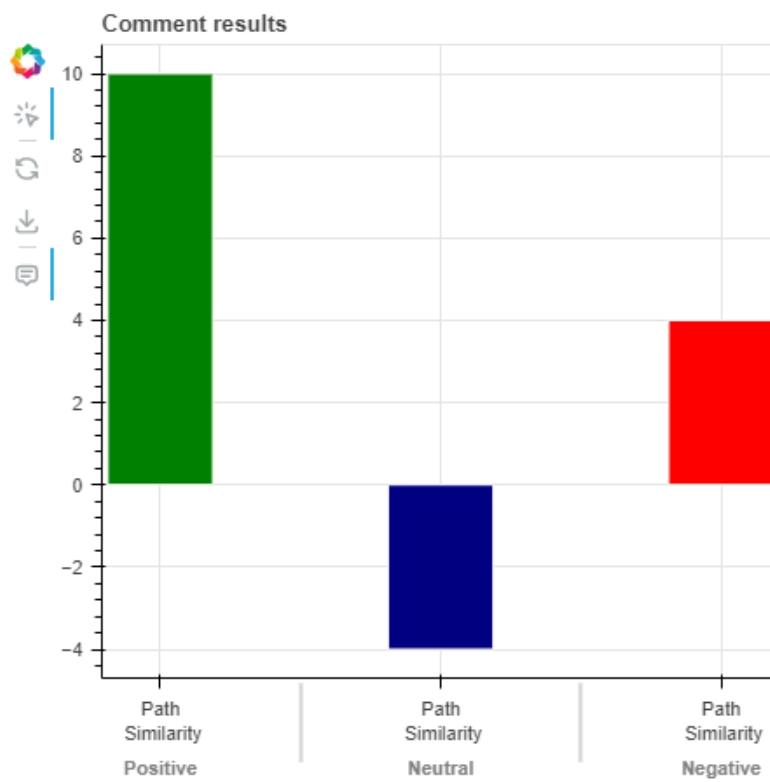


Figure 4.1: Results from running the test artificial dataset.

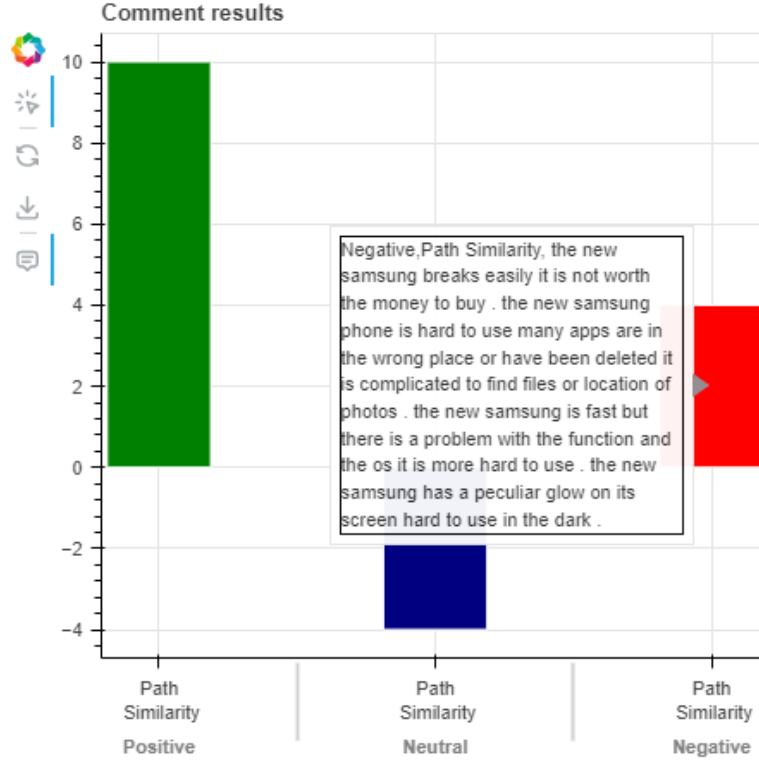


Figure 4.2: Hovering over the bar for the negative results.

$$recall = \frac{TP}{TP+FN}$$

[38]

where TP is the number of true positives and FN is the number of false negatives.

F1 or NDCG (Normalized Discounted Cumulative Gain) is the harmonic mean of recall and precision.

$$F1 = \frac{2*precision*recall}{precision+recall}$$

[38]

Accuracy is another measure. It is the ratio of the total correct classification to the sum of total classifications, correct and incorrect, positive and negative.

[38]

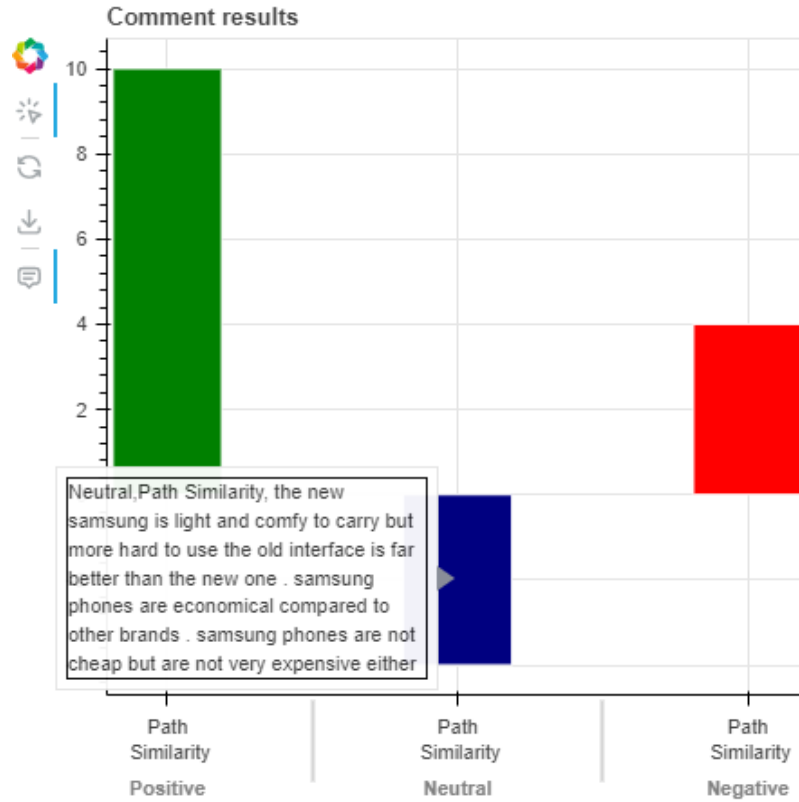


Figure 4.3: Hovering over the bar for the neutral results.

High score in both precision and recall means that the algorithm is returning high accuracy results. It should be noted that in these measures, there is the absence of neutral classification except for the measure of accuracy.

4.2 Datasets

At first, we ran experiments using the artificial dataset we had created. In this way, we would be able to see the performance of the system for all the different similarity measures. The artificial dataset comprises 45 comments which have been created by ourselves, of which 17 are positive, 22 are negative and 6 are neutral. As it has already been stated above, the reason for running the algorithm with this dataset was to test the operation of the algorithm and not to get any useful results about the operation of the system. The comments themselves were written specifically as comments for praising or criticising the attributes of certain products with clear, straightforward expressions, such as *good*, *bad*,



Figure 4.4: Hovering over the bar for the positive results.

costly, expensive or cheap. A few examples of the comments in this dataset are:

<i>The new ipad breaks easily. Don't drop it.</i>	Negative
<i>Samsung phones are economic compared to other brands.</i>	Positive
<i>The new samsung breaks easily. It is not worth the money to buy.</i>	Negative

The artificial dataset we created was used just for testing the function of the application. We have discussed above how and why it was created. Because it was created for testing purposes we couldn't be sure that similarity measuring with this dataset would yield useful results for evaluating similarity measures and how they could be used for other data. It was imperative that at least the algorithm for similarity classification to be tested with real data from twitter.

