

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

**Sentiment Analysis with Named Entity Recognition from Internet
Articles**

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics



Sentiment Analysis with Named Entity Recognition from Internet Articles

Diploma Thesis

Study programme: Cognitive Science
Field of study: 2503 Cognitive Science
Training workplace: Department of Applied Informatics
Supervisor: RNDr Kristína Malinovská PhD.

Bratislava, 2018

Michal Paleš



Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Michal Páleš
Študijný program: kognitívna veda (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: kognitívna veda
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Sentiment Analysis with Named Entity Recognition from Internet Articles
Analyza sentimentu pomocou rozpoznania pomenovaných entít z článkov na internete

Cieľ: Vytvorte inteligentný program na analýzu a porovnanie sentimentu článkov s rovnakou témou zverejnených na internete:
1. Preštudujte, vyberte a vyhodnoťte vhodné metódy strojového učenia na extrakciu pomenovaných entít z textu a analýzu sentimentu s dôrazom na metódy inšpirované kognitívnu vedou.
2. Implementujte skript (web-crawler) na extrahovanie dát a vybrané metódy strojového učenia, vyhodnoťte navrhovaný systém a jeho výsledky.


Literatúra: Mikolov T. et al. (2013). Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems. Nadeau D., Satoshi S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30,1: 3-26.
Wang S., Manning C. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. *Association for Computational Linguistics*.

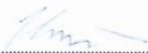
Anotácia: Problém mienkotvornosti médií sa stáva zaujímavým fenoménom našej doby. Analýza sentimentu je spôsob, akým sa dá určiť postoj autora voči opisovaným udalostiam alebo objektom. Novinové portály predstavujú kvalitný zdroj dát umožňujúci dolovanie v dátach a experimentovanie s prirodzeným jazykom. Preto je vytvorenie systému, schopného automatizovane spracovať články publikované na internete a zanalyzovať ich obsah pomocou extrakcie pomenovaných entít a analýzy sentimentu výzvou v tejto oblasti.

Vedúci: RNDr. Kristína Malinovská, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.
Dátum zadania: 01.03.2016

Dátum schválenia: 01.10.2016

prof. Ing. Igor Farkaš, Dr.
garant študijného programu


.....
študent


.....
vedúci práce



Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname: Bc. Michal Páleš
Study programme: Cognitive Science (Single degree study, master II. deg., full time form)
Field of Study: Cognitive Science
Type of Thesis: Diploma Thesis
Language of Thesis: English
Secondary language: Slovak

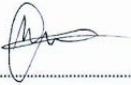
Title: Sentiment Analysis with Named Entity Recognition from Internet Articles
Aim: Design an intelligent application for analysis and comparison of sentiment of articles with identical subject published on the internet:
1. Find, choose and evaluate acceptable machine learning methods for the given task by using named entity recognition and sentiment analysis with focus on approaches inspired by cognitive science.
2. Implement a script (web-crawler) for extracting data and selected methods, and evaluate the proposed system.


Literature: Mikolov T. et al. (2013). Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems. Nadeau D. , Satoshi S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30,1: 3-26.
Wang S., Manning C. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics.

Annotation: The phenomenon of media being able to manipulate the public opinion is becoming an intriguing aspect of our society. Sentiment analysis allows to estimate the author's attitude towards the described event or object. News portals are a first-rate source of data which allows data mining and experimenting with natural language processing. A system capable of automatically processing the articles published on the internet by classifying named entities and the sentiment with help of distributional hypothesis is an interesting challenge.

Supervisor: RNDr. Kristína Malinovská, PhD.
Department: FMFI.KAI - Department of Applied Informatics
Head of department: prof. Ing. Igor Farkaš, Dr.
Assigned: 01.03.2016
Approved: 01.10.2016

prof. Ing. Igor Farkaš, Dr.
Guarantor of Study Programme


.....
Student


.....
Supervisor

I declare that this thesis is the result of my own work and I have cited all the sources.

Signature:

Acknowledgments

I wish to express my gratitude to my supervisor RNDr Kristína Malinovská PhD., who helped me with, apparently, endless amounts of goodwill, patience and support. I also thank my friends, colleagues and family for supporting me. Sometimes in the most fascinating and unexpected ways.

Abstrakt

S príchodom Internetu nastala doba, kedy má prístup k obrovskému množstvu informácii úplne každý. Spolu s tým, musia existovať aj ľudia, či skupiny, ktoré obsah webu naplňajú a tvoria. Tieto informácie môžu, ale nemusia byť pravdivé, a tak som sa rozhodol preskúmať daný problém cez niektoré dostupné postupy.

Mojím zámerom je vytvoriť aplikáciu, ktorá bude schopná stiahnuť on-line dáta z webov niekoľkých veľkých spravodajských denníkov, kvantifikovať sentiment pre vybrane entity a porovnať články relatívne k sentimentu voči spoločným entitám.

Za týmto účelom sme preskúmali prechádzajúce práce a prístupy k problematike “Sentiment Analysis” a “Named Entity Recognition”. Následne sme vybrali z týchto prác také, že spĺňali kritéria, ktoré sme považovali za dôležité a zaujímavé. Následne sme ich implementovali.

Po implementácii, sme urobili experimenty výsledných skriptov, zmenami na úrovni hyper-parametrov, reprezentáciami slov a taktiež vrstiev siete. Následne sme porovnali výslednú úspešnosť daných sietí.

Ďalej sme vyrobili aplikáciu schopnú sťahovať “live” dáta z novinových portálov.

Na záver sme ešte uviedli niekoľko možných vylepšení alebo úprav, ktoré by mohli napomôcť riešeniu alebo uľahčiť prácu v tomto okruhu výskumu.

Kľúčové slová: *Named Entity Recognition, Sentiment Analysis, Web crawler, Data mining*

Abstract

With the dawn of the Internet, came the time, where almost everybody has access to almost non depletable amount of information. But with this, there are people, or groups of people that create the content of the web itself. Such information can, but not necessarily have to, be true and so I decided to research this problem through several existing approaches.

My goal here is, to create an application, that is able to download data from online sources, like big newspaper sites, quantify the sentiment towards specific named entities and compare the results between the different online sources.

For this purpose, we researched previous works and approaches to “Sentiment Analysis” and “Named Entity Recognition”. Thereafter, we selected, from the previous works such approaches, that we deemed important and interesting to work with. Afterwards we implemented and tested the aforementioned network architectures.

After implementation, we experimented with the resulting scripts, by changing the network hyper-parameters, word level embeddings as well as the layers of the networks. Then, we proceeded to compare the resulting success rates.

Additionally, we developed an application capable of downloading “live” data from news agency.

Finally, we discussed several other possible improvements, which could potentially improve or ease the work on similar challenge.

Keywords: *Named Entity Recognition, Sentiment Analysis, Web crawler, Data mining*

Contents

1. Introduction	10
1.1. Motivation for analyzing sentiment of texts	11
1.2. Truth and Sentiment	13
1.3. Background	16
1.4. Information retrieval and the past	17
2. The challenges of our work	18
2.1. The data	20
2.1.1. Training data domain	21
2.1.2. Test and train data split	22
2.2. Scraping	23
2.3. Processing	23
2.4. Related works	25
2.4.1. Criteria of choice	25
2.4.2. Named Entity Recognition	26
2.4.3. Sentiment analysis	27
3. Implementation	29
3.1. Word embeddings	29
3.2. Web Crawler	31
3.3. Named Entity Recognition	32
3.4. Sentiment Analysis	34
3.5. Technical Details	35
3.5.1. Utilised python libraries	35
3.6. Usage/Documentation	37
3.6.1. Scraping	37
3.6.2. Network training	39
4. Results and Experiments	40
4.1. Named Entity Recognition	40
4.2. Sentiment Analysis	42
5. Conclusion and Future work	46
5.1. Conclusion	46
5.2. Future work	47
5.2.1. New datasets	47

5.2.2. Aspect Level approach	47
5.2.3. w2v prediction	48
5.2.4. Sarkasm	48
5.2.5. Pronouns	48
5.2.6. HTML Tags and article images	49
5.2.7. Topic analyzer	49
5.2.8. Improving the code	50

1. Introduction

The problem of Sentiment Analysis (SA) is a quite an old question, the research in various fields has posed long before I was born. Yet even the question is difficult to define, as it approaches the world of feelings and opinions, rights and wrongs, and as a consequence the search for the Truth in the fields of Natural Language Processing (NLP) and Machine Learning (ML). We do not assume know, nor try to answer these questions, but we would like, nonetheless, add our piece of work to the heap.

