Aarhus University

Bachelor Project

Advisor: Carsten Eie Frigaard

BA project no.: 2021E79

Mitigation of stigma, prejudice and discrimination during early stages of recruitment using machine learning - Annex Report

Navn:	*Underskrift:*	*Studienr.:*
Morten L. Hansen	Morten Lerschow Hansen	201805227
Rasmus F. Sørensen	Rasmus Føgh Sørensen	201806493

Submission deadline: Wednesday 15. December 2021 12:00

Contents

Preface

This report serves as an appendix to the main report, *Mitigation of stigma, prejudice and discrimination during early stages of recruitment using machine learning - Main Report.*

This appendix report is executed by Rasmus Føgh Sørensen and Morten Lenschow Hansen, both students of Aarhus University School of Engineering's BSc Software Engineering, in cooperation with the assigned advisor Carsten Eie Frigaard.

The project is to be handed in December the 15th, 2021, and is to be orally defended in Janurary 2022.

Conventions used

The following typographical conventions are used:

Italic
: indicates quotes, references equations and initial third party mentions.

Bold
: indicates new names and terms, except when used in figures, tables and listings, defined in *Glossary*.

Glossary

Here explanations of terms and abbreviations used in the report can be found.

Term	**Explanation**	**Comment**
Gendered wording	A context dependent measurement of a word or phrase's masculine and feminine polarity.	Masculine examples: 'he', 'competitive', 'active', 'confident', 'We are looking for a strong...', 'candidates who are aggresive' Feminine examples: 'she', 'support', 'nurture', 'communicate', 'nurture and connect with customers', 'build relationships' [1]
SaaS	Software as a service	
A	Applicant	Used consistently throughout report.
E	Employer	Used consistently throughout report.
Linguistic analysis	Analysis of machine learning algorithm to carry out named entity recognition.	
Sentiment analysis	Analysis of machine learning algorithm to carry out sentiment analysis.	
LA	Abbreviation of linguistic analysis.	
SA	Abbreviation of sentiment analysis.	
ML Algorithms	Used as a contraction of linguistic analysis and sentiment analysis	
NLP	Natural Language Processing	
NE	Named entity	
ML	Machine Learning	
NER	Named Entity Recognition	
XAI	Explained AI	In contrast to black box in ML, XAI is an implementation used to understand why an ML algorithm arrives at its conclusions.
POS	Part-of-speech	
Corpus	Dataset	In ML datasets are often refered to as corpus, but in general terms it is a collection.
GMB	Groningen Meaning Bank	Provider of dataset for the linguistic analysis.
IOB	Inside, outside, beginning	Named entity format.
SOS	start-of-string	
EOS	end-of-string	
RNN	Recurrent neural network	
LSTM	long-short-term-memory	
biLSTM	bidirectional LSTM	

Demographic characteristics & Demographic features		Are used interchangeably.
Employer, organization & recruiter		Are used interchangeably.

Table 1: Glossary explaining terms and abbreviations used in the report.

Version history

Date	Version number	Initials	Comments
31/08-21	1.0	MH	Initial iteration of annex report.
03/09-21	1.1	RFS	Added *Introduction*.
06/09-21	1.2	RFS	Added *Problem statement* and started on *Requirement specification*.
06/09-21	1.3	MH	Added *Conceptualization & Requirements* and subsection *Scope*.
08/09-21	1.4	MH & RFS	Finished system specification 1.0
13/09-21	1.5	RFS	Finished revised Introduction 2.0
13/09-21	1.6	RFS	Finished revised System specification 2.0
13/09-21	1.7	RFS	Added Analysis section, *Open-source NLP libraries* subsection and the start of OCR subsection
01/11-21	2.0	RFS & MH	Updated sections mentioned above to have a more scientific approach. Removed OCR subsection
15/11-21	2.1	RFS & MH	Added *Datasets* subsection to *Analysis*.
19/11-21	2.2	RFS	Added subsection *Machine learning frameworks* to *Analysis*.
22/11-21	2.3	RFS	Added subsection *Neural network architectures* to *Analysis* and *State-of-the-art*.
24/12-21	2.4	RFS & MH	Started on subsection *Methodology* for sections *Linguistic Analysis* and *Sentiment analysis*. Added subsection *Linguistic features* to *Analysis*.
29/12-21	2.5	RFS	Started on subsection *Implementation of neural networks* for section *Linguistic analysis*. Added section *Project description*.
30/12-21	2.6	RFS & MH	Started on sections *Discussion* and *Results*.
03/12-21	2.7	MH	Added section *Humanistic and ethical insight*.

Table 2: Version history

1 Introduction

1.1 Description of problem

People are subject to discrimination in the initial process of job recruitment, especially ethnic and immigrant-origin minority groups are disadvantaged to access the labour market [2]. The discrimination is typically based on demographic characteristics, socioeconomic perceptions of the applicant [2], or heuristics made by the employer [3]. Studies with correspondence experiments have been carried out throughout the last two decades. A recent study carried out in the Danish labour market found that the ratio of callbacks between a Danish-sounding name and a middle Eastern-sounding name is 1.52 for the same equally qualified application, implying the minority would have to apply 52% additional applications to get the same amount of callbacks as the majority. [2]. The same study also found discrimination of intersections between gender and ethnicity statistically significant between female minorities and female majorities in female dominated occupations and especially male minorities and male majorities in male dominated occupations [4]. The empirical context can also be extended to other countries. In America, an extensive research on racial discrimination in the American labour market in 2003 found, that the ratio of callbacks between a white-sounding name and a black-sounding name was 1.5 in favor of a white-sounding name [3].

The common denominator between mentioned studies are the employers performing the screening process of the applications, either manually by screening the application or automatically by use of a hiring system. A model explaining this screening process behaviour could be lexicographic search by the employer [3], where applicants are discarded after quick heuristics, based on names or photos, leading to cognitive bias instead of rational choices. Also, if an employer uses an automated hiring system, it shows that the underlying system can have special bias towards one gender, leading to exclusion of the opposite gender [5].

These studies only show observations based on one-way communication from applicant to employer in form of job applications, but going the other way around, job advertisements tend to contain **gendered wording**, which sustains gender inequality [6]. Gendered wording is not a defined quantity, but a context dependent measurement of a word or phrase with respect to a masculine and feminine polarity. Examples of gendered wording, which causes gender bias, can be obvious gender references like he/she or fireman/firewoman, but often occurs less apparent as people of specific genders tend to subconsciously prefer certain ways of expressing themselves in terms of words and phrasing [1]. In male-dominated occupations there is a statistical significance for job advertisements containing greater masculine wording [6], which ultimately affects job seekers pursuit intentions, due to anticipated belongingness. If a job advertisement contains greater feminine wording, men tend to have higher perception of gender diversity, which subsequently lead to lower pursuit intentions [6].

All things considered, some sort of discrimination is happening all over the line, even if it being an unconscious or conscious action in a manual or automated process. As of modern days, recruitment tends to go through an intermediary, such as recruitment platforms or social media [7], where perhaps this is an evident place to take action. Co-author of *'Monitoring hiring discrimination through online recruitment platforms'*, Dr. Dominik Hangartner, said:

> *"We are optimistic that at least part of the discrimination that we document in this study can be overcome by re-designing recruitment platforms. For example, more relevant information such as a candidate's work experience and education could be placed at the top, and details which might indicate ethnicity or gender, such as name or nationality, could appear much lower down the CV"* [8].

1.2 Problem statement

To mitigate discrimination in the initial job recruitment process, either it being manual or automated, it requires a change of behaviour of how the two parties, applicant (**A**) and employer (**E**) respectively, approach each other. Reflecting on the problems described in *1.1 Description of problem* and what the aim of the project should be, two key points can be deduced:

Key point 1. discrimination of A in regard to demographic characteristics and socioeconomic status, and

Key point 2. discrimination of A in regard to gendered wording, both in terms of wording in job applications, potentially leading to implicit cognitive perceptions of A, as well as in job advertisements, potentially discouraging A from applying for the job.

These key points play an essential role in the cognitive perception of one another, that fosters a subconscious bias. Especially during job recruitment, where discrimination towards A based on individual bias by E shouldn't be present. Therefore, the key points are vital in suppressing discrimination during entry to the labor market, so a more diverse and inclusive field of candidates can be assembled. This leads to a more generalized problem statement, that can be shortened to:

How can an unbiased recruitment process be established between A and E, where demographic characteristics, socioeconomic status and perception of gender are disregarded, and people are instead evaluated based on qualifications and experience?

1.3 State-of-the-art

Before going in depth with the project, this dedicated section will unfold already state-of-the-art services in the industry. These services are already used by many large-scale organisations and are an integrated part of the job recruitment process, whether it be in the screening process of candidates or processing of job advertisements before publication.

1.3.1 Automated hiring systems

In the job recruitment domain, machine learning is already playing a vital part, especially in the early stages of recruitment. The most commonly applied machine learning fall into the pits of *talent recruitment*, *talent sourcing* and *candidate screening* [9]. The manual screening process of job application is considered a high-volume task for recruiters, which is why many large-scale organizations implement an automated screening process to shortlist candidates. The automated screening process performs machine learning on job applications to rank candidates based on sentiment analysis, and extracted criteria down to distance in commute and gaps in employment history [10]. However, it has recently surfaced, that such automated hiring systems are not without bias, and they can reject millions of viable job candidates, simply based on picky requirements with no regard to qualifications or experiences of the applicant[10]. Another reason of exclusion could be due to the machine learning algorithms being trained on previous job application data, with special bias towards one gender [10][5].

1.3.2 Gender-biased language recognition

Textio is a powerful service used by companies to be more inclusive and diverse when recruiting. Textio supports gender-biased language measurement, by processing text and then highlighting it and suggesting changes to help recruitment personnel come across less gender-biased, when writing job advertisements. The service is built on datasets of more than 350 million job posts and their hiring outcome [11]. With former job descriptions skewing masculine, *Johnson & Johnson* reported an increase of 9% of female applicants in 2017, by using Textio during job recruitment [12].

Thus, Textio is considered the state-of-the-art **SaaS** for sentiment analysis with regard to gender-biased polarization, and is used by many organizations such as *Spotify*, *McDonald's*, *Twitter* etc. Textio is only used by E's recruitment personnel to come across unbiased, when writing job advertisements, whereas this report's sentiment analysis aims to help both A and E.

2 Project description

This project will revolve around the key points from *1.2 Problem statement* and analyze how machine learning, specifically neural networks, can be applied to aid solving the problem stated. More specifically, this project is separated into two individual neural network analyses and implementation; the **linguistic analysis** and the **sentiment analysis**, henceforth referenced as **LA** and **SA** respectively. The LA is dedicated to *Key point 1.*, whereas the SA is dedicated to *Key point 2.*.

Each analysis will be designed with regard to their intended purpose, stated below:

The intention of the LA is to recognize demographic characteristics in A's job application, so they can be censored for the initial round of candidate screening by E. This approach is used to emphasize the skills and experiences that A possess.

The intention of the SA is to recognize gendered wording, both in job applications as well as in job advertisements, so gendered wording can be highlighted and pointed out for the author, allowing for rephrasing. This approach is used to establish a gender neutral two-way communication line between A and E, so conscious and subconscious discrimination based on wording can be mitigated.

These analyses will play a fundamental part of the project, which will have an analytical and scientific perspective, rather than the traditional system development perspective. The LA and SA will both be proof of concept neural networks, and are both based on a thorough underlying analysis of feature engineering and different types of relevant neural network architectures. Each neural network will be measured by score metrics to evaluate its applicability, and if the score metrics are found satisfactory, this project could be the foundation of future work.

3 Conceptualization & requirements

As stated in *2 Project description*, the LA and SA are proof of concept neural networks, however designed with the intended purpose of solving the key points stated in *1.2 Problem statement*. Therefore, the first part of this section will act as an exposition for how the LA and SA can be combined with the intended job recruitment context. As mentioned, this project does not contain any system development, but to provide a better understanding of how the LA and SA can be applied in the real world, they have been integrated in a recruitment platform, which is one of the many ways the results of this report can be applied to a system if the conclusion is proven satisfactory for future work. In the second part of this section, requirement specifications are specified for the LA and SA for what is considered satisfactory for future work.

3.1 Concept description in a domain context

In the domain context of a job recruitment platform, machine learning could be an integrated part of the processing of job applications and job advertisements. The LA and SA are ideally part of a processing pipeline on the recruitment platform as seen in figure 1, where the process of A uploading a job application and E uploading a job advertisement is shown. The figure also makes it clear how the communication between A and E is decoupled, since the recruitment platform acts as a mediator between the two actors. The grey areas highlighting the LA and SA is what this report revolves around, and the remaining parts are part of the domain context.

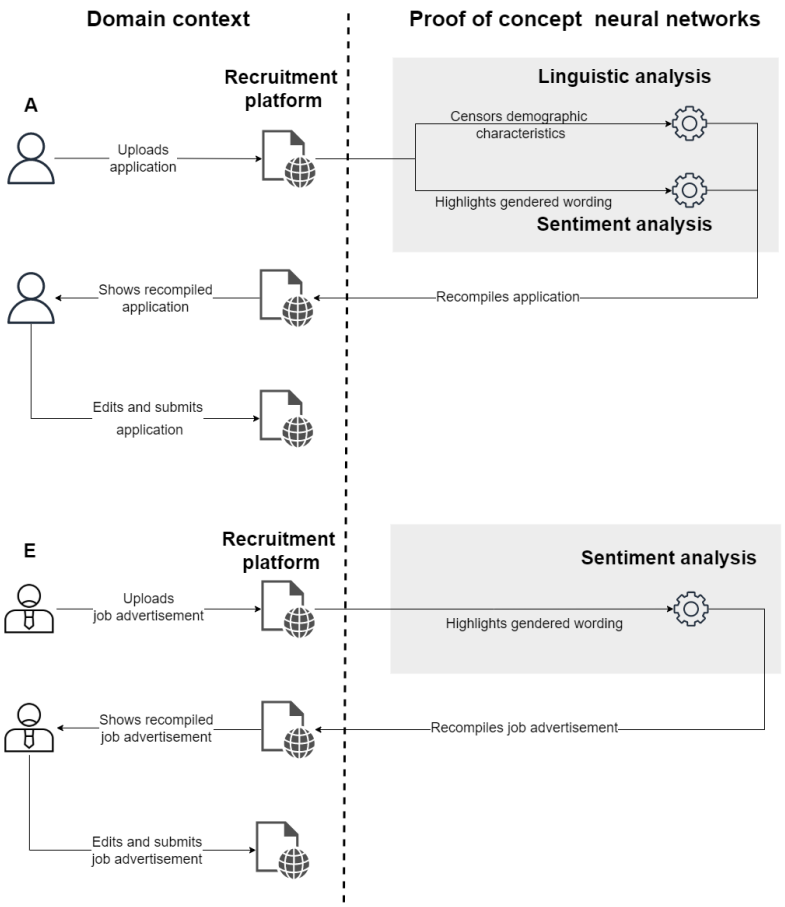


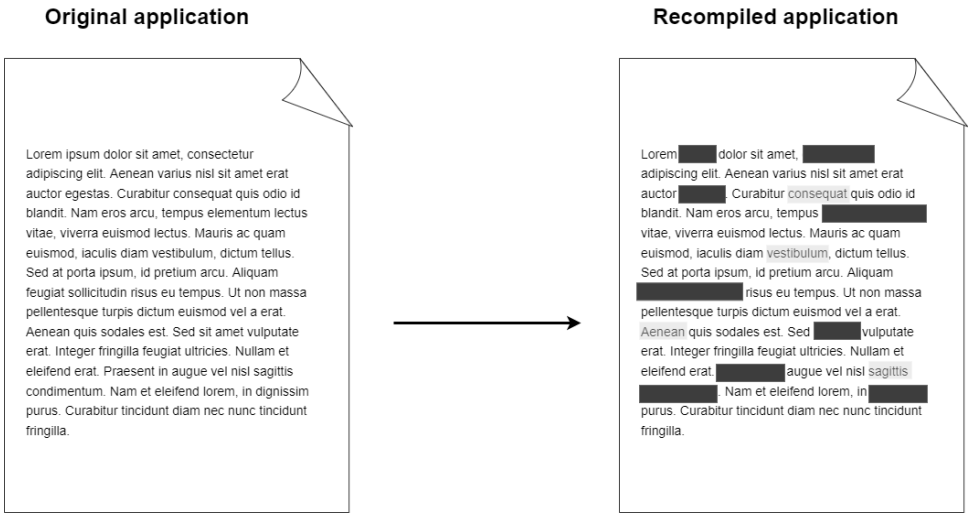
Figure 1: Rich picture showing the LA and SA in a domain context. The grey area depicts the processing pipeline which this report is intended to analyze.

3.1.1 Domain flow

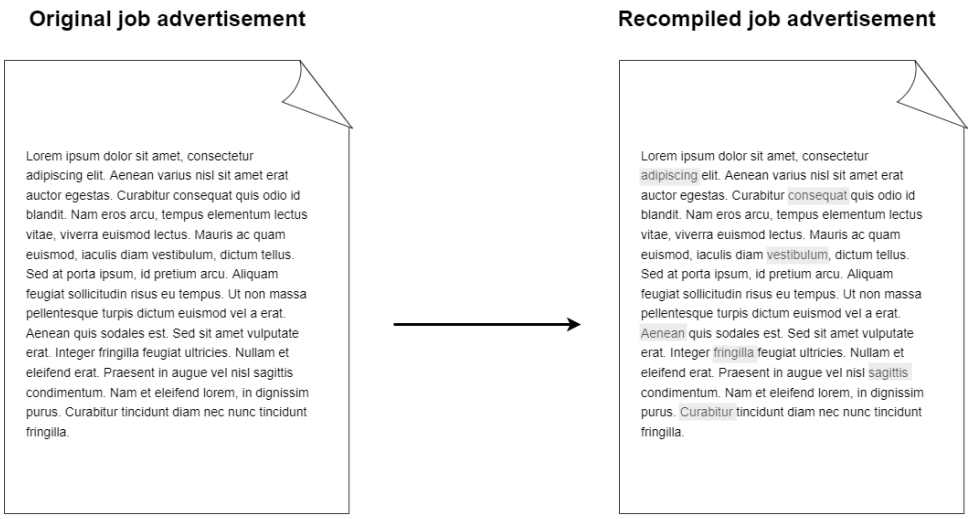
To shed some light on how the LA and SA can be part of a production-ready environment, a simple flow in the domain context, depicted in figure 1, is described below for A and E:

- As A it is possible to upload a job application to a recruitment platform, which then will be processed by the LA and SA. The processed application is then returned with censored demographic characteristics and with highlighted gendered wording for the applicant to review and potentially rephrase. At last, A can approve and submit the processed application for only E to view.
- As E it is possible to upload job advertisements to a recruitment platform. Before final submission, the job advertisement will be processed by the SA to recognize and highlight gendered wording, allowing for rephrasing by E. At last, E can approve and submit the processed job advertisement for all A's to view.

Zooming in on the conception of the recompiled documents (application and job advertisement), figure 2 shows a mock-up of how a recompiled application and job advertisement could look like. For subfigure 2a, demographic characteristics are censored and gendered wording are highlighted, whereas for subfigure 2b only gendered wording is highlighted.



(a) Recompilation of an original application written by A to a recompiled version, where demographic characteristics are censored and gendered wording is highlighted in light grey.



(b) Recompilation of an original job advertisement by E to a recompiled version, where gendered wording is highlighted in light grey.

Figure 2: Sketch of how recompilation of a job application and job advertisement could look like in a fully developed system.

As already stated, this given domain context is only to provide a conceptualization of how the LA and SA are applicable in context with a already developed system. This report will not delve deeper into the job recruitment context, which is also further specified in *3.2 Scope*.

3.2 Scope

Due to the time frame of this project, this report will from now on only focus on the LA and SA, and for that reason the domain context depicted in figure 1 is discarded. Therefore, this section is dedicated to state the scope of the LA and SA, as well as putting up formal requirements for these analyses in the shape of a MoSCoW analysis.

So by delimiting the project down to the LA and SA, the scope of these can be outlined as:

- Perform natural language processing, **NLP**, to conduct a linguistic analysis of sequences of words to classify words as named entities, **NE**'s (further described in *4.1.1 Named entity recognition*).
- Perform NLP to conduct a sentiment analysis of sequences of words to classify sentences based on gender-biased polarization.

Both analyses will be implementations of multiple neural networks, with respect to their individual NLP task, using supervised learning to predict a classification of their input. The requirements of each analysis are stated in the following section.

3.3 Requirement specification

This section will cover functional requirements and non-functional requirements for the LA and SA. As described in *2 Project description*, these analyses are proof of concept and their black box requirements are depicted in figure 3.

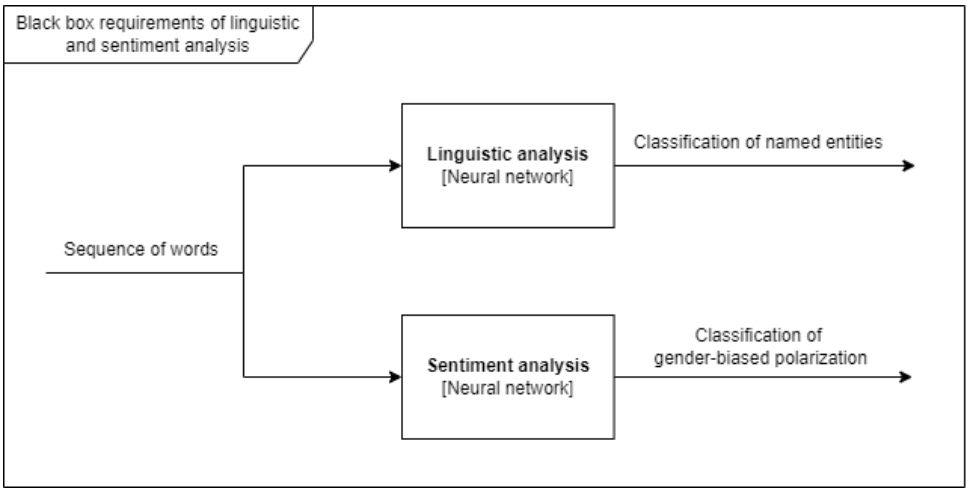


Figure 3: Black box requirements for the LA and SA.

3.3.1 MoSCoW analysis

Functional and non-functional requirements of the LA and SA are found and ranked by employing the MoSCoW method, resulting the MoSCoW analysis seen in table 3. The MoSCoW analysis is used to set requirements for both analyses, most importantly the non-functional requirements regarding score metrics, which will constitute a base for what is considered satisfactory for each analysis, in regard to further development.

Must	
1	The LA must be able to take a sequence of words as input, and output a sequence of predicted NE's.
2	The SA must be able to take a sequence of words as input, and output a binary sentiment classification based on the entire sequence.
3	The SA must be able to take a sequence of words as input, and output a multi sentiment classification based on each word in given sequence.
4	The SA using multi classification must be able to output a total sentiment value representing the gender bias of the entire sequence of words.
5	The LA must have contextual awareness when training on sequences of words.
6	The LA must be trained on an English dataset.
7	The SA must be trained on an English dataset.
8	The SA must implement the concept of Explained AI (**XAI**) to describe to the end user, what the classification was based on.
Should	
1	The LA should have a recall score metric for each named entity of at least 75%.
2	The SA should have a precision score metric of at least 70%.
3	The SA should have contextual awareness when training on sequences of words.
4	The LA should have other syntactical features as part of the input.
5	The SA should have other syntactical features as part of the input.
Could	
1	The SA could implement dependency parsing to identity which words correlate to a contrasting polarized gendered word, so gendered phrasing can be recognized.
Won't	
1	The LA won't be trained on a dataset of existing job applications.
2	The SA won't be trained on a dataset of existing job applications.

Table 3: MoSCoW method for the LA and SA, mostly in regard to non-functional requirements.

As it it already mentioned in *2 Project description*, the LA and SA will be measured by score metrics to evaluate their applicability, and the score measurements will be foundation for potential future work. The score metrics are some of the standard metrics of neural networks, which are precision, recall and F1-score. The trade-off between precision and recall is important for both analyses, since each metric plays a specific role in the outcome of the predictions. The LA requires good scoring recall measurements of each classification of a NE, since it remains important as for the job recruitment context, that as many words representing demographic characteristics and socioeconomic status are recognized (respecting *Key point 1.*), and that the margin of error remains as low as possible. The trade off is then precision, which is a compromise of the high recall, and it is by then accepted that rather censoring one too many words giving away demographic characteristics and socioeconomic status.

Contrary to the LA, the SA requires good scoring precision measurements, since it remains important as for the job recruitment context, that rather one too few gender-biased sentences/words are recognized (respecting *Key point 2.*), but with the pay off of being accurate on the most polarized sentences/words.

What is worth pointing out from table 3 is the wont's, where it is stated, that the LA and SA won't be trained on datasets of existing job applications. This might seem contradictory to the intended job recruitment context, and it is in fact. This is further discussed in *4.2 Datasets*, which also is the reason the LA and SA only are proof of concepts.

4 Analysis

To follow up on the requirements defined in *3.3.1 MoSCoW analysis* and the scope of the project defined in *3.2 Scope*, this section will cover an analysis of relevant conventions used in the linguistic field, tools, frameworks and available datasets to carry out the LA and SA. Additionally, an ethical and humanistic aspect of the project is also considered. The domain for which the analyses are intended is also kept in mind, since they have to be scalable in this context.

The LA is also what one would consider named entity recognition. NER is a widely used NLP task for information extraction and is based on a one-to-one relation between a word and a named entity as mentioned in *4.1.1 Named entity recognition*. For this report, as mentioned, NER will be used to identify named entities, with the intention to anonymize A.

Sentiment analysis, also known as opinion mining, uses NLP, text analysis, computational linguistics and biometrics to identify and extract subjective information. It is widely adopted by public relations and social management to understand the social sentiment of services or products, by automatizing monitoring and efficiently understanding customer feedback, reviews and surveys. Most commonly it is used to determine the polarity of a sentence or phrase's sentiment as positive, negative or neutral [13] [14].

In this report, the SA approach will be adapted to determine the sentiment of a sentence as a gender polarity and as classification of masculine, feminine or neutral.

4.1 Linguistic features

This section will serve as an introduction to relevant linguistic conventions used throughout this report, as well as it will cover an analysis of what syntactical features are pivotal for sentence construction and could prove beneficial as features for training purposes.

4.1.1 Named entity recognition

Named entity recognition, **NER**, is an annotation of words as entities describing the words as the object they represent such as time, organization, person, money etc. There is no specific set of entities, as this is context dependent. In table 4, an example is given of NER.

Thousands	of	demonstrators	have	marched	through	London	to	protest	the	war	in	Iraq
O	O	O	O	O	O	B-geo	O	O	O	O	O	B-geo
CARDINAL						GPE						GPE

Table 4: Example of named entity recognition, NER, with the **IOB** scheme used in the second row, whereas the words are just annotated with labels in the last row, which is the practice *SpaCy* does, mentioned in *Annex - 4.3 Open-source NLP libraries*. The different schemes used for NE tagging is described in *Annex - 4.1.1 Named entity recognition*.

In the second row, the format of the named entities is **IOB**, short for *inside, outside & beginning*. So the above mentioned tags are annotated with either *I* or *B* as a prefix. Words that are not named entities are annotated with an *O*. The format is not further discussed, but is essentially a format for indicating position and correlation among tags.

In the last row, the named entities are just labelled, which is the practise used by *SpaCy* in *4.3 Open-source NLP libraries*. So the format varies, which is mentioned due to SpaCy being used as a prepro-

cessing NLP library, described in *4.3.1 NLTK vs. spaCy*.

Since NER is essential for the LA, this feature is a requirement for the dataset in *4.2.1 Dataset for linguistic analysis*.

4.1.2 Part-of-speech tagging

Part-of-speech tagging, **POS** tagging, is the process of marking up a word as either a noun, adjective, verb, etc [17]. The same word can have different POS tags depending on the context of the sentence, therefore, this feature is important to distinguish between different meanings of a single word. In table 5 an example is given of POS tagging.

Thousands	of	demonstrators	have	marched	through	London	to	protest	the	war	in	Iraq
NNS	IN	NNS	VBP	VBN	IN	NNP	TO	VB	DT	NN	IN	NNP
NOUN	ADP	NOUN	AUX	VERB	ADP	PROPN	PART	VERB	DET	NOUN	ADP	PROPN

Table 5: Example of part-of-speech tagging, POS tagging, with both the *Penn Treebank* phrase structure in the second row and the universal phrase structure in the last row. The different schemes for POS-tagging are described in *Annex - 4.1.2 Part-of-speech tagging*. POS tagging proves great syntactical value, and is therefore an optional input for the LA in addition with the NER, whereas POS will be used for the SA.

What is special about POS-tagging is, that there are different schemes for the annotation of words. For the example given in table 5, the second row is *Penn Treebank* phrase structure and in the last row it is the universal phrase structure. The reason both are mentioned is, that in *4.3 Open-source NLP libraries*, *SpaCy* is supporting the universal phrase structure, which is used for the sentiment dataset in *4.2.2.4 Findings*, whereas for the linguistic dataset in *4.2.1.4 Findings*, the Penn Treebank phrase structure is used. The different use of schemes will not be further discussed, since the LA and SA are independent from each other, and the use of different schemes won't collide.

POS tagging proves great syntactical value, and is therefore an optional input for the LA in addition with the NER, whereas POS will be used for the SA.

4.1.3 Lemmatization

Lemmatization is a calculated process to return a word's base/dictionary form by removing inflectional endings such as *"-ed"* on *worked* to return the base form, *work*, which is known as the lemma [19]. In table 6, an example is given of lemmatization.

Thousands	of	demonstrators	have	marched	through	London	to	protest	the	war	in	Iraq
thousand	of	demonstrator	have	march	through	London	to	protest	the	war	in	Iraq

Table 6: Example of lemmatization.

Lemmatization is beneficial for the SA, and information retrieval in general, since it allows the algorithm to know that *is* and *be* are the same thing [20]. Lemmatization requires POS-tagging to work, and often leads to bad results in other languages than English.

Another similar syntactical feature is *stemming*, which also reduces words, but "guesses" where to chop off words, typically rule-based. It reduces words to an equal or shorter form, and is the quicker, but not as accurate, way of reducing words in a corpus. For instance *"University"* is reduced to *"Universe"*,

even though the two words are not correlated [21].

Lemmatization will be used, since it provides the more accurate results, which is prioritized. The trade-off is it requires more computational power.

4.1.4 Stop words

Stop words are words that not always provide syntactical value when performing NLP. Typical stop words are *"and"* or *"I"* and so on.

Thousands	of	demonstrators	have	marched	through	London	to	protest	the	war	in	Iraq
	x		x		x		x		x		x	

Table 7: Example of stop words, where the stop words are marked with *x*.