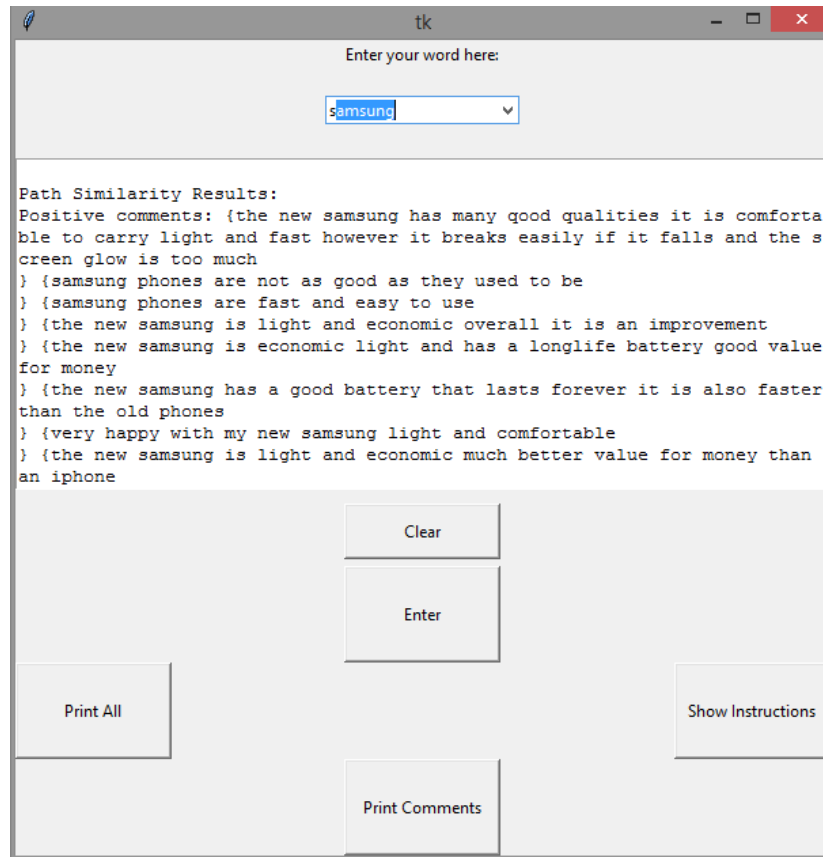


Figure 4.5: Having chosen a product from the list, the user can press the button **Print Comments** and the comments concerning this product will appear in the textbox, classified by sentiment.

The next logical step was to extract data from twitter. For reasons that we will discuss below we opted for the solution of using a twitter dataset with already classified posts. We found this dataset in the website <https://data.world>, the creator is the user crowdfunder, which is an account created by data.world. [33] The name of the dataset in data.world is Brands and Product Emotions and from now on we will call it crowdfunder dataset for clarity. The crowdfunder dataset we used has text from real posts that were extracted from twitter and is organized like a table in three columns. The columns are:

- The text of the post.
- The name of the product that is referred in the text of the post.
- The classification of the post.

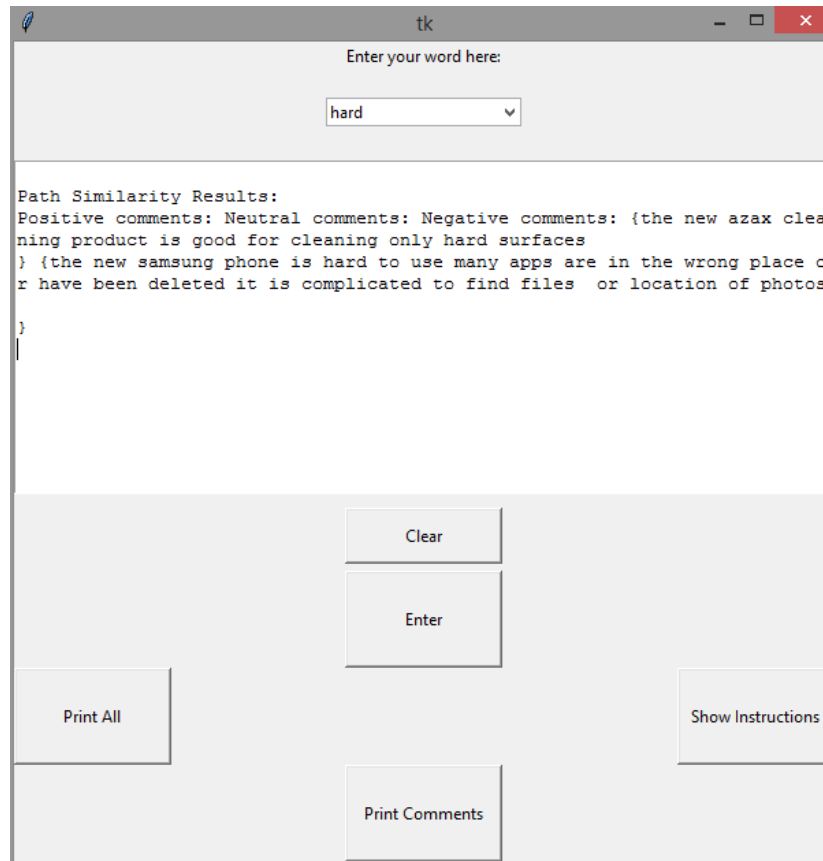


Figure 4.6: Having typed in and entered a word , the user can press the button **Print Comments** and the comments concerning this word, regardless of product name, will appear in the textbox, classified by sentiment.

The posts are classified as: 2978 positive emotion, 5545 unrelated or unclassified, 570 negative emotion. The total number of posts is 9093. Because we just wanted to test the classification system, we chose not to use the product name at these experiments, so we used only the text and the classification of each twitter post.

For reasons of testing our system with more datasets, in order to see if its performance would have significant differences depending on the nature of data, we started searching for datasets with similar structure, even if they weren't focused on product sentiment. We found this dataset in the website www.kaggle.com, the main uploader and the owner is M Yasser H, but there are also others who contribute and it is in the public domain. [34] The name of the dataset in Kaggle is Twitter Tweets Sentiment Dataset and from now on we will call it Yasser dataset for clarity. The Yasser dataset we used has text from real posts that

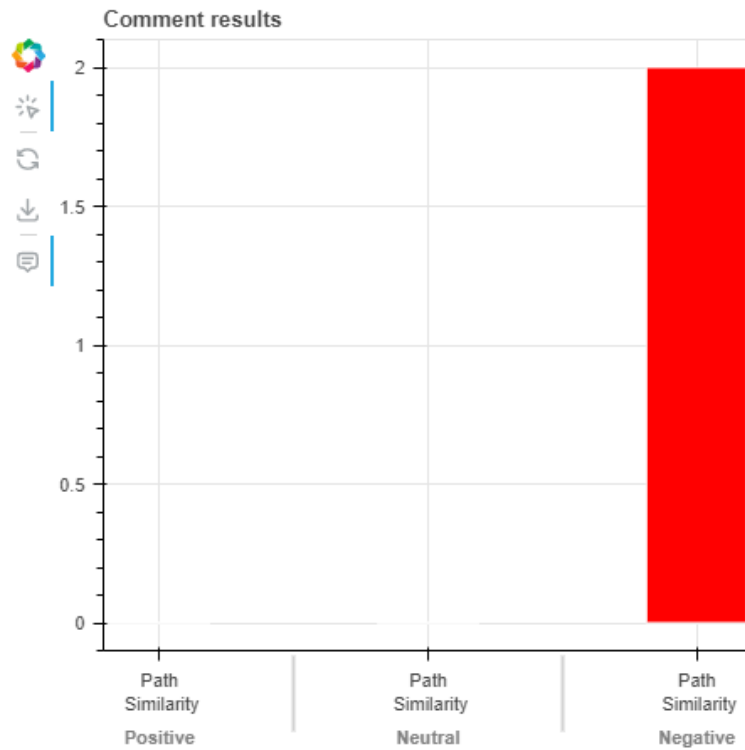


Figure 4.7: Results for the entered word **hard**.

were extracted from twitter and is organized like a table in four columns. The columns are:

- A unique id.
- The text of the tweet.
- Selected text from the tweet.
- The sentiment of the tweet.

The Yasser dataset comprises 27481 comments, 8582 of them positive, 7781 negative and 11118 neutral. The tweets are not limited to products but they can vary in topic, so we expected that there would be some expressions and words for sentiment that our system wouldn't detect as they probably were not included in the compiled lists of positive and negative words. It was expected that we would observe a difference in the performance metrics of the system, but before running it we assumed that the accuracy would drop noticeably if not significantly.

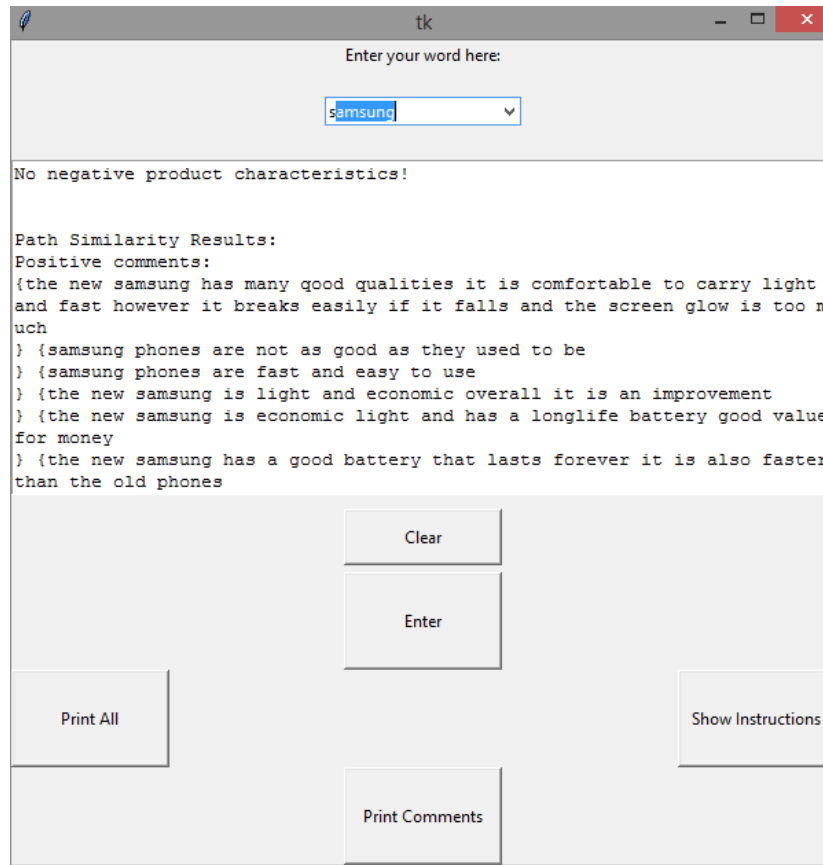


Figure 4.8: Having chosen a product from the list, the user can press the button **Print All** and the prevalent negative attributes of the product will appear in the textbox. Below them, the comments concerning this product will appear, classified by sentiment.