This master thesis work can be considered partially implementational and partially a research work.

The master thesis itself is divided to several chapters. In the first chapter, we will try to introduce the reader to the motivation and background of the problem we are facing. In the second chapter, we explain, in detail, what steps we choose to undertake, in hope of solving the “questions” our problem poses. We also add the reasoning behind those choices. In the implementational chapter number three, we will proceed to implementation and experimenting, to see how our solution performs and how we could change it, for better performance. Afterwards, we analyze the results and experiments we did with our proposed solution. In the end, we conclude our work in some final thoughts and words, followed by a list of the work we could do, to improve it further, or add another layer of sentiment mining. Furthermore, we will try to introduce the reader to the interdisciplinary background, touching the fields of Machine Learning, Philosophy of language and a small questioning of human behavior.

1.1. Motivation for analyzing sentiment of texts

As cognitive science is inherently interdisciplinary, we were motivated and inspired, by several fields, that are recognized within the scope of aforementioned science. Namely, fields relevant to sentiment analysis are linguistic, philosophy of language, and computer science.

Before answering the “how to do this” question, I would like to answer the question: “why did I decide to do this”, as this is, at least for me, the more important part. Since the “how” may fail, may be substituted by an better model or, indeed, an better approach altogether, yet the idea stays the same. This whole work could be considered my personal quest for finding the Truth. There is a reason for why it had to become a personal quest, instead of this being just another work I had to do. The reason is pretty simple actually. I find it extremely difficult to force myself, to do any work, in which I can't seem to find at least some application into practical reality. So a solely theoretical or research oriented work was mostly out of question. (Of course, a whip could do the job, yet this would require another person with a lot of motivation, time and energy.)

As a result, I decided to find something for myself to work on, and I had this idea (which I describe later) for some time, and figured that I might as well as use the time, I am supposed to work on the thesis in a way which coincided with my personal interests. This could make the work less a “do it, because you have to” type of work and more of “this is interesting and (partially) fun”. (Not the writing, though.)

As for what I wish to accomplish, in the long term plan, the basic concept goes, as follows. I would like to build an application with access to the internet, that can, after some user input, automatically scrape a web page, let the user know the topics (we use this word interchangeably with named entities) mentioned in the text and, in the case of relevant topics, it should also offer the sentiment towards those topics. Additionally it should also offer similar websites with comparable topics and its respective sentiments. The user now can compare the web articles with similar or identical topic, by sentiment toward the aforementioned topic and find out, if there is a pattern in the sentiment towards specific topics, by a specific new agency or

web page. There is some other functionality as well, but it is not important, or meaningful, to mention it here.

So, to oversimplify, the question I pose here is, whether there are specific news agencies, web pages or groups of agencies responsible for media, that connect a specific sentiment, with specific entities. Hence, they could have some level of impact over how such entities are perceived by their readers. Especially if the language they use is subtle and which would make an intention to influence less obvious. I would not dare to assume to be able to build such an, in my eyes, advanced system, which would be able to disclose such intention. Yet I deem this to be first of necessary steps to approach this question.

As reported by Cheryan, Sapna, and Galen V. Bodenhausen even an simple expectation, like “Overall, my race is considered good by others” can affect the people and impair their intellectual performance [1]. If we generalise and make an indirect statement that subtly suggests that “good people” tend dislike certain entity, there is a chance that people will be inadvertently influenced by such statement. Especially if we encode it into the context of the article, making it less obvious [2].

While I do not believe, that the system I am building would be able to successfully detect such subtle manipulation, I do believe, that such manipulation uses a specific subset of language, hence there should be a pattern. As Machine Learning (ML) is nothing but a way to find patterns in data, this should be an effective first step to accomplish this.

Such system, would then be able to let the user know whether these web pages have some sentiment attached to the topics or entities mentioned in the article, and compare it to similar articles with comparable topic.

1.2. Truth and Sentiment

To continue, we need first to explore what the Truth is, and how it correlates with either sentiment or opinion. I do believe that Truth is exceptionally important, especially nowadays, as almost everyone, with access to the internet, can make a public statement, that will be readable for anyone with the the same access. As my empirical observation would suggest that the human nature is first to believe, then to validate, it is not that difficult to miguide someone. Especially on topics that are difficult to attest. (If we are told that “It rains” we first believe the statement and only later we seek to validate this “truth”.)

So what is the Truth? For most people, probably the first definition that comes into mind, is a tautology. Truth is truth and it's as simple as that. While this definition is as true as it gets, for our needs, we will define the Truth as “the body of real things, events, and facts” and as “the state of being the case”.^[3] In the first definition, the part I want to concentrate on is “the body of real ... facts”. Meaning that the Truth, in our case is always a thing of the past, that is unchangeable and solid. It is not open to interpretation, it's simply the reality we live in and it cannot be altered. This kind of truth cannot be applied to predictions, as they can turn out to be true, just as the probabilistic nature of quantum superposition and its collapse into (our) reality.^[4] Let's not follow this path, as it leads towards Theory of Relativity, time dilation, Schrödinger's cat and some questions I can not even try to answer.

Sentiment towards anything, on the other hand, might be true or false, in its interpretation of objective reality as this is something, that is quite often, if not entirely, dependent on a feeling of an individual. The definition I would like to go with, for this arguments sake, is “an attitude, thought, or judgment prompted by feeling”^[5]. It is more of an personal preference, relative to some object or fact. To quickly illustrate, some people deem snakes to be something horrifying, disgusting or just scary, while others have them like pets, but neither of those people can be considered right or wrong in their sentiment towards snakes. Both can argument, within reason, for their standpoint, and neither of them would be wrong.

While I, consider pure sentiment, without reasoning, something of an inherent flaw in human race, especially in this age, there seems to be an evolutionary reason for this mechanics, as it is a faster response. Sentiment, or emotional processing, can be traced back to many parts of the brain, specifically the limbic system from which the most prominent would be amygdala [6]. There are also other, somewhat even older parts, responsible for emotional responses or namely cerebellum, as reported by Wolf, Uri, Mark J. Rapoport, and Tom A. Schweizer [7] in their study. It mentions that cerebellum, while most often associated with motor functions, also connects with brain stem nuclei, which supply the limbic system with important neurotransmitters, namely serotonin which do affect the mood [8] and dopamine, which takes care of the reward process [9] of the person.

From evolutionary standpoint, this impulsive (used as a synonym for emotionally based) kind of behavior has a seemingly simple explanation. According to Paul D. MacLean's triune brain theory [10], the human brain contains three theoretical components, categorised by their evolutionary age. The reptilian, old mammalian, and new mammalian “sub-brains”. It is further hypothesized that the limbic system, which belongs to the old mammalian brain, is responsible for “flight-or-fight” [11] survival adaptation. Without the immediate reflexive response to danger, we might have not survived, as it is much quicker in register danger, or negative stimuli [12]. From pragmatic point of view, an individual has better chances of survival, if he starts running as soon as he thinks he might have spotted a tiger in the savanna, compared to an individual, that tries to figure out whether it is a tiger or just the grass and wind playing mind tricks.

As for my disdain for impulsive and strictly reflexive behavior in our age, even after the perceived danger passed, it exists, because there is a certain unwillingness to alter the previously acquired opinion. (Because, apparently, thinking hurts... or at least it uses uncomfortably high amounts of energy, which the person is not willing to sacrifice.) As an simplistic example, I would like to illustrate this on a frequently occurring overreaction in case of bee, or wasp, in the room. While the animal might be somewhat agitated by the fact that it thinks to be trapped, the

human does not need to be, yet, no amount of explanation (in some cases) seems to help. The fact that the “dangerous” animal might be even more agitated by such behavior just adds insult to the injury. To sum it up, it is the absence, or unwillingness, to exercise self-control, that makes me question the human intelligence. Especially when we could manage it [13].

But does this mean, that all emotionally influenced data or opinions are wrong? Most certainly not, but the information could be always given in a neutral way, which would make it more valuable, as anybody would be able to extract the information with minimal bias. So this application we are building, could show the user the bias added to the topic by the author of the article, and let him choose to either accept, or agree with, the bias, or, if he is already informed about it, ignore it. On the other hand, not all bias neutral data is true either.

So, to summarize it, my motivation to do this work is following:

- To make the users alert to coupling of specific sentiment and named entity in the articles or articles as my personal quest, or search, for the “true, unbiased data”. (Yes, a unicorn. I know.)
- Try to help to remove emotional bias from data, by pointing it out.
- To improve and build my personal skills in fields of Natural Language Processing, Data-mining and Machine Learning.