Stop words are typically removed in information retrieval and sentiment analyses [22], which also is going to be the case for the SA of this report, as mentioned in *4.2.2.2 Considerations* for the SA.

4.2 Datasets

To carry out both the LA and SA, solid datasets are required with some of the linguistic features described in *4.1 Linguistic features*. Since both analyses need contextual awareness in a sequence of words, specific features for training the ML algorithms are required. In section *4.2.1 Dataset for linguistic analysis* and *4.2.2 Dataset for sentiment analysis*, datasets are evaluated and analyzed in the light of requirements, considerations, availability and findings for both the LA and SA.

4.2.1 Dataset for linguistic analysis

This section covers four steps to find the appropriate dataset for the lingusitic analysis. It serves as a link between the requirements in *3.3.1 MoSCoW analysis* and considerations about useful features introduced in *4.1 Linguistic features*. The section also studies availability of already constructed datasets and at last, presents the findings of available dataset which meets the given requirements.

4.2.1.1 Requirements

As mentioned in *3.3.1 MoSCoW analysis*, it is required that the LA must have contextual awareness and should have other syntactical features as part of the input. This is not only requirements to the implementation, but also to how the dataset should be designed. To obtain contextual awareness, a feature for the dataset has to be every word and special characters in a sentence, and every word and special character have to have a annotated named entity. This would compose the data matrix and target matrix respectively, for training a ML algorithm. An additional syntactical feature could be POS, described in *4.1.1 Named entity recognition*. This feature could potentially give additional value for the ML algorithm when finding a pattern in sequences of words.

4.2.1.2 Considerations

The ideal dataset would be a **corpus** of real-world job applications in English, labelled with named entities and POS-tags for each word and special character occurring. The named entities would then be relevant for the context, and could therefore be categorized demographic features such as person, age,

organization, ethnicity, religion and so on.

As an alternative, if there was a corpus of unlabelled real world job applications in English, then it could be possible to use a NLP library, such as those mentioned in *4.3 Open-source NLP libraries*, to label words with their respective POS tag and named entity (those provided by the library). This would carry along some issues, since using another trained ML algorithm to label a dataset would mean whatever bias and error margin that ML algorithm would have, would be passed on to this dataset.

4.2.1.3 Availability

Generating or writing dummy job applications and label them manually would first of all introduce bias, and the diversity of the dummy applications would be low, since they would probably all look like the same John Doe in the end. Second of all, it would be too time consuming compared to the time span of this project. Since job applications contain personal information and would most likely violate GDPR legislation, it isn't possible to find a corpus of English-written job applications, let alone a corpus where every word and special character is annotated with a named entity. So, considering this report is a proof of concept, a domain specific dataset is not required, as stated in *3.3.1 MoSCoW analysis*, and the LA of this report will for that reason be based on a public dataset of public domain English texts, annotated by *Groningen Meaning Bank*, **GMB** [23]. An alternative or supplement could also be the CoNLL-2003 dataset[24], which is very identical to GMB dataset, but has a limited amount of features and less samples, and is therefore not part of this report.

4.2.1.4 Findings

The dataset provided by GMB consists of 1,354,149 words, all annotated with a named entity. Furthermore, the dataset consists of 22 additional features such as descriptive, syntactical and contextual information of the word and its surroundings. In figure 4, a preview of the first 10 samples is given with selected features such as POS, named entity tag, the word itself and the word coming before and after.

lemma	next-word	pos	prev-word	shape	word	tag
thousand	of	NNS	__START1__	capitalized	Thousands	O
of	demonstrators	IN	Thousands	lowercase	of	O
demonstr	have	NNS	of	lowercase	demonstrators	O
have	marched	VBP	demonstrators	lowercase	have	O
march	through	VBN	have	lowercase	marched	O
through	London	IN	marched	lowercase	through	O
london	to	NNP	through	capitalized	London	B-geo
to	protest	TO	London	lowercase	to	O
protest	the	VB	to	lowercase	protest	O
the	war	DT	protest	lowercase	the	O

Figure 4: Preview of first 10 samples in the dataset provided by Groningen Meaning Bank, with seven out of 25 total features.

The dataset is avaiable on the public forum *Kaggle.com*[25]. The named entities for the dataset, also referenced as *tags* in the dataset, are:

- *geo* - Geographical Entity
- *org* - Organization
- *per* - Person
- *gpe* - Geopolitical Entity
- *tim* - Time
- *art* - Artifact
- *eve* - Event
- *nat* - Natural Phenomenon

The remainder of words are annotated *O*, which is just a padding. The format of the named entities is IOB, which is described in *4.1.1 Named entity recognition*. The POS scheme is following the *Penn Treebank* phrase structure, which also is described in *4.1.1 Named entity recognition*.

4.2.2 Dataset for sentiment analysis

This section accounts for a dataset's minimum requirements to build a sentiment analysis, considerations made before and underway, what's available and served the purpose best.

4.2.2.1 Requirements

A dataset with capabilities to help understand the differences in phrasing and vocabulary for how men and women communicate is needed, enabling to predict the gender bias. The absolute bare minimum requirements for such a dataset are samples consisting of sentences or entire texts labelled with the respective gender of the author.

For a better contextual understanding each sentence could be broken up into words, each joined by its semantic role labels and lemma.

4.2.2.2 Considerations

As per *4.2.1.2 Considerations* regarding the LA, the ideal dataset would be a corpus of real-world job applications in English, and in regards to sentiment analysis, ideally it should be labelled with gender bias or polarization, meaning three labels - male, female or neutral or a polarity varying between -1 and 1. Ultimately any other dataset with texts or sentences labelled as author by gender of each sample would do for a prototype.

Another way to approach the problem would be using seed words, which could look like the ones illustrated in table 8. They can be used for gender tagging entire sentences, if a sentence contains a greater number of a gender's seed words, it can be labelled as a male or female sentence. They can also be used for gender association, to create an embedding model, acting as a dictionary mapping semantically close words, helping the model understand the contextual dependencies [26]. Either way training a model using seed words to understand gender bias, would result in understanding how a gender is portrayed and not the actual underlying conscious or subconscious way of how men and women write.

Seed words: M	Seed words: W
man	woman
boy	girl
sir	ma'am
gentleman	lady
he	she
his	hers
him	her

Table 8: Seed words list. [26]

Datasets containing gendered words, which directly includes or excludes one or the other gender, exists online, such as *Gendered Words Dataset* [27]. An example of gendered words is using fireman or firewoman, instead of the neutral and inclusive alternative term, firefighter. It is is one of the most direct

forms of gender in-/exclusion, and is not something that the SA will include, since it is not equivalent of how male and female write, but instead a common and obvious gender bias in today's speech.

As in any machine learning algorithm, noise only introduces confusion and greater loss. In sentiment analysis and information retrieval in general, noise primarily stems from:

- URLs, email addresses, HTML tags, numbers and multiple spaces. None of these add any meaningful value or context [28] [29].
- Punctuation signs, which confuse a model to not recognise "sure" and "sure!" as the same thing [29] [30].
- Stop-words, which are extremely common words of little value, in English it could be "a", "by", "it", etc. [22]

It is also recommended to use stemming or lemmatization to reduce inflectional forms and derivationally related forms of a word to a common base form [19]. All these noise factors are removed in the preprocessing stage of a dataset, and can be done using custom functions and/or using NLTK or spaCy, analized in *4.3.1 NLTK vs. spaCy*.

4.2.2.3 Availability

No dataset with texts of any kind, labelled with gender of the writer, are currently publicly available. This seem to be a common issue among gender-oriented sentiment analyses [31], with a team of research interns, part of the humanitarian AI internship at Mila - Quebec AI Institute, going as far as openly crowd-sourcing labelling of gender biases, *biaslyai.com*. Other works, such as [26] uses film summaries from Wikipedia, using seed words to understand how male and female characters are described, hence 87-90% of "Wikipedians" are male, it is more accurately how men describe male and female characters. Lastly research papers like [32] "*..build a human-curated evaluation benchmark for the rewriting task, comprising 500 gendered and genderneutral sentence-pairs..*" to rewrite gendered pronouns with the corresponding gender neutral pronoun.

Considering a domain specific dataset is not required, as per *3.3.1 MoSCoW analysis*, reasoned the restrictions by GDPR doesn't allow for such to be publicly available, no useful dataset for the intended purpose publicly exists and we do not have the time to handcraft a significantly-sized labelled dataset. The SA of this report will for these reasons be based on the public dataset *News Category Dataset* by *Rishabh Misra* [33]. The dataset consists of 200,000 news headlines from the year 2012 to 2018 obtained from *HuffPost*, the author envisions that a model trained on this dataset could be used to identify news articles categories.

Instead the dataset will be used for web scraping of the linked articles, roughly assigning chosen categories to represent male and female by using the gender of the expected target audience, which is the chosen methodological approach to determine what is male and female writing.

4.2.2.4 Findings

The dataset consists of 200,853 samples with six features; category, headline, authors, link, short description and date. Category and link being the only relevant features for this analysis.

	category	headline	authors	link	short_description	date
0	CRIME	There Were 2 Mas...	Melissa Jeltsen	https://www.huff...	She left her hus...	2018-05-26
1	ENTERTAINMENT	Will Smith Joins...	Andy McDonald	https://www.huff...	Of course it has...	2018-05-26
2	ENTERTAINMENT	Hugh Grant Marri...	Ron Dicker	https://www.huff...	The actor and hi...	2018-05-26
3	ENTERTAINMENT	Jim Carrey Blast...	Ron Dicker	https://www.huff...	The actor gives ...	2018-05-26
4	ENTERTAINMENT	Julianna Marguli...	Ron Dicker	https://www.huff...	The "Dietland" a...	2018-05-26
5	ENTERTAINMENT	Morgan Freeman '...	Ron Dicker	https://www.huff...	"It is not right...	2018-05-26
6	ENTERTAINMENT	Donald Trump Is ...	Ron Dicker	https://www.huff...	It's catchy, all...	2018-05-26
7	ENTERTAINMENT	What To Watch On...	Todd Van Luling	https://www.huff...	There's a great ...	2018-05-26
8	ENTERTAINMENT	Mike Myers Revea...	Andy McDonald	https://www.huff...	Myer's kids may ...	2018-05-26
9	ENTERTAINMENT	What To Watch On...	Todd Van Luling	https://www.huff...	You're getting a...	2018-05-26

Figure 5: Head of *News Category Dataset*.

The dataset is worth at least 200,853 sentences, using short_description shown in Figure 5, which is short summaries consisting of one or more sentences.

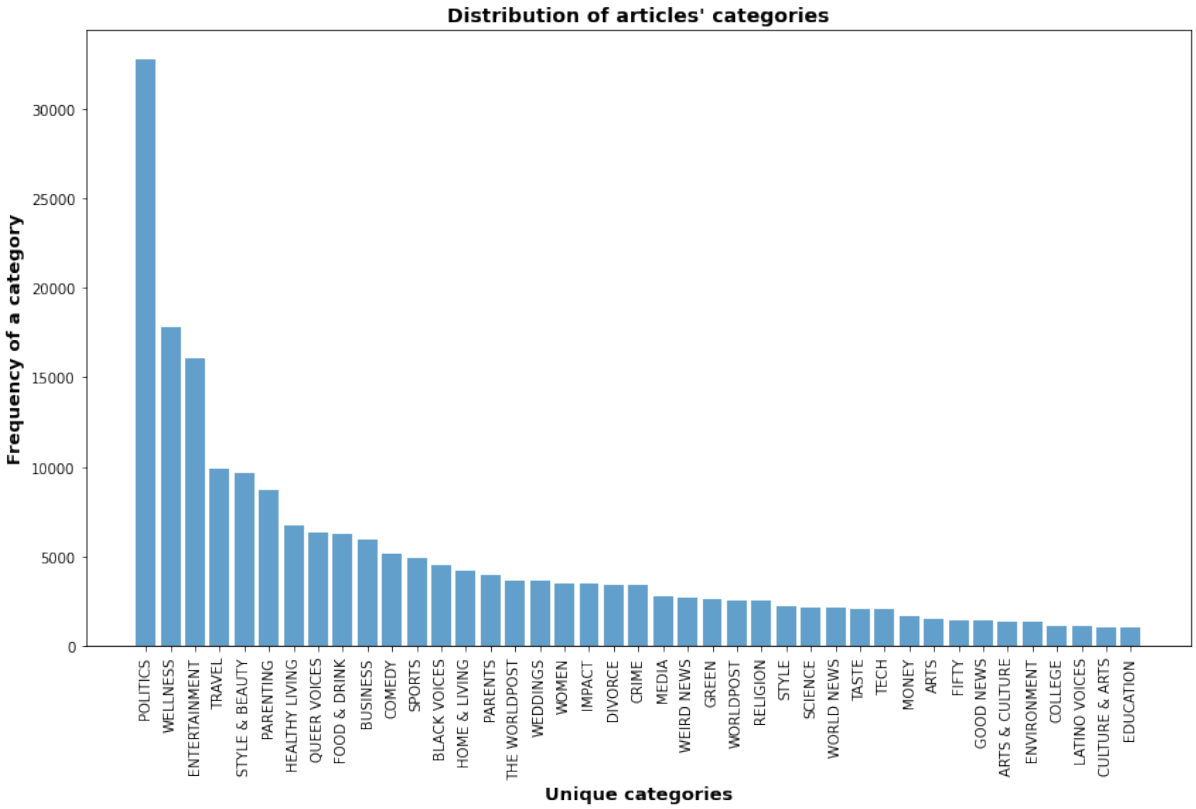


Figure 6: Categories found in *News Category Dataset* column *'categories'*.

Looking at figure 6, considering frequency of each category, level of nuanced language and expected biased wording, *SPORTS*, *MONEY* and *BUSINESS* has been chosen to represent male gendered phrasing and wording. *WOMEN* and *STYLE & BEAUTY* has been chosen to represent female gendered phrasing and wording. These choices are made, because it's presumed that the target audience is the expected gender, no scientific or theoretical theory validates it, it is simply chosen as a first iteration. So for now the labels are accepted as the truth and results in 12,528 and 13,139 articles for men and women respectively.

Since it is chosen that a given category is representative of gender, the dataset can be expanded beyond just short summaries, by web scraping each article for its body text. Then to create a clean dataset

a trained NLP pipeline, such as spaCy's *en_core_web_lg*, can be used for data augmentation using NLP to breakdown entire article's texts into sentences, and sentences into words. Furthermore it can identify named entities, stop words and semantic role labels [31] [34], which, as described in *4.2.2.2 Considerations*, only cause noise.

4.3 Open-source NLP libraries

As part of feature engineering the dataset for the SA, a NLP library is required to do some text classification and processing. There are a variety of specialized tools on the market, among these, which also are the most relevant to this project, are *NLTK*[35] and *spaCy*[36]. Both are open-source libraries in Python, which allows for integration in the development of this project.

4.3.1 NLTK vs. spaCy

NLTK is one of the leading open-source NLP libraries out there, supporting classification, tokenization, stemming, tagging, parsing, and semantic reasoning, and is used for both education and research across many professions [35]. The library is a toolkit, which allows it for the developer to use any kind of model in the toolkit, to try and apply to their use case.

SpaCy is also a leading open-source NLP library, and a direct contender to NLTK, supporting roughly the same amount of features as NLTK, but what makes the two differ is, that NLTK was initially developed for research and education, giving the user the ability to choose between multiple algorithms, whereas spaCy chooses the best suited algorithm for the specific NLP task, leading to better performance and less overhead [37].

Relevant for the SA is NLP tasks such as NER, POS tagging and lemmatization as well as identifying stop-words, all of which SpaCy supports. Therefore, due to its seamless integration, less overhead and many NLP tasks supported, SpaCy will be the NLP library used to compile the final dataset for the SA.

4.4 Machine learning frameworks

This section will consider what neural network framework is best suited for the intention of the project. There are two main factors to keep in mind, when deciding upon what framework to use for performing the LA and SA: *1.* the time factor of the project and *2.* the intention of the ML algorithms. Since the purpose is to adapt known architectures and algorithms to a new problem setting, a well-documented framework with a decent entry-level and high-level API which allows for relative quick benchmarking between models is favourable.

4.4.1 Scikit-learn

Scikit-learn is a framework build upon Python libraries such as *NumPy*, *Pandas* and *Matplotlib* and is among others used for preprocessing tasks, regression tasks and classification tasks. It is not the most suited framework for composing neural networks, where it only offers a few neural networks, where only a handful of hyper parameters are accessible. Since scikit-learn is very user-friendly and well-documented, its utility to preprocessing tasks are worth considering, but as a main framework for composing tailored neural networks, it's not the most favourable.

4.4.2 Keras

Keras is a deep learning framework interfacing TensorFlow and is known for fast experimentation. Keras has an intuitive interface, and allows for sequential and arbitrary modelling of neural networks. It offers a modular approach to assemble input layers, hidden layers and output layer as a complete model, and besides it is possible to decide which activation functions are appended to each neuron for each layer. Furthermore, Keras gives the ability to set multiple hyper parameters for each layer, decide what optimizer and loss-function are used for the model and it is possible to add callbacks to the model when training, allowing for close realisation of training and validation metrics. These are just a handful of options that Keras offers, which very much fit the desired framework.

4.4.3 TensorFlow & PyTorch

Both *TensorFlow* and *PyTorch* are deep learning frameworks, which are more low level than above-mentioned scikit-learn and Keras. The features offered by Tensorflow and PyTorch reach way beyond the scope of this project, with implementation in web applications and mobile applications and allowing for many downstream tasks. They are both strong contenders for research work and complex modelling of neural networks, but for the intended analyses of this project, low-level frameworks are not needed.

4.5 Neural network architectures

For NLP tasks such as NER and sentiment analysis, some neural network architectures are more compatible than others. Especially recurrent neural networks, **RNN**'s, are common for NLP tasks, since they among others are strong on gaining contextual awareness and recognizing patterns in time sequential data. Many-to-one RNN architectures are commonly used for sentiment analysis to perform binary or ternary classification, since the input can be a sequence of words and output a single sentiment for that sequence. Many-to-many RNN architectures usually are used for NER[38], since the input can be a sequence of words and the output can be a sequence of resulting named entities for each word. Additionally, convolutional neural networks, **CNN**'s, have proven great success with NLP tasks, despite being mostly known for image classification. Both architectures are discussed in the following sections.

4.5.1 Recurrent neural networks

Compared to common feed forward neural networks, RNN's have the same feed forward mechanism, but then backpropagates through the network, computing the gradient of the loss function and adjusts the weights one layer at a time. Since RNN's compute on sequential data, the architecture has the advantage of keeping state of previous input. When training a RNN at an arbitrary input at *t*, the network has knowledge about previous state at *t-1*. But the greater the time sequential data becomes, the harder it becomes for the network to recognize long-term dependencies. When the sequential data reaches a point where it is too long, RNN's can suffer from vanishing gradients, meaning the gradient becomes so small that the weights won't change during backpropagation.

Fortunately, long-short-term-memory, **LSTM**, networks are able to withhold these long-term dependencies and also stamp out the vanishing gradient problem. LSTM is an extension of RNN, where additional units have been added to the original RNN. As depicted in figure 7 the LSTM cell is partitioned in 3 parts, where division 1 is the forget gate, division 2 is the input gate and division 3 is the output gate. Additionally, the cell has the long-term-memory C_{t-1}, also known as the cell state, and it has the short-term-memory h_{t-1}, also known as the hidden state.

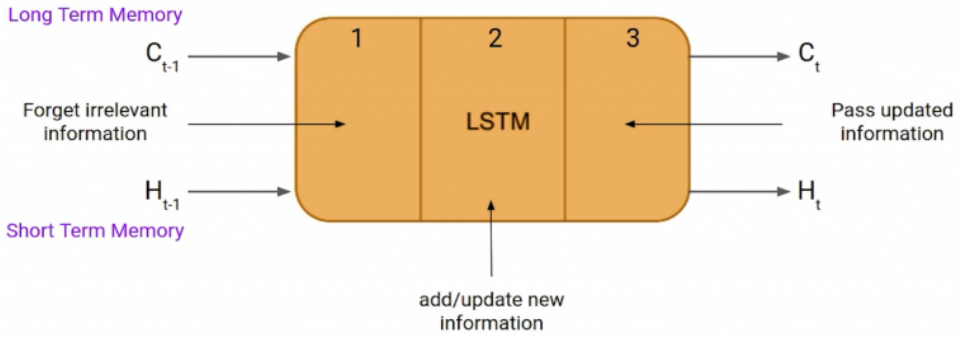


Figure 7: Overview of the of LSTM cell, where division 1 is the forget gate, division 2 is the input gate and division 3 is the output gate. Source: *www.analyticsvidhya.com* [39].

Breaking down the LSTM cell in figure 8, it becomes visible how the three gates are connected and operating. The forget gate consists of the cell state C_{t-1} and hidden state h_{t-1}, the input gate consists of the input X_t and the output gate consists of the updated cell state C_t and hidden state h_t which also reflects the output.

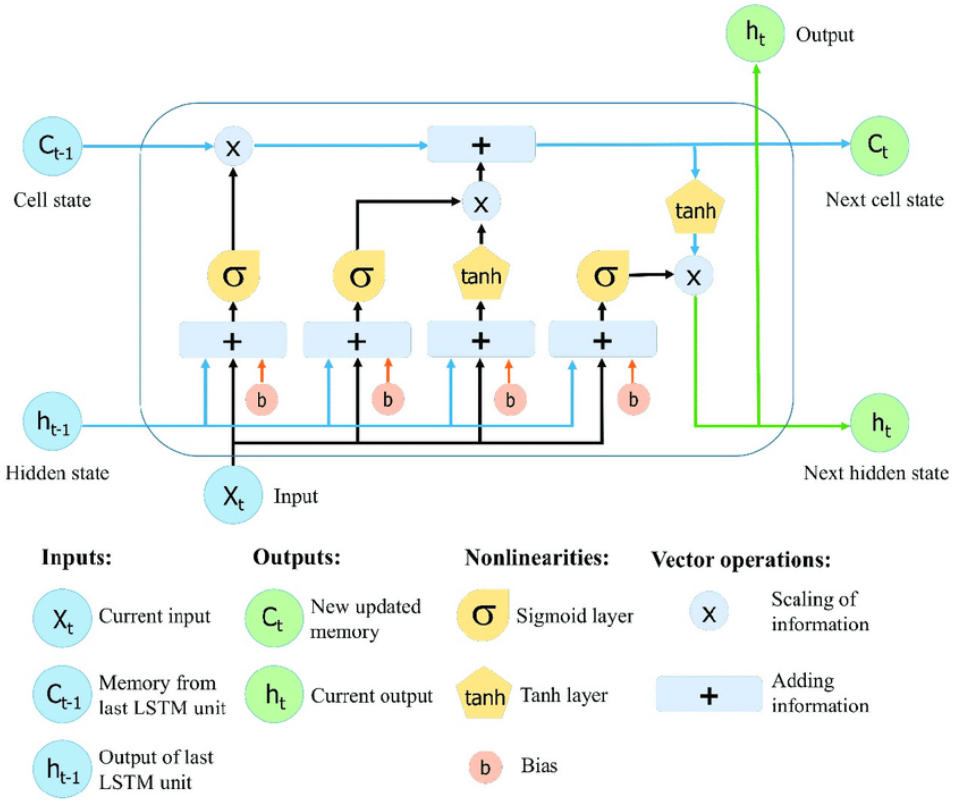


Figure 8: Structure of LSTM cell showing how information is persisted with the hidden state and cell state. The cell state C_{t-1} and hidden state h_{t-1} make up the forget gate, the input X_t makes up the input gate and the output h_t, next cell state C_t and next hidden state h_t make up the output gate. Source: *www.researchgate.net* [40].

The forget gate decides what information from h_{t-1} and X_t to persist, where it applies a sigmoid function and results in a number between 0 and 1, where 0 is that it forgets everything from the previous timestamp and 1 it persists everything from the previous timestamp. The output from the sigmoid function is multiplied with the cell state C_{t-1} [39][41].

The input gate decides what information will be stored as the new cell state C_t, by multiplying the output from a sigmoid function and tanh function for h_{t-1} and X_t [41].

Lastly, the output gate decides what information is outputted and saved as the new hidden state h_t, which is the new cell state C_t through a tanh function multiplied with h_{t-1} and X_t through a sigmoid function [41].

The structure of LSTM allows for contextual awareness of sentences as per requirement in *3.3.1 MoSCoW analysis*, since the combination of the hidden state and cell state acts as short-term and long-term memory, and therefore has the ability to recognize patterns in sequence data. However, LSTM networks only has long-term dependencies from the past, whereas it could be favourable to also know about the future. For instance, taking an arbitrary sentence from the dataset at *t*, the network would have knowledge about previous state *t-1*, but it could be beneficial for the prediction if the network also had knowledge about future state *t+1*. This is allowed by bidirectional LSTM, **biLSTM**.

Opposite to LSTM, which flows in only one direction and has knowledge of only previous state, biLSTM flows in both directions; past to future and future to past. This means the network will have knowledge of both previous and future state, which is beneficial when computing on sentences where context could be given in both the start and the end of a sentence.

4.5.2 Convolutional neural networks

Convolutional neural networks are commonly used in multi-dimensional image classification problems to detect complex features, but have also proven strong in detecting complex features in two-dimensional text classification problems. CNN's consists in its simplicity of a convolution layer and a pooling layer. The convolution layer consists of a kernel that shifts over the input with its receptive field, computing a feature map by detecting the most prominent features within the receptive field[42]. The pooling layer then subsamples the feature map by performing an operation on a receptive field of the feature map step by step and lastly outputs a matrix of fixed size, which is a necessity for classification. If a $n \times m$ matrix is considered in a classification problem, then each row represent a vectorized word, also considered a word embedding as depicted in figure 9[43].

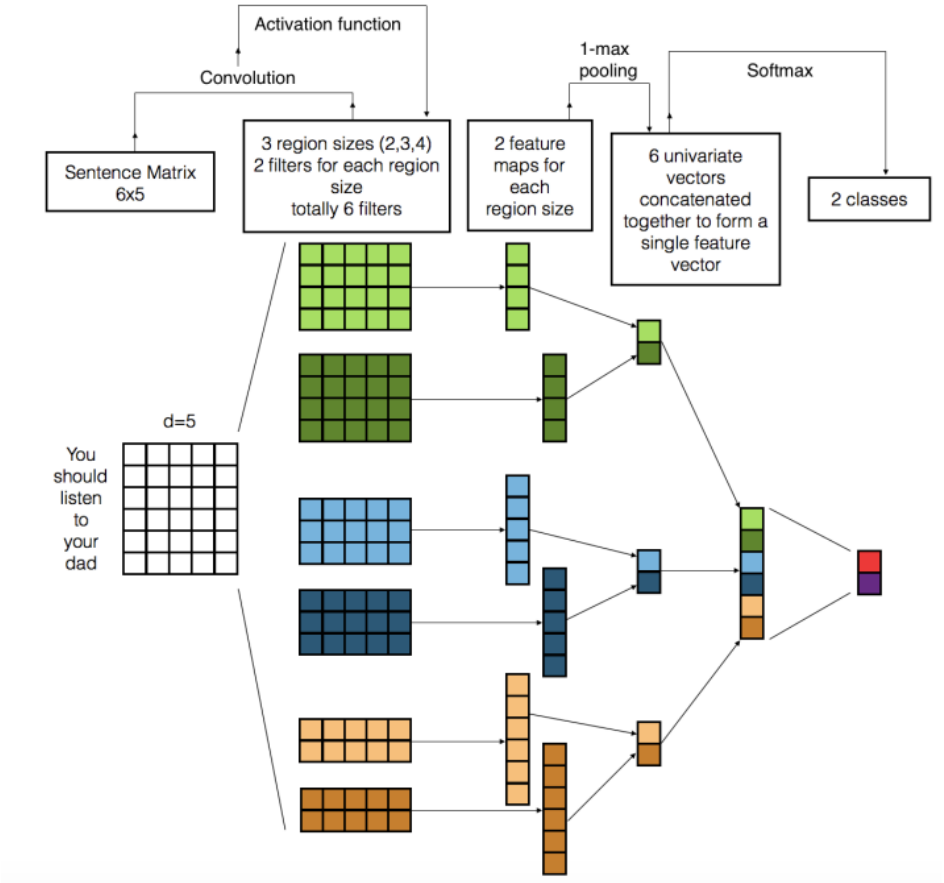


Figure 9: CNN layer handling of a text classification task. The sentence matrix contains every row representing an embedding (a vector) of a word. The embeddings are of the dimensionality of five, meaning every vector representing a word has five elements. Source: `https://arxiv.org/pdf/1703.03091.pdf`.

Depending on the embedding implementation, the embedding basically vectorizes a one-hot encoded word into a vector that represents that specific word. When the embedding of all words is done, a lookup table has been created, where the tokenized word maps to the vectorized word, which is used for optimizations of downstream tasks. An embedding example could look like the following:

you	$\longrightarrow$	7	$\longrightarrow$	[4.2, 1.7, 2.3, 0.5, 5.9]
Word		Token		Vector

So for figure 9, each word is embedded with a dimensionality of five, meaning that each word has been transformed into a vector of fixed size five representing the respective word. In this case, there are six filters resulting in six feature maps, since each filter outputs a feature map. A max-pooling layer is applied, meaning the element with the greatest weight of each feature map is extracted and concatenated into a single feature vector on which a softmax function is applied to give a probability distribution of two classes. Figure 9 is a specific implementation of CNN, but gives an idea of how CNN can be applied to NLP tasks.

4.5.3 Comparison

Looking at *4.5.1 Recurrent neural networks* and *4.5.2 Convolutional neural networks*, LSTM and bidirectional LSTM networks have the ability to process long sequences of data, such as sentences, and then

utilize its long-term memory and short term memory to gain contextual awareness. This would provide great value for both the LA and SA, where the inputs are sequences of words. CNN networks will be explored as well for the LA, despite they normally are being used for image classification, also are able to identify complex patterns in sequence data. Either LSTM or BiLSTM with CNN could potentially also be used as a mixture.

4.6 Humanistic and ethical insight

In this section the project's intention, stated in *3.1 Concept description in a domain context*, is discussed in relation to humanistic and ethical considerations. Furthermore a survey is conducted to study the public's perception of the project's problem statement and whether the project's intended solution is of any interest.

4.6.1 Ethical dilemmas and concerns

The SA aims to predict gender bias in a binary form, masculine and female, knowing gender definitions increasingly are becoming ambiguous. Thereby already in setting the task, the project discriminates against people, who do not define themselves by the traditional binary gender roles [44]. Even more so, the gender is classified by a news category's presumed target audience's gender. The presumption is not made on any theoretical basis, but on intuition by the project's authors, who are two cisgender white males, which is highly probable to introduce bias.