4.3 Experimental results

In the tables below the results from the experiments with different value for similarity threshold are shown. The thesis code can be downloaded from <https://drive.google.com/drive/folders/1jnlivUIHJ6cSVJDvNA8m32vV0?usp=sharing>

4.3.1 Artificial dataset

The artificial dataset comprises 45 comments which have been created by ourselves, from which 17 are positive, 22 are negative and 6 are neutral.

Threshold	Method	Precision	Recall	F1
0.7	Path Similarity	0.65	0.85	0.73
	Wu-Palmer	0.75	0.56	0.64
	Leacock-Chodorow	0.61	1.0	0.75
	Jiang-Conrath	0.65	0.85	0.73
1.0	Path Similarity	0.65	0.85	0.73
	Wu-Palmer	0.65	0.85	0.73
	Leacock-Chodorow	0.61	1.0	0.75
	Jiang-Conrath	0.65	0.85	0.73

Table 4.1: Performance metrics results for artificial dataset.

Performance metrics

4.3.2 crowdflower dataset

The crowdflower dataset has text from real posts that were extracted from twitter and is organized like a table in three columns. The posts are classified as: 2978 positive emotion, 5545 unrelated or unclassified, 570 negative emotion. The total number of posts is 9093.

Performance metrics

Threshold	Method	Precision	Recall	F1
0.3	Path Similarity	0.40	0.004	0.009
	Wu-Palmer	0.42	0.002	0.005
	Leacock-Chodorow	0.48	0.53	0.50
	Jiang-Conrath	0.48	0.53	0.50
0.5	Path Similarity	0.48	0.47	0.48
	Wu-Palmer	0.24	0.02	0.03
	Leacock-Chodorow	0.48	0.53	0.50
	Jiang-Conrath	0.48	0.53	0.50
0.7	Path Similarity	0.48	0.53	0.50
	Wu-Palmer	0.37	0.16	0.22
	Leacock-Chodorow	0.47	0.56	0.51
	Jiang-Conrath	0.48	0.53	0.50
1.0	Path Similarity	0.48	0.53	0.50
	Wu-Palmer	0.48	0.53	0.50
	Leacock-Chodorow	0.47	0.56	0.51
	Jiang-Conrath	0.48	0.53	0.50

Table 4.2: Performance metrics results for crowdflower dataset.

From what we can see in the results for the crowdflower dataset, for path and

Threshold	Method	Precision
0.3	Leacock-Chodorow	0.48
0.3	Jiang-Conrath	0.48
0.5	Path Similarity	0.48
1.0	Jiang-Conrath	0.48
0.5	Leacock-Chodorow	0.48
0.5	Jiang-Conrath	0.48
0.7	Path Similarity	0.48
0.7	Jiang-Conrath	0.48
1.0	Path Similarity	0.48
1.0	Wu-Palmer	0.48
1.0	Jiang-Conrath	0.48
0.7	Jiang-Conrath	0.48
1.0	Path Similarity	0.48
1.0	Wu-Palmer	0.48
1.0	Leacock-Chodorow	0.47
0.7	Leacock-Chodorow	0.47
1.0	Leacock-Chodorow	0.47
0.3	Path Similarity	0.40
0.3	Wu-Palmer	0.42
0.5	Wu-Palmer	0.24
0.7	Wu-Palmer	0.37

Table 4.3: Precision results for crowdflower dataset organized from best to worst.

Wu-Palmer similarity measures, when the similarity threshold is low, precision has a low value, of around 0.41, but recall has a very low value close to 0.0 which makes us think that for threshold values at 0.3 and 0.5, there are more false negatives that take advantage of the low threshold and appear at the final results. For path and Wu-Palmer measures, this is not the case as threshold value increases and they show better results for precision and recall at higher values of threshold. Leacock-Chodorow and Jiang-Conrath similarity measures show consistently for all values of similarity threshold around the same value for precision and recall. In fact, for Leacock-Chodorow, precision value is slightly better at lower threshold values and recall value is slightly better at higher threshold values. Jiang-Conrath similarity measure performance is the same throughout the experiments with different values of similarity threshold.

4.3.3 Yaaser dataset

In order to get more results and observe and analyze the behaviour of our system, in case the results were heavily influenced by the data of the specific dataset, we run another experiment with a different dataset. The Yaaser dataset, doesn't

Threshold	Method	Recall
1.0	Leacock-Chodorow	0.56
0.7	Leacock-Chodorow	0.56
0.3	Leacock-Chodorow	0.53
0.3	Jiang-Conrath	0.53
0.5	Leacock-Chodorow	0.53
0.5	Jiang-Conrath	0.53
0.7	Path Similarity	0.53
0.7	Jiang-Conrath	0.53
1.0	Path Similarity	0.53
1.0	Wu-Palmer	0.53
1.0	Jiang-Conrath	0.53
0.5	Path Similarity	0.47
0.7	Wu-Palmer	0.16
0.3	Path Similarity	0.004
0.3	Wu-Palmer	0.002
0.5	Wu-Palmer	0.02

Table 4.4: Recall results for crowdfower dataset organized from best to worst.

Threshold	Method	F1
0.7	Leacock-Chodorow	0.51
1.0	Leacock-Chodorow	0.51
0.3	Leacock-Chodorow	0.50
0.3	Jiang-Conrath	0.50
0.5	Leacock-Chodorow	0.50
0.5	Jiang-Conrath	0.50
0.7	Path Similarity	0.50
0.7	Jiang-Conrath	0.50
1.0	Path Similarity	0.50
1.0	Wu-Palmer	0.50
1.0	Jiang-Conrath	0.50
0.5	Path Similarity	0.48
0.7	Wu-Palmer	0.22
0.5	Wu-Palmer	0.03
0.3	Path Similarity	0.009
0.3	Wu-Palmer	0.005

Table 4.5: F1 results for crowdfower dataset organized from best to worst.

have text from posts that users just wrote to comment on products. The Yasser dataset comprises 27481 comments, 8582 of them positive, 7781 negative and 11118 neutral. The posts in this dataset were written to express sentiment

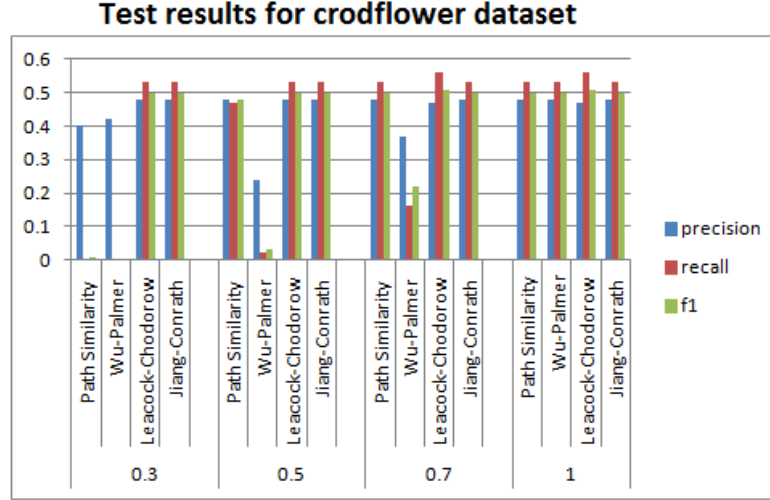


Figure 4.9: Test results from using crowdflower dataset for all threshold values.

in general, even to state if one is happy or unhappy for completely arbitrary reasons, or for reasons having to do with one's personal life, i.e. the birth of a child. The list of words we have created in our system for using in similarity measures computations have been specifically chosen for products and attributes of products, even though some more general words in the lists, such as good, great or bad are more generally used in everyday life. The reason for choosing this dataset over others was that the comments were already classified in the same way we are classifying results in our system, that is in three categories, positive, neutral and negative and so, with a similar classification system it would be easy to assess the performance of the similarity classification stage.

Because we started by analyzing the whole text of a post and we already have created a text preprocessing stage, at first we used only the text and the classification of each twitter post and not the data from the selected text column.

Performance metrics

From what we can see from these results, Leacock-Chodorow and Jiang-Conrath similarity measures have consistently better results in precision and recall values from Wu-Palmer except for when the value of similarity threshold is 1.0.