1.3. Background

The amount of netizens grew in the last five years by more than 1.7 billions [14]. It is not only web content consumers, but also internet content creators. This gives everybody with the access to the internet the opportunity to use the freely available data any way one wishes to. As a result, I would argue, there is no better time to start with machine learning and data mining, as these techniques are the future of data, especially “big data”, manipulation. As for why I believe that now is the best time, it is the merge of several factors outlined below.

- Almost everybody has access to the internet, so he can grab the data and do the magic.
- Most people have access to hardware that makes such efforts possible, even at home, there is no need to own a datacenter.
- The knowledge is free, there is enormous amount of tutorials, “how-tos” and video introductions into the “data science”.
- Lastly, the emergence of easy to use computational frameworks, that make the data analysis accessible and, almost, straightforward. Just to name few: tensorflow [15], keras [16], scikit-learn [17], theano [18], etc.

The content of the web is filled by people, and as we know, people can err. Be it intentional, or unintentional, not all data on the internet are equal. Be it because they are influenced by some bias, or the information entirely made up, a system, that can analyse and decide the verity of any given data would be extremely useful. While I do not claim to be able to, in my current state, or even wish, to build such a system during this thesis, there are necessary steps for building the foundations. One of them is being able to spot the sentiment, another being able to spot the aspect, to which the sentiment is coupled.

1.4. Information retrieval and the past

The intention of the field of “Information Retrieval” (IR) is to develop a method of automatic management of text (in terms of automatically obtaining an information from some information resources, or documents), instead of just storage or display. For this purpose, the discipline of Information retrieval was founded, which, several decades ago, tried to solve this problem by mostly statistically based methods. The parallel work on “Natural Language Processing” (NLP) sought to model these methods after human language processing. This consisted mostly of hand written rules. The genesis of these disciplines can be traced back to Alan Turing's “Computing machinery and intelligence” [19] in case of NLP and Vannevar Bush's “As we may think” for IR [20]. One of the semantic sub-task of NLP is also Sentiment Analysis.

A younger field of “Information Extraction” (IE) was spawned, in between the previous two, in the late 1970. Its goal was to allow computation to extract information, or find a pattern, in a previously unstructured data. To illustrate what was the goal of IE, it is simply put, to understand a set of domain specific documents well enough to be able to answer a imaginary questionnaire. If the document domain would be terrorism, the goal for IE would be to fill in the information like who the terrorist were, who the victims were, what was the goal of the attack, etc [21].

IE evolved over the last two decades with help of the “Message understanding Conference” (MUC) which was the premier forum for research in IE. Later researchers in Machine Learning (ML) noticed that IE offers a great set of interesting problems, that ML could try to solve. [21] At the 6th MUC the sub-task of IE was created, which is now known as Named Entity Recognition.

2. The challenges of our work

This chapter contains the introduction to the problem we face, the analysis of the said problem and then the possible solutions, which were utilised. We also analyse the data we have at our disposition and surveyed other options, in terms of different datasets. One more important information here, is the fact that we will do this work strictly for English language, as datasets are difficult to get by. In case we had to solve this for different natural languages (like German or Slovak), it would be even more problematic, as public datasets are virtually non-existent as are higher quality word representations, for the aforementioned languages.

Our challenge itself can be divided into several sub-problems, which will require us to solve separately and in the following sequence, as otherwise, the result would be meaningless, as it require the output of the previous step. First issue we want to tackle are the data and the respective solution. The issue with data is finally, at least, 3-fold. First of, we require tagged data for training purposes, this is again more problematic, as we were unable to find data of sufficient quality that would help us in solving our challenge. As a result of this, we ended up splitting our task, into 2 different subtasks. One is for solving to figure out the entities the “real life” data (In the scope of this document, we may call them “demo” data as well.) contains and the second is for its respective sentiment towards the topics, which were discussed. Lastly, we needed a way to acquire the demo data.

To make our problem more “digestible”, we need to break it down into atomic parts, so we can focus on one of the issues at the time and then proceed to another one, when the previous is finished.

- We want a way to find the articles, about the same event/fact from different sources.
- We want to find the core topics, or rather whom or what does the article discuss (these will be further called Named Entities, or just Entities and the method of extracting it is known as Named Entity Recognition)
- We want to figure out a way, how to quantify the standpoint of the author towards the various topics and Entities (this technique is further called Sentiment Analysis)

2.1. The data

For our implementation, we require properly tagged data, as this is commonly required for supervised training. In an ideal world, the data would contain a sentence in English language, with its respective tags. As mentioned previously, we would like to obtain the sentiment towards specific Named Entities which we mined beforehand. So the text should contain all the possible named entities, regardless of their domain. And another tag for the sentiment of the sentence, or even better, partial tags for parts of the sentence for, as we hope, greater sentiment granularity.

As we were unable to obtain such data we needed to expand our search onto more task-specific datasets, which was less problematic to find and obtain, compared to a complete dataset with all the required tags.

As we try to keep the data as close to the real use-case, we would want to deploy the resulting networks in, we decided to keep the data length exactly one sentence long, so we can try and make our network learn in the worst case scenario.

For Named Entity Recognition, we were able to find such data, which contained 1000 sentences, each sentence broken down to one word per line, followed by a tabulator and its respective tag. After each sentence there was an empty newline. The origin of this dataset is unknown, it has been used by several projects located on github [22][23].

There were several other possibilities, in terms of choice for this particular task. Specifically the CoNLL challenges from years 2002[24] and 2003[25]. While the data itself was usable, some circumstances made it a less viable choice, since it contained sentences with different languages. This was an issue for us, as we planned, in the later stages, to use pre-trained word representations [26][27], that do not contain pre-trained vector representation [28] for any other language, except for English.

The single snippet of this dataset, is formatted as follows. Each sentence is followed by an empty newline, which indicates a new snippet. All the data is in already in lowercase. There were 9 different tags, for each respective class category.

kaiima	B-KEYWORDS
merger	I-KEYWORDS
news	B-NEWSTYPE
stories	I-NEWSTYPE
in	O
the	O
kootenai	B-PROVIDER
valley	I-PROVIDER
Record	I-PROVIDER

For Sentiment Analysis, we had several datasets, to choose from. Again, the intention was to have the “worst-case-scenario” data length, to see how our model could perform on “live” data. Hence, most of the datasets freely available, were inappropriate for our use case, as they usually tend to be articles, instead of single sentences. It was first used by Bo Pang and Lillian Lee[29], in their work. The data consist of two files, each file contains 5331 sentences. The structure of data is straightforward, so it is unnecessary to explain it further. We also used our model on Keras IMDB [16], for validation purposes and to find out if the performance of the network has anything to gain from a dataset that is much larger compared to our selected train dataset.

2.1.1. Training data domain

Our intention is to build an application that is universal in its approach all data domains, hence we wanted our data to be as variable as possible. But the training data domain specificity for the Sentiment Analysis remains somewhat questionable, since we did not use, nor have, any domain-non-specific training data. The dataset for SA was comprised of one line movie reviews,

so the trained network will perform better on review-like sentences, and might be less useful on data, that have different structure, domain, or use the language in a previously (by the network) unwitnessed manner. [30][31] The data for our NER task was not domain specific, as least as far as we could tell.

2.1.2. Test and train data split

As for splitting the data, we used 2 different data split options, as Leave-p-out exhaustive Cross Validation (LPO-CV) is not a great option in our case, since it is computationally infeasible, even on datasets, such as the ones we used. We decided to use some sort of approximations of the previously mentioned LPO-CV, namely the K-Fold Cross Validation (KF-CV) [26] and Monte-Carlo Cross Validation [42].

The Monte-Carlo Cross Validation is simple random split of the dataset in which one part will be used as training data, and another part as testing. Usual ratio is 8:2, we used 9:1 because we wanted the network to see as much data as possible. Second reason for this, is that we are using 10 Fold CV where applicable. It is also known as repeated random subsampling [32].

K-Fold Cross Validation is a method, in which we split the whole data into “K” chunks of similar length, and we iterate “K”-times over the training. We change the current chunk incrementally so it is used as testing data and the rest is used for training. The usual “K” is 10[33].

As KF-CV is more systematic in its approach, we preferred this method to Monte Carlo, but used both during implementation and training.