The project's intention, stated in *3.1 Concept description in a domain context*, is to reduce or completely remove possible discrimination between A and E, but in straight pursuing an solution without further humanistic or ethical considerations, contradictory the project itself may be discriminatory and contain a lot of bias.

4.6.2 Study of public demand

The project aims to help both job A's and E's, by making the early stages of the job recruitment process more anonymous, allowing A's to be evaluated and E's to evaluate job candidates without any prejudice based on their demographic characteristics, socioeconomic status. Two questionnaire surveys have been developed, one to study A's perception of the problem and proposed solution, and vice versa for E. Due to the project group's inexperience with questionnaires and the liberal arts in general, the questionnaires were developed in collaboration with three cand. mag. students of Media Studies.
The questionnaires consists of simple 'yes'/'no'/'don't know' questions for simplicity's sake. Below the selected questions and answers of highest interest are presented, but all questions and answers can be found in *8.3 Appendix C*.

	Questions	Yes	No	Don't know
1	Would you let your application be run through an algorithm, which censors your personal information (gender, age, ethnicity, race and religion), to not be picked/discarded based on it, but contrarily your competences and experience?	9	1	0
4	Would you let your application be run through an algorithm, which makes you aware of words with a high gender bias, which could give away your gender?	2	4	4
8	Would you be more prone to send an application for a job, in which the job advertisement uses a gender neutral language without gender biased words?	5	2	3

Table 9: 3 out of 8 of the questions articulated for A's, which students around the campus were chosen as, since it is highly probable that they are job seeking at the moment, recent past or the near future. The survey group consisted of 5 males and 5 females, and was primarily white. It is evident that the majority are willing to use a solution, such as the LA aims to provides, to be evaluated solely on competences and experience. While there doesn't seem to be a majority interested in using a solution such as SA, to help make an application more gender neutral, it seems from A's viewpoint, that E would benefit from applying the SA while writing job advertisements.

	Questions	Yes	No	Don't know
1	Is the applicant's gender a part of your considerations in relation to the screening process?	3	7	0
4	Would you utilize an algorithm to hide an applicant's personal information, such as gender, ethnicity, race, age and religion?	6	4	0
6	Do you think that an applicant's gender, ethnicity, race or religion may influence your choice subconsciously?	6	3	1
14	Would you be willing to make your job advertisement more gender neutral, if you were made aware that it contains gender biased wording?	9	1	0

Table 10: 3 out of 14 of the questions articulated for job employers, which the teachers in the AU building Edison were chosen as, since they seem probable to have the age and experience, which job recruiters often has. The survey group consisted of 10 lecturers, who were primarily white males. The majority agrees to not let gender influence their decision when screening candidates, but also agrees they think that demographic characteristics, socioeconomic status or gender may influence their decision subconsciously. The same partition of the group agrees to utilize an algorithm to hide such information. Lastly, there is an overwhelming amount willing to make their job advertisement more gender neutral, if a solution such as SA, was to make them aware of gender biased wording.

5 Linguistic analysis

The LA was accounted for shortly in *4 Analysis*, where related linguistic conventions were introduced and explained and the dataset, which the analysis will center about, is presented. This section will cover the methodological approaches taken in use step by step, both as part of the preprocessing part as well as the training of networks an presenting results. The preprocessing steps will remain unchanged, whereas for the composition of networks, layers will be substituted such that LSTM, BiLSTM and CNN will be explored. Looking at figure 10, the pipeline of methodological approaches is shown.

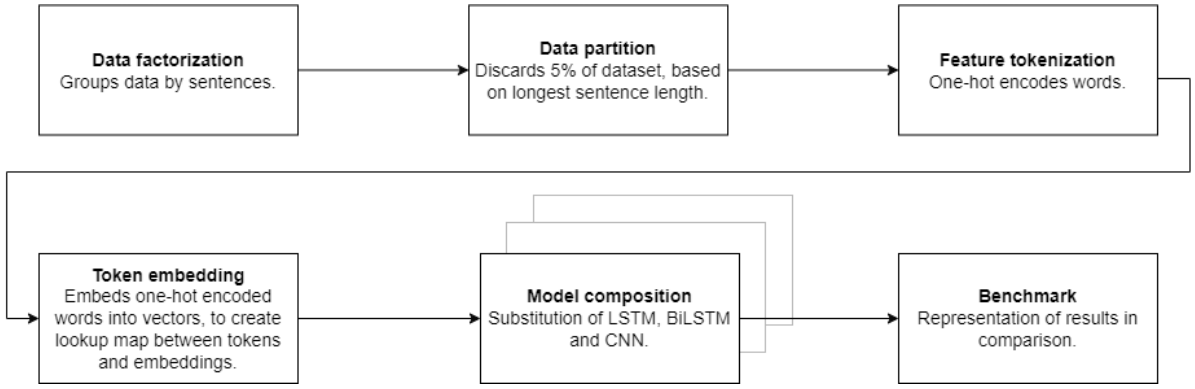


Figure 10: Pipeline of methodological approaches.

An exposition of these methodological approaches are given in *5.1 Methodology* and an implementation is covered during *5.2 Feature engineering* and *5.3 Implementation of neural networks*, where the flow of the pipeline is visualized for better understanding. The results are unfolded in *7.1 Linguistic analysis: Results*.

5.1 Methodology

This section covers the methodology used to process the dataset analyzed in *4.2.1 Dataset for linguistic analysis* and the methodological approach to gain the results of the LA in *7.1 Linguistic analysis: Results*. Additionally, it covers an evaluation of the methodology and its validity and reliability.

5.1.1 Data collection

The dataset from GMB is based on public domain English texts, which has been processed by GMB itself, to annotate each word and punctuation within each text with a named entity. The dataset comes in a *.csv* format and is comma-separated. A summary of the dataset is given in *4.2.1.4 Findings*. A compiled version of the dataset was fetched from *Kaggle.com*[25] and no further initial preprocessing of the dataset was performed. As a start, the dataset was compressed to only consist of words, POS-tags and NE-tags.

5.1.2 Methodological approach

Sequential preprocessing steps was initially performed to prepare the data to train the neural networks described in *5.3 Implementation of neural networks*. These methodological approaches of data preparation, as well as for neural network composition, are described in the following sections.

5.1.2.1 Data factorization

Since the neural networks should be trained on sequences of words, as described in *3.3 Requirement specification*, the dataset was transformed into divisions of sentences, so each new sample consisted of a sequence of words, sequence of POS-tags and sequence of NE-tags.

5.1.2.2 Partition of data

The distribution of length of all sentences were measured against their frequency to identify sentences with a particularly high word count. A decision boundary was set in the distribution, partitioning the distribution at 95% of the word accumulation for all sentences. This was done to discard the remaining 5% of sentences with the highest word count, thus reducing the required padding of all sentences in *5.1.2.3 Feature tokenization* and subsequently reducing training time. The target value of the decision boundary was based on an empirical estimate of the dataset with no specific theoretical argumentation.

5.1.2.3 Feature tokenization

Every sequence of words, POS-tags and NE-tags were tokenized as integers, so each sequence would consist of integers that can be mapped to the value it is representing. The decision boundary was then used as a cap in combination with a padding process, so samples containing sequences with encoded elements greater than the boundary would be discarded and the remaining samples were padded with a padding token, so they all had the length of the decision boundary.

Finally, every sequence of NE-tags were transformed into binary class matrices, meaning a binary representation of each NE-tag in a sequence as a matrix.

5.1.2.4 Token embedding

Since the input sequences of words and POS-tags all were encoded and of equal length, they where embedded, so each integer representation of a word was embedded into a vector of a fixed dimensionality. This creates a lookup table during training where the tokenized words map to the vectorized token. Embeddings allow for the neural network to understand semantically similar words over time, and is build in the idea that similar words occur together more frequently than dissimilar words [45]. Semantic similarity between words is crucial for gaining contextual awareness, which is why token embedding is an important step prior to model composition.

5.1.2.5 Model composition and benchmarking

After the methodological approaches in *5.1.2 Methodological approach* were carried out, only the input layer was changed. At this point, different layers were appended to the embedding layer, composing different neural networks varying in input size, hyper parameters, callbacks etc. The neural networks where benchmarked and compared with respect to score metrics and confusion matrices.

5.1.3 Evaluation

This methodology is a widespread approach for NER, but how data is preprocessed varies. Especially the tokenization approach could have some side effects, since every sequence is padded with a padding token. Another approach could also be adding a start-of-string token, **SOS**, and a end-of-string token, **EOS**. These could help the model to recognize patterns in the sentences.

Both words and punctuation were tokenized and trained upon, so another approach could be to discard punctuation since these possibly could just create more complexity than advantage while training the

neural networks.

If more time was available for the project, tuning hyperparameters, learning rate scheduler and early stopping could have been favourable for a better result. Due to training times of the neural networks, the analysis consisted mostly of analyzing viable models and less of finding the right parameters, so this is merely a noting.

5.2 Feature engineering

The compressed dataset from *4.2.1.4 Findings* consisting of words, POS-tags and NE-tags was loaded into a Python environment as a *dataframe* using the *Pandas* library. This allows for easy data manipulation and visualization. Additionally, as part of the feature engineering, the *NumPy* library and Keras framework was used, analyzed in *4.4.2 Keras*.

5.2.1 Data visualization

To gain an insight into the distribution of target values, the NE's of the dataset was plotted against their occurrence. Since some NE's of the dataset are more frequent than others, this would give a suggestion to what to expect from the results regarding each predicted NE.

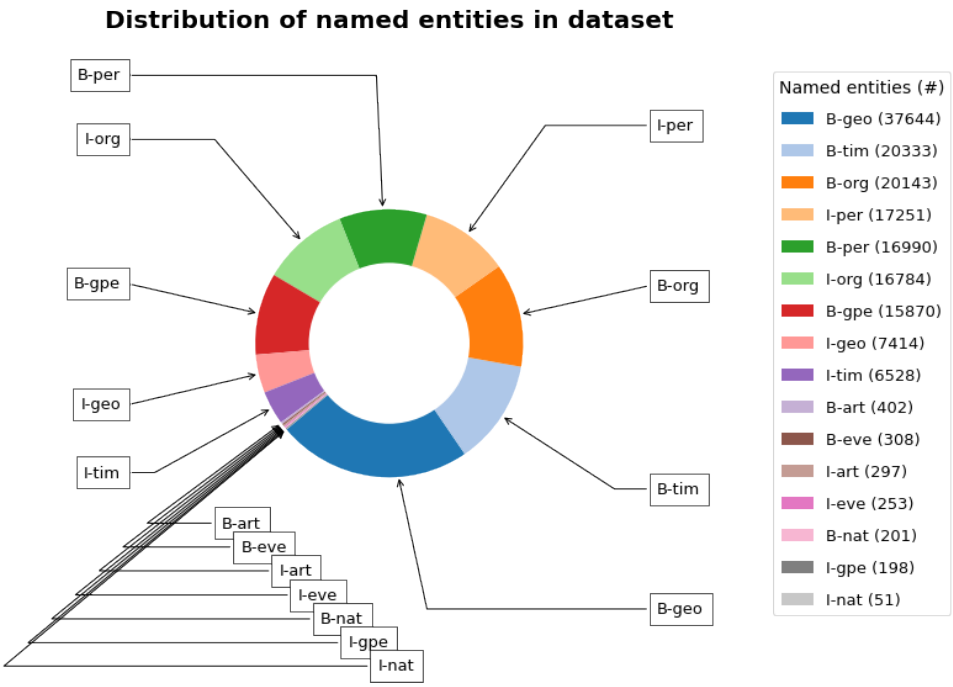


Figure 11: Distribution of named entities in dataset, with *O* tags being omitted since they account for 887,908 instances.

In figure 11, the distribution of named entities can be seen from the dataset. Despite it being a dataset consisting of 1,354,149 words, some of the NE are only appearing a few hundred times and less, which is an expected occurrence. Reasoned a sentence consists of many stop words, which aren't assigned a named entity. Moreover, named entities such as *natural phenomenon* is presumably more seasonal in articles, whereas *organizations* is most likely mentioned on a daily basis. Therefore, the neural network won't have many occurrences of these NEs to train on. However, from *I-tim* and upwards, there is a strong representation. All words marked with *O* is omitted from the chart, so a better relative comparison between the real NE was possible. *O* accounts for 887,908 instances, and would give a bad visual

representation of the desired NE tags. A description of the NE annotation scheme is in *4.1.1 Named entity recognition*.

The dataset was factorized, so words were grouped by the sentence they belonged to, resulting in each sample becoming a sequence of words, sequence of POS-tags and sequence of NE-tags as depicted in figure 12.

	Word	POS	Tag
0	[Thousands, of, demonstrators, have, marched, ...	[NNS, IN, NNS, VBP, VBN, IN, NNP, TO, VB, DT, ...	[O, O, O, O, O, O, B-geo, O, O, O, O, O, B-geo...
1	[Iranian, officials, say, they, expect, to, ge...	[JJ, NNS, VBP, PRP, VBP, TO, VB, NN, TO, JJ, J...	[B-gpe, O, O, O, O, O, O, O, O, O, O, O, O,...
2	[Helicopter, gunships, Saturday, pounded, mili...	[NN, NNS, NNP, VBD, JJ, NNS, IN, DT, NNP, JJ, ...	[O, O, B-tim, O, O, O, O, O, B-geo, O, O, O, O...
3	[They, left, after, a, tense, hour-long, stand...	[PRP, VBD, IN, DT, NN, JJ, NN, IN, NN, NNS, .]	[O, O, O, O, O, O, O, O, O, O, O]
4	[U.N., relief, coordinator, Jan, Egeland, said...	[NNP, NN, NN, NNP, NNP, VBD, NNP, ,, NNP, ,, J...	[B-geo, O, O, B-per, I-per, O, B-tim, O, B-geo...

Figure 12: First five samples of factorized LA dataset, grouped by sentences.

The factorization was mainly done as a step towards generating viable input data for the neural networks, but it also allows for further insights into the distribution of sentence lengths measured against frequency, as depicted in figure 13.

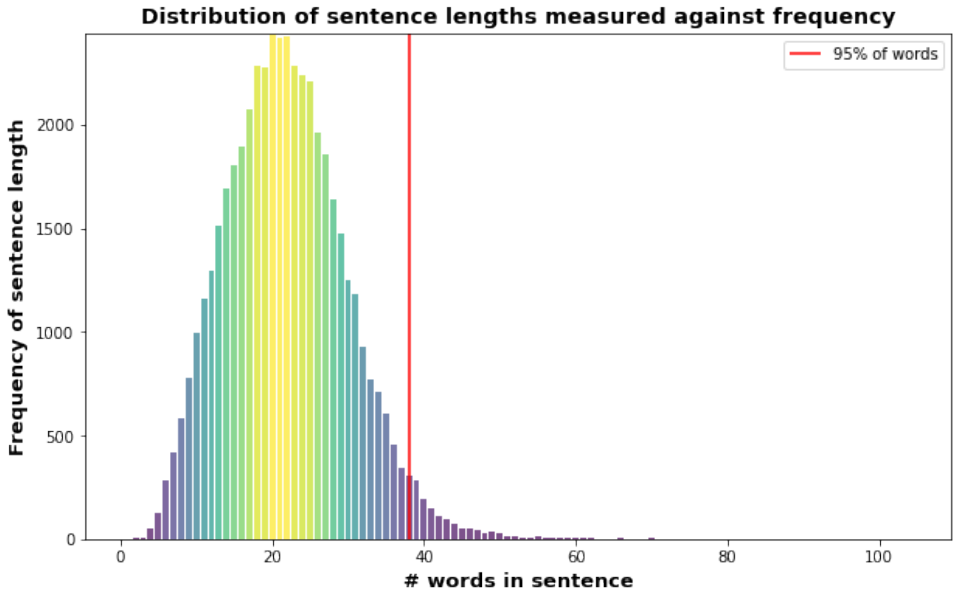


Figure 13: Distribution of sentence lengths with a decision boundary partitioning the sentences at a word and punctuation count at 38, discarding the remaining 5% of the dataset.

In figure 13 a decision boundary was set, partitioning the distribution at approximately 95% of the accumulated word count. The value of the decision boundary was 38, meaning that sentences with a word and punctuation count greater than 38 would be discarded during training. The reason behind this methodological approach is further explained in *5.2.2 Word tokenization*.

5.2.2 Word tokenization

Since neural networks only take integers as input, it is required to tokenize the sequences of words, POS-tags and NE-tags from figure 12. The first step of the tokenization process consisted of creating a

pair of dictionaries for each column in figure 12. One of the dictionaries in a pair acted as a mapping between the integer index and a unique value and vice versa for the second dictionary. An example of a pair of mapping dictionaries is given in figure 14.

0 :	<PAD>
1 :	Thousands
2 :	of
3 :	demonstrators
4 :	have
...	...

(a) Example of mapping dictionary from index to word, with padding token added as first index.

<PAD> :	0
Thousands :	1
of :	2
demonstrators :	3
have :	4
...	...

(b) Example of mapping dictionary from word to index, with padding token added as first index.

Figure 14: Examples of mapping dictionaries in both directions. Non of the above dictionaries resembles the actual dictionaries, these are just for comprehension.

On index 0 of each dictionary in figure 14, a padding token was inserted as part of a masking process in *5.3.1.1 Token embedding*. As a padding token for word and POS sequences, the string *<PAD>* was used, which is just a meaningless value not expected to be encountered in the dataset. For sequences of NE tags, the padding value is *O*, which is the annotation already used for non-NE's and therefore makes sense to use.

A combined transformation was then performed on the dataset, where the sequences of words, POS tags and NE tags from figure 12 where assessed in proportion to the decision boundary of 38 from *5.2.1 Data visualization*. If the length of any sequence was greater than 38, they were discarded, due to reasons mentioned in *5.3.1.1 Token embedding*. The remaining sequences where then padded up to and including 38 with their respective padding token. So sequences of words an POS tags less than 38 would be padded with *<PAD>* and sequences of NE tags would be padded with *O*. When all remaining sequences were padded, they were tokenized utilizing their respective dictionary mapping from strings to indexes, mapping each word, POS tag and NE tag to their respective index in their respective mapping dictionary. An extract of the tokenized sequences is shown in figure 15.

	Word_index	Tag_index	POS_index
0	[15078, 27701, 20970, 24219, 26435, 33390, 968...	[0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 5, 0, 0, ...	[20, 11, 20, 36, 35, 11, 18, 30, 32, 8, 17, 11...
1	[8194, 27728, 31034, 33290, 22578, 33465, 2372...	[7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[12, 20, 36, 23, 36, 30, 32, 17, 30, 12, 12, 2...
2	[7599, 24040, 13560, 28906, 26766, 24371, 2485...	[0, 0, 15, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 11...	[17, 20, 18, 33, 12, 20, 11, 8, 18, 12, 17, 2,...
3	[15050, 25893, 16916, 16575, 33180, 24593, 323...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[23, 33, 11, 8, 17, 12, 17, 11, 17, 20, 3]
4	[15425, 30228, 20255, 8421, 5853, 30958, 14608...	[5, 0, 0, 13, 14, 0, 15, 0, 5, 0, 7, 0, 7, 0, ...	[18, 17, 17, 18, 18, 33, 18, 2, 18, 2, 12, 6, ...

Figure 15: First five samples of tokenized LA dataset, grouped by sentences.

In figure 16 an arbitrary example is given of a mapping between a sequence of tokenized words and its real value, utilizing the mapping dictionary mapping from index to string value, given an example of in figure 14. Here it comes to show how the padding token is appended to the sequence to get a sequence length of 38, and that the padding tokens map to the same unique token representation. Furthermore, in figure 16, there are two occurrences of the word "to", where it also becomes apparent, that they have the same unique token representation.

[8194, 27728, 31034, 33290, 22578, 33465, 23724, 16666, 33465, 31143, 31320, 28268, 27701, 33247, 28647, 16053, 22, 16916, 17350, 7925, 32880, 32986, 18239, 23556, 25, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

⇕

['Iranian', 'officials', 'say', 'they', 'expect', 'to', 'get', 'access', 'to', 'sealed', 'sensitive', 'parts', 'of', 'the', 'plant', 'Wednesday', ',', 'after', 'an', 'IAEA', 'surveillance', 'system', 'begins', 'functioning', '.', '<PAD>', '<PAD>', '<PAD>', '<PAD>', '<PAD>', '<PAD>', '<PAD>', '<PAD>', '<PAD>', '<PAD>', '<PAD>', '<PAD>', '<PAD>']

Figure 16: Example of mapping between a tokenized sequence and its real string value. It comes to show the padding with the padding token *<PAD>* is carried out. From the sequence it is also visible that two occurrences of "to" has the same unique encoding.

Now, the *Word_ index* and *POS_ index* from figure 15 are going to be input values for the neural networks, whereas *Tag_ index* is the target value. In that case, the *Tag_ index* was encoded into binary class matrices, meaning that every encoded NE-tag in a sequence is transformed into an encoded sequence itself with the length of all unique NE-tags. So to wrap it up, a sequence of tokenized NE tags had the length of 38 due to the decision boundary. Each element is transformed into its own binary class, as depicted in table 11, so the sequence with length 38 now consisted of 38 new sequences, each with a length of 17, which is the amount of unique NE tags overall. The reason behind the encoding of tokenized NE tags to binary class matrices is described in *5.3.1.4 Crossentropy loss function*.

Tag	**Tokenized tag**	**Binary class matrix**
O	0	[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
B-art	1	[0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
I-art	2	[0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
B-eve	3	[0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0]
I-eve	4	[0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0]
B-geo	5	[0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0]
I-geo	6	[0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0]
B-gpe	7	[0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0]
I-gpe	8	[0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]
B-nat	9	[0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0]
I-nat	10	[0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0]
B-org	11	[0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0]
I-org	12	[0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0]
B-per	13	[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0]
I-per	14	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0]
B-tim	15	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0]
I-tim	16	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]

Table 11: Visualization of tokenization and binary class matrix transformation of NE-tags, also known as one-hot encoding. The binary class matrix has the length of the sum of all NE tags, and each element is an index representation of a NE tag. Considering the NE tag *B-geo*, the tokenized tag value 5 is the index value in the binary class matrix, where there has to be a 1, meaning the matrix represents that specific NE tag.

Last but not least, the dataset is split into a 60% training partition and 40% testing partition. From the 60% training partition, a 20% validation partition is created during fitting for all neural networks.

5.3 Implementation of neural networks

5.3.1 Prerequisites

Leading up to training the neural networks on the processed data from *5.2 Feature engineering*, a few prerequisites are set beforehand. The prerequisites cover embedding of preprocessed data and implementations of callbacks in the training process which are crucial to limiting overfitting, reducing training time and getting better predictions.

5.3.1.1 Token embedding

During the steps in *5.2 Feature engineering*, viable inputs were generated, so the inputs consisted of sequences of tokenized sequences, padded to a length of 38. These preprocessing steps were required prior to embedding, since the embedding layer only takes input of equal length. Additionally, the lesser the length of sequences, the lesser the training time.

The embedding layer, which is a sequence processing layer, vectorizes each element in a tokenized sequence to a fixed dimensionality. Therefore after embedding, every tokenized value of a word can be mapped to a vector of the fixed dimensionality. During training, weights of the layer are updated, as similiar semtantics among tokenized words a acquired.
During embedding while training the neural network, the padding values for all sequences are masked out, meaning that along the output from the embedding layer, a mask tensor is given along too, which specifies which values should be processed and which values should be ignored. What important is, that the tokenized padding value must be 0 for the masking to work properly.

5.3.1.2 Learning rate scheduler with exponential decay

As part of the training process of a neural network, Keras allows for adding callback functions to both monitor the training process, but also to dynamically set values with respect to epoch. Therefore, a customized learning rate scheduler is injected in the training process as a callback, allowing for dynamically setting the learning.

During training of all neural networks, a learning rate scheduler with exponential decay is implemented as in equation 1.

$$\text{learning rate} = lr_0 \cdot 0.1^{epoch/s} \tag{1}$$

For every epoch going by, with lr_0 being the initial learning rate value and s being the step size, the number of epoch is injected in the learning rate scheduler, calculating a new learning rate, which then is returned to the optimizer. The impact of the learning rate scheduler can be seen in *7.1 Linguistic analysis: Results*. The values set for the learning rate scheduler are described in *5.3.2 Architectural composition*.

5.3.1.3 Early stopping

Another callback allowed by Keras is early stopping. Early stopping is implemented for each neural network, which monitors the loss of the validation partition, opting for as low a loss value as possible without overfitting. Its given a patience, meaning that if the validation loss begins to increase or plateau, the patience indicates how many epochs will be allowed before returning the best weights for the optimal model. The patience is set to three to avoid unnecessary training time, based on observed behavior of many training periods. The values for early stopping are described in *5.3.2 Architectural composition*.

5.3.1.4 Crossentropy loss function

A crossentropy loss function is used, due to the target values being transformed into binary class matrices. To predict probability distributions of the binary class matrices, the crossentropy loss function is required, since it computes the crossentropy loss between the target value and predicted value, which is used when there is two or more target classes. Considering *B-org* in table 11, its binary class matrix is [0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0]. When predicting a NE-tag for a word annotated *B-org*, this would be 0% loss, due to correctly bitwise prediction. In combination with a softmax activation function in the last hidden layer (further described *5.3.2.1 LSTM* under the section *Time distributed layer*), a prediction would look like the class matrix function for *B-org*, however the softmax activation function would distribute the predicted probability for each NE-tag over the binary class matrix, where the sum of the sequence is 1. So a combination of the transformation of the tokenized NE-tags to binary class matrices, using a crossentropy loss function and a softmax activation function on the last hidden layer, makes it possible to predict a probability distribution over all possible NE-tags for each element in an input sequence.

5.3.2 Architectural composition

As elaborated in *4.5 Neural network architectures*, this report will set out to explore RNN's and CNN's with single and multiple input to perform multi-label classification. Especially the ability of LSTM networks, and how to take advantage of its long-term memory will play a significant role. Single-input networks will have preprocessed sequences of words as input, whereas multi-input networks will have preprocessed sequences of words and POS tags as input. At this point, it is expected that the preprocessing from *5.2 Feature engineering* and setup from *5.3.1 Prerequisites* is conducted, so table 12 is true.

Prerequisites		
Learning rate scheduler:		
	Initial learning rate, lr_0:	0.001
	Step size, s:	10
Early stopping:		
	Patience:	5
	Monitor:	validation loss
	Mode:	Minimal loss
	Restore best weights:	true
Input layer:		
	Input dimensionality:	*batch size* $\times$ 38
Embedding layer:		
	Output dimensionality:	128
	masking paddings:	true
Loss function:		
	Loss:	Categorical crossentropy
Optimizer:		
	Type:	Adam

Table 12: Prerequisites prior to architectural composition of the neural networks. Early stopping is monitoring the validation loss every each epoch and ensuring that is decreasing and opting for a minimal loss value. If the validation loss plateau or increase over five epochs, the network will restore its best weights from a previous epoch.

As stated in *5.3.1.1 Token embedding*, the first hidden layer of each neural network is an embedding layer,

which vectorizes tokenized/encoded words and punctuation into a fixed dimension. The dimension is set to be 128, resulting in a matrix of $batch\,size \times 38 \times 128$ as in figure 17, where 38 is the decision boundary from *5.2.1 Data visualization* and batch size, also annotated *None* in the following figures, is the amount of grouped sentences from *5.2.2 Word tokenization*. In the following sections, LSTM networks, BiLSTM networks, CNN and a mix of BiLSTM and CNN are composed and elaborated upon.

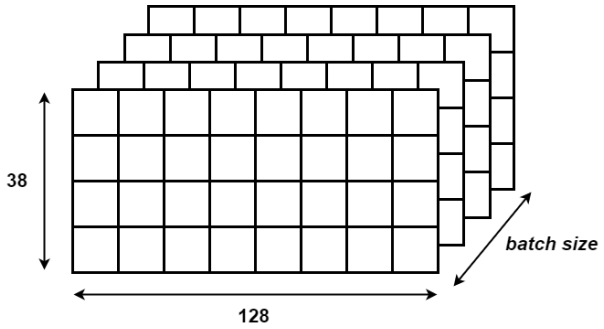
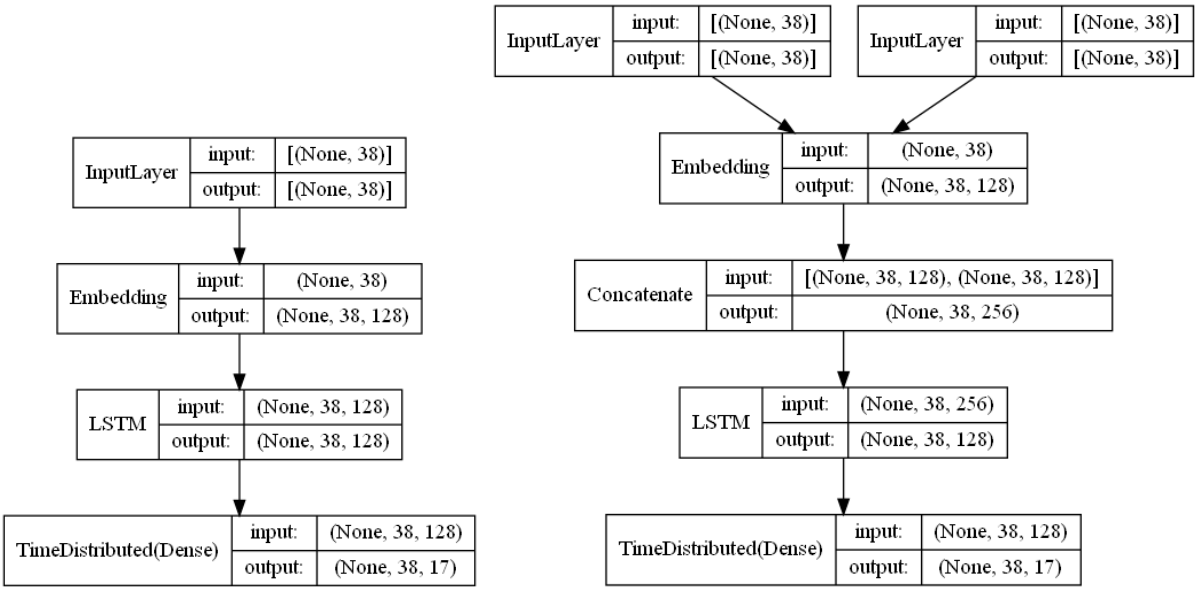


Figure 17: Visualization of $batch\ size \times 38 \times 128$ output from the embedding layer where 38 is the amount of tokens in a sentence, 128 the dimension of the embedding and lastly the batch size.