Threshold	Method	Precision	Recall	F1
0.3	Path Similarity	0.81	0.06	0.10
	Wu-Palmer	0.39	0.04	0.08
	Leacock-Chodorow	0.64	0.73	0.68
	Jiang-Conrath	0.63	0.73	0.68
0.5	Path Similarity	0.63	0.71	0.67
	Wu-Palmer	0.23	0.01	0.02
	Leacock-Chodorow	0.64	0.73	0.68
	Jiang-Conrath	0.64	0.73	0.68
0.7	Path Similarity	0.64	0.73	0.68
	Wu-Palmer	0.33	0.21	0.26
	Leacock-Chodorow	0.65	0.77	0.70
	Jiang-Conrath	0.64	0.73	0.68
1.0	Path Similarity	0.64	0.73	0.68
	Wu-Palmer	0.64	0.73	0.68
	Leacock-Chodorow	0.65	0.77	0.70
	Jiang-Conrath	0.64	0.73	0.68

Table 4.6: Performance metrics results for Yasser dataset.

Threshold	Method	Precision
0.3	Path Similarity	0.81
1.0	Leacock-Chodorow	0.65
0.7	Leacock-Chodorow	0.65
0.7	Jiang-Conrath	0.64
1.0	Path Similarity	0.64
1.0	Wu-Palmer	0.64
1.0	Jiang-Conrath	0.64
0.3	Leacock-Chodorow	0.64
0.5	Leacock-Chodorow	0.64
0.5	Jiang-Conrath	0.64
0.7	Path Similarity	0.64
0.3	Jiang-Conrath	0.63
0.5	Path Similarity	0.63
0.3	Wu-Palmer	0.39
0.7	Wu-Palmer	0.33
0.5	Wu-Palmer	0.23

Table 4.7: Precision results for Yasser dataset organized from best result to worst.

Path similarity has good results but not consistently. Overall, it seems that for lower values of similarity threshold it is more difficult to identify and clas-

Threshold	Method	Recall
1.0	Leacock-Chodorow	0.77
0.7	Leacock-Chodorow	0.77
0.3	Leacock-Chodorow	0.73
0.3	Jiang-Conrath	0.73
0.5	Leacock-Chodorow	0.73
0.5	Jiang-Conrath	0.73
0.7	Path Similarity	0.73
0.7	Jiang-Conrath	0.73
1.0	Path Similarity	0.73
1.0	Wu-Palmer	0.73
1.0	Jiang-Conrath	0.73
0.5	Path Similarity	0.71
0.7	Wu-Palmer	0.21
0.3	Path Similarity	0.06
0.3	Wu-Palmer	0.04
0.5	Wu-Palmer	0.01

Table 4.8: Recall results for Yasser dataset organized from best result to worst.

Threshold	Method	F1
1.0	Leacock-Chodorow	0.70
0.7	Leacock-Chodorow	0.70
0.7	Jiang-Conrath	0.68
1.0	Path Similarity	0.68
1.0	Wu-Palmer	0.68
1.0	Jiang-Conrath	0.68
0.3	Leacock-Chodorow	0.68
0.3	Jiang-Conrath	0.68
0.5	Leacock-Chodorow	0.68
0.5	Jiang-Conrath	0.68
0.7	Path Similarity	0.68
0.5	Path Similarity	0.67
0.7	Wu-Palmer	0.26
0.3	Path Similarity	0.10
0.3	Wu-Palmer	0.08
0.5	Wu-Palmer	0.02

Table 4.9: F1 results for Yasser dataset organized from best result to worst.

sify correctly negative sentiment posts for Path similarity and especially for Wu-Palmer measures.

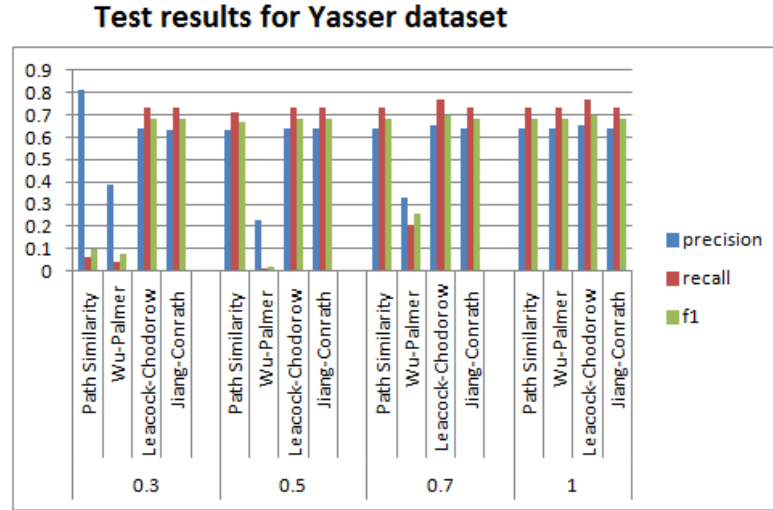


Figure 4.10: Test results from using Yasser dataset for all threshold values.

At this point, we should also mention that we performed the same experiments with the data in the selected text column. The results in performance measures were exactly the same. This makes us think that we used similar preprocessing techniques with the creators of the dataset, or at the very least that at the sentiment classification stage, our system gives the most importance to the words in the selected text column. The experiments with the data in the selected text column were carried out exactly because we wanted to see the performance of our system with already preprocessed data.

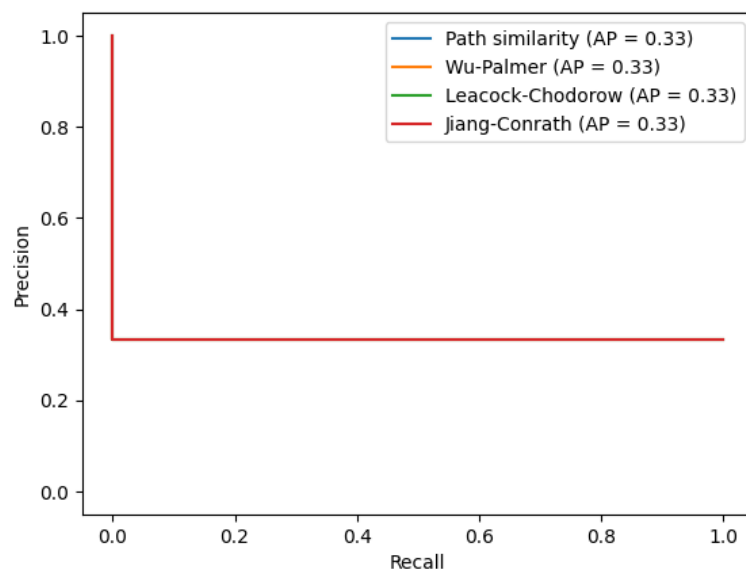


Figure 4.11: Precision-recall diagram micro-averaged from running the crowd-flower dataset with the threshold at 0.3.

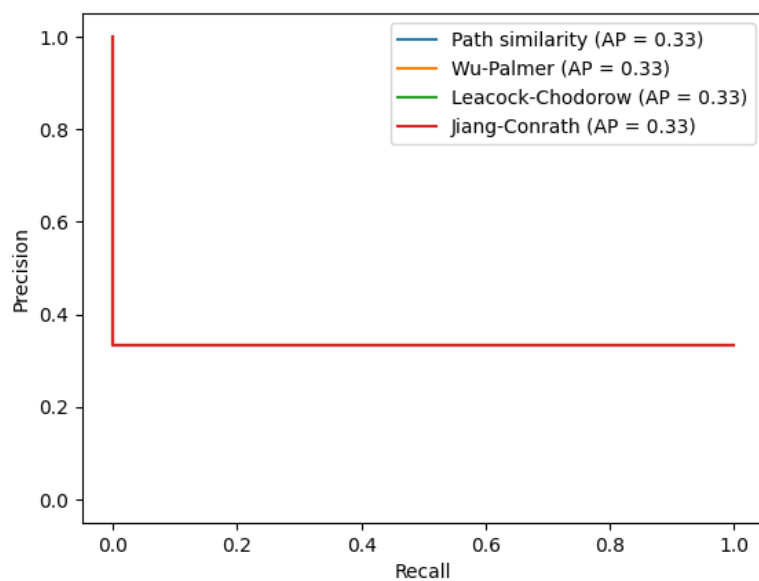


Figure 4.12: Precision-recall diagram micro-averaged from running the crowd-flower dataset with the threshold at 0.5.