2.2. Scraping

For our demo (or real) use, we need to acquire the real, untagged data by scraping some of the better known web news agency web pages. As the data from web pages comes as pure html, with lot of undesired data, we needed a way to data-mine it and select only the desired parts. So we removed all the unnecessary HTML tags, links, ads, pictures and other type of data that we could not use it the current state of our data-mining workflow. Some of the discarded HTML data might be of some importance, but we haven not built-in the ability to work with data of such structure, so this will be discussed in the Future Work.

2.3. Processing

After scraping the data, we detect the Named Entities in the text, and select keep the sentences containing the same Name Entities for later use in Sentiment Analysis. So as a result, the Sentiment Analysis runs only on chunks of the text, that contain the one specific Named Entity. This approach could be probably improved by finding sentences which mention the Named Entity by pronoun, but we will devote some time to it, in the later chapters.

After we got the data, we need to pre-process it for each algorithm, in a different manner. While we need the whole text for our Named Entity Recognition, because the Named Entities can contain words, that are usually discarded before Sentiment Analysis, we do not require all the word contained in the sentence for Sentiment Analysis. This is because we remove lot of stop-words or too often used words.

The Named Entity requires a few adjustments to the text corpus, as any word, can be a part of a Named entity. The only adjustment is to remove non alphabetical characters and make all word lowercase.

For Sentiment Analysis we remove the “stopwords” or “most common words”, which counter intuition, and are not important, as they are used way too often, so we are unable to

assign them any value. Such words as “the” or “and,” help build ideas, but do not carry any significance in the text.

These are formally defined as by a fraction also known as TF.IDF (Terms Frequency times Inverse Document Frequency) [34]. Mathematically speaking, Term Frequency, is the fraction of the frequency (freq) of word $w_{[i]}$ in document [d] divided by the occurrence (or frequency) of the most common word, let's call it $w_{[max]}$, in the document [d].

$$\mathbf{TF} = (\text{freq}(w_{[i]})_{[d]} / \text{freq}(w_{[max]})_{[d]})$$

The IDF part is defined as $\log_2(N/n_{[i]})$ for which N is the total number of documents we have at our disposition and $n_{[w]}$ is the number of documents in which observe the word $w_{[i]}$.

$$\mathbf{IDF} = (\log_2(N/n_{[i]}))$$

So a high weight of TF.IDF is reachable only if the TF is high and the IDF is low.

We then proceed and tokenize [35] the text and start the analysis of sentiment in the text chunks, which we will discuss further in the implementation and experiments.

2.4. Related works

This chapter is dedicated to the previous solutions from which we want to choose one to implement and test. The chosen approach will have to fulfil several criteria, which are considered important, with different weights on them.

2.4.1. Criteria of choice

We choose our favorites according to several criteria, that the chosen approaches should fulfil.

First of all, the selected machine learning (ML) technique should be within the scope of abilities of the chosen language. This was an easy decision, as Python is one of the best supported languages with regards to ML and my default choice.

As Python is well supported with libraries, the next thing I wished to explore is Keras[16] as previous quick attempts proved this library superior to others in terms of quick prototyping, so I wished to see how far this can bring me.

The last of the criteria, that is interest-bound is the utilisation of Convolutional Neural network (CNN). Also, they are strongly influenced by the principles of mammalian visual cortex, which makes them even interesting, because if it works well in the brain, it should also work in computation.

The last few criteria I impose upon this choice, are more of pragmatic character. The chosen models should be practical and the greater the performance, the better. Furthermore the model should ideally work as a baseline model with the opportunity to expand upon the said model with additional ideas, improvements or layers of neurons.

2.4.2. Named Entity Recognition

Named Entity Recognition (NER) is the task of identifying Named Entities in the text corpus and classifying them into predefined categories such as persons, organisations, locations, dates, and similar. The “Named Entity” aims to restrict the task only to find Saul Kripkes “rigid designators” [36]. Simply put, the “Named Entity” denotes only the one thing it denotes in all possible world it might or might not exist in. Even simpler, the thing that NE denotes is only the one thing that comes into mind, when we use the word. We require this technique to select our named entities in the text for later classification with SA.

The earliest approaches to this problem included mainly handcrafted rule-based algorithms which were later substituted by techniques borrowed from the ML field. The question of Named Entities was asked through the whole spectrum of ML standards for the supervised learning the most prominent approaches since the begin of this type of classifications are Hidden Markov Models, Decision Trees, Maximum Entropy models Support Vector Machines and Conditional Random Fields. These approaches require a vast amount of tagged data to learn and they try to generalise the rules they found in the data.

The Semi-Supervised Learning path involves a small amount of supervision and then the algorithm tries to generalise from those data. Usually this done by something like regular expressions and, for our use-case, this technique would not bring any fruits. [37] Fully unsupervised learning, on the other hand, induces opportunities, that allow us to combine data from unsupervised learning with supervised learning for greater success. In the last 4 years there had been plenty of data from unsupervised learning, that were able to represent the words in a many dimensional vector space, that give them computational meaning[28].

In the last few years, the most prominent and successful approach was witnessed with the use of Long Short term Memory Recurrent Neural Network with various adjustments and additions [27][44][45].

The systematic approach proposed by Zhiheng Huang, Wei Xu and Kai Yu [27] was very much to my liking, because it was systematic, and they went from the simplest useful architecture to those more complex ones. So from the possible set of approaches I saw, I chose this one.

In the time of writing we also found a mention of Iterated Dilated Convolutional Neural network [43] for NER, that I didn't have time to understand nor the time to try to implement it, as it is fairly nouveau in its approach to the field. Other than that, I found it interesting as it promised strong optimisation for graphics card hardware, which is kind of my hobby and I probably will try to figure it out later.

2.4.3. Sentiment analysis

Sentiment Analysis (SA), or also known as opinion mining, is simply put, the way, how to identify the subjective information in the text with help of Natural Language Processing and ML. It is, in the case of our “real data”, the attitude of the author of the article towards some described topic. A form of emotional or subjective reaction. It is a sub-task of Natural Language Processing, we mentioned in the first chapter. So it is normally a binary classification, but there are datasets, that have tags representing a degree of sentiment, for the humble beginnings, the binary classification seems to be sufficient, especially, we need to keep in mind that the dataset has to be as we specified. In the last years, the sentiment analysis created a great number of sub-tasks that are being solved by SA. Sentiment incorporates also Opinion Mining and Objectivity Mining.

The previous research do tell us, that almost any supervised learning method could be applied, within reason. Just to name the most used: Support Vector Machines and Naive Bayes Classifier. Like in any supervised learning work, the main objective is to find an effective set of features which best describe our data and the pattern in our data [38].

In some more recent works we found that there was an initiative to build a simple yet effective CNN architecture for supervised learning that would perform reasonably well, as reported by Yoon Kim [26] and Collobert and colleagues [39]. With this in mind and the fact that we wanted to utilise the CNN, not to mention the seemingly easy to implement architecture, we decided to use this approach for our Sentiment Analysis task.

3. Implementation

As we outlined in chapter 2 the problem of combined SA and NER incorporates several different task, that need to be solved in sequence, else they do work as intended. We will hence address the

This chapter is devoted to explain our implementation step by step with the respective decision for our application to work as intended. The required libraries with its respective dependencies are mentioned before the implementational part, as without them, it is impossible to run any of the scripts contained within this work.

We will first address the web crawler, that gets our “real” data, afterwards we describe our proposed models for NER and SA.

The last part of this chapter belong to technical issues we did run into a system requirements.

3.1. Word embeddings

For SA, we tried different word representations. NER used only the pretrained. The first alternative we worked with, was random word embedding. This creates, for every word, an n-dimensional word-vector to represent it. This approach worked, but was slightly limiting as we would end up with always growing word-vector storage that would be of low quality, as the incoming data is not sufficiently massive, in our training, for this kind of tasks.

We decided in the earlier stages, that we want to utilize the pre-trained, publicly available, vectors of word2vec (w2v) to see, if this unsupervisedly learned word representation [28] indeed improves the accuracy of our implementation of the selected approaches. Especially when our training data corpus is as small as it is. Other than that, the vector-space was trained on “100 billion words” [26] which was a little bit unclear, as the definition of billions is, at least, two-fold. One would wish the authors of [26] used powers of 10. Not a rigid designator, then.

The w2v model is, in a sense, an implementation of “distributional hypothesis” [40]. To put it simply, it implies that words, that are used and are found in similar context, tend to have similar meaning.