5.3.2.1 LSTM

Input layer

For the LSTM network, both single-input and multi-input multi-label classification have been composed. Considering figure 18a and 18b, their network layers are almost identical, only difference is that the input varies. In figure 18a, the input layer are the sequences of tokenized words whereas for figure 18b, the input layers are sequences of tokenized words and sequences of tokenized POS-tags. For 18b, both input layers are embedded separately, then concatenated before being inputted to the LSTM layer. All input layers are of the same shape, which is $batch\,size \times 38$, where batch size corresponds to *None*. The input sequences are embedded into vectors of fixed length, so all embeddings are of equal size. As mentioned in *5.3.1.1 Token embedding*, during embedding all padding values are masked out.



(a) Layer architecture of a single-input multi-label classification LSTM neural network.

(b) Layer architecture of a multi-input multi-label classification LSTM neural network.

Figure 18: Layer architecture of LSTM networks of multiple sized input.

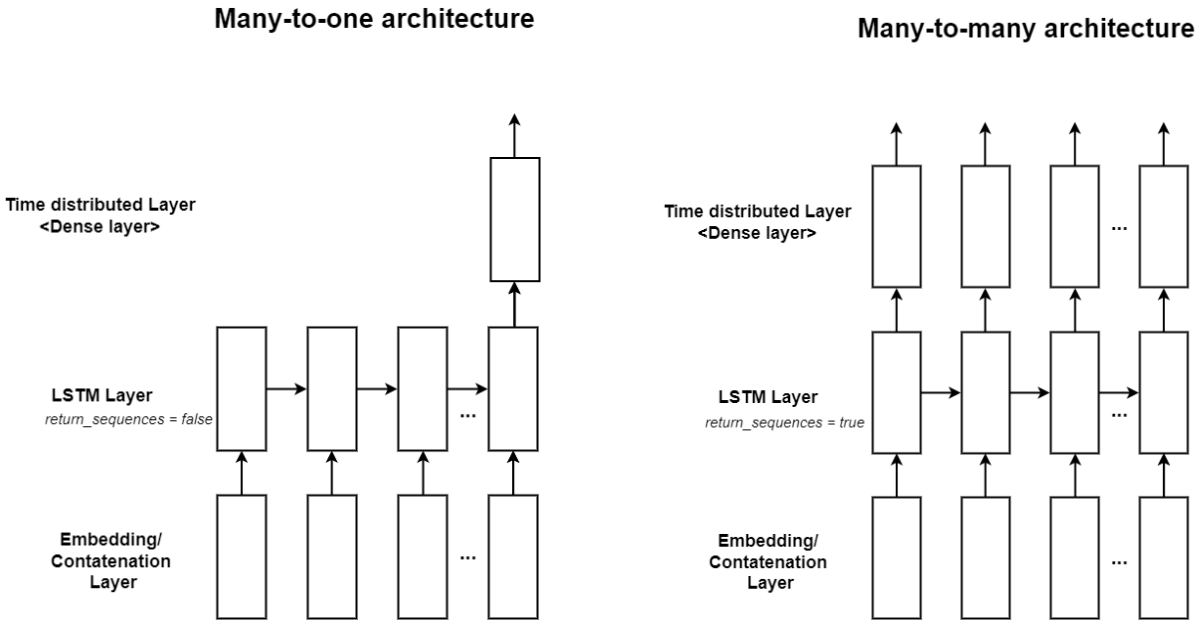
LSTM layer

The LSTM layers are configured almost identically for subfigure 18a and 18b, only difference is the input shape, due to the concatenated input embeddings in 18b. The layer is set to have a dropout of 20% and a recurrent dropout of 20%, which is a dropout executed inside the LSTM layer on the internal states mentioned in *4.5.1 Recurrent neural networks*, whereas the regular dropout is of the input of the LSTM layer. Both are set to avoid overfitting.

The activation function of the LSTM layer remains unchanged and therefore is *tanh*, which helps diminishing the vanishing gradient problem, since its second derivative $tanh''(z)$ can sustain longer before reaching zero [46].

Time distributed layer

The last layer is a time distributed layer wrapping a dense layer that can be applied to every temporal slice of the input, meaning every element in a sequence input. The output will then be a sequence for every given sequence as input from the LSTM layer as depicted in 19b. Since the parameter *return_sequences* is set to true for the LSTM layer, every hidden state at time t will be returned, allowing for a many-to-many architecture when applying a time distributed layer afterwards. If the *return_sequences* was set to false instead, only the last hidden state from the LSTM layer will be returned and the use of a time distributed dense layer would not be necessary, and a dense layer would be appropriate.



(a) Many-to-one architecture for LSTM network with a time distributed layer wrapping a dense layer as last hidden layer. *return_sequences* is set to false.

(b) Many-to-many architecture for LSTM network with a time distributed layer wrapping a dense layer as last hidden layer. *return_sequences* is set to true.

Figure 19: LSTM network architecture with a time distributed layer as the last hidden layer in the network. Subfigure 19a shows a many-to-one architecture, due to *return_sequences* being false, whereas subfigure 19b shows a many-to-many architecture, due to *return_sequences* being true.

Another important feature of the time distributed layer is, that the wrapped dense layer has a softmax activation function. As pointed out in *5.3.1.4 Crossentropy loss function*, the applied softmax function distributes the predicted probability of NE-tags for each element in a input sequence, resulting in a sequence with the length of the sum of all unique NE-tags in the dataset. So to elaborate on this, looking at subfigure 19b, each output token from the time distributed layer contains a sequence of predicted NE-tags, with values varying between 0 and 1, and the sum of all prediction in the output sequence is 1. Therefore, the output at time t is the prediction of each NE-tag measured in percentage for that respective token.

Summary

In table 13, the final setup, in addition to the prerequisites in table 12 can be seen for both the single-input and multi-input LSTM network. The table displays actively set hyper parameters where the majority of default hyper parameters are not displayed.

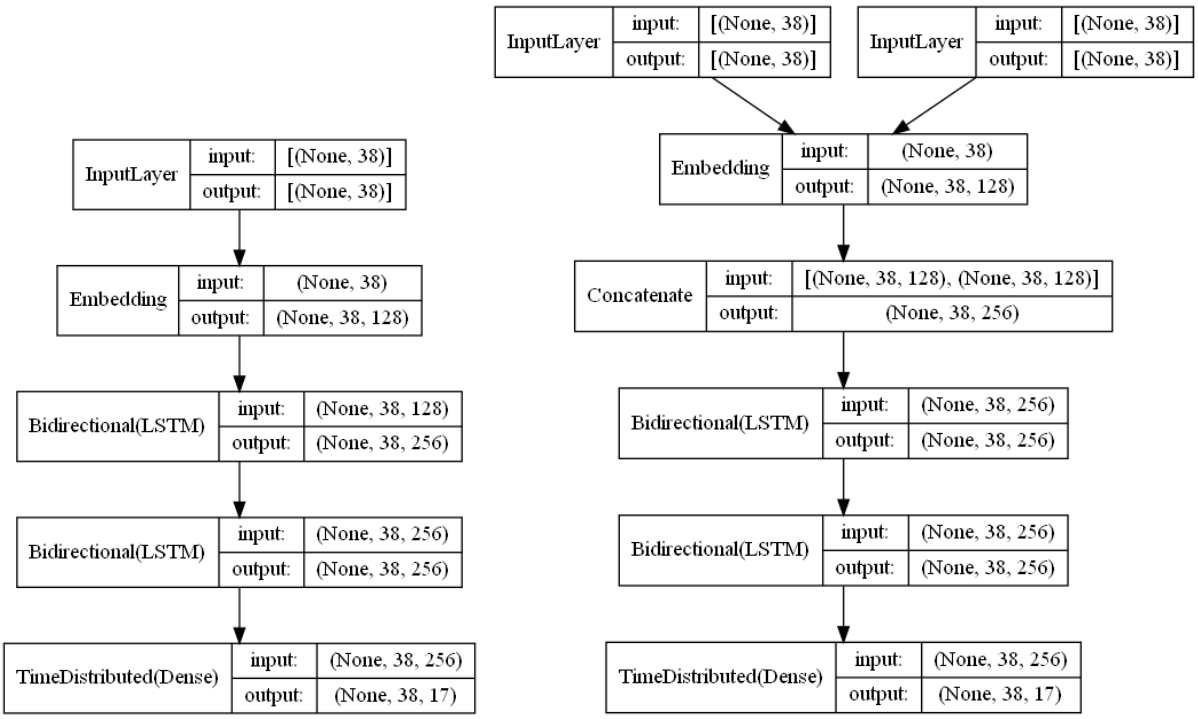
Architectural composition of LSTM network

Single-input multi-label classification			**Multi-input multi-label classification**		
			Concatenate layer:		
				Output dimensionality:	256
LSTM layer:			**LSTM layer:**		
	Dropout:	0.2		Dropout:	0.2
	Recurrent dropout:	0.2		Recurrent dropout:	0.2
	Return sequences:	true		Return sequences:	true
	Output dimensionality:	128		Output dimensionality:	128
	Activation function:	tanh		Activation function:	tanh
Time distributed layer (dense):			**Time distributed layer (dense):**		
	Activation function:	softmax		Activation function:	softmax

Table 13: Architectural composition of both single-input (subfigure 18a) and multi-input (subfigure 18b) LSTM network. Displaying actively set hyper parameters, whereas many hyper parameters are default values for the layers provided by Keras.

5.3.2.2 BiLSTM

The bidirectional LSTM network is very much like the LSTM network in *5.3.2.1 LSTM*, the LSTM layers have just been wrapped by a bidirectional layer and afterwards duplicated, so there are two consecutive bidirectional layers, wrapping a LSTM layer. These changes are visible in figure 20. Since the LSTM network and BiLSTM network are very much alike, their shared layers are not being described again, only the bidirectional layers.



(a) Layer architecture of a single-input multi-label classification BiLSTM neural network.

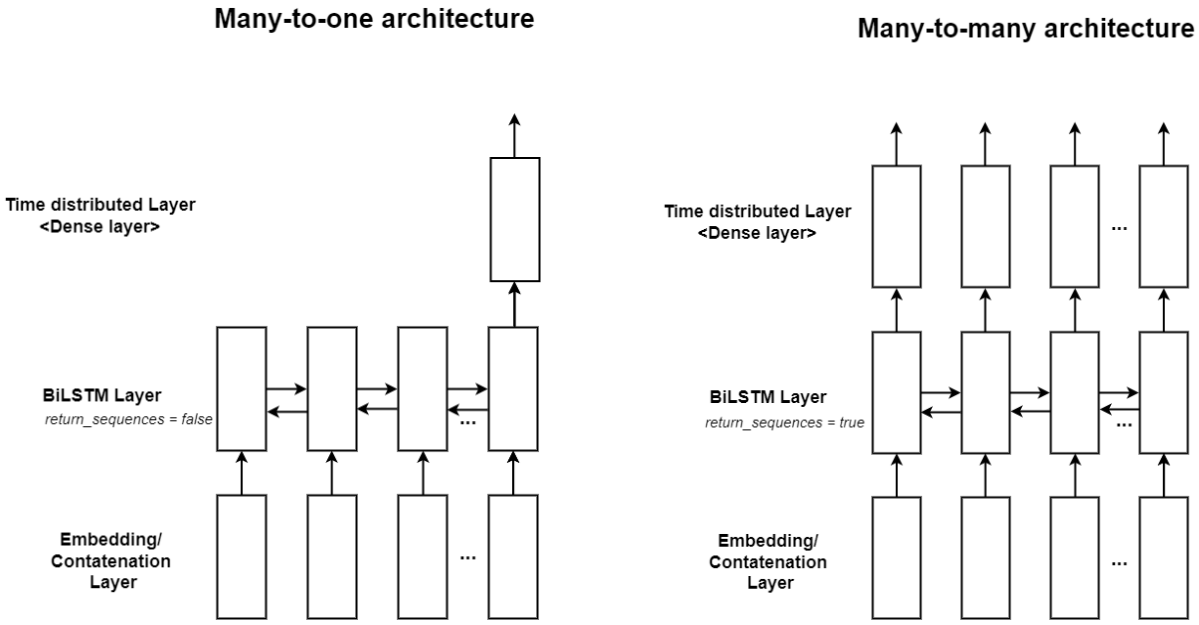
(b) Layer architecture of a multi-input multi-label classification BiLSTM neural network.

Figure 20: Layer architecture of bidirectional LSTM networks of multiple sized input.

Bidirectional layer

The bidirectional layer, wrapping a LSTM layer, allows for forwards and backwards traversal of a sequence, since it applies the LSTM layer in both directions, so that information about a sequence (such as words or POS tags) is preserved both from the past and the future. Therefore, it becomes easier for the model to first of all preserve information from a previous timestamp, but also from a future timestamp, so that when predicting about the current input, the network has knowledge about past and future context.

Figure 21 resembles figure 19 for LSTM; but the LSTM layer is substituted with a bidirectional layer wrapping a LSTM layer, and the forwards and backwards traversal is depicted as well.



(a) Many-to-one architecture for BiLSTM network with a time distributed layer wrapping a dense layer as last hidden layer. *return_sequences* is set to false.

(b) Many-to-many architecture for BiLSTM network with a time distributed layer wrapping a dense layer as last hidden layer. *return_sequences* is set to true.

Figure 21: BiLSTM network architecture with a time distributed layer as the last hidden layer in the network. Subfigure 21a shows a many-to-one architecture, due to *return_sequences* being false, whereas subfigure 21b shows a many-to-many architecture, due to *return_sequences* being true.

Summary

The same parameter configuration is preserved for the LSTM layer (wrapped by the bidirectional layer), meaning that a many-to-many architecture is employed. The configured hyper parameters along with a few default parameters can be seen in table 14.

Architectural composition of BiLSTM network

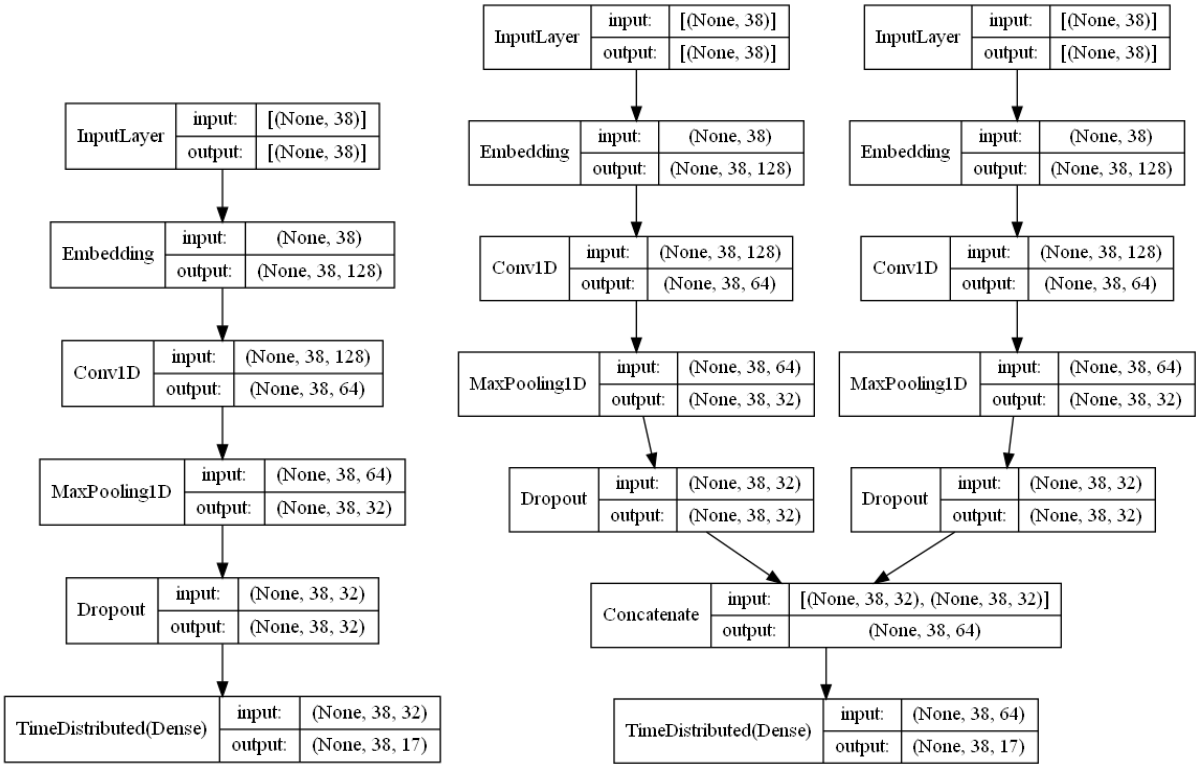
Single-input multi-label classification			**Multi-input multi-label classification**		
			Concatenate layer:		
				Output dimensionality:	256
Bidirectional layer (LSTM):			**Bidirectional layer (LSTM):**		
	Dropout:	0.2		Dropout:	0.2
	Recurrent dropout:	0.2		Recurrent dropout:	0.2
	Return sequences:	true		Return sequences:	true
	Merge mode:	concat		Merge mode:	concat
	Output dimensionality:	128		Output dimensionality:	128
	Activation function:	tanh		Activation function:	tanh
Bidirectional layer (LSTM):			**Bidirectional layer (LSTM):**		
	Return sequences:	true		Return sequences:	true
	Merge mode:	concat		Merge mode:	concat
	Output dimensionality:	128		Output dimensionality:	128
	Activation function:	tanh		Activation function:	tanh
Time distributed layer (dense):			**Time distributed layer (dense):**		
	Activation function:	softmax		Activation function:	softmax

Table 14: Architectural composition of both single-input (subfigure 20a) and multi-input (subfigure 20b) BiLSTM network. Displaying actively set hyper parameters, whereas many hyper parameters are default values for the layers provided by Keras.

Notice that for the bidirectional layer, merge mode is set to *'concat'*, meaning that for the forwards and backwards traversing LSTM layer, an element wise concatenation happens for each LSTM layer at the output. Due to concatenation, the output dimensionality of the wrapped LSTM layers are doubled, as seen in figure 20.

5.3.2.3 CNN

The approach to the CNN networks is somewhat alike the LSTM and BiLSTM where both a single-input multi-label classification and a multi-input multi-label classification network are composed. However for the multi-input solution, instead of concatenating the the inputs after embedding, each input will run separately in their own separate lane of CNN network, where the outputs then will be concatenated in the end before the time distributed layer. The input layer, embedding layer and time distributed layer for the CNN networks are identical to the previous networks, and for further reading on these layers, go to *5.3.2.1 LSTM*. Therefore, this section will focus on CNN specific layers.



(a) Layer architecture of a single-input multi-label classification CNN.

(b) Layer architecture of a multi-input multi-label classification CNN.

Figure 22: Layer architecture of bidirectional LSTM networks of multiple sized input.

Convolutional layer

As depicted in figure 22, the layer after the embedding layer is the convolutional layer, which is called 1D convolutional layer, since its filter(s) will convolve with the input with respect to the temporal dimension [47]. The temporal dimension is also called the dimension of time, which is convenient for sequences of words, since the time dimension is what is considered the batch size. As it can be seen in subfigure 22a for the convolutional layer, the output dimension is reduced by half from $batch\ size \times 38 \times 128$ to $batch\ size \times 38 \times 64$. This is due to the amount of filters (64) which is equal to the output dimensionality. The filters have an equally defined size of 3 that slides across the 38×128 input with respect to the time dimension (batch size).

Pooling layer

The pooling layer is added after the convolutional layer to reduce its output dimensionality. This is done by performing a max pooling operation on the output from the convolution layer, reducing the input from $batch\ size \times 38 \times 64$ to $batch\ size \times 38 \times 32$, due to the pool size being 2. Even though this does reduce the dimensionality and therefore logically compresses the input, one has to remember that the convolution layer outputs the most outstanding features rendered by the filters, from which the pooling layer reduce by half, where that half now consists of the most 32 distinguishable features, which should be the NE's.

Dropout layer

At last, the dropout layer is apppended, which randomly sets some of the input to 0 to prevent overfitting. The dropout rate is set to 20%, so during training 20% of the input will be set to 0.

Summary

In table 15 the final composition of both the single-input and multi-input CNN network is shown. One thing to notice is the hyperparameter *data_format* which is set to *'channels_first'* for the pooling layer. This is important, otherwise it would be the wrong dimensionality of the input that would get reduced. So *data_format* ensures that the output dimensionality of the convolution layer goes from *batch size* $\times 38 \times 64$ to *batch size* $\times 38 \times 32$, if considering subfigure 22a.

Architectural composition of CNN

Single-input multi-label classification			**Multi-input multi-label classification**		
Conv1D layer:			**Conv1D layer:**		
	Filters:	64		Filters:	64
	Kernel size:	3		Kernel size:	3
	Activation function:	Relu		Activation function:	Relu
	Padding:	same		Padding:	same
MaxPooling1D layer:			**MaxPooling1D layer:**		
	Pool size:	2		Pool size:	2
	Data format:	channels_first		Data format:	channels_first
Dropout layer:			**Dropout layer:**		
	Rate:	0.5		Rate:	0.5 & 0.3
			Concatenate layer:		
				Output dimensionality:	64
Time distributed layer (dense):			**Time distributed layer (dense):**		
	Activation function:	softmax		Activation function:	softmax

Table 15: Architectural composition of both single-input (subfigure 22a) and multi-input (subfigure 22b) CNN. Displaying actively set hyper parameters, whereas many hyper parameters are default values for the layers provided by Keras.

5.3.2.4 CNN BiLSTM

The exciting thing about neural networks is that they can be bent and stretched in all kinds of directions, being composed of all kinds of layers, different network architectures, as long as the dimensionality of the inputs and outputs throughout the network is aligned correctly. This allows for indefinite many combinatory network compositions. Therefore, this section will attempt with two different network compositions that feature both CNN and BiLSTM. As the previous sections, both single-input and multi-input multi-label classification is explored.

Sequential CNN BiLSTM

The first of the two CNN BiLSTM networks composed, is a sequentially layered single-input multi-label classification network as depicted in figure 23.

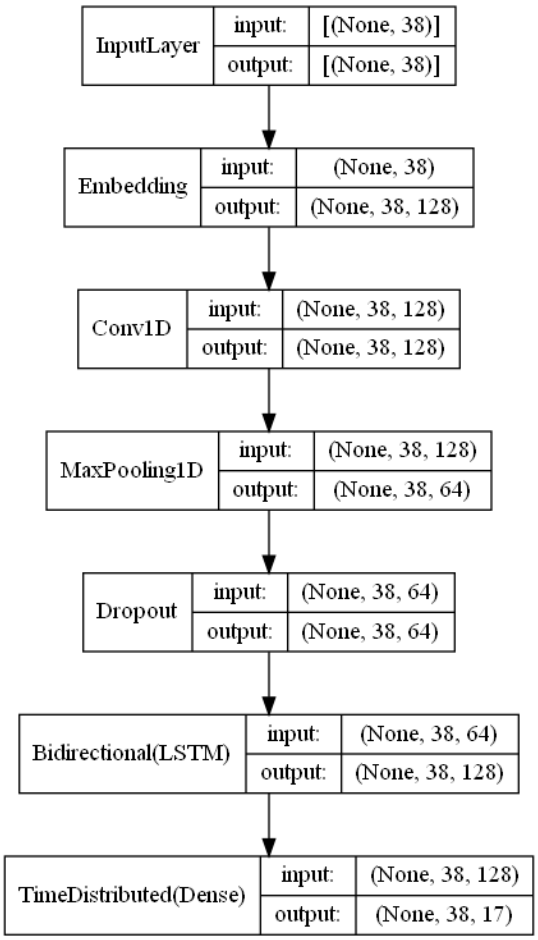


Figure 23: Layer architecture of a single-input multi-label classification CNN BiLSTM.

First and foremost, the network is composed of a input layer, embedding layer and time distributed layer similar to the previous single-input networks in *5.3.2.1 LSTM*, *5.3.2.2 BiLSTM* and *5.3.2.3 CNN*. Afterwards, the network is composed of a convolutional layer, max pooling layer and dropout layer similar to the composition of the networks in *5.3.2.3 CNN*. At last, a bidirectional layer, wrapping a LSTM layer, is appended.

Concurrent CNN BiLSTM

The second of the two CNN BiLSTM networks composed, is a partial concurrently layered multi-input multi-label classification network as depicted in figure 24.

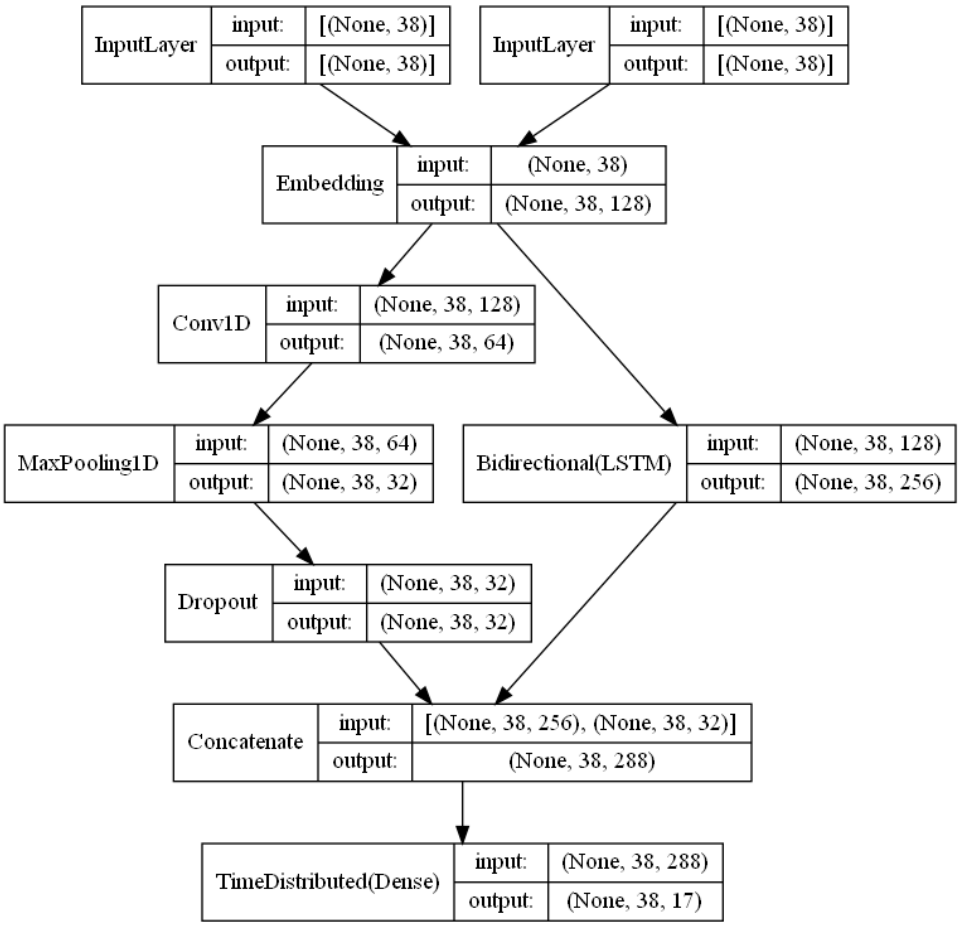


Figure 24: Layer architecture of a multi-input multi-label classification CNN BiLSTM.

This network is somewhat similar to the sequentially layered single-input CNN BiLSTM network. As already described, the input is sequences of tokenized words and POS tags, but for this network, these inputs are handled separately by either CNN or BiLSTM. Both inputs are embedded, but from then on, the tokenized POS tags are going through the CNN layers and the tokenized words are going through the BiLSTM layer. Afterwards, the output from the dropout layer in the CNN lane is concatenated with the output from the bidirectional layer in the BiLSTM lane.

Summary

Both the sequential and concurrent implementation of NN could prove exciting networks, since the CNN layers will recognize the most outstanding features in the tokenized POS tags and the bidirectional layer will utilize its memory sections to recognize patterns in the sequences of tokenized words. The combined, perhaps a satisfying result is achieved. In table 16, the actively set hyper parameters can be seen, along with some of the default parameters for both CNN BiLSTM networks.

Architectural composition of BiLSTM

Single-input multi-label classification			**Multi-input multi-label classification**		
Conv1D layer:			**Conv1D layer:**		
	Filters:	64		Filters:	64
	Kernel size:	3		Kernel size:	3
	Activation function:	Relu		Activation function:	Relu
	Padding:	same		Padding:	same
MaxPooling1D layer:			**MaxPooling1D layer:**		
	Pool size:	2		Pool size:	2
	Data format:	channels first		Data format:	channels first
Dropout layer:			**Dropout layer:**		
	Rate:	0.2		Rate:	0.2
Bidirectional layer (LSTM):			**Bidirectional layer (LSTM):**		
	Dropout:	0.2		Dropout:	0.2
	Recurrent dropout:	0.2		Recurrent dropout:	0.2
	Return sequences:	true		Return sequences:	true
	Merge mode:	concat		Merge mode:	concat
	Output dimensionality:	256		Output dimensionality:	256
	Activation function:	tanh		Activation function:	tanh
			Concatenation layer:		
				Output dimensionality:	64
Time distributed layer (dense):			**Time distributed layer (dense):**		
	Activation function:	softmax		Activation function:	softmax

Table 16: Architectural composition of both single-input (figure 23) and multi-input (figure 24) CNN BiLSTM. Displaying actively set hyper parameters, whereas many hyper parameters are default values for the layers provided by Keras.

6 Sentiment analysis

The SA, commonly used to determine if a input is positive, negative or neutral, approach will be adapted to determine the gender polarization in a sentence. In this section the methodology of choosing and crafting both the ML sentiment algorithm and sentiment corpus will be accounted for. Furthermore it will account for the feature engineering and implementation of neural networks.

6.1 Methodology

This section covers the methodology used to define the input and output of the sentiment ML algorithms, define and craft the sentiment corpus, feature engineer the new corpus and implement the ML algorithms. Lastly, it will cover an evaluation of the methodology's validity and reliability.

6.1.1 Features and classification

The text will be classified on a sentence and word level. On a sentence level the only input feature will be the sentence's text. On a word level the features will consist of the word and strategically picked features. The manually picked features are the word's lemma, sentence number, POS-tag and polarity. These are thought able to help provide a contextual understanding of each sentence, as found in *4.1 Linguistic features*.

Both text depth levels will be classified on binary and multi level. The binary classification output is masculine or feminine, a direct representation of chosen genders of each article category. The multi classification output is value between -1 (masculine) and 1 (feminine), with 0 being neutral. The stepsize from -1 to 1 is 0.1, to limit the number of classes, resulting in 21 different classes.

This results in four different ML algorithms:

Sentiment algorithm 1. Sequence of lemmas as single feature input and binary-classification output.
Sentiment algorithm 2. Sequence of lemmas as single feature input and multi-classification output.
Sentiment algorithm 3. Sequence of lemmas, sentence number, POS-tag and polarity as multiple feature input and binary output.
Sentiment algorithm 4. Sequence of lemmas, sentence number, POS-tag and polarity as multiple feature input and multi-classification output.