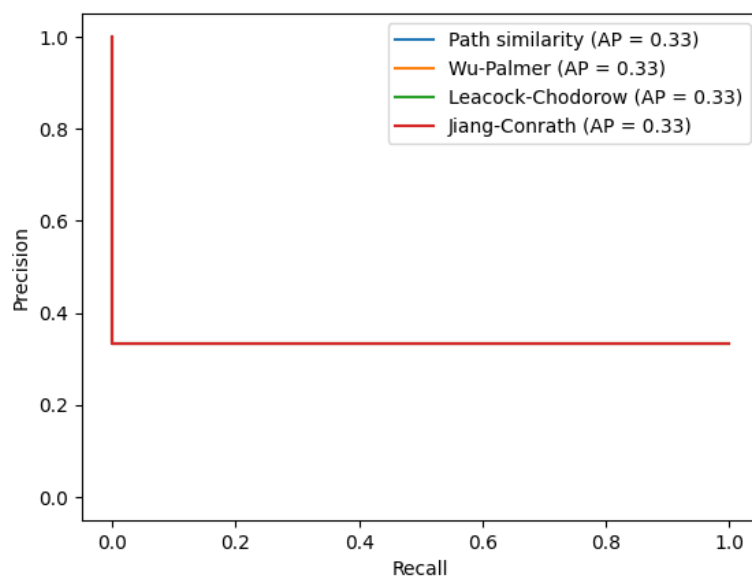


Figure 4.13: Precision-recall diagram micro-averaged from running the crowd-flower dataset with the threshold at 0.7.

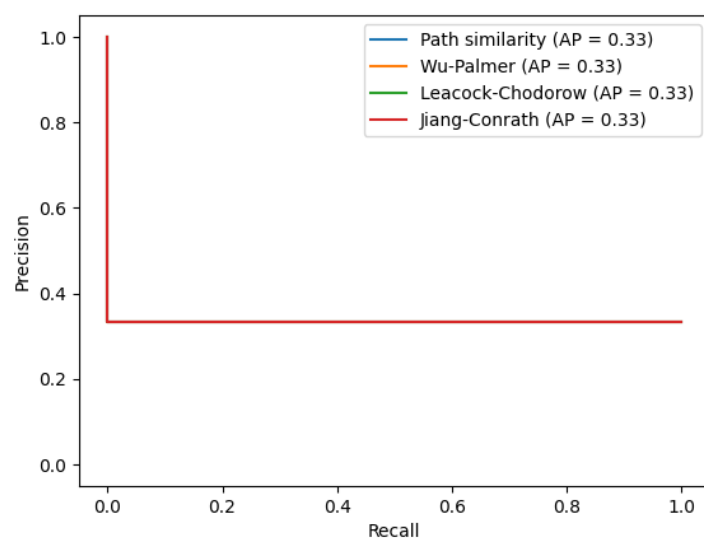


Figure 4.14: Precision-recall diagram micro-averaged from running the crowd-flower dataset with the threshold at 1.0 for Path, Wu-Palmer, Jiang-Conrath, Leacock-Chodorow similarity.

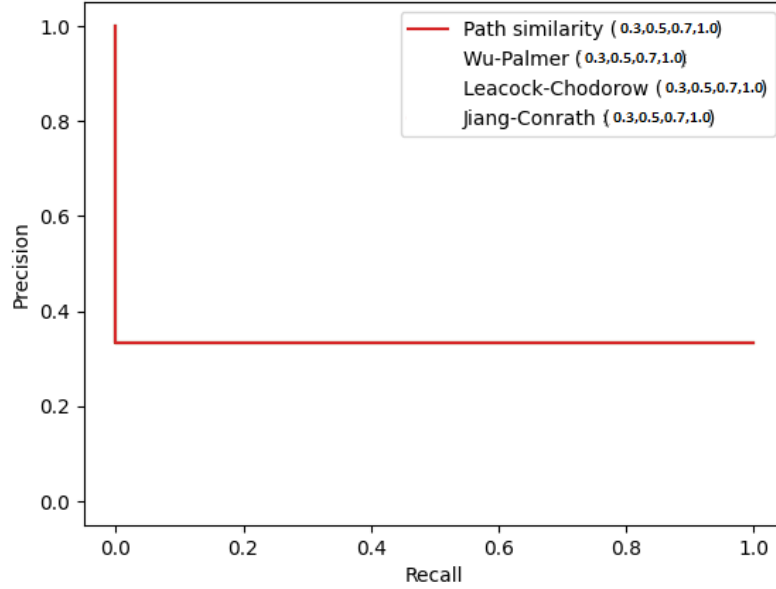


Figure 4.15: Precision-recall diagram micro-averaged for the crowdflower dataset, where the accumulated results for all threshold values (0.3,0.5,0.7,1.0) are shown.

4.4 Analysis of results

In general, for the two last datasets, the system performs almost the same. The results of similarity measures are similar and we can conclude with some certainty that Leacock-Chodorow and Jiang-Conrath similarity measures will be more useful in the operation of the application, due to the consistent higher scores of accuracy and other performance metrics for every similarity threshold value and the lower execution time for both datasets. In both datasets we can see that there was a difficulty to classify large number of comments correctly, even though for most cases in larger similarity threshold values the relationship of positive to negative comments, that is the fact that positive sentiment comments are more than negative is correct. The ratio of positive to negative comments is not representative of the true relationship in the original data in any case.

Of note is the bad performance of Wu-Palmer similarity measure in lower similarity threshold values. Wu-Palmer, Path and Leacock-Chodorow all take into account the length of the shortest path between the two synsets. However, for Leacock-Chodorow the calculation takes into account a logarithmic value and this reduces the calculation cost significantly. Also, for Wu-Palmer similarity measure the length of the path from the least common subsumer to the root is taken into account in the calculation which adds to the calculation cost.

Overall, the unexpectedly low performance can be explained by factors we

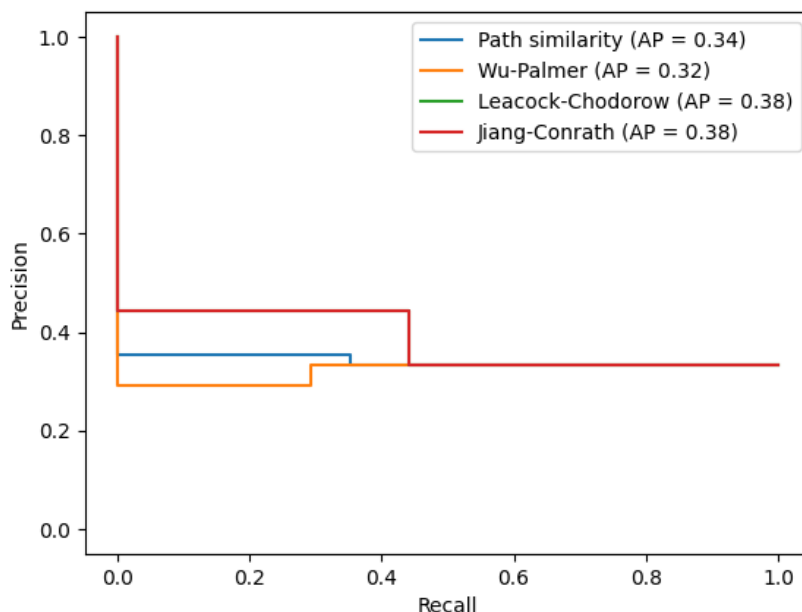


Figure 4.16: Precision-recall diagram micro-averaged from running the yasser dataset with the threshold at 0.3.

have not taken into account in the preprocessing stage. These include the provisions for slang and negation words. By the system we have created right now, a comment with the following text is more likely to be classified as negative:

The new ipad is not bad.

Apart from these cases, another problem may be caused by the pool of good (of positive sentiment) and bad (of negative sentiment) words for classification. We have created these lists with the aim to be as representative, or at least indicative, as possible. However, synset distance for some and lack of synsets in the library for abbreviations and slang may cause problems in the assessment of the sentiment for each text.

The vertical decline in all the precision-recall diagrams shows that the micro-averaged precision values are mostly stable. After a point the value of precision takes another decline at most diagrams and the value of recall increases. Overall, the system after a point starts to yield results with smaller accuracy. As time progresses and recall increases the ratio of true positive increases. It should be noted that the micro-averaged results seem similar for most similarity measures, depending on the similarity threshold value.

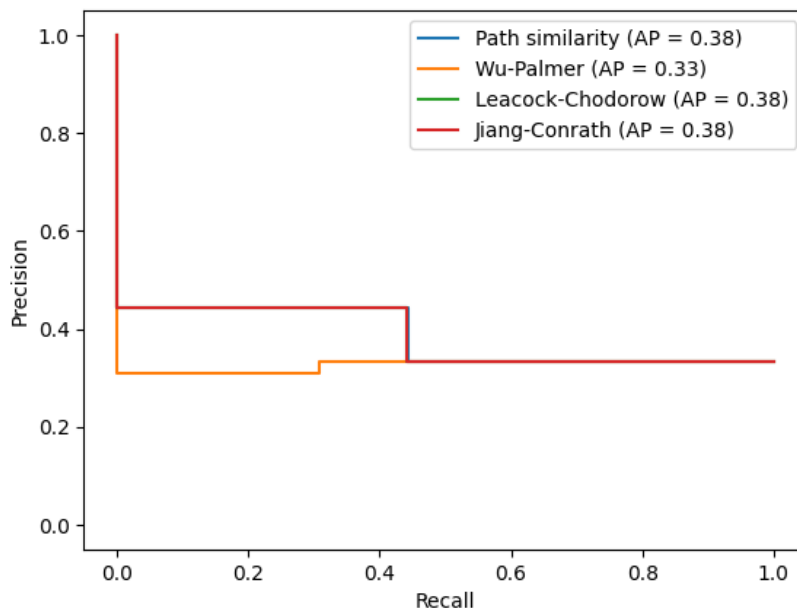


Figure 4.17: Precision-recall diagram micro-averaged from running the yasser dataset with the threshold at 0.5.