The w2v was trained by the approximation of skip-gram model, proposed by Mikolov et al. in 2013 [28], also known as the negative sampling. The skip-gram model, as it stands is very useful in predicting the words $w_{[i-n]}$ and $w_{[i+n]}$ for the word $w_{[i]}$ (while n is floor $(0.5 * k)$ k is the size of the windows responsible for the skip-gram training), but, as we proved later on, also useful for SA and NER.

The skip-gram model uses a window in which it travels and selects the words, that are together as inputs. In comparison, n -gram selects only the n words that follow each other, this approach is modeled based on the distributional hypothesis. Even with my limited knowledge in languages, this idea sounds pretty convincing, as German uses entirely different word-order, compared to Slovak, yet we are able to translate between these two languages, so the meaning in the sentence stays the same. The same can be applied to Hungarian language.

The pretrained binary blob containing the pretrained word-vectors was trained with the help of negative sampling, which is an approach that reduces the enormous amount of computations required to train a true skip-gram network [41]. The basic idea is to adjust the weights of a small amount of “negative” samples, which are 2-grams that are almost never seen together, instead of adjusting all the weights. [28]

The word2vec embeddings are freely available from the internet, they are not included into this thesis.

3.2. Web Crawler

The first part of our implementation is getting the untagged “live” data, from the internet. It is done by aforementioned scrapy and beautifulsoup, which are 2 python frameworks for working with scraping, spiders and HTML files.

The first step we needed to do, was to define the actual crawlers, which scrapy calls “spiders”. Each site we wanted to scrape needed one, so it can travel through the sites and look for news. The implemented default two sites, to compare, were “theguardian.com” and “independent.co.uk” with their respective world related news.

At the time of writing, the most favourite word (empirically) of both of the news agencies is “Trump” with not much of an positive sentiment.

It is quite simple to append additional spiders to the list and use them for scraping as well.

After the spider finishes the crawling part, the user is presented with a list of items he may wish to download and compare. He may choose one from each spider/website to make them comparable.

The choice activates a fetcher script that downloads the raw HTML script representing the site and then removes all the unusable data we do not need. This will eventuate in a human-readable chunk of text that represents the actual article.

Again, the scripts responsible for this, are handcrafted and cannot cope with changes in the structure of the website, so we need to write each of these “fetcher” scripts manually. As every website has its own structure and conventions, it is not possible to make an universally functional script for all the web content. Actually, it is nearly impossible to make one script work for two different web pages, not to mention more of them.

The last few pages of this chapter are dedicated to documentation like descriptions that should make it easier to understand the process.

This implementation is kind of an eternal fight, as it works with an ever-changing content of the web. A direct result of this, is that the content structure might change which eventually breaks the crawlers ability to effectively download any data for us to analyze.

3.3. Named Entity Recognition

In the previous chapter we introduce the implementation we wish to explore. We are going to address it here in more detail, as it is required and simpler to re-implement. We chose the model from Zhiheng Huang and colleagues [27].

Recurrent Neural Networks (RNN) have been successfully deployed in the past to tackle a variety of tasks. Named Entity Recognition, as a sequence tagging problem, is also within the capability of such architecture. The problem that an RNN suffers from, is usually the vanishing gradient problem, which makes the network lose the error backpropagation (signal) over time.

This is fought by various approaches, one of them is also the deployment of LSTM [46][47] cells, which mitigate the vanishing gradient problem, by replacing the hidden layer with a number of LSTM cells.

The network topology of an RNN is, as follows: input layer \mathbf{x} , hidden layer \mathbf{h} and output layer \mathbf{y} . The \mathbf{x} keeps the same dimensionality, as the input feature size. The output layer \mathbf{y} , represents the probability of the distribution of the of labels in time \mathbf{t} . The output layer has also the same dimensionality as the size of labels. The RNN has also, in comparison to feedforward network, a connection from the past network state of \mathbf{h}_{t-1} to the present \mathbf{h} . The \mathbf{W} , \mathbf{V} and \mathbf{U} are weight matrices of the connections, computed at training time.

The value of hidden layer \mathbf{h} , in time \mathbf{t} is

$$\mathbf{h}(\mathbf{t}) = \text{sigmoid_activation}(\mathbf{U}\mathbf{x}(\mathbf{t})+\mathbf{W}\mathbf{h}(\mathbf{t}-1))$$

And the value of the output layer is

$$\mathbf{y}(t) = \text{softmax_activation}(\mathbf{V}\mathbf{h}(t))$$

The LSTM memory cell is implemented as follows [27]:

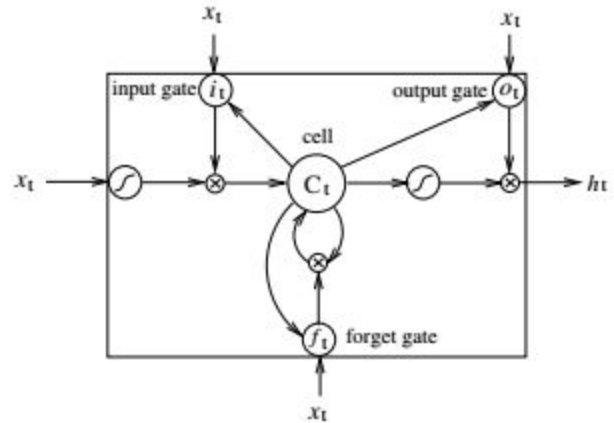
$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f)$$

$$\mathbf{c}_t = \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t\tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_o)$$

$$\mathbf{h}_t = \mathbf{o}_t\tanh(\mathbf{c}_t)$$



Where \mathbf{i} is the input gate, \mathbf{f} is the forget gate, \mathbf{o} is the output gate and \mathbf{c} is the cell vector. It is better to try to figure out the inner workings of the LSTM unit, by imprinting the picture next to the implementation definition from Zhiheng Huang and colleagues [27].

σ denotes a logistic sigmoid function. The subscript indices of the weight matrix represent the gate matrices. \mathbf{W}_{cf} would be the cell-forget gate matrix.

Bidirectional LSTM network contains two layers of LSTM units between \mathbf{x} and \mathbf{y} layers, and the network sees is able to make use of past features, by utilizing the forward states as well as backward states for future features.

3.4. Sentiment Analysis

The idea of CNN is usually coupled with computer vision, or image classification as the idea behind the network is biologically inspired. Namely by the mammalian visual cortex. In the more recent history, it also started to be applied to NLP problems, especially now, that we can represent the words with n-dimensional word-vectors. The layers of this network are following.

The first layer is an embedding layer, which contains the word-representations in form of concatenated word vectors. Keras enables this layer to be trainable as well, so we utilised this later on, for testing purposes.

To create this layer, let \mathbf{x}_i be the that the n-dimensional word vector representing the i-th word in the sentence. Then we have a concatenation of all the word vectors, in one sentence with its respective padding, to the length on the longest observed sentence,

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \dots \oplus \mathbf{x}_n,$$

Further, let the \mathbf{w} , be the filter, in a convolutional operator, which is applied to the window or words with the length \mathbf{h} . So a feature \mathbf{c}_i generated from a window with length \mathbf{h} . A concatenation of words is represented as $\mathbf{x}_{i:i+h-1}$. \mathbf{f} stands for a non linear functions and \mathbf{b} for bias, if applicable.

$$\mathbf{c}_i = \mathbf{f}(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + \mathbf{b})$$

Such filter is applied on each possible window \mathbf{h} in the sentence, producing a feature map.

$$\mathbf{c} = [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{n-h+1}]$$

After creating several feature maps, in our case, the exact number was 16 with \mathbf{h} of 4, the output is fed into a global max pooling layer over time [39], which selects the one most prominent of

the features in the feature map. This is then fed into a fully connected layer with a dropout, as overfitting protection, and softmax output, for classification.

We used the most obvious metrics, as this task was a binary classification and accuracy makes perfect sense, as the predictions might only be right or wrong and there is no middle ground.

3.5. Technical Details

These scripts were tested on Python3.5, on Windows x86_64 and Python3.6 on Linux operating systems with the same memory bus width. The scripts will not work on Python2.7 or older.

3.5.1. Utilised python libraries

The libraries used to complete this work contain mainly the following set, but the dependency set is much larger (as is obvious in the req_*.txt mentioned later on). For the running these scripts, following libraries are required:

- keras
- ntkl (SA specific)
- tensorflow (as keras backend we tested with -gpu)
- gensim
- scikit-learn
- scrapy
- beautifulsoup

Additional requirements may apply and problems might occur. These are described in the later chapter “Technical Details”. Also, worth mentioning is the fact that all of the training was done on a CUDA capable video card, so for the training, we strongly recommend to start it in a CUDA capable environment with the respective video card. There is also an alternative for AMD cards through HIP-ify, but was not tested. We do not know how long the scripts would run on a CPU.