6.1.2 Data collection

As a start, each sample of the corpus, *News Category Dataset* [33] analyzed in *4.2.2 Dataset for sentiment analysis*, contains an article link, category and other (irrelevant) key/value pairs. The goal for the end corpus, is to have as many samples as possible, noise-reduced as described in *4.2.2.2 Considerations*, labeled with the gender of the of the targeted audience, as required in *4.2.2.1 Requirements*, in accordance to categories chosen for each gender in *4.2.2.4 Findings*.

The end corpus is defined as:

Sentiment corpus. Each sample consists of a sentence number, word, lemma, POS-tag, the lemma's polarity and gender.

The process to create *Sentiment corpus* has been split into several steps, as seen in figure 25. In this following subsections the steps will be explained in a chronological order, according to the figure.

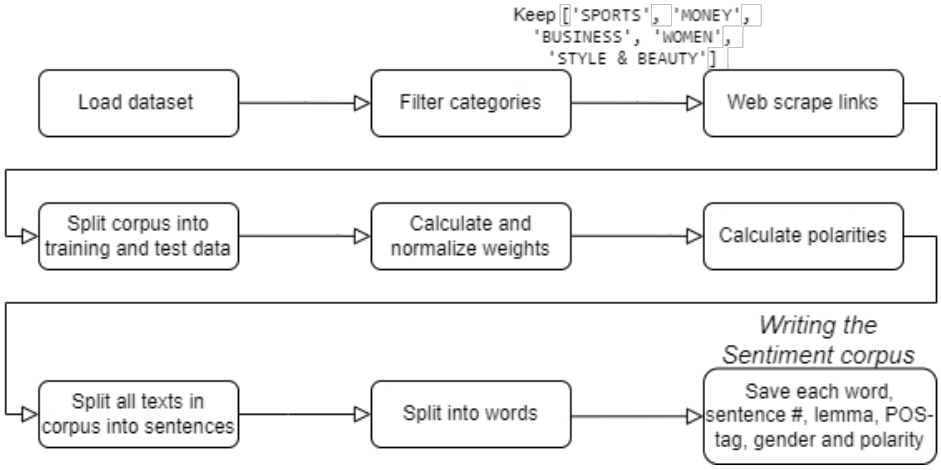


Figure 25: Visualizing the steps in methodology of creating the *Sentiment corpus*. In short, the methodology is to load the corpus with all articles, filter them to match the selected news categories and web scrape the links of the remaining articles to extract each article's body text, creating the foundation for the *Sentiment corpus* with news articles' body text paired with the presumed gender of the target audience. Afterwards it is split, 75% to training and 25% to test, a polarity dictionary is computed from the training data, and lastly the *Sentiment corpus* is written by splitting entire articles' texts into sentences, sentences into words, and saving each word with its sentence number, lemma, POS-tag, gender and polarity as a sample.

Each step is a iteration of the corpus, meaning the dataset is saved as it is, allowing each component to be interchangeable and decoupled of each other.

6.1.2.1 Web scrape links

All samples containing one of the categories *SPORTS*, *MONEY*, *BUSINESS*, *WOMEN* or *STYLE & BEAUTY* are filtered, after which each sample's link, directing to a HuffPost article as shown in *4.2.2.4 Findings*, is web scraped for its body text. If the returned body text isn't empty, then the text is saved and gender-labelled according to category's presumed targeted audience, M (masculine) or F (feminine), as stated in *4.2.2.4 Findings*. The result is the start of a new dataset with even more data, with each sample being a text and a gender.

6.1.2.2 Calculate weights

The gender-label, "M" or "F", is sufficient for binary classification, but something more nuanced is needed for multi classification, being a gender polarity on a continuous scale between masculine and feminine, as stated in the introduction of *4 Analysis*. The solution, thought appropriate, is a dictionary of the most important words used by men and women, each word with a polarity somewhere between -1 (*masculine*) and 1 (*feminine*), with 0 being *neutral*. The weights and polarities are computed only from the training data, to prevent overfitting.

To determine the importance of words, *tf-idf weighting* is used, which combines *term frequency* and *inverse document frequency*. It is a statistical measure used to determine the mathematical significance of words in documents [48], and commonly used in search engines to filter documents from a query term and in word embedding [49]. Throughout this section, a mini-example corpus c consisting of 6 made up short sentences, used as examples of an article's body text, which in reality consists of one or more sentences. Furthermore the corpus is split by gender and computed independently, resulting in two temporary corpus, corpus c_m containing masculine categorized articles and corpus c_f containing feminine

categorized articles, as seen in table 17. Throughout this section multiple tables will be shown, which are all connected, with 17 being the root. This allows for visualization and explanation of each step from corpus, to weights, to polarities.

corpus c	
corpus c_m	corpus c_f
"John Johnson is stronger than strong Jane"	"Jane Smalls is prettier than pretty John"
"I am a very strong person"	"I am a very supporting person"
"I thrive in a competitive setting"	"I thrive in a nurturing setting"

Table 17: Base mini-corpus, examples of gendered sentences. Left is categorized by masculine, right is categorized as feminine.

Each article is referred as a document d consisting of one or more words. Each document is processed by spaCy, to differentiate all words from one another, identify stop words and named entities. The document is then filtered for special characters and named entities of type *['TIME', 'DATE', 'GPE', 'CARDINAL', 'PERSON', 'MONEY', 'PERCENT']*, since they are thought to not be representative of any gender's language usage. Lastly, each word is lemmatized by spaCy, and the resulting all lowercase lemma is referred to as a term t. Occasionally words will not be lemmatized by spaCy, since lemmatization is designed with recall in mind [20], therefore correct lemmatization are prioritized above number of lemmatization. In such cases the word is still appended.

Table 17

⇓

['be', 'strong', 'than', 'strong']	['be', 'pretty', 'than', 'pretty', 'John']
['I', 'be', 'a', 'very', 'strong', 'person']	['I', 'be', 'a', 'very', 'support', 'person']
['I', 'thrive', 'in', 'a', 'competitive', 'setting']	['I', 'thrive', 'in', 'a', 'nurture', 'set']

Table 18: The base-mini corpus from table 17 is preprocessed, by filtering each document and lemmatizing each word.

To determine a term's importance in a document, a document being a text/article, a mechanism is used to compute a score between a term t and document d, based on the weight of t in d. The chosen weighting scheme is quantitative digest, that assigns the weight based on number of occurrences of t in d, referred to as term frequency $tf_{t,d}$. This type of perceiving a document is called a *bag of words model*, which refers that the order of terms is ignored [50].

Table 18

⇓

[1, 2, 1, 2]	[1, 2, 1, 2, 1]
[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]

Table 19: Term frequency values computed of each term in each doc/sentence of the corpus from table 18.

Using $tf_{t,d}$ it can be assumed that stop words, such "a", "by", "it", etc., would have high scores, but does not contribute in any way to how men and women write. All terms in a document are not equally important, and stop words are not the most important, therefore a mechanism is needed to attenuate the effect of terms that appear too often across all documents in the corpus. Thus, inverse document frequency idf_t, which is the inverse of document frequency df_t, is defined to be the amount of documents

in the collection that contains term t [51]. The rarer a term is, the higher idf_t. It is defined in equation (2).

$$idf_t = log(\frac{N}{df_t}) \tag{2}$$

N being the total number of documents in c_m or c_f and df_t the document frequency. When calculating idf_t each unique term from a corpus is computed, resulting in a dictionary as seen in table 20.

Table 18

⇓

Masculine idf_t dictionary		Feminine idf_t dictionary	
'than', 'very', 'person', 'in', 'thrive', 'setting', 'competitive'	1.79	'than', 'pretty', 'John', 'support', 'very', 'person', 'in', 'thrive', 'set', 'nurture'	1.79
'strong', 'be', 'I', 'a'	1.1	'be', 'I', 'a'	1.1

Table 20: Inverse document frequency values computed of each unique term across each corpus, c_m or c_f from table 18. Equal values are grouped for simplicity, but are not affiliated in any way.

By combining the definitions term frequency and inverse document frequency, a composite weight can be computed for each term in each document, accounting for its significance in a given document. The $tf\text{-}idf$ weighting scheme for term t in document d is given by equation (3) [52].

$$tf\text{-}idf_{t,d} = tf_{t,d} \cdot idf_t \tag{3}$$

When computing $tf\text{-}idf_{t,d}$ for such a small dataset, where every term's $tf_{t,d}$ is 1, the result would equal to idf_t in table 20. Furthermore $tf\text{-}idf_{t,d}$ is highest when t occurs many times in a small number of documents, lower when the term occurs fewer times in a document or occurs in many documents, and lowest when the term occurs in all documents [52].

Generally when using $tf\text{-}idf$, it is used to find the significance of a term in a document, therefore denoted $tf\text{-}idf_{t,d}$, which allows for finding the most relevant document in a corpus according to a query term. No publicly available equation to find the significance of a term in the whole corpus could be found. To solve this issue equation (3) has been further developed, to a first iteration of an equation to solve this very problem. The self-developed equation uses the mean of each term's $tf\text{-}idf_{t,d}$, as seen on equation (4), thereby generalizing each gender's language usage.

$$tf\text{-}idf_{t,c} = \frac{\sum_{n=1}^{|d|} tf_{t,d} \cdot idf_t}{N} \tag{4}$$

N being the total number of documents in c_m or c_f. Using this equation we can compute a masculine and feminine $tf\text{-}idf_{t,c}$ dictionary.

Table 19 & table 20

⇓

Masculine $tf\text{-}idf_{t,c}$ dictionary		Feminine $tf\text{-}idf_{t,c}$ dictionary	
'strong'	0.92	'pretty'	1.19
'be', 'I', 'a'	0.37	'be', 'I', 'a'	0.37
'than', 'very', 'person', 'thrive', 'in', 'competitive', 'setting'	0.3	'than', 'John', 'very', 'support', 'person', 'set', 'thrive', 'in', 'nurture',	0.3

Table 21: $tf\text{-}idf_{t,c}$ values computed of each unique term across each corpus c_m or c_f from table 18, using the term frequencies table 19 and invert document frequencies from table 20. Equal values are grouped for simplicity, but are not affiliated in any way.

As shown in figure 25, showing each step in the data collection methodology, the weights are only computed on the corpus' training data, to prevent overfitting. Many terms of low significance will appear in both the masculine and feminine weight dictionaries close or equal to 0, therefore all weights equal or lower than 0.05 is cut off, since anything lower is thought to be noise. To put it into contrast a value of 1.4 is high, 0.1 is very low, and 0 being the lowest meaning no significance. There is no maximum, but the lowest value possible will always be 0. To prevent bias the length of each dictionary is cut to match each other.

All weights are normalized, to span between 0 and 1. This allows them be used to compute the polarities of each term, since we want a values between -1 and 1. The general formula for normalizing arrays is used, equation (5).

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{5}$$

6.1.2.3 Calculate polarities

No publicly available equation to find a term's polarity on a continuous scale between -1 and 1, representing masculine and feminine respectively, could be found. To solve this issue equation (6) has been self-developed, which is a first iteration, that does it by subtracting masculine weights from feminine weights.

$$polarity_{t,c} = tf\text{-}idf_{t,c_f} - tf\text{-}idf_{t,c_m} \tag{6}$$

Equation (6) is then used to compute the polarity dictionary for the whole corpus.

Table 21

⇓

'pretty'	1.0
'than', 'John', 'very', 'support', 'person', 'thrive', 'in', 'set', 'setting', 'nurture', 'competitive'	0.0
'be', 'I', 'a'	0.03
'strong'	-1.0

Table 22: Polarity dictionary, 1 being most feminine, -1 being most masculine. It is calculated using equation 6 and the $tf\text{-}idf_{t,c}$ dictionary from table 21. Equal values are grouped, but are independent of each other.

6.1.2.4 Writing the *Sentiment corpus*

Lastly, it is all wrapped up into a final dataset, as shown in *25 Visualizing the steps in methodology of creating the Sentiment corpus. In short, the methodology is to load the corpus with all articles, filter them to match the selected news categories and web scrape the links of the remaining articles to extract each article's body text, creating the foundation for the Sentiment corpus with news articles' body text paired with the presumed gender of the target audience. Afterwards it is split, 75% to training and 25% to test, a polarity dictionary is computed from the training data, and lastly the Sentiment corpus is written by splitting entire articles' texts into sentences, sentences into words, and saving each word with its sentence number, lemma, POS-tag, gender and polarity as a sample.*. It is created iterating each sample outputted from *6.1.2.1 Web scrape links*, with each sample containing a text and a gender, the text being the full body text scraped from the article link, and the gender being the presumed targeted audience.

Each text is processed by spaCy to identify all sentences, then they are iterated. Each word in the sentence is iterated. If the word isn't noise in form of a stop word or one of named entity types *['TIME', 'DATE', 'GPE', 'CARDINAL', 'PERSON', 'MONEY', 'PERCENT']*, it is then prepossessed to remove URLs, emails, special characters and HTML tags. Finally if it isn't a empty string after being prepossessed, it is appended to the corpus. Each word is appended along with its sentence number, lemma, POS-tag, gender and polarity, which is 0 if not found in the polarity dictionary.

6.1.3 Feature engineering

This section will cover all preprocessing methods used, when the corpus is first loaded, before it is used to fit a model.

6.1.3.1 Data factorization

The sentiment algorithms should be trained on sequences of words, as described in *3.3.1 MoSCoW analysis*. Therefore the *Sentiment corpus* is transformed, divided by each sentence number, so that each sample consists of a sequence of words, lemmas, POS-tags, genders and polarities.

6.1.3.2 Partition of data

To prevent any bias towards one of the two genders, all samples with gender-label *M* (man) and *W* (woman) is counted. Then they are cut to an equal count of samples.

Most sentences are of different lengths, but input sequences in a ML algorithm are required to be equal. It is solved by padding, but as padding introduces noise for a ML algorithm, it has been chosen to cut off the sentences with the highest word counts. The distribution of length of all sentences are measured, the sentence length in which 80% of sentences fit is found, and a decision boundary is set to disregard all sentences that do not fit within it.

6.1.3.3 Feature tokenization

Keras Tokenizer is fit on all lemma sequences, which updates its internal vocabulary with each unique word, ranked by its term frequency. The closer to 0, the more frequent a word is, 0 is reserved for padding [53]. Each sentence is then transformed to a sequence of integers based on their index in the tokenizer's internal vocabulary. POS-tags are tokenized with same process. Each sequence is then padded with a padding token to match the decision boundary.

For binary classification the gender is encoded to integers, -1 if the gender-label is M and 1 if F, and then one-hot encoded.

For multi-classification the polarity is rounded to one decimal, resulting in a maximum of 21 classes (it goes from -1 to 1 with a stepsize 0.1). Then every polarity is one-hot encoded to binary class matrix, as shown on table 23.

Polarity		Polarity		Polarity
-0.9742		-1.0		[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
-0.9221		-0.9		[0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
-0.8212	⇒	-0.8	⇒	[0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
-0.7195		-0.7		[0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
-0.5976		-0.6		[0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
...		...		...

Table 23: An example shown on all 21 possible polarities for multi-classification, cut short to only show the first 5 for simplicity's sake. First the raw polarities are shown, then rounded to one decimal, and lastly one-hot encoded to binary class matrices.

Then the sequences are padded to match the decision boundary.

6.1.3.4 Token embedding

The tokenized sequences are embedded to a fixed size, stretching each integer to a vector of that size, thereby changing each sequence from 1D to 2D. Embedding is an alternative to one-hot encoding, that represents words in a better way with reduced dimensions. It converts each word to a vector of a defined size with real values, instead of 0's and 1's, which reduces need for large storage capacity and increases model efficiency. To clarify, if one was to one-hot encode all words, the resulting vector size of each would be equal to the number of unique words in the corpus.

The embedding is the hidden layer in the neural network, and the size will be determined in *6.3 Feature engineering*.

6.1.4 Implementation

At this point all input is set, and different layers are appended to the first hidden layer, the embedding layer, of the neural network. One model is composed based on the research in *4.5.3 Comparison*, and used for all 4 sentiment algorithms. The composed model will use biLSTM, which has properties to understand complex contextual correlations.

Furthermore a first iteration of XAI will be tested, that will first calculate an overall polarity for a sentence. No publicly available equation to calculate an overall polarity for a sentence could be found. To solve this issue an equation has been self-developed, equation (8), that calculates a total weight of a sentence's bias. As part of equation (8), equation (7) is defined, which is a first-order logic predicate symbol.

Put in natural language, equation (8) calculates the following: the polarity of a sentence p_s is given by the sum of the predicted polarities of the lemmas at the specified indexes in the sentence s, divided by the amount of lemmas in the sentence s, that has a polarity.

$$P(t) = \{t | t \neq 0.0\} \tag{7}$$

$$p_s = \frac{\sum_{s_i=0}^{|s|} p_{t_{s_i},c}}{\sum_{s_i=0}^{|s|} \exists s_i(P(s_i))} \tag{8}$$

If the overall polarity exceeds a fixed decision boundary, which could be anything the user judges as below or above neutral e.g -0.2 and 0.2, the lemmas with the biggest spikes in polarity are presented.

6.1.5 Evaluation

The methodology of the sentiment algorithms and the selected features is nothing new, and is used in established classic sentiment analyses, determining positive, negative or neutral. Thereby the approach is using tested and iterated methods, known to be effective in classic sentiment analysis, and thereby adapted to gender sentiment analysis.

The methodology of the data collection and processing is mostly self-developed, which is a necessity, since no appropriate data could be found for gender sentiment analysis. All choices made in the process are first iterations of the Sentiment corpus' first iteration, and made based on the research and knowledge gained by reading studies of other gender sentiment analyses.

The choice to remove any special characters and punctuation can be debated, although it is the norm in information retrieval to reduce noise, it isn't always so in sentiment analysis, since a punctuation may carry a sentiment, although not concluded with certainty [54] [55].

6.2 Creating the Sentiment corpus

The methodology of creating the *Sentiment corpus* is explained in detail in *6.1.2 Data collection*, and in this section the actual execution is covered. The code is made as a Python Jupyter Notebook document, found in the source code as *sentiment/data_dataset_producer.ipynb*. The document is split in code cells, each with with a title correlating to the methodology steps shown in figure 25, meaning it should be easy to follow. The code is made to be easily reproduceable, and can be run at any time, only prerequisites is the *News Category Dataset* analyzed in *4.2.2 Dataset for sentiment analysis* located in the source code at *sentiment/datasets/1_newsDataset.json* and all imported external modules installed. Lastly, a custom helper class, called *DataPrepper*, is created to simplify some common functionality, and all self-developed equations.

Firstly, the dataset is loaded, and each sample contains a news category, headline, authors, article web link, short description and publishing date. By the end the *Sentiment corpus* is created, where each sample contains a sentence number, word, lemma, POS-tag, polarity and gender.

	articles
0	{'category': 'CRIME', 'headline': 'There Were 2 Mass Shootings In Texas Last...
1	{'category': 'ENTERTAINMENT', 'headline': 'Will Smith Joins Diplo And Nicky ...
2	{'category': 'ENTERTAINMENT', 'headline': 'Hugh Grant Marries For The First ...
3	{'category': 'ENTERTAINMENT', 'headline': 'Jim Carrey Blasts 'Castrato' Adam...
4	{'category': 'ENTERTAINMENT', 'headline': 'Julianna Margulies Uses Donald Tr...
...	...
200848	{'category': 'TECH', 'headline': 'RIM CEO Thorsten Heins' 'Significant' Plan...
200849	{'category': 'SPORTS', 'headline': 'Maria Sharapova Stunned By Victoria Azar...
200850	{'category': 'SPORTS', 'headline': 'Giants Over Patriots, Jets Over Colts Am...
200851	{'category': 'SPORTS', 'headline': 'Aldon Smith Arrested: 49ers Linebacker B...
200852	{'category': 'SPORTS', 'headline': 'Dwight Howard Rips Teammates After Magic...

Figure 26: The base corpus, *News Category Dataset* [33], *1_newsDataset*.

The base corpus, shown in figure 26 is filtered to only keep categories of *SPORTS*, *MONEY*, *BUSINESS*, *WOMEN* or *STYLE & BEAUTY*, in accordance to to categories chosen for each gender in *4.2.2.4 Findings*.

	category	headline	authors	link	short_descrisption	date
0	WOMEN	Morgan Freeman D...	Mary Papenfuss	https://www.huff...	Both Visa and Va...	2018-05-25
1	WOMEN	The Joy Of Watch...	Emma Gray	https://www.huff...	There's a delici...	2018-05-25
2	WOMEN	The 20 Funniest ...	Hollis Miller	https://www.huff...	"Welcome to adul...	2018-05-25
3	WOMEN	Morgan Freeman A...	Sebastian Murdock	https://www.huff...	Eight people tol...	2018-05-24
4	SPORTS	Jets Chairman Ch...	Ron Dicker	https://www.huff...	"I never want to...	2018-05-24
...	...	...	...	...	...	...
24076	BUSINESS	Positive Custome...	Ernan Roman, Con...	https://www.huff...	"Analysts at Ado...	2012-01-28
24077	SPORTS	Maria Sharapova ...		https://www.huff...	Afterward, Azare...	2012-01-28
24078	SPORTS	Giants Over Patr...		https://www.huff...	Leading up to Su...	2012-01-28
24079	SPORTS	Aldon Smith Arre...		https://www.huff...	CORRECTION: An e...	2012-01-28
24080	SPORTS	Dwight Howard Ri...		https://www.huff...	The five-time al...	2012-01-28

Figure 27: *2_filtered_category* corpus.

Web scraping

An self-developed web scraping algorithm was made, tailored to scrape HuffPost web articles, using *requests* and *bs4*'s *BeautifulSoup*. The algorithm was developed by following few of the samples' links, each directing the browser to a HuffPost web article. The page's HTML elements was inspected using Chrome DevTools, and elements, containing the articles' body text, classes were extracted. The *divs* containing body text was found to be of three variants, e.g in figure 28.

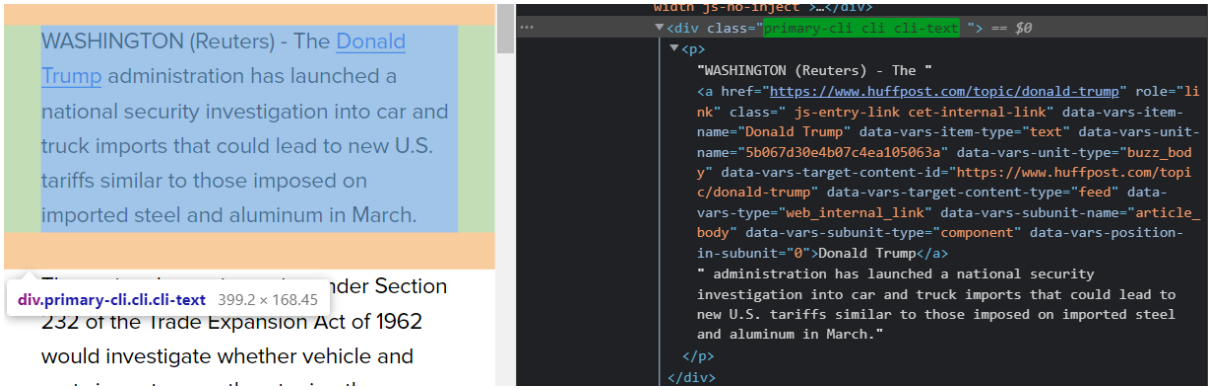


Figure 28: Example of HTML *div* element one of the three class variants. In this article, it can then be expected that each paragraph of the body text is wrapped with an identical div.

Some links was no longer functional, presumably due the links being at least 3 years old, and redirecting the browser to HuffPost's frontpage, but their frontpage didn't contain any elements with one of the three class variants, which no text was extracted. Also some articles was just empty for text, multiple just contained *iframe*s to Facebook, Instagram, etc. 770 articles didn't contain any text for one of those reasons, and was discarded.

The corpus, shown in figure 27, was iterated and each sample's link was web scraped, after each scraping the extracted body text and the category's gender was appended to a new corpus, *3_text_and_gender*, shown in figure 29. By using web scraping the articles the sample size is increased immensely, although 770 articles are discarded due to being textless, the resulting corpus contains 23310 body texts. Initially each of 24080 samples in the base corpus, figure 26, contained a short description of a given article with at least one sentence, but as it will be seen later, now 668.175 sentences are acquired.

	text	gender
0	At least two organizations have deci...	F
1	The best way to watch Harvey Weinste...	F
2	The ladies of Twitter never fail to ...	F
3	Multiple women have accused actor Mo...	F
4	When it comes to New York Jets playe...	M
...	...	...
23306	The Challenge: How do you build a bu...	M
23307	By Steve Tignor, Tennis.com\n\nMELBO...	M
23308	Leading up to Super Bowl XLVI, the m...	M
23309	49ers rookie and rising star Aldon S...	M
23310	Judging by the comments that Magic c...	M

Figure 29: *3_text_and_gender* corpus, each sample containing an article's body text and presumed target audience's gender.

The corpus *3_text_and_gender* is then split for training and test data, the first 75% is used for training. Resulting in 17483 training and 5828 test samples.

Weights and polarities

Based on the training data, the weight for each word is computed using a self-developed algorithm based on publicly known and some self-developed equations, which is covered in detail in *6.1.2.2 Calculate weights*. The essence being the training data is split into a masculine and feminine corpus, each containing texts with their respective gender label. Then a weight, $tf\text{-}idf_{t,c}$, is computed to mathematically define a word's significance across the given masculine or feminine corpus. Since each unique word's lemma is given is found and computed, an comprehensive weight dictionary assembled for each gender.

$tf\text{-}idf_{t,c}$ **dictionary**			
masculline		*feminine*	
company	1.28	woman	1.80
say	1.28	like	0.87
work	1.09	say	0.81
people	0.99	look	0.80
business	0.98	man	0.79
...		...	

Table 24: The 5 highest weighted lemmas for masculine and feminine dictionary, *5_BEFORE_SLICE_word_weight_m* and *5_BEFORE_SLICE_word_weight_f*.

The masculine dictionary consists of 56255 lemmas, and the feminine 46851 lemmas. When distribution of weights is plotted, as shown in figure 30, it is seen that the weights are primarily distributed at or close to zero.

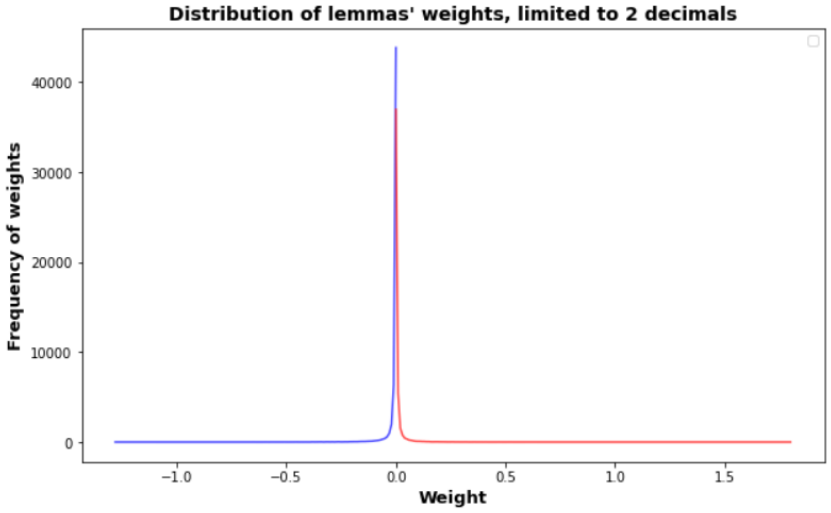


Figure 30: Distribution of weights, blue is masculine weights set negative, red is feminine weights, *5_BEFORE_SLICE_word_weight_m* and *5_BEFORE_SLICE_word_weight_f*.

All weights below 0.05 are of low significance, considered as noise and is cut off, as stated in *6.1.2.2 Calculate weights*, resulting in 2260 masculine and 1505 feminine weights. These weights are cut off to equal length, 1500, to prevent any bias of one gender. To qualify the weights for computing polarities of each term, they are normalized to span between 0 and 1. The resulting distribution of the sliced and normalized weights are shown in figure 31.

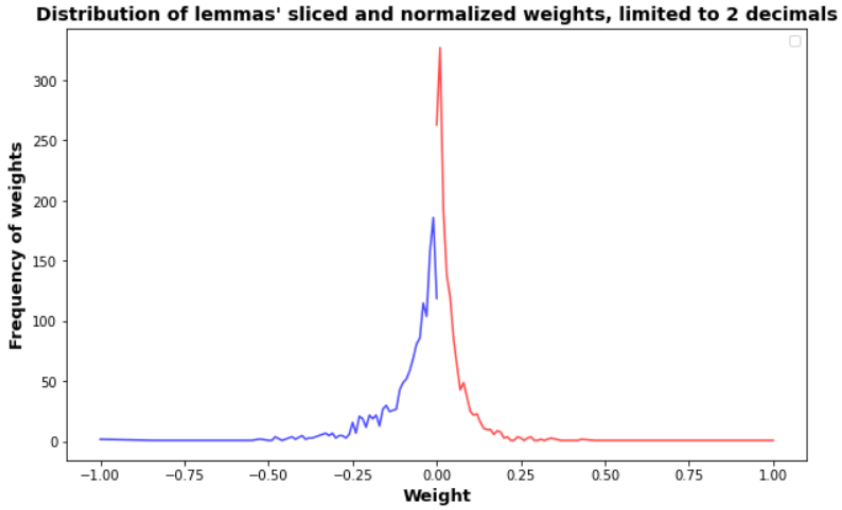


Figure 31: Distribution of sliced and normalized weights, blue is masculine weights set negative, red is feminine weights, *5_word_weight_m_norm* and *5_word_weight_f_norm*.

It is visible that the weights are still close to 0 and primarily below 0.25. It presumably is attributable to the a few highly weighted lemmas in each dictionary, but those can't be discarded to get a better distributed dictionary, since it would be forgery. The polarity is then computed by subtracting masculine from feminine weights, resulting in the polarity dictionary.

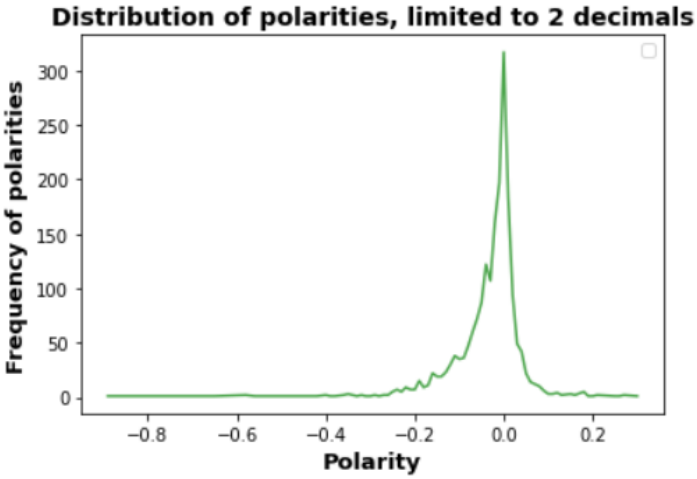


Figure 32: Distribution of polarities, *6_polarity_dict_norm*.