4.5 Comparison with related work

In most cases of related work that we examined, a greater effort has been made to make a more careful preprocessing of the data for analysis. Also, preprocessing techniques have been selected through trial and error among a larger pool of available methods. Through repetition of experiments with different parameters, the performance of the systems has been improved by taking into account of factors that reduce accuracy and performance measures at every stage.

The use of machine learning methods is another factor of the good performance metrics that are being shown in the results of related work. Analyzing data through multiple cycles of training and testing help increase the accuracy of the results and create models specifically designed for data of certain nature. The repeated training cycles also help taking into account certain language features such as special expressions, slang or technical terms, even though for the vast majority of twitter users it is more likely to write posts with slang than technical terms, although the latter is not impossible.

The use of machine learning methods is not limited to the sentiment classification stage. Many of these systems include machine learning methods in the preprocessing stage and reinforcement techniques so that the system can be aware of the impact of certain preprocessing options for text.

In the results section, we have to also note that our system is simple in its

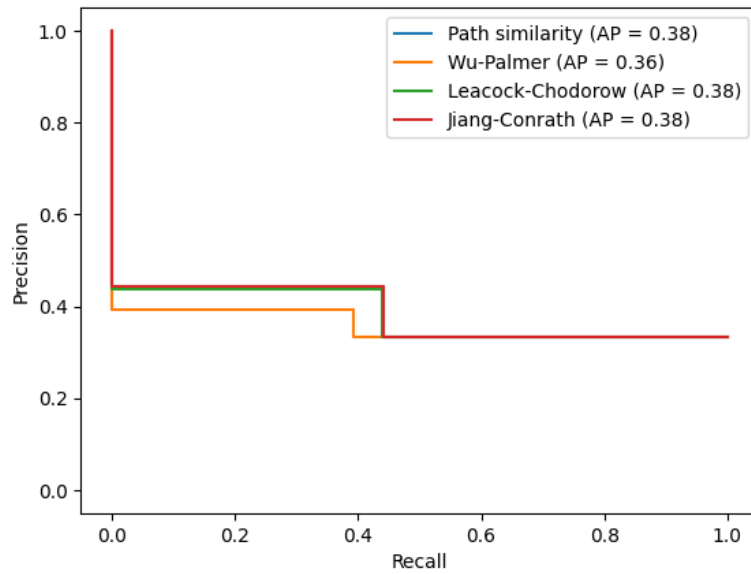


Figure 4.18: Precision-recall diagram micro-averaged from running the yasser dataset with the threshold at 0.7.

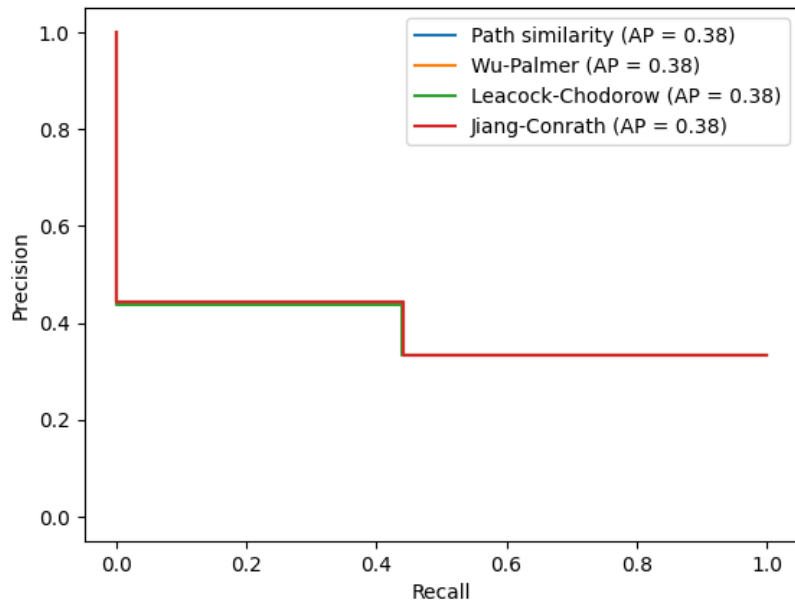


Figure 4.19: Precision-recall diagram micro-averaged from running the yasser dataset with the threshold at 1.0.

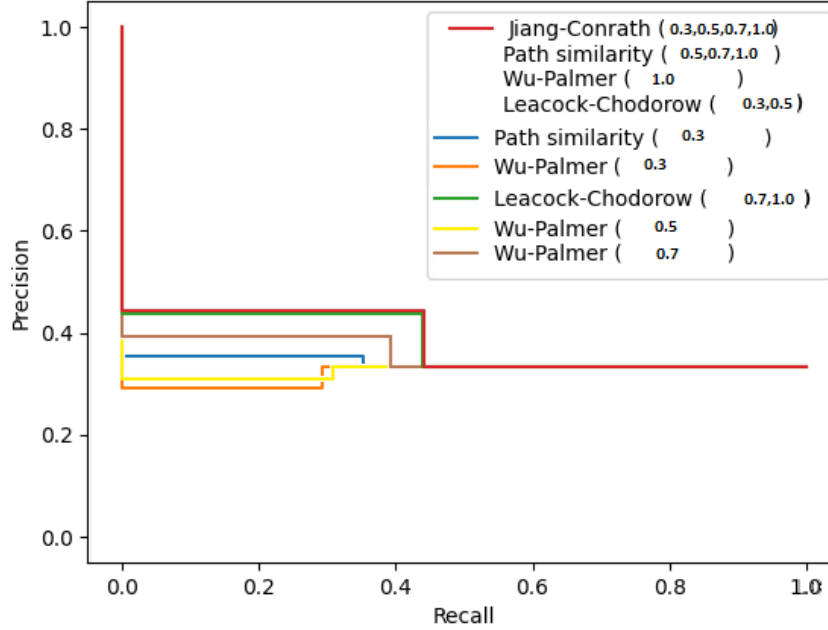


Figure 4.20: Precision-recall diagram micro-averaged for the Yasser dataset, where the accumulated results for all threshold values (0.3,0.5,0.7,1.0) are shown.

conception. This design choice includes the assessment of the results of the sentiment classification stage. In our system, every value of similarity above the similarity threshold takes one mark. In theory that should be accurate in the representation of the sentiment for each word. In practice, in lower similarity thresholds, various values of similarity have different values and they all count the same. This, not only seems to be wrong, but in view of our results, is proven to be wrong, but it doesn't affect every similarity measure in the same way. The most extreme case in our experiments is Wu-Palmer similarity measure. In higher similarity thresholds it normally shouldn't count as much because we take into consideration only the highest values. In comparison with other work that uses different marking process, or none at all and focuses on similarity values themselves, or gives special weight to certain expressions our system struggles to achieve similarly accurate results. This problem may be the main cause for the large difference in accuracy between the experiments we run with the two datasets, regarding the accuracy results in relation to the neutral classification. Many systems, though, do not take into account the existence of middle ground or neutral sentiment and focus on classifying text as positive or negative ignoring the neutral sentiment completely. If we ignore the neutral classification in our system, the results for higher similarity thresholds give an approximately good

representation of the relation between positive and negative sentiment in the posts but not an accurate image, especially taking into account the number of posts that is falsely categorized.

4.6 Problems with the implementation and the experiments

A few problems came up at the course of the implementation, problems that were unforeseen for people with experience in writing code but now with working with specific coding platforms. Our code was written in Python language 3.6.0 in the Jupyter Lab environment. Certain libraries in python had issue being installed or working correctly in the Jupyter Lab. One of those is matplotlib, that for static plots works correctly but for dynamic plots that we wanted to produce for the application does not. This, forced us to search for long until we found hvplot which worked correctly but had a quirk. In order for the hvplot library graph to be dynamic, we also had to write a few lines of code in xml, otherwise the system wouldn't be able to communicate with the graph and send data to it. Concerning the libraries used for text generation, it was impossible to find libraries that would operate without problems in Jupyter lab for the version of Jupyter lab and of Python we were using.

In order to ascertain the overall performance of our system, we wrote code to produce precision-recall diagrams. The main issues we faced with these diagrams was that there was need to take into account the fact that we had three categories instead of two. The results we got using the standard functions weren't representative of the performance of the system because the functions had been created for two categories. This led us to take into account all three categories and use an average for each pace (value of results) of the system. This may affect the way the diagrams are presented.