It is strongly recommended to install this in a python virtual environment. The libraries are for both of the scripts, there was only one problematic library, and that being networkx, as hyperas required specific version, which is not automatically downloaded.

The libraries necessary for replication our implementations are in the root folder of this thesis called req_lin.txt and req_win.txt. Both these files are required to be installed by:

```
pip install -r req_{OS}.txt
```

Where {OS} is the OS where one wishes to test the script.

There were several issues one might encounter while trying to run any part of the code. First of all, if the target machine is windows with command line, one needs to fix the issue, that it has, and make it utilise the win-unicode-console python library for better UTF-8 support. Otherwise, any use of an unicode only character will break the workflow and result in error. Where this was observed, the additional import was added. Alternatively, maybe a better solution would be to use the most recent PowerShell Core, as it defaults to UTF-8, at long last.

For Windows, to have everything working, one needs to have Windows Subsystem for Linux (WSL) installed, otherwise, the shell scripts for the data scraping will not work.

Another big issue for most of the computers is the extreme RAM utilisation, that is a direct result of pretrained word2vec vectors, which take up to 7.5GB of space when loaded into the computer's memory. As we rely heavily on the “vocabulary” it provides, it is virtually impossible to run it without at least 12GB or RAM. We tested it on a 8Gb linux machine with no

success, as it can not break the memory file into smaller chunks and utilise the virtual memory/swap space. It is dependent on the embedding building, if there are too many words, it will easily grow to unrestrained

3.6. Usage/Documentation

In this chapter, I would like to log the different uses of various standalone parts of the application we have implemented. This should be considered a short version of the documentation/how-to. I will go from “fetcher” scripts to training and explain how to run each part of the code separately, if needed.

All of these command must be run from the project root, with the specified libraries installed and functional.

3.6.1. Scraping

As mentioned previously, in the scraping part, we deploy the spiders, to get the web links for us. These spiders are located in the sub-directory “./crawl” of our project. Each web-page has its own spider. Each spider need to be set up concretely for its destination web, it should crawl. All the spiders are located in

```
./crawl/crawl/spiders/<web_destination_name>.py
```

If there is a need to test the spider, the directory from which it should be started is the first ./crawl directory. This directory tree was created be scrapy. The exact command to start the crawling process is

```
scrapy runspider <relative path to the spider>
```

This command will, if called by the main scraping script “scrape.sh” result in a log file and some <timestamp>-<web_destination_name>.csv in “./scraped” that contains all the current news headlines and links from the selected web pages. Otherwise it just shows the raw output of the said spider.

The last part of the scraping is the data “fetching” script, that removes the unnecessary HTML tags, etc. This script requires an argument to be passed down, as it word as mediator between scraping and the resulting text.

```
python .\fetch\fetch-<web_destination_name>.py ARG1
```

Where ARG1 is the URL address to the article we want to download and process, so that we may get a simple and clean data in form of text only.

This whole workflow is integrated in the “scrape.sh” script found. It also incorporates the user input regarding the identity of topic in the two journals, so we may scrape the correct articles. Later on we plan on substituting the user input with a network capable of analyzing the topics from the headline. This is mentioned the Future Works chapter later. The starting of the script is straightforward, as it requires only the following command to run.

```
./scrape.sh ARG1?
```

If everything goes as planned, the user is then presented with the choice of articles, from which he has to choose articles commenting on the same event or affair. ARG1? stands for an optional search string, that can be used if there is a certain word involved and the script is supposed to look for it.

3.6.2. Network training

The training of the networks is as simple as to run a python script with no arguments whatsoever. On one side, it is convenient, on the other, it makes it more time-consuming to adjust the core code. The code has to be run from the document root, because relative paths were used, also, there are several implementations in terms of layers and word embeddings. To start the training the following command is required:

```
python .\training\SA\sa_w2v.py
```

respectively

```
python .\training\NER\ner_BiLSTM.py
```

Which starts the training from scratch with the preset hyperparameters. There are also various other scripts, that can be used to train hyperparameters, with the help of hypers or just different network architectures.

The word2vec file, with pre-trained word embeddings has to be placed on the document root, so the training scripts can access it, if required.

4. Results and Experiments

As previously mentioned, this chapter contains the various results we encountered, while on the search for semi optimal hyper-parameters and network layers. Chronologically, first we discuss the results of NER, then proceed to SA.

4.1. Named Entity Recognition

For named entity recognition task we implemented the LSTM and Bi-LSTM model as described in chapter 3.2. As mentioned in chapter 2.1.2, in order to compute the final accuracy over the selected dataset, we did use the K-Fold and Monte-Carlo cross validation. In the former, the K was set to 10, and for the latter, we used 20 consecutive runs.

From pragmatic point of view, we were mostly interested in accuracy, as this metrics is straightforward, but does not answer the penalty for a wrong prediction in a sequence, as, obviously, a partial prediction in a sequence, makes the whole prediction of said sequence wrong.

In the end we reimplemented the metrics for Keras (F1) but it works only if we use the batch size of 1, which makes the training time explode 20 fold compared to that of normally witnessed previous training time. It was actually the reason, why Keras developers decided to use a different default metrics. (In before version 2.0, the default metrics were F1 score, precision, and recall, per batch.)

Result from the experiments are displayed in the table 1. The Dat1 was the data, we described in chapter 2.1 for our NER task. The experimental setup on Dat1 was surprising, as, after 25 epoch of training, we were able to get above 99%. From practical viewpoint, this makes the data useless in further training, as any improvement would be immeasurable. (Or, at least

difficulty so.) We tried to improve the model, by adding another layer of BiLSTM, but the only difference was that it converged slightly faster to the final 99.64%

table 1.

Experimental setup	F1 score
LSTM Dat1 w2v	98.49%
Bi-LSTM Dat1 w2v	99.64%

As the data was rather small we were surprised that this network managed to get such high F1 score. We assume, that the data contained easily recognisable patterns which we were able to extract without much problem. Hence we need to find a better data for training purposes. I touch this topic in Future work chapter more.

We used the aforementioned word embedding from freely available w2v, this might have been the culprit behind the way-too-good performance of the selected network. As a result, we moved on to the SA.

4.2 Sentiment Analysis

For SA we implemented a Convolution Neural Network with word embedding and Max-over-time pooling, as described in the chapter 3.3. Afterwards, we tested several variations of data as well as architectures. Most notably, after hyperparameter training on our initial dataset, we presented the network with different embeddings of the words. For one variation, we initialize a random uniform vector, per word, and then train it via a word embedding layer in our network.

Results from experiments on SA are displayed in table 2. The formal description of the various implementation of the our model, are as follows. CNN SA is the core description, REmb and w2v are word representations which were used to various degree of success, Dat stands for data. The number 1 indicates the data first used by Bo Pang and Lillian Lee [29], which are the sentence log reviews, or number 2 stands for imdb dataset. The rest of the description in case of w2v differentiates between trainable and not trainable embeddings and the representation of unknown words.

table 2

Experimental setup	accuracy
CNN SA REmb Dat1 1 layer	75.36%
<i>CNN SA REmb Dat2 1 layer (MC CV 20x)</i>	<i>85.28%</i>
CNN SA w2v-stat-unk-zero Dat1	77.28%
CNN SA w2v-stat-unk-random Dat1	77.50%
CNN SA w2v-dyn-unk-zero Dat1	78.39%
CNN SA w2v-dyn-unk-random Dat1	78.77%

There are two winners in two categories in table 2, at least for our application of this model and its abilities. As mentioned previously, we deployed 2 different word embeddings, which were, performance-wise, quite distinct. This was to be expected, as the source paper claimed even greater disparity. The first was random uniform word embedding and the second was pre-trained w2v, as described in the chapter 3.1.

Also, we experimented with another dataset, namely the imdb, which is included in keras for such purposes. We did use only 300 dimensional random word embeddings, for this purpose, because if we would have tried to map w2v onto this amount of data, the memory would not be sufficient. We assumed, that the rather poor performance of our network, compared to other datasets [26] was caused by the data-set itself, rather than the inability of the network. The result, found in the Table 2 confirms this assumption, as we did little to no fine tuning, when compared to all the other models.

During the training, we also encountered massive overfitting, as observable on picture 1, on the dataset labeled as dat1, so we had to deploy a quite aggressive dropout coefficient, as seen in table 3. The dropout was insufficient, so we had to adjust the learning rate and decay, to find the best possible configuration of parameters. In our case, we used an adam optimiser with the learning rate of 0.0005 and decay of 0.0058. The overfitting was absent on the Dat2.