It can be seen on the distribution of polarities, figure 32, that no polarities go lower than 0.9 and higher than 0.3. Compared to the normalized weights, the span is decreased, due to highly significant words appearing on for both genders, equaling them out.

Polarity dictionary			
top masculline		*top feminine*	
company	-0.89	hair	0.30
business	-0.65	woman	0.27
team	-0.58	dress	0.27
game	-0.58	wear	0.26
say	-0.56	fashion	0.25
...		...	

Table 25: The 5 most masculine/feminine lemmas in the polarity dictionary, *6_polarity_dict_norm*.

In table 25 the polarity dictionary's most masculine and feminine lemmas shown, which seem to be heavily influenced by the news categories chosen for each gender.

The sentiment corpus

Lastly, the sentiment corpus can be assembled using *3_text_and_gender* and *polarity_dict_norm*, appending each word of each sentence of each text with its lemma, POS-tag, gender and sentence number.

	Sentence #	Word	Lemma	POS	Polarity	Gender
0	1	organizations	organization	NOUN	-0.223025	F
1	1	decided	decide	VERB	-0.039008	F
2	1	drop	drop	VERB	-0.058436	F
3	1	women	woman	NOUN	0.272143	F
4	1	accused	accuse	VERB	0.006282	F
...	...	...	...	...	...	...
4970462	668175	named	name	VERB	-0.030568	M
4970463	668175	options	option	NOUN	-0.051049	M
4970464	668175	potential	potential	ADJ	-0.096828	M
4970465	668175	landing	landing	NOUN	0.000000	M
4970466	668175	spot	spot	NOUN	-0.005818	M

Figure 33: The final *Sentiment corpus*, stretching 4.970.466 samples, each with a sentence number, word, lemma, POS, polarity and gender. Created from 668.175 sentences from 23310 articles.

6.3 Feature engineering

This section accounts for the execution of the feature engineering methodology covered in *6.1.1 Features and classification* on the *Sentiment corpus*. Following the methods described, the corpus will be preprocessed and readied for all four sentiment algorithms. The four sentiment algorithms, declared in *6.1.1 Features and classification*, are as follows.

Sentiment algorithm 1. Sequence of lemmas as single feature input and binary-classification output.
Sentiment algorithm 2. Sequence of lemmas as single feature input and multi-classification output.
Sentiment algorithm 3. Sequence of lemmas and POS-tag as multiple feature input and binary output.
Sentiment algorithm 4. Sequence of lemmas and POS-tag as multiple feature input and multi-classification output.

From the *Sentiment corpus*, consisting of 4.970.466 samples, 3.000.000 samples from the head is extracted for training dataset and 1.000.000 samples from the tail for test dataset. It is essential to extract test data from the tail to prevent misleading test results, since the polarity dictionary is computed from the first 75% of the *Sentiment corpus* to prevent overfitting.

	Sentence #	Word	Lemma	POS	Polarity	Gender
0	1	organizations	organization	NOUN	-0.223025	F
1	1	decided	decide	VERB	-0.039008	F
2	1	drop	drop	VERB	-0.058436	F
3	1	women	woman	NOUN	0.272143	F
4	1	accused	accuse	VERB	0.006282	F
...	...	...	...	...	...	...
2999995	404384	cool	cool	ADJ	0.037064	F
2999996	404384	factors	factor	NOUN	-0.055300	F
2999997	404384	like	like	ADP	-0.129606	F
2999998	404384	pore	pore	NOUN	0.000000	F
2999999	404384	refiner	refiner	NOUN	0.000000	F

Figure 34: The allocated training dataset.

The training and test datasets are separated for simplicity and to prevent overfitting, but will roughly go through the same preprocessing steps. Both datasets are then grouped in rows per their sentence numbers, *Sentence #*, equalling grouped words, lemmas, POS, polarities and genders, as sequences. The gender label is transformed to a single value representative of the entire sentence for binary classification, and polarities are kept as a sequence for multi classification for each word in a sentence.

	Sentence #	Word	Lemma	POS	Polarity	Gender
0	1	[organizations, decided, d...	[organization, decide, dro...	[NOUN, VERB, VERB, NOUN, V...	[-0.223024829414031, -0.03...	F
1	2	[Women, previously, worked...	[woman, previously, work, ...	[NOUN, ADV, VERB, VERB, NO...	[0.27214283999099703, -0.0...	F
2	3	[response, allegations, Vi...	[response, allegation, Vis...	[NOUN, NOUN, PROPN, VERB, ...	[-0.028053754978966002, 0....	F
3	4	[aware, allegations, Mr]	[aware, allegation, Mr]	[ADJ, NOUN, PROPN]	[-0.01571544139416, 0.0046...	F
4	5	[point, Visa, suspending, ...	[point, Visa, suspend, mar...	[NOUN, PROPN, VERB, NOUN, ...	[-0.22872995655083203, 0.0...	F
...	...	...	...	...	...	...
362024	404378	[Philosophy, Help, Cream]	[philosophy, help, cream]	[NOUN, VERB, NOUN]	[0.0, -0.25124934299454604...	F
362025	404379	[Philosophy, gem, great, b...	[Philosophy, gem, great, g...	[PROPN, NOUN, ADJ, ADJ, NO...	[0.0, 0.0, -0.192702886367...	F
362026	404381	[Benefit, POREfessional, P...	[benefit, porefessional, p...	[VERB, NOUN, NOUN, VERB, N...	[-0.16728961099418502, 0.0...	F
362027	404383	[Pores]	[Pores]	[PROPN]	[0.0]	F
362028	404384	[Pore, Refiner, notch, pro...	[Pore, Refiner, notch, pro...	[PROPN, PROPN, NOUN, NOUN,...	[0.0, 0.0, 0.0, -0.1630039...	F

Figure 35: The training dataset grouped per *Sentence #*.

Present each sample in the training dataset and test dataset represents a sentence and its features as sequences, and to prevent further prevent overfitting both datasets samples' are shuffled. To prevent any bias towards one gender, all samples' genders are counted and samples are cut to equal the counts. To prevent noise, all sentences' lengths are counted and the length 80% of them are within are found, called the decision boundary, which is used when padding sequences.

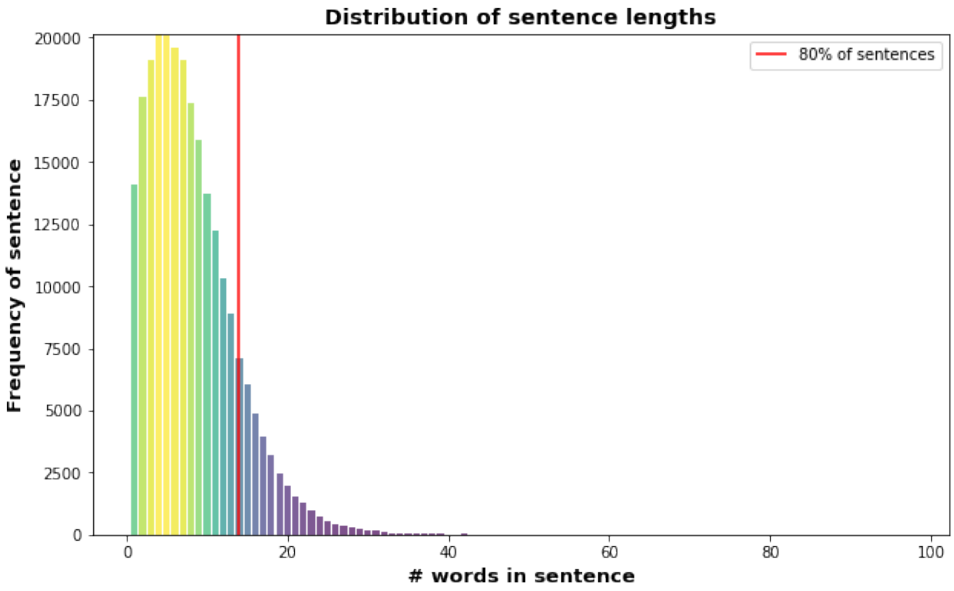


Figure 36: The distribution of sentence lengths and a vertical line marking the decision boundary, 14.

6.3.1 Input engineering

The input engineering is split in two, single input for *Sentiment algoritm 1* and *2*, multiple inputs for *Sentiment algorithm 3* and *4*:

Single feature input consisting of a sequence of lemmas. Keras' text preprocessing tool *Tokenizer* is fit on the training datasets lemmas, meaning it iterates through each sequence of lemmas, iterating through each lemma, building a vocabulary of unique values mapped to integers, ordered by frequency in a descending order. The tokenizer is then used to encode all lemma sequences in the training and test dataset, the resulting sequences are saved as a new feature in each dataset. Lastly, Keras' preprocessing's tool *pad_sequences* is used to pad or truncate all sequences to match the decision boundary, resulting in training input, and vice versa for the test dataset, resulting in test input *X_test*.

Base	['government', 'official', 'consider', 'game', 'vital', 'nation', 'morale']
Encoded	[371, 591, 209, 34, 2398, 596, 5200]
Padded	[371., 591., 209., 34., 2398., 596., 5200., 0., 0., 0., 0., 0., 0., 0.]

Table 26: E.g of a sequence of lemmas being preprocessed.

Multiple features input consisting of a sequence of lemmas and sequence of POS-tags. The procedure for lemmas in *Single input feature* is replicated for lemma sequences and POS-tag sequences. The result is two input features each for training and test, *X_train_lemma* and *X_train_pos*, and *X_test_lemma* and *X_test_pos*.

The vocabulary size is saved for each fitting, since it is the input dimension to be used in the implementation for each input feature.

6.3.2 Output engineering

The ouput engineering is split in two, binary sentiment classification for *Sentiment algoritm 2* and *4*, multiple sentiment classification for *Sentiment algorithm 1* and *3*:

Binary sentiment classification is a target value, which can be either 'M' *masculine* or 'F' *feminine*. Scikit-learn's preprocessing tool *LabelEncoder* is fit on a array with the gender labels, *['M', 'F']*, mapping each value to an unique integer. The *Gender* feature in training and test dataset is then iterated, encoded using the LabelEncoder and one-hot encoded using Keras' utility *to_categorial*, resulting in training target matrix *y_train* and test target matrix *y_test*.

	M	**F**
Base	'M'	'F'
Encoded	0	1
One-hot	[1., 0.]	[0., 1.]

Table 27: Gender values encoded and one-hot-encoded to binary class matrices.

Multiple sentiment classification is a target value, which be one of 21 classes, representing polarities from -1.0 to 1.0 with a stepsize of 0.1. The polarity sequences are iterated rounding each value to one decimal, encoding to integers and one-hot-encoding to binary class matrices. Lastly, the polarity sequence is padded to match the decision boundary, since it has to match the input dimensions.

Tag	**Tokenized tag**	**Binary class matrix**
PAD	0	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
-1.0	1	[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
-0.9	2	[0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
-0.8	3	[0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
-0.7	4	[0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
-0.6	5	[0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
-0.5	6	[0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
-0.4	7	[0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
-0.3	8	[0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0]
-0.2	9	[0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0]
-0.1	10	[0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0]
0.0	11	[0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0]
0.1	12	[0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0]
0.2	13	[0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]
0.3	14	[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0]
0.4	15	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0]
0.5	16	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0]
0.6	17	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0]
0.7	18	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0]
0.8	19	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0]
0.9	20	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0]
1.0	21	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]

Table 28: Visualization of encoding and binary class matrix transformation of polarities.

6.4 Implementation of neural networks

As stated in *4.5 Neural network architectures* RRN's are common for NLP tasks, since they are strong on contextual awareness. BiLSTM, accounted for in *4.5.1 Recurrent neural networks*, is especially strong,

due to its unique capabilities of using long-short-term-memory to compute sentences from the start and end simultaneously. The first iterations of the four sentiment algorithms will essentially use the same model, the biLSTM, reasoned it the projects time span.

6.4.1 Architectural composition

Before implementing the models, it is required that the preprocessing from *6.3 Feature engineering* is executed, and the Keras' callback early stopping is implemented, described in *5.3.1 Prerequisites*.

Prerequisites		
Early stopping:		
	Patience:	5
	Monitor:	validation loss
	Mode:	Minimal loss
	Restore best weights:	true
Input layer:		
	Input dimensionality:	*batch size* $\times$ 14
Embedding layer:		
	Output dimensionality:	128
	masking paddings:	true
Loss function:		
	Loss:	Categorical crossentropy
Optimizer:		
	Type:	Adam

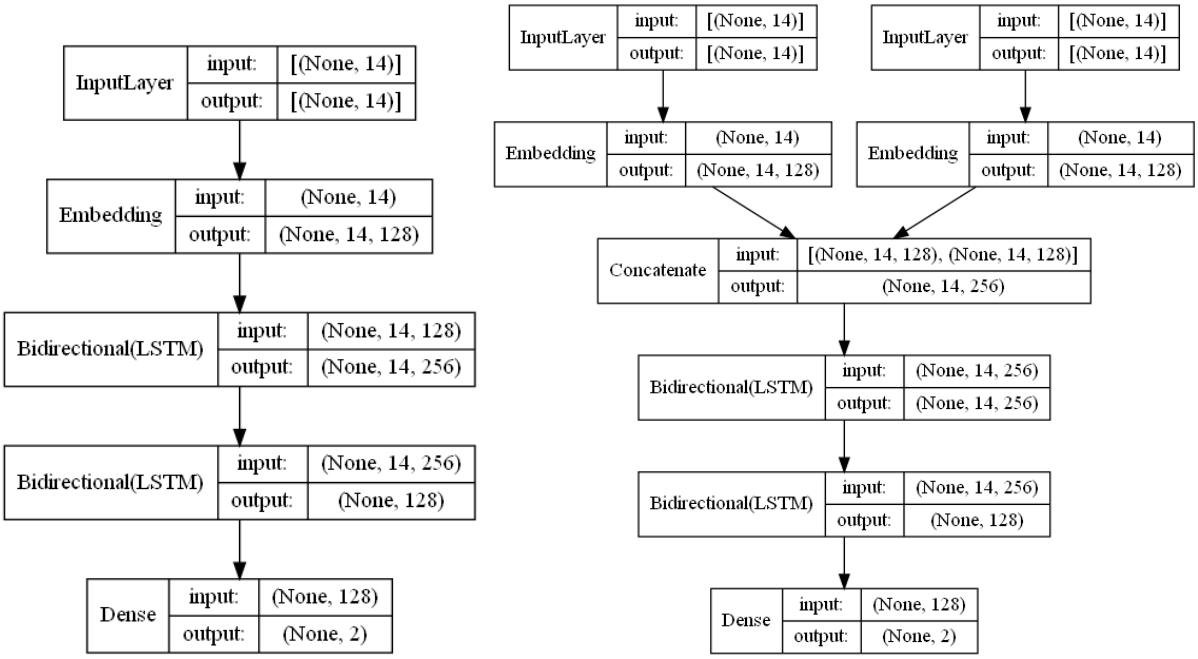
Table 29: Prerequisites prior to architectural composition of the neural networks. Early stopping is monitoring the validation loss every each epoch and ensuring that is decreasing and opting for a minimal loss value.

If table 29 is true, all four sentiment algorithms can now be implemented. The algorithms are grouped by the classification problem, as following.

Sentiment algorithm 1. Sequence of lemmas as single feature input and binary-classification output.
Sentiment algorithm 3. Sequence of lemmas and POS-tag as multiple feature input and binary output.

Sentiment algorithm 2. Sequence of lemmas as single feature input and multi-classification output.
Sentiment algorithm 4. Sequence of lemmas and POS-tag as multiple feature input and multi-classification output.

6.4.1.1 Binary sentiment classification



(a) Layer architecture of *Sentiment algorithm 1* BiLSTM neural network.

(b) Layer architecture of *Sentiment algorithm 3* BiLSTM neural network.

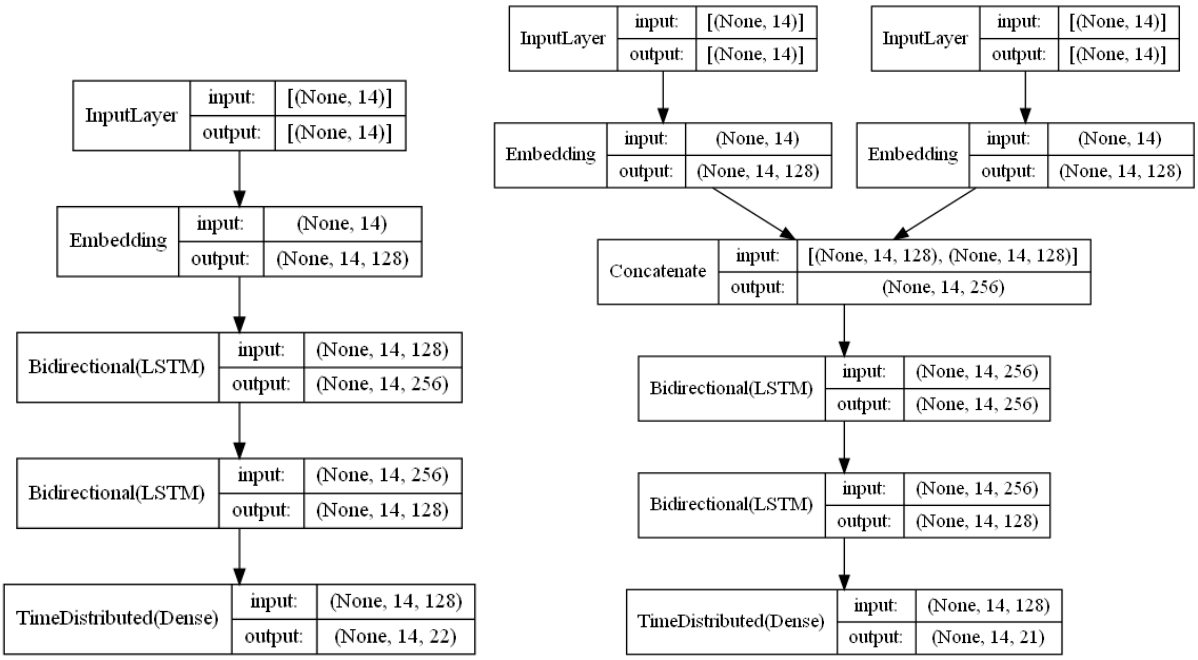
Figure 37: Layer architecture of bidirectional LSTM networks for binary sentiment classification.

Architectural composition of binary sentiment classification network

Single-input: Lemma			**Multi-input: Lemma and POS-tag**		
			Concatenate layer:		
				Output dimensionality:	256
Bidirectional layer (LSTM):			**Bidirectional layer (LSTM):**		
	Dropout:	0.3		Dropout:	0.3
	Recurrent dropout:	0.3		Recurrent dropout:	0.3
	Return sequences:	true		Return sequences:	true
	Merge mode:	concat		Merge mode:	concat
	Output dimensionality:	128		Output dimensionality:	128
	Activation function:	tanh		Activation function:	tanh
Bidirectional layer (LSTM):			**Bidirectional layer (LSTM):**		
	Dropout:	0.1		Dropout:	0.1
	Recurrent dropout:	0.1		Recurrent dropout:	0.1
	Return sequences:	false		Return sequences:	false
	Merge mode:	concat		Merge mode:	concat
	Output dimensionality:	64		Output dimensionality:	64
	Activation function:	tanh		Activation function:	tanh
Dense:			**Dense:**		
	Output dimensionality:	2		Output dimensionality:	2
	Activation function:	softmax		Activation function:	softmax

Table 30: Architectural composition of both single-input (subfigure 37a) and multi-input (subfigure 37b) BiLSTM network for binary sentiment classification. Displaying actively set hyper parameters, whereas many hyper parameters are default values for the layers provided by Keras.

6.4.1.2 Multi sentiment classification



(a) Layer architecture of *Sentiment algorithm 2* BiLSTM neural network.

(b) Layer architecture of *Sentiment algorithm 4* BiLSTM neural network.

Figure 38: Layer architecture of bidirectional LSTM networks for multi sentiment classification.

The output dimension, *(None, 14, 22)*, of the last layer is the 21 unique polarities (-1.0 to 1.0 with stepsize 0.1) added by 1, to account for the unique padding integer.

Architectural composition of binary sentiment classification network

Single-input: Lemma			**Multi-input: Lemma and POS-tag**		
			Concatenate layer:		
				Output dimensionality:	256
Bidirectional layer (LSTM):			**Bidirectional layer (LSTM):**		
	Dropout:	0.3		Dropout:	0.3
	Recurrent dropout:	0.3		Recurrent dropout:	0.3
	Return sequences:	true		Return sequences:	true
	Merge mode:	concat		Merge mode:	concat
	Output dimensionality:	128		Output dimensionality:	128
	Activation function:	tanh		Activation function:	tanh
Bidirectional layer (LSTM):			**Bidirectional layer (LSTM):**		
	Dropout:	0.1		Dropout:	0.1
	Recurrent dropout:	0.1		Recurrent dropout:	0.1
	Return sequences:	true		Return sequences:	true
	Merge mode:	concat		Merge mode:	concat
	Output dimensionality:	64		Output dimensionality:	64
	Activation function:	tanh		Activation function:	tanh
Time distributed layer (dense):			**Time distributed layer (dense):**		
	Output dimensionality:	22		Output dimensionality:	21
	Activation function:	softmax		Activation function:	softmax

Table 31: Architectural composition of both single-input (subfigure 37a) and multi-input (subfigure 37b) BiLSTM network for multi sentiment classification. Displaying actively set hyper parameters, whereas many hyper parameters are default values for the layers provided by Keras.

7 Results

This section will cover the results found in *5 Linguistic analysis* and *6 Sentiment analysis* for all networks composed. The results are gathered from both training results and testing results. Training results cover the progression of networks measured by accuracy and loss, showing the states of networks throughout training. The testing results cover the predictions made by the networks, which will be measured by precision, recall and f1-score from which confusion matrices are constructed, to visualize recall and precision in an intuitive way. These score metrics are dependent on true positives, true negatives, false positives and false negatives, which is deduced from the true and predicted values when fitting a network.

7.1 Linguistic analysis: Results

The results during and after the implementations of the LSTM, BiLSTM, CNN and CNN BiLSTM networks will be presented in this section. The results will be presented in the same chronological order as they were implemented in *5.3 Implementation of neural networks* for the LA. At first, the results during training will be presented and thereafter the results after prediction.

7.1.1 Training results

Measurements of batch accuracy and batch loss were made during training. The reason for measuring accuracy and loss pr batch instead of epoch is, that you get a more detailed view of the training process and how the learning rate scheduler from *5.3.1.2 Learning rate scheduler with exponential decay* has an impact on the training phase. For figure 39, 40, 41 and 42 It is clear how the learning rate scheduler helps getting faster to the optimal accuracy and minimizes training time. Each vertical step represents an epoch, which is where the new learning rate is injected into the optimizer.

Additionally, the early stopping callback is also visible for figure 39, 40, 41 and 42, as the networks stop at different epochs, which is due to monitoring the validation loss, as mentioned in *5.3.1.3 Early stopping*.

Figure 39 and figure 40 depict the batch accuracy and loss, measured against timesteps during training, for the single-input LSTM, BiLSTM, CNN and CNN BiLSTM networks, whereas figure 41 and figure 42 depict the same metrics for the multi-input LSTM, BiLSTM, CNN and CNN BiLSTM networks.

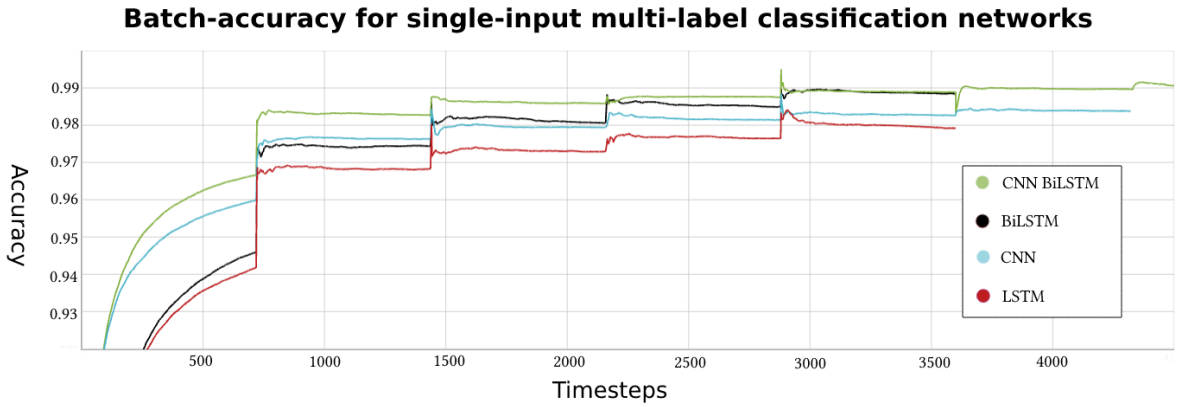


Figure 39: Batch accuracy for single-input LSTM, BiLSTM, CNN and CNN BiLSTM networks measured against timesteps during training.

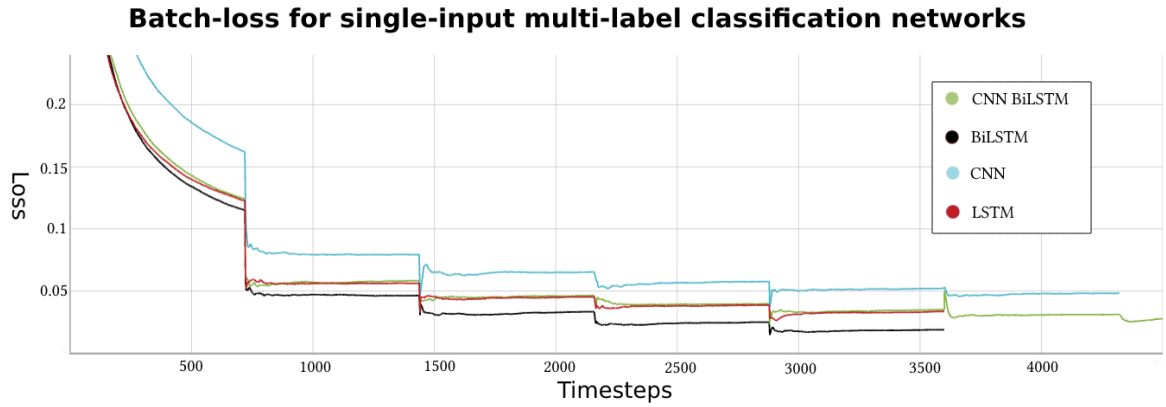


Figure 40: Batch loss for single-input LSTM, BiLSTM, CNN and CNN BiLSTM networks measured against timesteps during training.

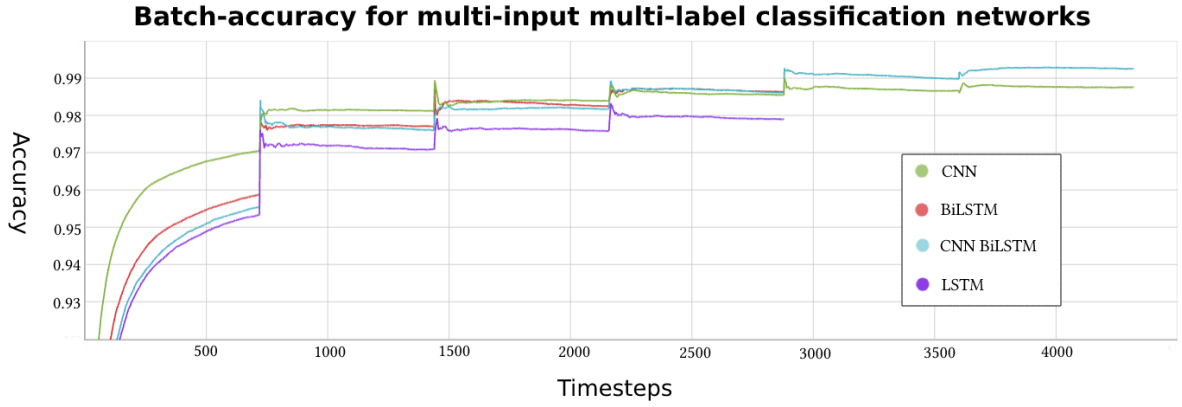


Figure 41: Batch accuracy for multi-input LSTM, BiLSTM, CNN and CNN BiLSTM networks measured against timesteps during training.

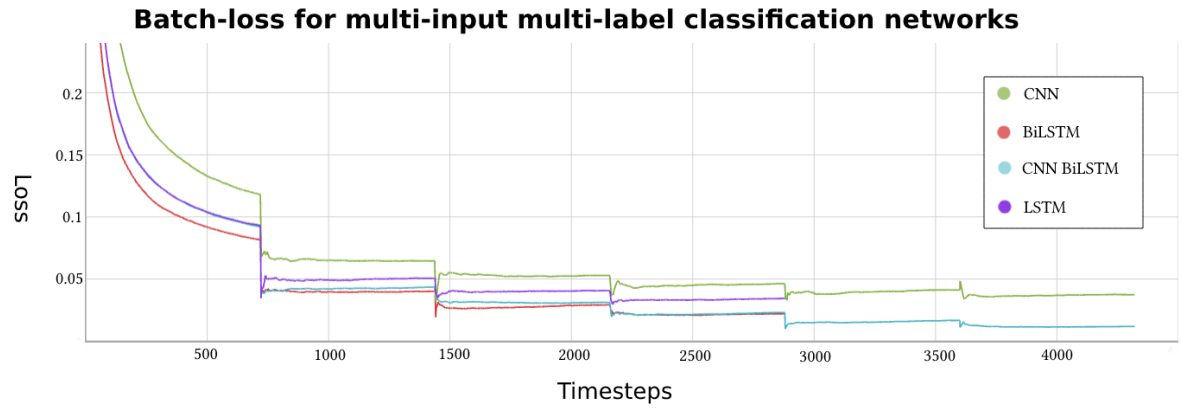


Figure 42: Batch loss for multi-input LSTM, BiLSTM, CNN and CNN BiLSTM networks measured against timesteps during training.

7.1.2 Testing results

The testing results consists of score metrics and confusion matrices that visually represents the predictions against true values for all neural networks.