For the purposes of our project, we wanted to produce results with more similarity measures and with more libraries other than WordNet. In the case of similarity measures Resnik and Lin that use information content for calculating similarity there was a problem with adjectives. Apparently in some collections there are no entries for adjectives. We could have worked our way around it by disregarding adjectives, but adjectives make up almost the entirety of our lists of words with positive and negative sentiment. In effect we would have to run separate experiments for different similarity measures and this would produce a variety of results that we would have no way of comparing with results from other similarity measures. Also, adjectives are used in the text of posts too and this would lead to a peculiar behaviour from our system with these similarity measures. So, even though we had written code for this similarity measures as well, we decided to include them in the study of similarity measures but to disregard them from the implementation.

In the case of cosine similarity, because it had nothing to do with synsets and information content, but with vectorization of words and texts in order to

get a similarity value, a library that transforms sentences into vectors had to be installed and used. The most obvious choice was word2vec, but there was a problem with the installation. Certain files were unavailable for Windows operating system and the library could not be installed. As things stood, we decided to include cosine similarity in the study of similarity measures but to exclude it from the implementation.

The datasets we used were chosen because they had the same logic in the sentiment classification of posts, so we could measure the results of the system in comparison of the classification of the posts in the datasets. Other datasets had a different grading system where the system was shown with numbers, or where the text in the posts had to do with expressing opinion about different issues that required compiling of a different list of words for sentiment analysis. Our initial goal was to use data directly from twitter. In order to do this it was required to create a developer account in twitter in order to have credentials to use with twitter API. The application for getting a developer account was submitted but it was rejected. Furthermore, it took a long time to receive a response. All this happened in the middle of the takeover of twitter by Elon Musk and the laying off of thousands of employees. In this problematic situation when everything was being reevaluated and user dissatisfaction was rising it had to be expected that there would be problems with new developer status applications. As time had already gone by and we had already developed a small dataset of our own to be able to test the functionality of the code, it was decided that we would go on with running experiments with data from twitter datasets and would not try once more to apply for developer status twitter account which, even if accepted, there was a real risk to get a response to very late for the development of this system. It was a problem unrelated with coding but very important for everyday applications and the people who develop them. We should keep in our minds that the market does not always operate in the way we would like it and that not only technological developments, but also financial or social developments may render our work or a part of it obsolete and irrelevant.

Chapter 5

Conclusions

5.1 System assessment

We have created a system that makes sentiment classification using similarity measures. It is a system that can use data from twitter and return an assessment of a product based on the prevalent sentiment in the posts it analyzed. At the moment the accuracy of the results shows that there is room for improving in the preprocessing and the sentiment classification stages. The system appears to be able to give representative results of the data it gets as input but still not an accurate depiction of the sentiment distribution. The issue with the neutral sentiment classification may have had a greater impact than what we thought initially. Still, we have managed to demonstrate the function of such a system and of the simple application we developed. Also, we have managed to produce comparably good results with other work in the field, especially if we take into account that we haven't implemented a machine learning method in our system. We have shown that for an effective application in real time there are similarity measures (Leacock-Chodorow and Jiang-Conrath) that can return reliably accurate results in a short amount of time even when there is a large volume of data to process. Depending on what a user asks from the application, if the number of comments that is included is the most recent ones, the results should be produced in a very short amount of time, perhaps shorter than 142 seconds that is the shorter amount of time that Jiang-Conrath similarity measure run successfully. Overall, we have shown that it is possible to create such an application with specific characteristics. Also, it is possible to get useful results that can influence the choice of a user to purchase or not to purchase a device or a program or a service, even if they are not 100% accurate. The need for fast information in this case is of greater importance. Besides, we have to remember that this is an opinion mining tool and people's opinions vary and may even change in time.

5.2 Future research and ideas for improvements

Seeing how the system performs in the experiments we run, we can point to some improvements.

One basic improvement is the use of machine learning methods in order to improve the performance of our system. Seeing how the accuracy of other work can be as high as 0.96, or 96%, and ours is at around 0.60 to 0.62 depending on the dataset, there is certainly room for improvement. Machine learning methods can be used in the sentiment classification stage but also in the text preprocessing stage. Any computational costs and delays in the function of the system will have to be dealt with in some way. We have to remember that a user needn't run an application from the start and ameliorate its results by using it. This can be done in the development stage without having to burden the users' devices.

In the preprocessing stage we have chosen a simple preprocessing procedure with eliminating parts of speech such as articles for the sake of time and also eliminating stop words from the analysis. We have not delved into handling expression or phrases but only words. We haven't dealt with expressions, sarcasm, slang or technical terminology. We have not used a system where certain words have a greater gravity than others in preprocessing but only in the sentiment classification stage when that gravity appears in the form of similarity value. From what we have seen from previous work there is a large body of work dedicated to handling everyday expressions and slang. We could also handle negations to avoid situations when comments like not good, not bad and not terrible can give confusing and false results.

In the sentiment classification stage, maybe the use of a special list with words with neutral sentiment can give more accurate results and reduce the difference in the accuracy results between taking into account the neutral results and not taking them into account.

Running experiments with other similarity measures and using different libraries from WordNet can yield different results. In these experiments time measuring is of the essence if we want our results to mean something for everyday users. If a very accurate method produces results at 500 seconds or 600 seconds and occupies the resources of the smartphone or tablet until then, it is practically unusable. Even if the accuracy of the results reaches 99%, it will do no favours for the popularity of such a slow application. So, we have to remember that we are aiming for an accurate application but we can sacrifice a bit of the accuracy because of time constraints.

Similarity results are used for sentiment classification. In our system we give one mark for every similarity value that exceeds a similarity threshold we ourselves have set. This isn't a perfect approach and may not be an optimal one. More research is needed in this procedure and alternate methods may be developed with the help of machine learning techniques.

Regarding the functionality of the application, another interface may be more commercial and user friendly. The one we have developed gets the job done and is easy to use but people, especially young people, don't choose applications

just because they are easy to use. The plots that the application produces open in the default browser of the user's device. So, have to remember that two programs are needed for the application to function properly, the application and the browser.

For a user, the appearance of the results could also be very important. The results in the graph could be shown in different ways. If a user wished it, a combination of similarity measures could be used or if a user would find it useful separate bars in the graph of results corresponding to the different similarity measures.

Bibliography

- [1] Tan, Kian Long, Chin Poo Lee, and Kian Ming Lim. "A survey of sentiment analysis: Approaches, datasets, and future research." *Applied Sciences* 13.7 (2023): 4550.
- [2] Yang, Zonglin, Xinya Du, Rui Mao, Jinjie Ni, and Erik Cambria. "Logical Reasoning over Natural Language as Knowledge Representation: A Survey." *arXiv preprint arXiv:2303.12023* (2023).
- [3] Gozalo-Brizuela, Roberto, and Eduardo C. Garrido-Merchán. "A survey of Generative AI Applications." *arXiv preprint arXiv:2306.02781* (2023).
- [4] Gohel, Prashant, Priyanka Singh, and Manoranjan Mohanty. "Explainable AI: current status and future directions." *arXiv preprint arXiv:2107.07045* (2021).
- [5] Alsaeedi, Abdullah, and Mohammad Zubair Khan. "A study on sentiment analysis techniques of Twitter data." *International Journal of Advanced Computer Science and Applications* 10.2 (2019): 361-374.
- [6] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Computational linguistics*, vol. 37, no. 2, pp. 267-307, 2011.
- [7] A. Harb, M. Plantié, G. Dray, M. Roche, F. Trouset, and P. Poncelet, "Web Opinion Mining: How to extract opinions from blogs?," in *Proceedings of the 5th international conference on Soft computing as trans-disciplinary science and technology*, 2008: ACM, pp. 211-217.
- [8] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report*, Stanford, vol. 1, no. 2009, p. 12, 2009.
- [9] M. Anjaria and R. M. R. Guddeti, "Influence factor based opinion mining of Twitter data using supervised learning," in *2014 Sixth International Conference on Communication Systems and Networks (COMSNETS)*, 2014, pp. 1-8.

- [10] A. Barhan and A. Shakhomirov, "Methods for Sentiment Analysis of twitter messages," in 12th Conference of FRUCT Association, 2012.
- [11] T. Zagibalov and J. Carroll, "Unsupervised classification of sentiment and objectivity in Chinese text," in Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I, 2008.
- [12] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," in LREc, 2010, vol. 10, no. 2010.
- [13] E. Kouloumpis, T. Wilson, and J. D. Moore, "Twitter sentiment analysis: The good the bad and the omg!," *Icwsn*, vol. 11, no. 538-541, p. 164, 2011.
- [14] H. Saif, Y. He, and H. Alani, "Semantic sentiment analysis of twitter," in International semantic web conference, 2012: Springer, pp. 508-524.
- [15] H. Hamdan, F. Béchet, and P. Bellot, "Experiments with DBpedia, WordNet and SentiWordNet as resources for sentiment analysis in micro-blogging," in Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), 2013, vol. 2, pp. 455-459.
- [16] F. Akba, A. Uçan, E. A. Sezer, and H. Sever, "Assessment of feature selection metrics for sentiment analyses: Turkish movie reviews," in 8th European Conference on Data Mining, 2014, vol. 191, pp. 180-184.
- [17] J. Lin and A. Kolcz, "Large-scale machine learning at twitter," in Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, 2012: ACM, pp. 793-804.
- [18] N. F. F. da Silva, E. R. Hruschka, and E. R. Hruschka, "Tweet sentiment analysis with classifier ensembles," *Decision Support Systems*, vol. 66, pp. 170-179, 2014/10/01/ 2014.
- [19] M. Hagen, M. Potthast, M. Büchner, and B. Stein, "Webis: An ensemble for twitter sentiment detection," in Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015), 2015, pp. 582-589.
- [20] E. Martinez-Cámara, Y. Gutiérrez-Vázquez, J. Fernández, A. Montejoráez, and R. Muñoz-Guillena, "Ensemble classifier for Twitter Sentiment Analysis," 2015.
- [21] T. Chalothom and J. Ellman, "Simple Approaches of Sentiment Analysis via Ensemble Learning," Berlin, Heidelberg, 2015: Springer Berlin Heidelberg, pp. 631-639.