The difference between letting the network learn and disable adjusting of the embeddings made an observable difference of 1.11% and 1.27% respectively

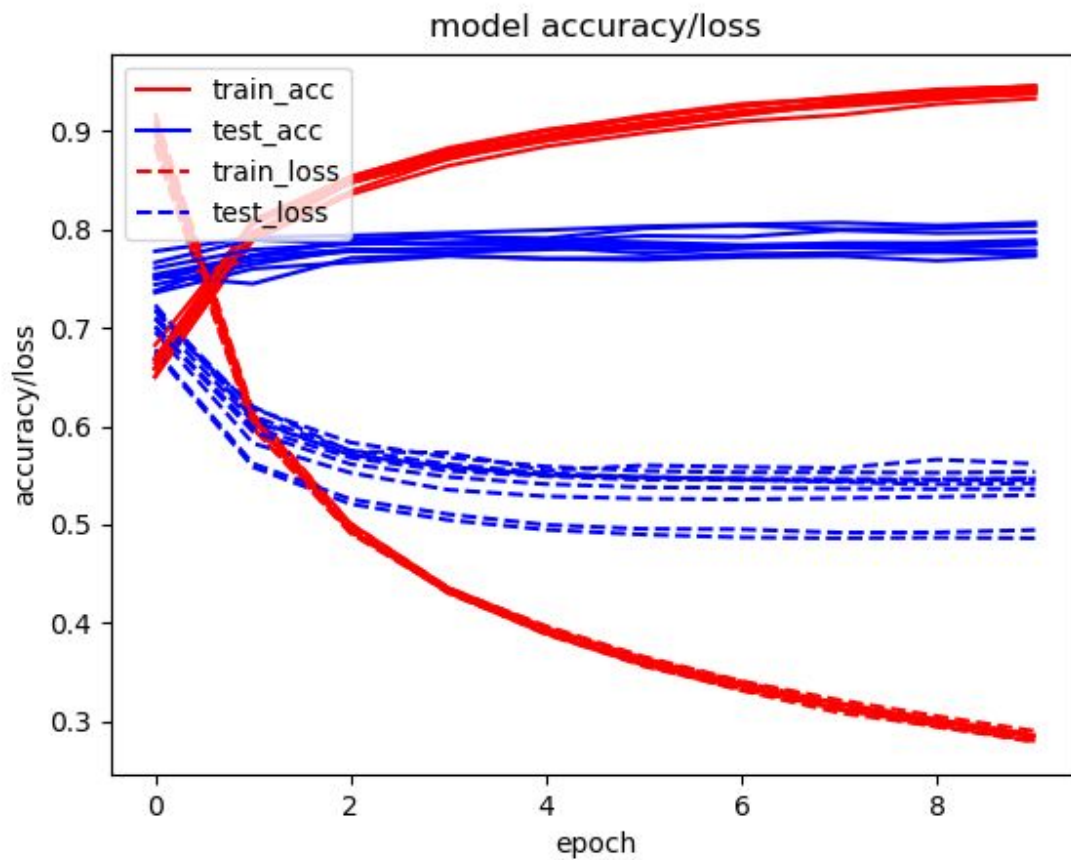
The difference between the implementation, which considered all unknown words to be meaningless and created an embedding of zeroes versus the approach that assigned, for each unknown word, a random normalised word vector was also observable. The numerical values were 0.22% and 0.38%.

The rest of the hyperparameters were only used, if we utilised the word embedding layer (REmb) and not the w2v representation.

table 3

Hyperparameter	Hyperparameter space	Final value
dropout	[0.1, 0.2, 0.3, 0.4, 0.5]	0.5
conv filters	[8, 16, 32, 64]	16
conv filter size	[4, 8, 12, 24]	4
REmb features	[300, 1500, 4000, 6000]	6000
REmb dimensionality	[64, 128, 256]	128

picture 1



As we stated earlier, we had 2 different winners in the table 2. The first one is the most capable model, we succeeded in training. with 78.77% accuracy. The only downside of this model, is the necessity to keep and train new “dictionary” file, because the word2vec representations.

The picture one represents the best model, we observer with our data and hyperparameters. We do not deemed necessary to include more pictures, as they are well represented by picture 1. The rest of the plots can be found inside the files of this thesis, in the folder ./runs_log.

5. Conclusion and Future work

In this final chapter, we have the last observations, regarding this thesis, and some ideas, how to augment this work even further.

5.1. Conclusion

We report, that we successfully made an application capable of downloading live data from the internet and train our networks containing our chosen ML algorithms and store the resulting networks for later use. Furthermore we explored my ideas, proved some of them useless or unattainable, but nonetheless the ground concept proved to be functional and I may proceed to expand it further.

While we could not report the same findings, as the original paper[26], the help of pre-trained word-vectors did indeed improve the final accuracy of the network on our dataset, as reported by the original paper.

As for NER, we found that the Bi-LSTM network worked really well, for our dataset, but there is still more room for improvement. Especially in finding a better suited data for training, which is longer, and more complex, as the currently used has problems to show the limits of chosen architecture.

Luckily, both of the used models could be easily swapped in the code, so we can improve on the work further with relative ease.

Worth mentioning is also the ability to work with data extraction tools for web more effectively and, I will admit, a newfound fascination with unexplainable word representations in form of w2v or glove.

5.2. Future work

This chapter is dedicated to our future plans with this work. It contains dreams, that may or may not come into fruition in the distant, or not so distant future.

5.2.1. New datasets

The first and probably generally most useful work, from the view of the ML community, would be to put this work onto a webpage with automatic downloading and classifying capabilities. It's not because this work in itself can be considered that great, but if we set it up, and people start to use it, we could ask them to tell us what they think (we are aware of the fact that approximately 90% of netizens will not answer) about whether the Named Entities are correctly classified as such, or if the prediction of the perceived sentiment is correct in their opinion. This way, we could get a great amount of tagged data that could be used in future projects, be it NER, SA or aspect-level SA.

5.2.2. Aspect Level approach

This approach could provide us with a more fine-grained data. As it's simpler to illustrate, I will use following sentence to do so: "The phone is great, but the battery life is terrible." Now consider the word "phone" and "battery" aspects of the sentence. If I would want to classify the sentence of such type for both of the aspects, in our current approach, we would get the full sentence for both aspects into the corpus for sentiment analysis. Implementing this correctly would lead to cutting the sentence into two sentences, one for each aspect, and assigning the relevant part into the corpus for SA, which would mitigate this problem. [n2]

5.2.3. w2v prediction

This idea was one of the earlier ones. It might be useful for Named Entities that are difficult to classify for sentiment analysis to be substituted by a word, predicted by word2vec [28]. This could prove useful in a situation, when the Named Entity was used in a sentence to imply something negative or positive. Or, if the sentence structure was used for this implication. Even though this should be more obvious to the classifying system, even without the prediction from w2v. I am hoping, that w2v would predict a word with strong “sentiment value”. To try to illustrate, “{ORG} tried to ruin our reunion.” where {ORG} stands for a NE of an organisation. Now let's suppose that the skip-gram model was trained upon data, that saw the words “evildoers” most often, in similar context. Which would obviously put such sentence into the more negative sentiment.

5.2.4. Sarkasm

Sarkasm is a very difficult problem to solve, as it is not obvious to people either. But such sentence might be perceived as inverse sentiment for the less sophisticated SA approaches. This problem is too far from solved, but would be useful to try and train on a additional ML algorithm.

5.2.5. Pronouns

Another addition could be, to find a way to discover the pronouns used in text and couple them with its respective Named Entity. To substitute the words like “it” and “he” or “she” with the NE it stands for, which would, connected to the Aspect Level Sentiment Analysis, provide additional data to classify, making the final SA easier, thanks to a bigger final.

5.2.6. HTML Tags and article images

Yet another addition to the data that is minable from the web-based articles are the HTML tags in the text. For example bold or italics in the text make a way to use html tags to scrape the importance of the data ``, `<i>`, links etc. Other than that, most articles have images assigned to them. Images usually carry some sentiment and, as a result, they should not be omitted in the final sentiment score either. The problem here would be to assign the correct image to its respective entity, if it had one and analyze the image sentiment, from which, neither of these actions, are easy.