7.1.2.1 Score metrics

The score metrics are split in two, where precision and recall are measured separately for single-input and multi-input networks. After, a conjoint comparison between single-input and multi-input networks is measured in the f1-score. In table 33 the measurements of precision and recall can be seen for single-input networks and in table 34 the measurements of precision and recall can be seen for multi-input networks. Table 35 shows all measured f1 scores. Each table is colored with either red, orange, yellow or green, which depends in the values in each cell. The colors adhere to the color scheme in table 32.

Red:	[0;0.25[
Orange:	[0.25;0.50[
Yellow:	[0.50;0.75[
Green:	[0.75;1.0]

Table 32: Color scheme for precision, recall and F1 score metrics.

	Score metrics for single-input networks							
	Precision				Recall			
	LSTM	BiLSTM	CNN	CNN BiLSTM	LSTM	BiLSTM	CNN	CNN BiLSTM
O	0.99	0.99	0.99	0.99	0.99	1.00	1.00	1.00
B-art	0.29	0.00	0.00	0.00	0.05	0.00	0.00	0.00
I-art	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
B-eve	0.49	0.66	0.00	0.79	0.28	0.19	0.00	0.23
I-eve	0.38	0.23	0.00	0.50	0.07	0.14	0.00	0.07
B-geo	0.81	0.86	0.80	0.85	0.85	0.87	0.90	0.88
I-geo	0.85	0.84	0.84	0.78	0.70	0.73	0.64	0.77
B-gpe	0.94	0.96	0.95	0.97	0.94	0.94	0.92	0.93
I-gpe	0.94	0.97	0.97	0.96	0.52	0.54	0.46	0.38
B-nat	0.45	1.00	0.00	1.00	0.25	0.01	0.00	0.01
I-nat	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
B-org	0.68	0.78	0.76	0.79	0.64	0.69	0.62	0.65
I-org	0.71	0.81	0.79	0.78	0.78	0.73	0.65	0.73
B-per	0.87	0.83	0.84	0.78	0.74	0.80	0.74	0.83
I-per	0.83	0.84	0.78	0.84	0.89	0.86	0.88	0.86
B-tim	0.92	0.93	0.92	0.88	0.79	0.81	0.82	0.88
I-tim	0.73	0.90	0.90	0.85	0.70	0.60	0.53	0.62

Table 33: Precision and recall for single-input LSTM, BiLSTM, CNN and CNN BiLSTM networks. Color scheme adheres to table 32.

	Score metrics for multi-input networks							
	Precision				Recall			
	LSTM	BiLSTM	CNN	CNN BiLSTM	LSTM	BiLSTM	CNN	CNN BiLSTM
O	0.99	1.00	0.99	0.99	0.99	1.00	0.98	1.00
B-art	0.46	0.21	0.00	0.40	0.06	0.04	0.00	0.03
I-art	0.00	0.00	0.14	0.00	0.00	0.00	0.01	0.00
B-eve	0.62	0.45	0.00	0.76	0.04	0.30	0.00	0.22
I-eve	0.26	0.35	0.00	0.42	0.23	0.27	0.00	0.05
B-geo	0.82	0.85	0.59	0.86	0.88	0.92	0.88	0.89
I-geo	0.82	0.80	0.74	0.81	0.72	0.81	0.58	0.77
B-gpe	0.97	0.98	0.93	0.98	0.94	0.94	0.92	0.92
I-gpe	0.68	0.93	0.81	0.90	0.55	0.58	0.54	0.58
B-nat	0.68	0.37	0.00	0.76	0.29	0.48	0.00	0.20
I-nat	0.57	1.00	0.00	0.26	0.18	0.09	0.00	0.41
B-org	0.70	0.83	0.60	0.78	0.68	0.68	0.52	0.69
I-org	0.71	0.77	0.61	0.84	0.79	0.80	0.60	0.69
B-per	0.83	0.80	0.66	0.78	0.81	0.86	0.71	0.87
I-per	0.84	0.87	0.87	0.87	0.89	0.87	0.38	0.87
B-tim	0.89	0.90	0.72	0.92	0.83	0.89	0.89	0.86
I-tim	0.78	0.82	0.49	0.82	0.68	0.73	0.57	0.73

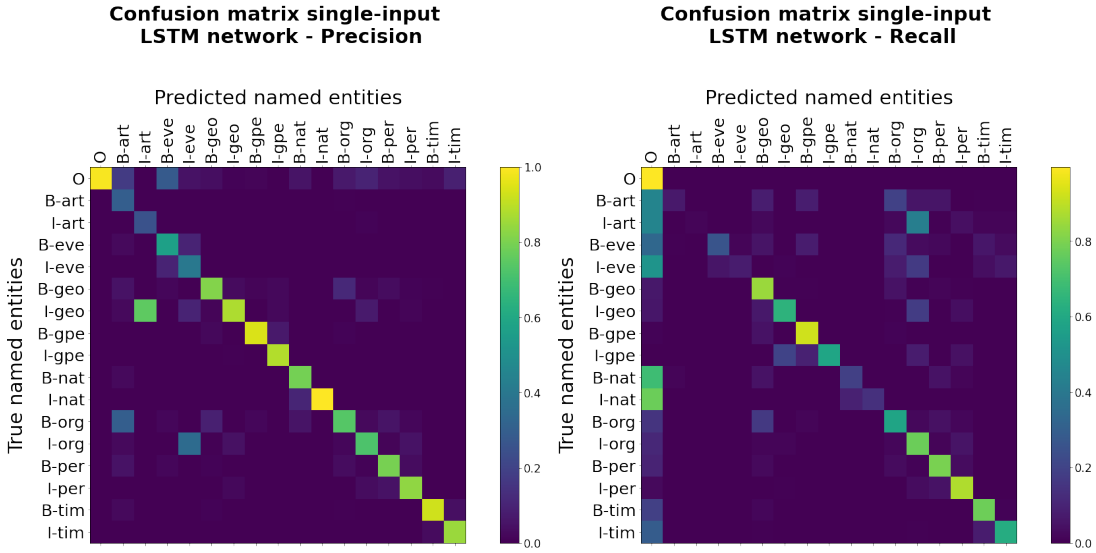
Table 34: Precision and recall for single-input LSTM, BiLSTM, CNN and CNN BiLSTM networks. Color scheme adheres to table 32.

	Score metrics for single-input and multi-input networks							
	F1-score for single-input networks				F1-score for multi-input networks			
	LSTM	BiLSTM	CNN	CNN BiLSTM	LSTM	BiLSTM	CNN	CNN BiLSTM
O	0.99	0.99	0.99	0.99	0.99	1.00	0.98	1.00
B-art	0.09	0.00	0.00	0.00	0.11	0.07	0.00	0.06
I-art	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
B-eve	0.36	0.29	0.00	0.35	0.07	0.36	0.00	0.34
I-eve	0.12	0.17	0.00	0.12	0.25	0.30	0.00	0.08
B-geo	0.83	0.87	0.85	0.86	0.85	0.88	0.71	0.88
I-geo	0.77	0.78	0.73	0.77	0.76	0.81	0.65	0.79
B-gpe	0.94	0.95	0.94	0.95	0.95	0.96	0.92	0.95
I-gpe	0.67	0.69	0.62	0.55	0.71	0.72	0.65	0.71
B-nat	0.32	0.02	0.00	0.02	0.40	0.42	0.00	0.32
I-nat	0.00	0.00	0.00	0.00	0.28	0.17	0.00	0.32
B-org	0.66	0.73	0.68	0.71	0.69	0.75	0.56	0.73
I-org	0.75	0.77	0.71	0.76	0.75	0.78	0.61	0.76
B-per	0.80	0.82	0.79	0.80	0.82	0.83	0.68	0.82
I-per	0.86	0.85	0.83	0.85	0.86	0.87	0.53	0.87
B-tim	0.85	0.87	0.87	0.88	0.86	0.90	0.80	0.89
I-tim	0.71	0.72	0.67	0.72	0.73	0.77	0.53	0.77

Table 35: F1-score for both single-input and multi-input LSTM, BiLSTM, CNN and CNN BiLSTM networks. The colors of each cell can be one of four dependent on the value of the cell. Red is [0;0.25[, orange is [0.25;0.50[, yellow is [0.50;0.75[and green is [0.75;1.00].

7.1.2.2 Confusion matrix

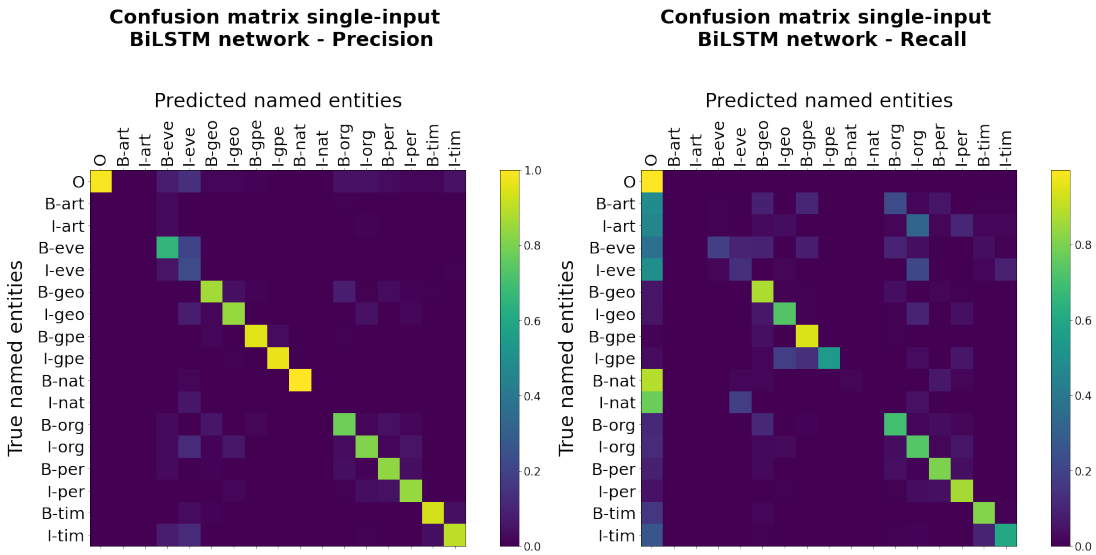
This section presents the results as confusion matrices. Each network, whether its a single-input or multi-input network, is presented with a pair of confusion matrices, one visualizing the precision and the other visualizing recall. The precision and recall is measured between predicted named entities and true named entities.



(a) Confusion matrix showing the precision values for the single-input LSTM network.

(b) Confusion matrix showing the recall values for the single-input LSTM network.

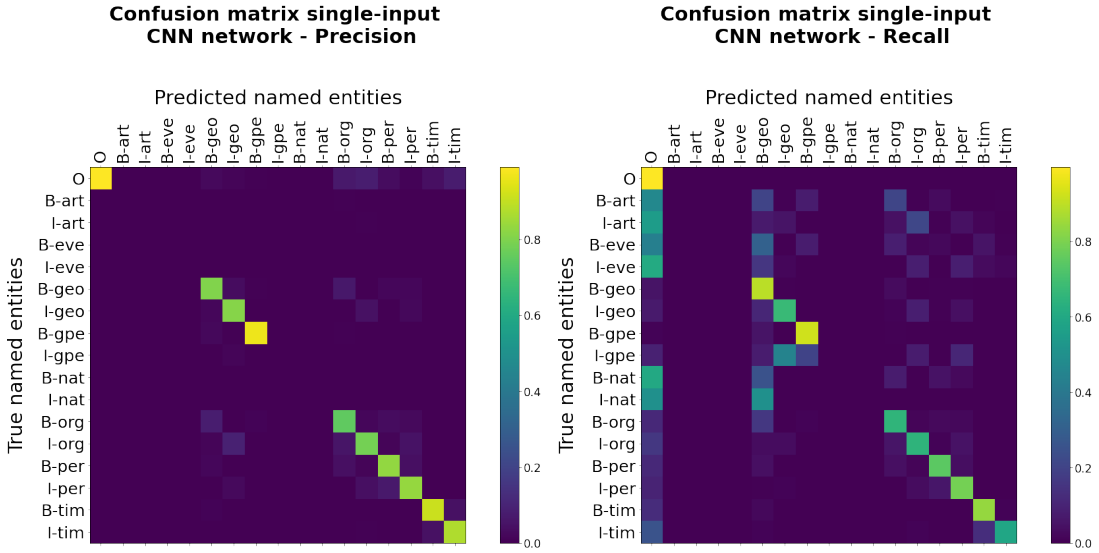
Figure 43: Confusion matrices for single-input LSTM network, where 43a is a visualization of precision and 43b is a visualization of recall.



(a) Confusion matrix showing the precision values for the single-input BiLSTM network.

(b) Confusion matrix showing the recall values for the single-input BiLSTM network.

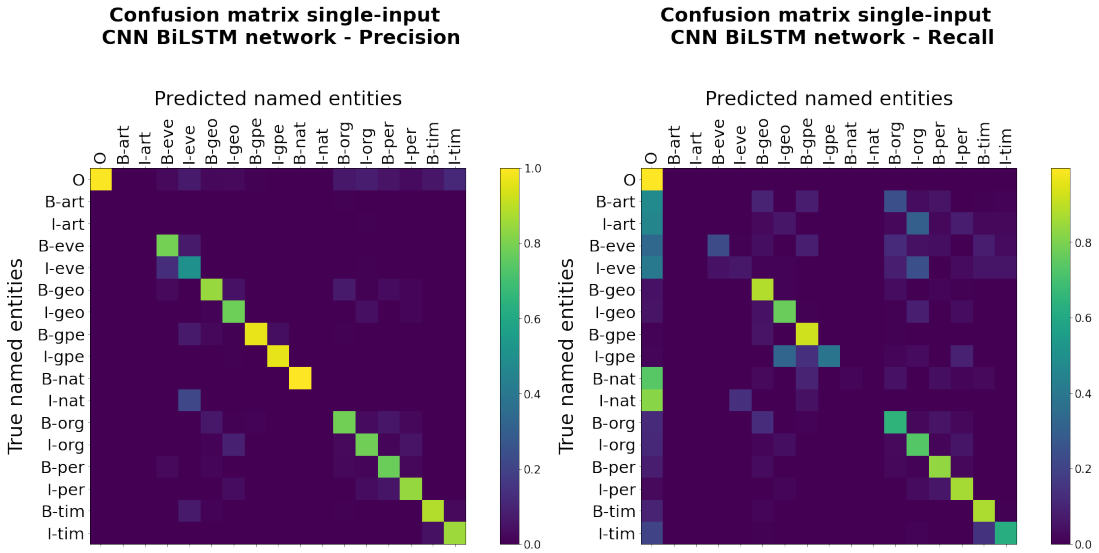
Figure 44: Confusion matrices for single-input BiLSTM network, where 44a is a visualization of precision and 44b is a visualization of recall.



(a) Confusion matrix showing the precision values for the single-input CNN.

(b) Confusion matrix showing the recall values for the single-input CNN.

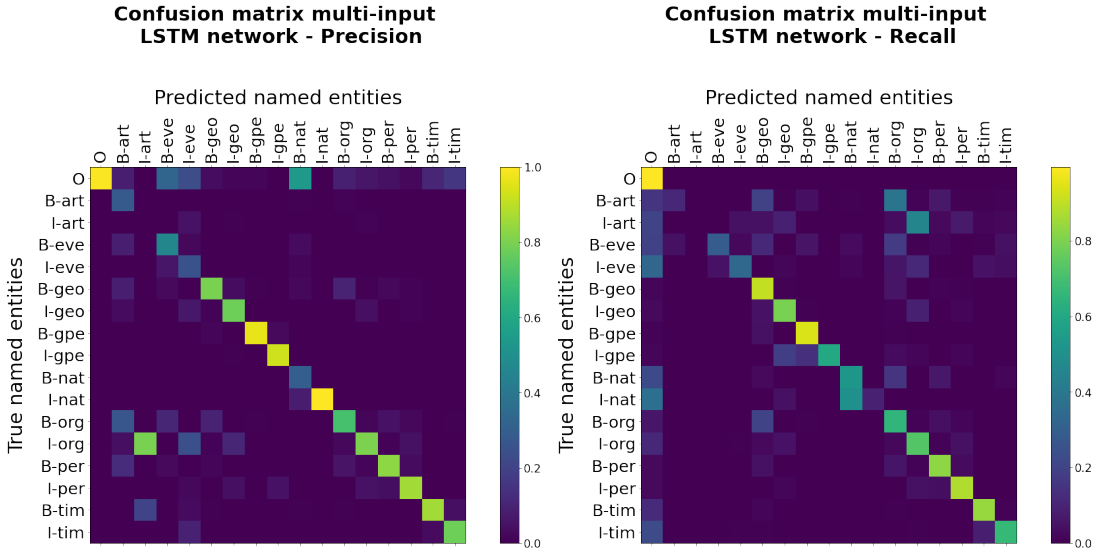
Figure 45: Confusion matrices for single-input CNN network, where 45a is a visualization of precision and 45b is a visualization of recall.



(a) Confusion matrix showing the precision values for the single-input CNN BiLSTM network.

(b) Confusion matrix showing the recall values for the single-input CNN BiLSTM network.

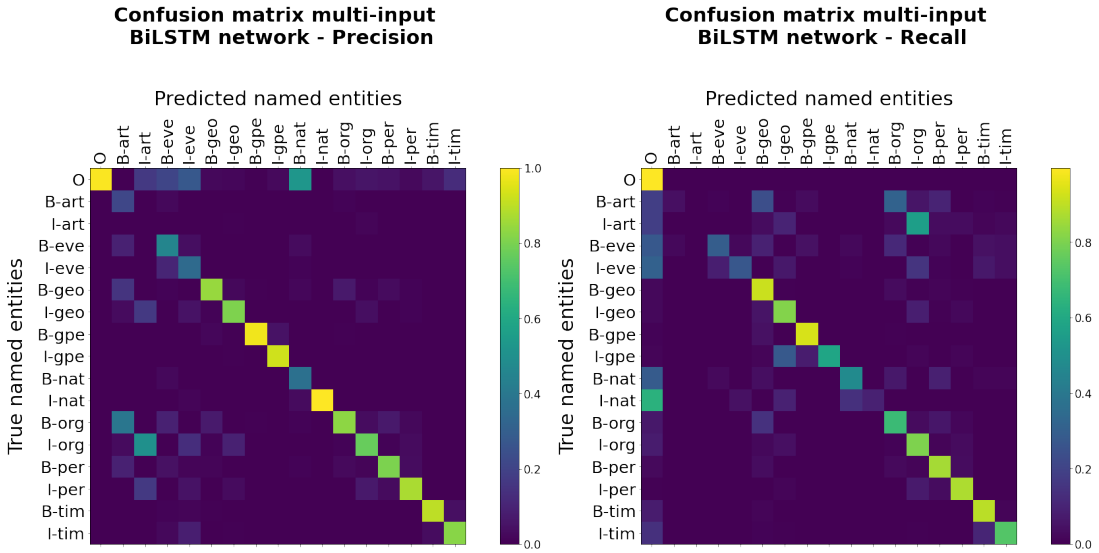
Figure 46: Confusion matrices for single-input CNN BiLSTM network, where 46a is a visualization of precision and 46b is a visualization of recall.



(a) Confusion matrix showing the precision values for the multi-input LSTM network.

(b) Confusion matrix showing the recall values for the multi-input LSTM network.

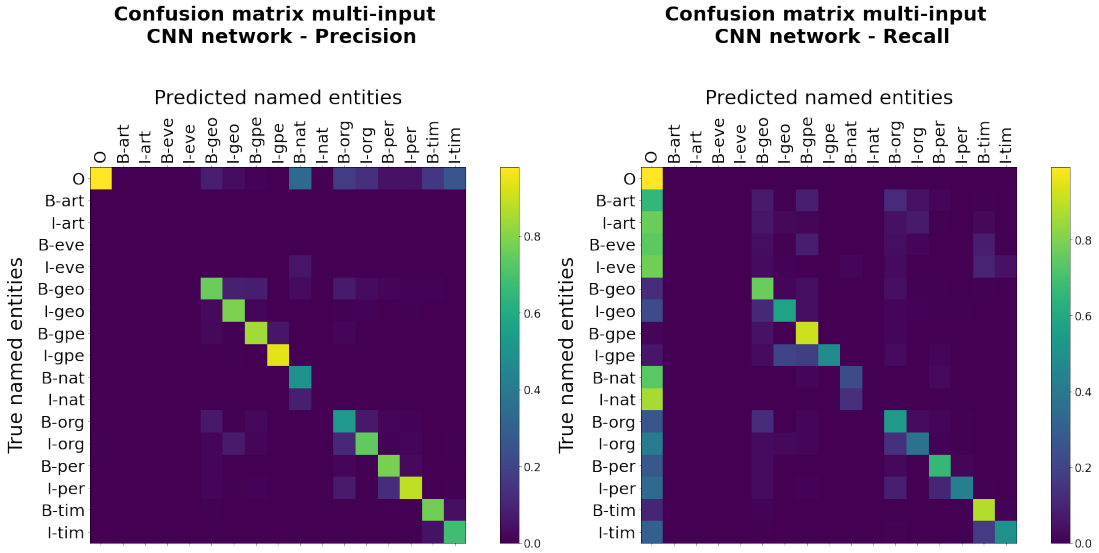
Figure 47: Confusion matrices for multi-input LSTM network, where 47a is a visualization of precision and 47b is a visualization of recall.



(a) Confusion matrix showing the precision values for the multi-input BiLSTM network.

(b) Confusion matrix showing the recall values for the multi-input BiLSTM network.

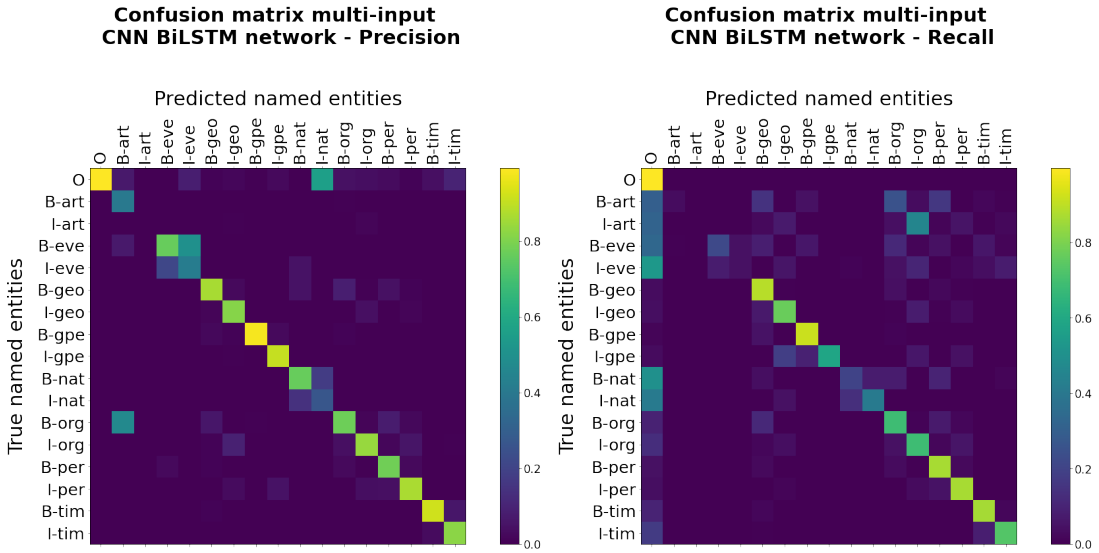
Figure 48: Confusion matrices for multi-input BiLSTM network, where 48a is a visualization of precision and 48b is a visualization of recall.



(a) Confusion matrix showing the precision values for the multi-input CNN.

(b) Confusion matrix showing the recall values for the multi-input CNN.

Figure 49: Confusion matrices for multi-input CNN, where 49a is a visualization of precision and 49b is a visualization of recall.



(a) Confusion matrix showing the precision values for the multi-input CNN BiLSTM network.

(b) Confusion matrix showing the recall values for the multi-input CNN BiLSTM network.

Figure 50: Confusion matrices for multi-input CNN BiLSTM network, where 50a is a visualization of precision and 50b is a visualization of recall.

7.2 Sentiment analysis: Results

The training and test results of *Sentiment algorithm 1*, *2*, *3* and *4* will be presented in this section. The results are ordered by the classification problem, meaning *Sentiment algorithm 1* and *3* are grouped together, since both are binary sentiment classification. *Sentiment 2* and *4* are grouped, since both are multi sentiment classification. At first, the results during training will be presented and thereafter the prediction results.

7.2.1 Training results

Measurements of batch accuracy and batch loss were made during training. Figure 51 and figure 52 depict the batch accuracy and loss, measured against timesteps during training, for *Sentiment algorithm 1* and *3*, whereas figure 53 and figure 54 depict the same metrics for *Sentiment algorithm 2* and *4*.

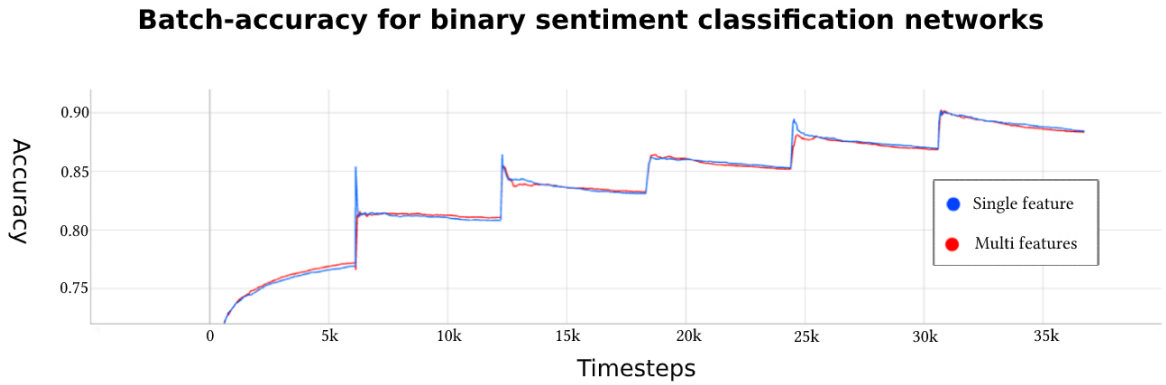


Figure 51: Batch accuracy for single and multi input binary sentiment classification networks, *Sentiment algorithm 1* and *3*, measured against timesteps during training. Both models seem to be very equal in performance, almost identical. They quickly become overfitted on the training data, and are stopped early of their planned 20 epochs, since the validation data accuracy falls, while the training data accuracy rises.

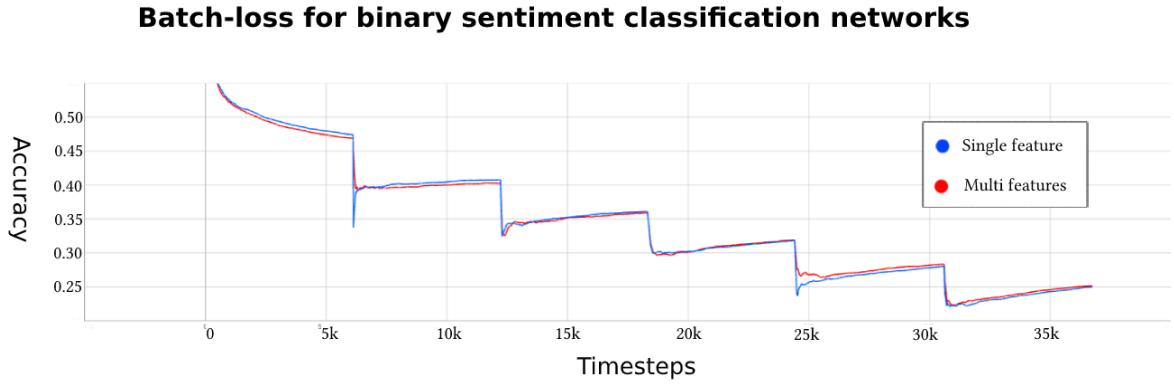


Figure 52: Batch loss for single and multi input binary sentiment classification networks, *Sentiment algorithm 1* and *3*, measured against timesteps during training.

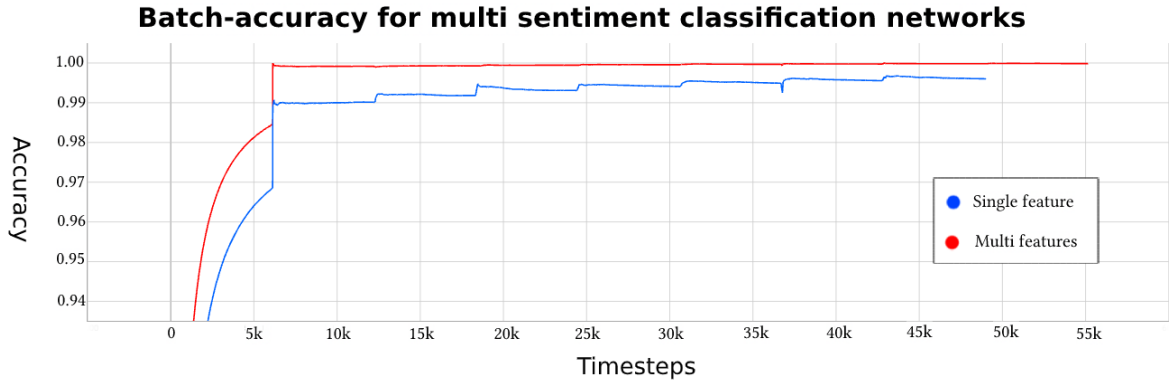


Figure 53: Batch accuracy for single and multi input multi sentiment classification networks, *Sentiment algorithm 2* and *4*, measured against timesteps during training. Both models quickly reach an accuracy score close of 1.00, it is evident that the models quickly become overfitted, already after one epoch. Still they both come closer to 1.00 after each timestep, but are stopped early, due to the validation accuracy falling, while they are becoming overfitted.

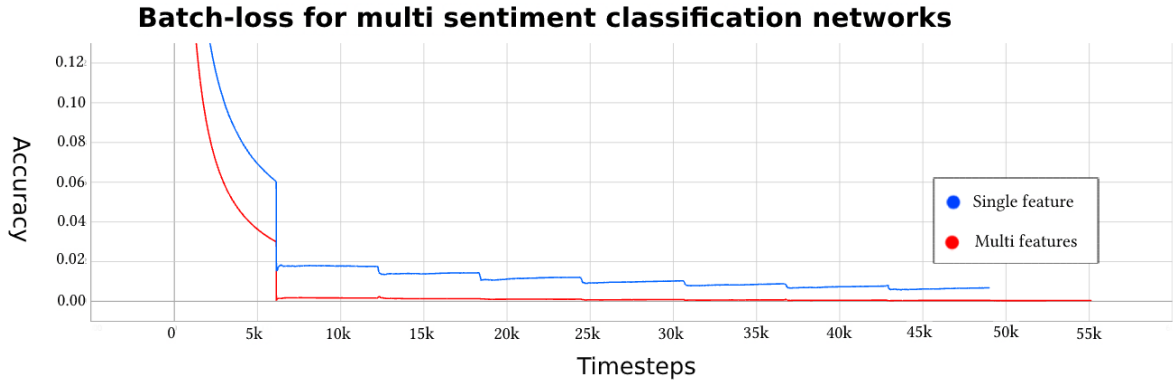


Figure 54: Batch loss for single and multi input multi sentiment classification networks, *Sentiment algorithm 2* and *4*, measured against timesteps during training.