- [22] M. M. Fouad, T. F. Gharib, and A. S. Mashat, "Efficient Twitter Sentiment Analysis System with Feature Selection and lassifier Ensemble," in International Conference on Advanced Machine Learning Technologies and Applications, 2018: Springer, pp. 516-527.
- [23] X. Hu, J. Tang, H. Gao, and H. Liu, "Unsupervised sentiment analysis with emotional signals," in Proceedings of the 22nd international conference on World Wide Web, 2013: ACM, pp. 607-618.
- [24] N. Azzouza, K. Akli-Astouati, A. Oussalah, and S. A. Bachir, "A real-time Twitter sentiment analysis using an unsupervised method," in Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, 2017: ACM, p. 15.
- [25] G. Paltoglou and M. Thelwall, "Twitter, MySpace, Digg: Unsupervised sentiment analysis in social media," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 3, no. 4, p. 66, 2012.
- [26] F. M. Kundi, A. Khan, S. Ahmad, and M. Z. Asghar, "Lexicon-based sentiment analysis in the social web," Journal of Basic and Applied Scientific Research, vol. 4, no. 6, pp. 238-48, 2014.
- [27] M. Z. Asghar, A. Khan, S. Ahmad, M. Qasim, and I. A. Khan, "Lexicon enhanced sentiment analysis framework using rule-based classification scheme," PloS one, vol. 12, no. 2, p. e0171649, 2017.
- [28] M. Ghiassi, J. Skinner, and D. Zimbra, "Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network," Expert Systems with applications, vol. 40, no. 16, pp. 6266-6282, 2013.
- [29] F. H. Khan, S. Bashir, and U. Qamar, "TOM: Twitter opinion mining framework using hybrid classification [1] Tan, Kian Long, Chin Poo Lee, and Kian Ming Lim. "A survey of sentiment analysis: Approaches, datasets, and future research." Applied Sciences 13.7 (2023): 4550.
- [30] N. Zainuddin, A. Selamat, and R. Ibrahim, "Hybrid sentiment classification on twitter aspect-based sentiment analysis," Applied Intelligence, pp. 1-15, 2017.
- [31] M. Z. Asghar, F. M. Kundi, S. Ahmad, A. Khan, and F. Khan, "T-SAF: Twitter sentiment analysis framework using a hybrid classification scheme," Expert Systems, vol. 35, no. 1, 2018.
- [32] Neogi, Ashwin Sanjay, et al. "Sentiment analysis and classification of Indian farmers' protest using twitter data." International Journal of Information Management Data Insights 1.2 (2021): 100019.
- [33] <https://data.world/crowdflower/brands-and-product-emotions>

- [34] <https://www.kaggle.com/datasets/yasserh/twitter-tweets-sentiment-dataset>
- [35] Meng, Lingling, Runqing Huang, and Junzhong Gu. "A review of semantic similarity measures in wordnet." *International Journal of Hybrid Information Technology* 6, no. 1 (2013): 1-12.
- [36] Elavarasi, S. Anitha, J. Akilandeswari, and K. Menaga. "A survey on semantic similarity measure." *International Journal of Research in Advent Technology* 2, no. 3 (2014): 389-398.
- [37] Vijaymeena, M. K., and K. Kavitha. "A survey on similarity measures in text mining." *Machine Learning and Applications: An International Journal* 3, no. 2 (2016): 19-28.
- [38] Powers, David MW. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." *arXiv preprint arXiv:2010.16061* (2020).
- [39] Ali, Sajid, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M. Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. "Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence." *Information Fusion* 99 (2023): 101805.
- [40] <https://wn.readthedocs.io/en/latest/api/wn.similarity.html>
- [41] <https://wn.readthedocs.io/en/latest/index.html>
- [42] Alicioglu, Gulsum, and Bo Sun. "A survey of visual analytics for Explainable Artificial Intelligence methods." *Computers & Graphics* 102 (2022): 502-520.
- [43] Resnik, Philip. "Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language." *Journal of artificial intelligence research* 11 (1999): 95-130.
- [44] Jiang, Jay J., and David W. Conrath. "Semantic similarity based on corpus statistics and lexical taxonomy." *arXiv preprint cmp-lg/9709008* (1997).
- [45] Lin, Dekang. "An information-theoretic definition of similarity." In *Icml*, vol. 98, no. 1998, pp. 296-304. 1998.
- [46] Li, Yuhua, Zuhair A. Bandar, and David McLean. "An approach for measuring semantic similarity between words using multiple information sources." *IEEE Transactions on knowledge and data engineering* 15, no. 4 (2003): 871-882.

- [47] Z. Wu and M. Palmer, “Verb semantics and lexical selection”, Proceedings of 32nd annual Meeting of the Association for Computational Linguistics, (1994) June 27-30; Las Cruces, New Mexico.
- [48] R. Rada, H. Mili, E. Bicknell and M. Blettner, “Development and Application of a Metric on Semantic Nets”, IEEE Transactions on Systems, Man and Cybernetics, vol. 19, Issue 1, (1989) January-February, pp. 17 - 30.
- [49] H. Bulskov, R. Knappe and T. Andreassen, “On Measuring Similarity for Conceptual Querying”, Proceedings of the 5th International Conference on Flexible Query Answering Systems, (2002) October 27-29, Copenhagen, Denmark.
- [50] C. Leacock and M. Chodorow, “Combining Local Context and WordNet Similarity for Word Sense Identification, WordNet: An Electronic Lexical Database”, MIT Press, (1998), pp. 265-283.
- [51] Bandi, Ajay, Pydi Venkata Satya Ramesh Adapa, and Yudu Eswar Vinay Pratap Kumar Kuchi. ”The Power of Generative AI: A Review of Requirements, Models, Input–Output Formats, Evaluation Metrics, and Challenges.” *Future Internet* 15, no. 8 (2023): 260.
- [52] Duan, Nan, Duyu Tang, and Ming Zhou. ”Machine reasoning: Technology, dilemma and future.” In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pp. 1-6. 2020.
- [53] Shet, K. C., and U. Dinesh Acharya. ”A new similarity measure for taxonomy based on edge counting.” *arXiv preprint arXiv:1211.4709* (2012).
- [54] Srilatha, Pulipati, and Ramakrishnan Manjula. ”Similarity index based link prediction algorithms in social networks: A survey.” *Journal of Telecommunications and Information Technology* 2 (2016): 87-94.
- [55] Ziogas, Ioannis Panagiotis, Errikos Streviniotis, Harris Papadakis, and Georgios Chalkiadakis. ”Content-based recommendations using similarity distance measures with application in the tourism domain.” In *Proceedings of the 12th Hellenic Conference on Artificial Intelligence*, pp. 1-10. 2022.
- [56] Nandwani, Pansy, and Rupali Verma. ”A review on sentiment analysis and emotion detection from text.” *Social network analysis and mining* 11, no. 1 (2021): 81.
- [57] Wankhade, Mayur, Annavarapu Chandra Sekhara Rao, and Chaitanya Kulkarni. ”A survey on sentiment analysis methods, applications, and challenges.” *Artificial Intelligence Review* 55, no. 7 (2022): 5731-5780.

- [58] Lighthart, Alexander, Cagatay Catal, and Bedir Tekinerdogan. "Systematic reviews in sentiment analysis: a tertiary study." *Artificial Intelligence Review* (2021): 1-57.
- [59] Bansal, Gaurang, Vinay Chamola, Amir Hussain, Mohsen Guizani, and Dusit Niyato. "Transforming conversations with AI—a comprehensive study of ChatGPT." *Cognitive Computation* (2024): 1-24.
- [60] Koubaa, A., Boulila, W., Ghouti, L., Alzahem, A., & Latif, S. (2023). Exploring ChatGPT capabilities and limitations: a survey. *IEEE Access*.
- [61] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- [62] Karimi, Akbar, Leonardo Rossi, and Andrea Prati. "Improving bert performance for aspect-based sentiment analysis." *arXiv preprint arXiv:2010.11731* (2020).
- [63] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., Polosukhin, I., "Attention is all you need." *Advances in Neural Information Processing Systems* (2017).