5.2.7. Topic analyzer

This addition to the web crawler we created, would solve the requirement of the scraper script for user input, and would make the script fully autonomous. As currently we depend on a user input, which tells us whether the articles are about the same occurrence, or not. This would allow us store and analyze great amount of data asynchronously, and only later depend on user input.

5.2.8. Improving the code

At last, but not least The last personal choice of improvement would be improve the codebase, as, in current state, it is somewhat unsightly and spaghetti-like to my great sorrow. It would be nice to have time to improve the modularity of the code to build more library like approach with greater variability and code modularity. Also the aforementioned Iterated Dilated Convolution [n3] looks interesting at least. First to understand, then try to implement.

Another fancy step forward, is to build a word representation, that does not strain the system memory in such a way, as the current implementation. The solution to this would be, to use the word embeddings only on the embedding keras layer. As this works as a vocabulary, instead of building the whole sentences out of concatenated 300 dimensional word vectors. Regretfully, this idea came too late to change anything.

Bibliography

- [1] Cheryan, Sapna, and Galen V. Bodenhausen. "When positive stereotypes threaten intellectual performance: The psychological hazards of "model minority" status." *Psychological Science* 11.5 (2000): 399-402.
- [2] Simon, George K. *In sheep's clothing: Understanding and dealing with manipulative people*. AJ Christopher & Company, 1996.
- [3] "Truth." *Merriam-Webster*, Merriam-Webster, 24 December 2017 Updated. 08 January 2018 Accessed. www.merriam-webster.com/dictionary/truth.
- [4] Faye, Jan, "Copenhagen Interpretation of Quantum Mechanics", *The Stanford Encyclopedia of Philosophy (Fall 2014 Edition)*, Edward N. Zalta (ed.), 03 May 2002. 08 January 2018 Accessed. <https://plato.stanford.edu/archives/fall2014/entries/qm-copenhagen/>.
- [5] "Sentiment." *Merriam-Webster*, Merriam-Webster, 08 January 2018 Updated. 08 January 2018 Accessed. <https://www.merriam-webster.com/dictionary/sentiment>.
- [6] LeDoux, Joseph. "The emotional brain, fear, and the amygdala." *Cellular and molecular neurobiology* 23.4-5 (2003): 727-738.
- [7] Wolf, Uri, Mark J. Rapoport, and Tom A. Schweizer. "Evaluating the affective component of the cerebellar cognitive affective syndrome." *The Journal of neuropsychiatry and clinical neurosciences* 21.3 (2009): 245-253.
- [8] Chilmonczyk, Zdzisław, et al. "Functional selectivity and antidepressant activity of serotonin 1A receptor ligands." *International journal of molecular sciences* 16.8 (2015): 18474-18506.
- [9] Berridge, Kent C., and Terry E. Robinson. "What is the role of dopamine in reward: hedonic impact, reward learning, or incentive salience?." *Brain research reviews* 28.3 (1998): 309-369.
- [10] MacLean, Paul D. *The triune brain in evolution: Role in paleocerebral functions*. Springer Science & Business Media, 1990.
- [11] Selye, Hans. "The physiology and pathology of exposure to stress." (1950).
- [12] Carretié, Luis, et al. "Automatic attention to emotional stimuli: neural correlates." *Human brain mapping* 22.4 (2004): 290-299.
- [13] Dolcos, Florin, Alexandru D. Iordan, and Sanda Dolcos. "Neural correlates of emotion–cognition interactions: A review of evidence from brain imaging investigations." *Journal of Cognitive Psychology* 23.6 (2011): 669-694.
- [14] Kemp, Simon. "The incredible growth of the internet over the past five years – explained in detail." *Thenextweb.com*, 6 Mar 2017. 08 January 2018 Accessed. thenextweb.com/insider/2017/03/06/the-incredible-growth-of-the-internet-over-the-past-five-years-explained-in-detail/.
- [15] Abadi, Martin, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems. 2016." *arXiv preprint arXiv:1603.04467* (2016).
- [16] Chollet, François. "Keras: Deep learning library for theano and tensorflow." URL: <https://keras.io/k> (2015).

- [17] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *Journal of Machine Learning Research* 12.Oct (2011): 2825-2830.
- [18] Bergstra, James, et al. "Theano: A CPU and GPU math compiler in Python." *Proc. 9th Python in Science Conf.* 2010.
- [19] Turing, Alan M. "Computing machinery and intelligence." *Mind* 59.236 (1950): 433-460.
- [20] Bush, Vannevar. "As we may think." *The atlantic monthly* 176.1 (1945): 101-108.
- [21] Freitag, Dayne. "Machine learning for information extraction in informal domains." *Machine learning* 39.2 (2000): 169-202.
- [22] Yuan, Xingdi. "Named-Entity Recognition using Deep Learning." *Eric Yuan's Blog*. 23 March 2015. 07 January 2018 Accessed. http://eric-yuan.me/ner_1/
- [23] Atkinson, Adam. Deep-Named-Entity-Recognition, *Github.com*, 20 Sept. 2016, 07 January 2018 Accessed. <https://github.com/aatkinson-old/deep-named-entity-recognition>.
- [24] "Language-Independent Named Entity Recognition (I)." *CoNLL*. conll.org, 08 May 2005 Edited. 07 January 2018 Accessed. <https://www.clips.uantwerpen.be/conll2002/ner/>.
- [25] "Language-Independent Named Entity Recognition (II)." *CoNLL*. conll.org, 05 December 2005 Edited. 07 January 2018 Accessed. <https://www.clips.uantwerpen.be/conll2003/ner/>.
- [26] Kim, Yoon. "Convolutional neural networks for sentence classification." *arXiv preprint arXiv:1408.5882* (2014).
- [27] Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging." *arXiv preprint arXiv:1508.01991* (2015).
- [28] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).
- [29] Pang, Bo, and Lillian Lee. "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales." *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005.
- [30] Chen, Minmin, Kilian Q. Weinberger, and John Blitzer. "Co-training for domain adaptation." *Advances in neural information processing systems*. 2011.
- [31] Ben-David, Shai, et al. "A theory of learning from different domains." *Machine learning* 79.1 (2010): 151-175.
- [32] Dubitzky, Werner, Martin Granzow, and Daniel P. Berrar, eds. *Fundamentals of data mining in genomics and proteomics*. Springer Science & Business Media, 2007.
- [33] McLachlan, Geoffrey, Kim-Anh Do, and Christophe Ambroise. *Analyzing microarray gene expression data*. Vol. 422. John Wiley & Sons, 2005.
- [34] Rajaraman, A. Ullman. "JD (2011)." *Data Mining*." *Mining of Massive Datasets*: 1-17.
- [35] Bird, Steven, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. "O'Reilly Media, Inc.", 2009.
- [36] Kripke, Saul. "Naming and Necessity (1972)." Cambridge, Mass.: Harvard (1980).

- [37] Nadeau, David, and Satoshi Sekine. "A survey of named entity recognition and classification." *Linguisticae Investigationes* 30.1 (2007): 3-26.
- [38] Liu, Bing. "Sentiment analysis and opinion mining." *Synthesis lectures on human language technologies* 5.1 (2012): 1-167.
- [39] Collobert, Ronan, et al. "Natural language processing (almost) from scratch." *Journal of Machine Learning Research* 12.Aug (2011): 2493-2537.
- [40] Sahlgren, Magnus. "The distributional hypothesis." *Italian Journal of Disability Studies* 20 (2008): 33-53.
- [41] McCormick, C. "Word2Vec Tutorial Part 2 - Negative Sampling" (2017).
<http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/>
- [42] Xu, Qing-Song, and Yi-Zeng Liang. "Monte Carlo cross validation." *Chemometrics and Intelligent Laboratory Systems* 56.1 (2001): 1-11.
- [43] Strubell, Emma, et al. "Fast and accurate entity recognition with iterated dilated convolutions." *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017.
- [44] Chiu, Jason PC, and Eric Nichols. "Named entity recognition with bidirectional LSTM-CNNs." *arXiv preprint arXiv:1511.08308* (2015).
- [45] Wang, Peilu, et al. "A unified tagging solution: Bidirectional LSTM recurrent neural network with word embedding." *arXiv preprint arXiv:1511.00215* (2015).
- [46] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.
- [47] Graves, Alex, and Jürgen Schmidhuber. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures." *Neural Networks* 18.5 (2005): 602-610.
- [48] Liu, Bing. "Sentiment analysis and opinion mining." *Synthesis lectures on human language technologies* 5.1 (2012): 1-167.
- [49] Strubell, Emma, et al. "Fast and accurate entity recognition with iterated dilated convolutions." *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017.