7.2.2 Testing results

7.2.2.1 Score metrics

The score metrics are split in two, where precision, recall and F1-score are measured separately for binary and multi classification networks. In table 37 the score metrics can be seen for binary classification networks, which is *Sentiment algorithm 1* and *2*. In table 37 the score metrics can be seen for multi classification networks, which is *Sentiment algorithm 3* and *4*. Each table is colored with either red, orange, yellow or green, which depends in the values in each cell. The colors adhere to the color scheme in table 36.

Red:	[0;0.25[
Orange:	[0.25;0.50[
Yellow:	[0.50;0.75[
Green:	[0.75;1.0]

Table 36: Color scheme for precision, recall and F1 score metrics.

	Score metrics for binary classification					
	Precision		Recall		F1-score	
	SF	MF	SF	MF	SF	MF
M	0.54	0.57	0.54	0.52	0.54	0.54
F	0.54	0.56	0.54	0.60	0.54	0.58

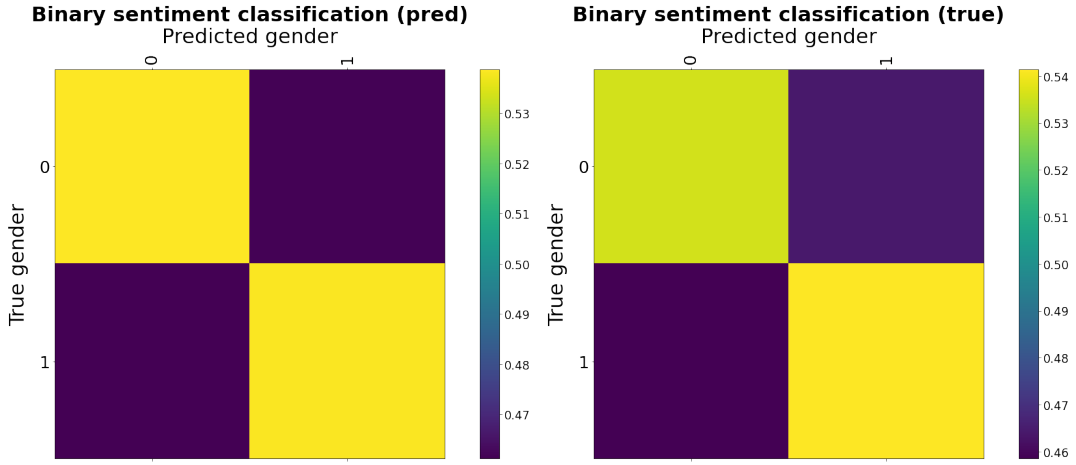
Table 37: Precision, recall and F1-score for binary sentiment classification single and multiple feature input, labeled SF and MF respectively. SF: *Sentiment algorithm 1*. MF: *Sentiment algorithm 3*. The colors of each cell can be one of four dependent on the value of the cell. Red is [0;0.25[, orange is [0.25;0.50[, yellow is [0.50;0.75[and green is [0.75;1.00].

	Score metrics for multi classification					
	Precision		Recall		F1-score	
	SF	MF	SF	MF	SF	MF
-0.9	0.00	0.00	0.00	0.00	0.00	0.00
-0.7	0.00	0.00	0.00	0.00	0.00	0.00
-0.6	0.89	0.93	0.91	0.91	0.90	0.92
-0.5	0.93	0.64	0.94	0.86	0.94	0.74
-0.4	0.60	0.63	0.66	0.53	0.63	0.58
-0.3	0.35	0.49	0.29	0.35	0.32	0.41
-0.2	0.46	0.45	0.47	0.42	0.47	0.43
-0.1	0.42	0.45	0.45	0.43	0.43	0.44
0.0	0.33	0.33	0.80	0.84	0.47	0.48
0.1	0.33	0.38	0.21	0.24	0.26	0.30
0.2	0.29	0.30	0.18	0.15	0.22	0.20
0.3	0.84	0.73	0.40	0.50	0.54	0.59

Table 38: Precision, recall and F1-score for multi sentiment classification single and multiple feature input, labeled SF and MF respectively. SF: *Sentiment algorithm 2*. MF: *Sentiment algorithm 4*. Color scheme adheres to table 32.

7.2.2.2 Confusion matrix

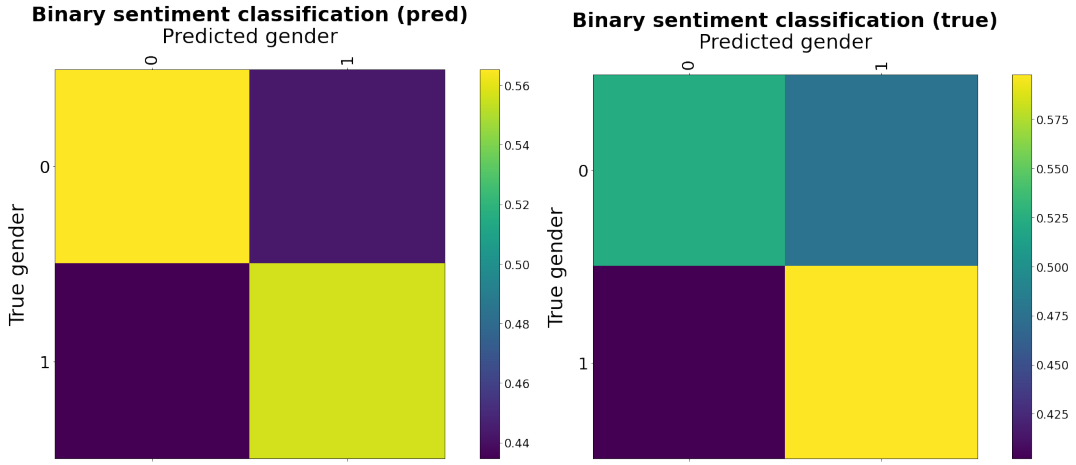
This section presents the results as confusion matrices. Each network is presented with a pair of confusion matrices, one visualizing the precision and the other visualizing recall. The precision and recall for binary sentiment classification is measured between predicted gender and true gender of the whole sentence. The precision and recall for multi sentiment classification is measured between predicted polarity and true polarity of each word index in a sequence.



(a) Confusion matrix showing the precision values for *Sentiment algorithm 1*.

(b) Confusion matrix showing the recall values for *Sentiment algorithm 1*.

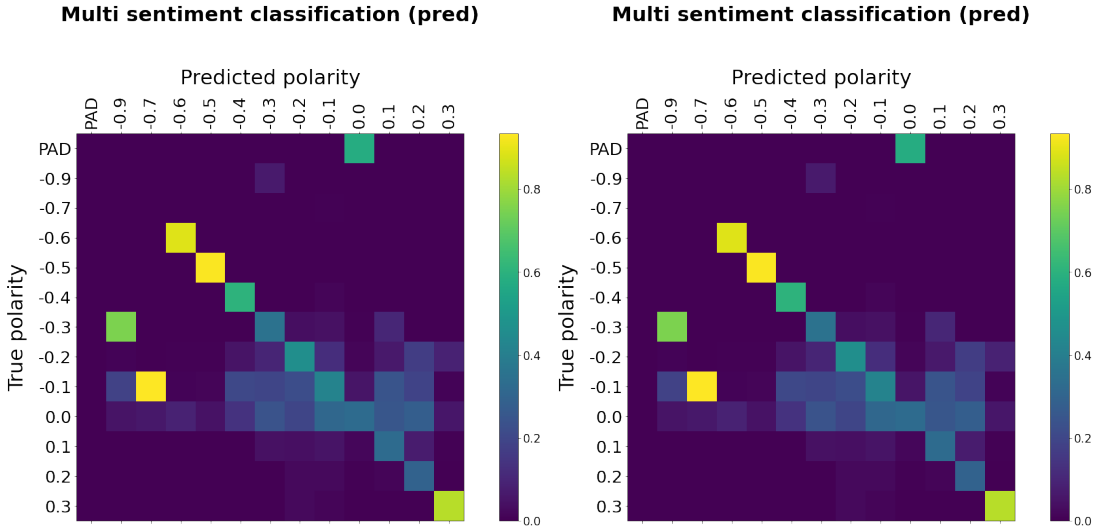
Figure 55: Confusion matrices showing prediction results for *Sentiment algorithm 1*, single feature (sequence of lemmas) as input and binary sentiment classification network. Confusion matrix (a) normalized over the predicted (columns), (b) normalized over the true (rows).



(a) Confusion matrix showing the precision values for *Sentiment algorithm 3*.

(b) Confusion matrix showing the recall values for *Sentiment algorithm 3*.

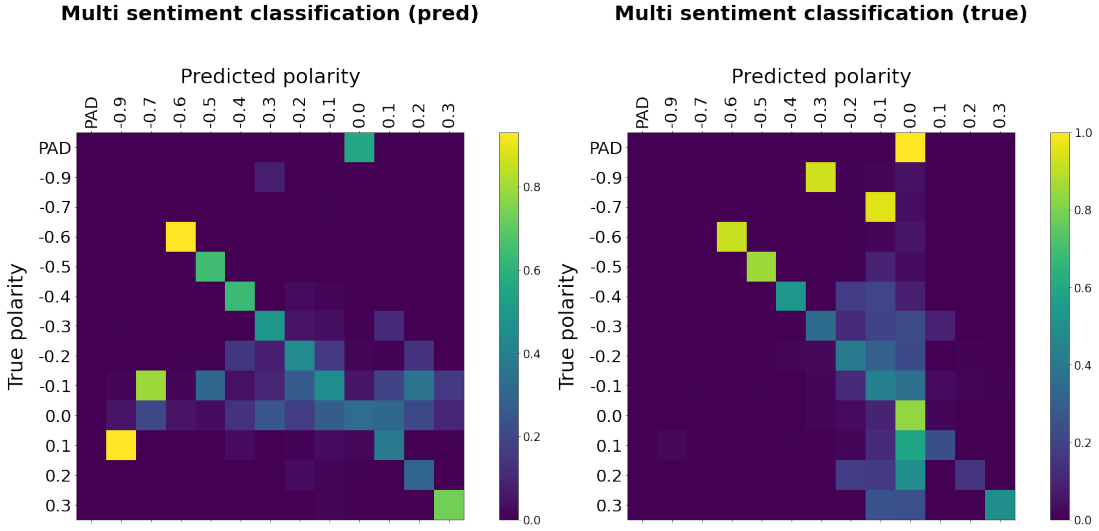
Figure 56: Confusion matrices showing prediction results for *Sentiment algorithm 3*, multiple features (sequence of lemmas and POS-tags) as input and binary sentiment classification network. Confusion matrix (a) normalized over the predicted (columns), (b) normalized over the true (rows).



(a) Confusion matrix showing the precision values for *Sentiment algorithm 2*.

(b) Confusion matrix showing the recall values for *Sentiment algorithm 2*.

Figure 57: Confusion matrices showing prediction results for *Sentiment algorithm 2*, single feature (sequence of lemmas) as input and multi sentiment classification network. Confusion matrix (a) normalized over the predicted (columns), (b) normalized over the true (rows).



(a) Confusion matrix showing the precision values for *Sentiment algorithm 4*.

(b) Confusion matrix showing the recall values for *Sentiment algorithm 4*.

Figure 58: Confusion matrices showing prediction results for *Sentiment algorithm 4*, multiple features (sequence of lemmas and POS-tags) as input and multi sentiment classification network. Confusion matrix (a) normalized over the predicted (columns), (b) normalized over the true (rows).

7.2.2.3 XAI

This section presents the results of the first iterations of XAI, a way of the algorithm explaining itself for its predictions. For each sentiment algorithm the XAI method is run on the 500 first predictions. From the 500, two explanations has been cherry-picked, one to present a sentence predicted to be masculine and one feminine.

The binary sentiment classification algorithms' XAI uses the polarity dictionary to present the most masculine or feminine words, dependent on the predicted gender. If the sentence is predicted masculine,

XAI will present the most masculine words.

The multi sentiment classification algorithms' XAI works independently of the polarity dictionary, only using the predicted polarities. If the overall sentiment, computed by equation 8, of the sentence exceeds the threshold (-0.1 and 0.1), XAI presents the most masculine or most feminine words, dependent of which way the overall polarity leans.

Sentiment algorithm 1

The sentence: "Cashing out on or taking a hardship withdrawal from a retirement fund is a very costly move that cuts into current savings in addition to wiping out future returns.".
Was predicted to be masculine (0).
Most masculine words in sentence are:
'fund' with a polarity of -0.19
'returns' with a polarity of -0.17

The sentence: "Do some price checking with your teen to help them identify what is better to purchase from home or online.".
Was predicted to be feminine (1).
Most feminine words in sentence are:
'checking' with a polarity of 0.11
'teen' with a polarity of 0.02

Sentiment algorithm 2

The sentence: "That's what Mitt Romney was talking about when he said he pays about 15 percent in taxes.".
Was predicted to be masculine (-0.4).
Most masculine words in sentence are:
'said' with a polarity of -0.6
'pays' with a polarity of -0.4

The sentence: "It is generally the party of the rich.".
Was predicted to be feminine (0.2).
Most feminine words in sentence are:
'party' with a polarity of 0.2
'rich' with a polarity of 0.0

Sentiment algorithm 3

The sentence: "But as the ill-fated Kardashian Card – which ceased shortly after launching due to public outcry about high fees – demonstrates, celebrity endorsements don't ensure fair, quality service.".
Was predicted to be masculine (0).
Most masculine words in sentence are:
'service' with a polarity of -0.25
'high' with a polarity of -0.24

The sentence: "Only four percent of women said that they would definitely go on a date with an unemployed man.".
Was predicted to be feminine (1).
Most feminine words in sentence are:
'women' with a polarity of 0.27
'man' with a polarity of 0.09

Sentiment algorithm 4

The sentence: ""A further order will be entered contemporaneous with the formal decision," the temporary order said.".
Was predicted to be masculine (-0.3).
Most masculine words in sentence are:
'order' with a polarity of -0.6
'formal' with a polarity of -0.3

The sentence: "In 2010, 27 percent of millionaires worldwide were women.".
Was predicted to be feminine (0.3).
Most feminine words in sentence are:
'women' with a polarity of 0.3
'worldwide' with a polarity of 0.0

8 Appendix

8.1 Appendix A

Appendix A is the the source code, and it is enclosed in the attached *Appendixes* zip-folder.

8.2 Appendix B

Text from first sample of *3_text_and_gender.json*, which is the web scraped body text from an article with category *WOMEN*.

At least two organizations have decided to drop Morgan Freeman after eight women accused him of inappropriate behavior and sexual harassment. Women who previously worked with the Oscar-winning actor told CNN that he repeatedly made comments about their bodies or their clothing and frequently engaged in inappropriate touching. In response to the allegations, Visa announced it had suspended him from its marketing campaign. We are aware of the allegations that have been made against Mr. Freeman. At this point, Visa will be suspending our marketing in which the actor is featured, Visa said in a statement. TransLink, Vancouver's public transit system, also decided to stop using Freeman's voice as part of an ad campaign to promote its Visa credit card and mobile payments on bus and Skytrain operations. In light of information we've learned ... of allegations regarding actor Morgan Freeman, TransLink has decided to pause his voice announcements as part of a Visa ad campaign on our transit system, the transit system said a statement. We will be reaching out to Visa to discuss further. A few hours after the report was released, Freeman, 80, issued an apology: Anyone who knows me or has worked with me knows I am not someone who would intentionally offend or knowingly make anyone feel uneasy, I apologize to anyone who felt uncomfortable or disrespected — that was never my intent. A production assistant working on the 2015 film Going in Style said Morgan made numerous comments about her body, asked her several times if she was wearing underwear and tried to lift her skirt. Another worker on the set said women wore loose-fitting clothing in an effort to avoid Freeman's attention. Entertainment Tonight released footage on Thursday evening of two on-air interviews with Freeman that also revealed his attitude toward women. During an interview for the 2016 film London Has Fallen, Freeman asked a young female reporter, My goodness, are you married? Fool around with other guys? I'm just asking. In 2015, he told author Janet Mock: You got a dress halfway between your knee and your hips, and you sit down right across from me and you cross your legs. Mock, who was a special correspondent for the interview, said Freeman's treatment of her was an exhibition of the casual nature at which men in positions of power believe that everything belongs to them, including women's bodies. Freeman's union, SAG-AFTRA, is considering what to do about Freeman's Life Achievement Award. The award is given to actors who represent the finest ideals of the acting profession, CNN reported. These are compelling and devastating allegations which are absolutely contrary to all the steps that we are taking to ensure a safe work environment, the union said in a statement. Any accused person has the right to due process, but it is our starting point to believe the courageous voices who come forward to report incidents of harassment ... we are reviewing what corrective actions may be warranted at this time.

8.3 Appendix C

Study of public demand, questions and answers grouped by the survey group.

Applicants

Opsætning:

Du er job ansøger, og skal til at sende dit CV og din motiverede ansøgning.

	Spørgsmål	Ja	Nej	Ved ikke
1	Ville du lade din ansøgning køre igennem en algoritme, som censurerer dine personlige oplysninger (køn, alder, etnicitet, race og religion), for at du ikke bliver valgt/fravalgt på baggrund af disse, men derimod dine kompetencer og erfaring?			
2	Tror du at man kunne mindske diskrimination i ansøgningsprocessen ved at censurere disse personlige oplysninger?			
3	Har du tidligere været nervøs for at blive diskrimineret i job-ansøgningsprocessen på baggrund af din køn, alder, etnicitet, race og/eller religion?			
4	Ville du lade din ansøgning køre igennem en algoritme, som gør dig opmærksom på kønsladede ord, som kan røbe dit køn? (eksempel: statistisk set er følgende sætning feminint ladet, grundet understregede ord: "Jeg er en sød og hjælpende person...")			
5	Ville du være villig til at gøre din ansøgning mere kønsneutral, hvis du bliver gjort opmærksom på at den indeholder kønsladede ord?			
6	Tror du at man kunne mindske diskrimination i ansøgningsprocessen ved at gøre ens ansøgning mere kønsneutral?			

Opsætning:

Du er job ansøger, og leder efter relevante jobs, du vil ansøge.

	Spørgsmål	Ja	Nej	Ved ikke
7	Ville du være *mindre* tilbøjelig til at sende en ansøgning til et job, hvor stillingsopslaget benyttede kønsladede ord, der *ikke* svarede til dit køn?			
8	Ville du være *mere* tilbøjelig til at sende en ansøgning til et job, hvor stillingsopslaget benyttede et neutralt sprog uden kønsladede ord?			

Figure 59: Questions asked in the applicant survey. The survey was printed and handed out to 10 students around the campus to be answered "anonymously". 5 males and 5 females answered, the group was primarily white.

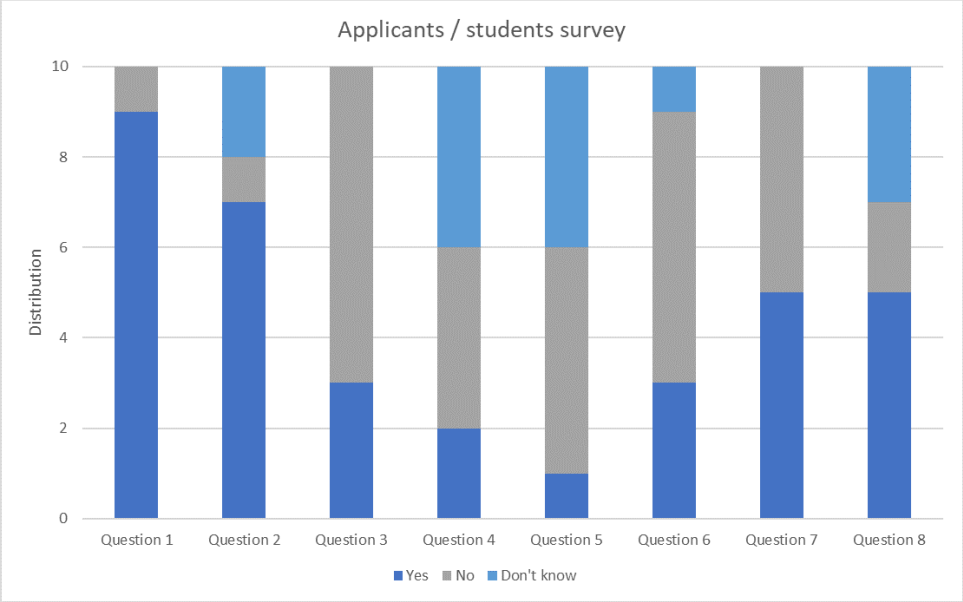


Figure 60: Distribution chart showing the answers of the applicant survey, shown in figure 59.

Employers

Opsætning:

Du er ansætter i en job ansøgningsproces, hvor du skal lave et job opslag og screene ansøgere til en første samtale.

	Spørgsmål	Ja	Nej	Ved ikke
1	Er ansøgerens køn en del af dine overvejelser ift. screeningen?			
2	Er ansøgerens etnicitet en del af dine overvejelser ift. screeningen?			
3	Er ansøgerens race en del af dine overvejelser ift. screeningen?			
4	Er ansøgerens alder en del af dine overvejelser ift. screeningen?			
5	Er ansøgerens religion en del af dine overvejelser ift. screeningen?			
6	Kunne du forestille dig, at du underbevidst lader ansøgerens køn, etnicitet, race eller religion påvirke din beslutning?			
7	Kunne du forestille dig, at du er mere tilbøjelig til at vælge kandidater af samme køn, etnicitet, race, alder og religion som dig selv?			
8	I en situation med mange ansøgere, ville du anvende algoritmer til at filtrere kandidater fra?			
9	I en situation med mange ansøgere, ville du anvende algoritmer til at fremhæve de bedst egnede kandidater? (baseret på algoritmens filtreringsparametre)			
10	Ville du anvende en algoritme til at skjule personlige oplysninger som ansøgerens køn, etnicitet, race, alder og religion?			
11	Ville du anvende en algoritme, som fremhæver ansøgerens kompetencer og erfaringer?			
12	Synes du, at det er en god idé at censurere ansøgerens personlige information i en ansøgning for at lægge fokus på kompetencer og erfaring og dermed mindske diskrimination af ansøgeren?			
13	Ville du være tilbøjelig til at anvende en algoritme, der markerede kønsladede ord i et job opslag? (eksempel: Følgende sætning indeholder ord der statistisk set er maskulint ladede: "Vi søger en stærk leder til...")			
14	Ville du være villig til at gøre dit job opslag mere kønsneutralt, hvis du bliver gjort opmærksom på at det indeholder kønsladede ord?			

Figure 61: Questions asked in the employers survey. The survey was printed and handed out to 10 teachers at their offices in the AU building Edison to be answered "anonymously". The group was primarily white males.

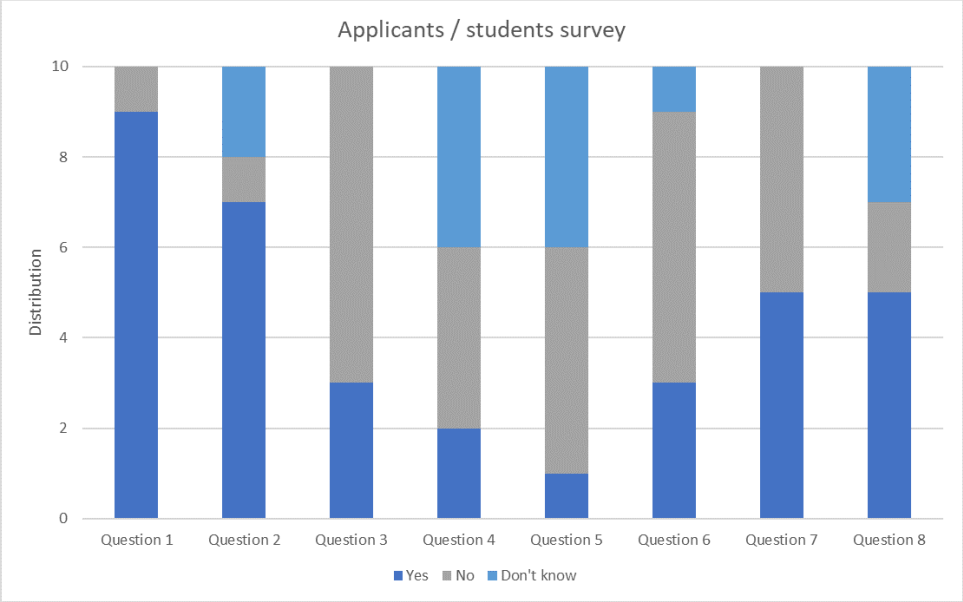


Figure 62: Distribution chart showing the answers of the employer survey, shown in figure 61.

8.4 Appendix D

Appendix D is the process report and is enclosed in the attached *Appendixes* zip-folder.

8.5 Appendix E

Appendix E is the XAI log txt files for each sentiment algorithm, and are enclosed in a folder in the attached *Appendixes* zip-folder.

List of Figures

List of Tables

References

[1] Signature Recruitment. *Gendered Words Dataset*. https://www.signaturerecruitment.co.uk/why-gender-neutral-language-improves-recruitment-ads/. [Online; Accessed 24/11-21]. 2021.

[2] Malte Dahl and Niels Krog. *Experimental evidence of discrimination in the labour market: Intersections between ethnicity, gender, and socio-economic status*. https://menneskeret.dk/sites/menneskeret.dk/files/media/dokumenter/malte_dahl_forskning.pdf. [Online; Accessed 30/08-21; Pages 83 - 114]. 2018.

[3] Marianne Bertrand and Sendhil Mullainathan. *ARE EMILY AND GREG MORE EMPLOYABLE THAN LAKISHA AND JAMAL? A FIELD EXPERIMENT ON LABOR MARKET DISCRIMINATION*. https://www.nber.org/system/files/working_papers/w9873/w9873.pdf. [Online; Accessed 30/08-21]. 2003.

[4] Malte Dahl. *Alike but different: How cultural distinctiveness shapes immigrant-origin minorities' access to the labour market*. https://menneskeret.dk/sites/menneskeret.dk/files/media/dokumenter/malte_dahl_forskning.pdf. [Online; Accessed 30/08-21; Pages 125-155]. 2019.

[5] Jeffrey Dastin. *Amazon scraps secret AI recruiting tool that showed bias against women*. https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G. [Online; Accessed 03/11-21].

[6] Danielle Gaucher, Justin Friesen, and Aaron C. Kay. *Evidence That Gendered Wording in Job Advertisements Exists and Sustains Gender Inequality*. https://www.hw.ac.uk/documents/gendered-wording-in-job-ads.pdf. [Online; Accessed 30/08-21; Pages 4-5]. July 2011.

[7] Society for Human Resource Management. *Using Social Media For Talent Acquisition - Recruitment and Screening*. https://www.shrm.org/hr-today/trends-and-forecasting/research-and-surveys/pages/social-media-recruiting-screening-2015.aspx. [Online; Accessed 30/08-21; Slide 3]. Jan. 2016.

[8] The London School of Economics and Political Science. *'Big data' from online recruitment platforms show discrimination against ethnic minorities and women – and sometimes men*. https://www.lse.ac.uk/News/Latest-news-from-LSE/2021/a-Jan-21/Big-data-from-online-recruitment-platforms-show-discrimination. [Online; Accessed 30/08-21;] Jan. 2021.

[9] Kumba Sennaar. *Machine Learning for Recruiting and Hiring – 6 Current Applications*. https://emerj.com/ai-sector-overviews/machine-learning-for-recruiting-and-hiring/. [Online; Accessed 03/11-21].

[10] James Vincent. *Automated hiring software is mistakenly rejecting millions of viable job candidates*. https://www.theverge.com/2021/9/6/22659225/automated-hiring-software-rejecting-viable-candidates-harvard-business-school?fbclid=IwAR2Kie4rlDVBQumbJ9iM-8PKLbUyTrOF5e2EspLquq8pUGHysDScoqtRCBk. [Online; Accessed 03/11-21].

[11] Tim Halloran. *Watch your (gender) tone*. https://textio.com/blog/watch-your-gender-tone/13035166463. [Online; Accessed 03/11-21].

[12] Ayn-Monique Klahre. *3 Ways Johnson & Johnson Is Taking Talent Acquisition to the Next Level*. https://www.jnj.com/innovation/3-ways-johnson-and-johnson-is-taking-talent-acquisition-to-the-next-level. [Online; Accessed 03/11-21].

[13] *Sentiment analysis*. https://en.wikipedia.org/wiki/Sentiment_analysis. [Online; Accessed 15/11-21].

[14] Srijani Chaudhury. *Building a Sentiment Analyzer With Naive Bayes.* https://medium.com/swlh/building-a-sentiment-analyzer-with-naive-bayes-c96cc8aa52a5. [Online; Accessed 15/11-21].

[15] *Annex - 4.3 Open-source NLP libraries.*

[16] *Annex - 4.1.1 Named entity recognition.*

[17] Wikipedia. *Part-of-speech tagging.* https://en.wikipedia.org/wiki/Part-of-speech_tagging. [Online; Accessed 29/11-21].

[18] *Annex - 4.1.2 Part-of-speech tagging.*

[19] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval.* https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf. [Online; Accessed 17/11-21; Chapter 2.2.4 Stemming and lemmatization]. 2009.

[20] Hunter Heidenreich. *Stemming? Lemmatization? What?* https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8. [Online; Accessed 29/11-21]. 2018.

[21] Jabir Jamal. *NLP-03 Lemmatization and Stemming using spaCy.* https://medium.com/mlearning-ai/nlp-03-lemmatization-and-stemming-using-spacy-b2829becceca. [Online; Accessed 29/11-21].

[22] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval.* https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf. [Online; Accessed 17/11-21; Chapter 2.2.2 Dropping common terms: stop words]. 2009.

[23] Groningen Meaning Bank. *Homepage.* https://gmbdemo.webhosting.rug.nl/about.php. [Online; Accessed 08/11-21].

[24] *Language-Independent Named Entity Recognition (II).* https://www.clips.uantwerpen.be/conll2003/ner/. [Online; Accessed 12/11-21].

[25] Groningen Meaning Bank. *Dataset for linguistic analysis.* https://www.kaggle.com/abhinavwalia95/entity-annotated-corpus. [Online; Accessed 08/11-21].

[26] Ellyn Rolleston Keith. *A Sentiment Analysis of Language & Gender Using Word Embedding Models.* https://academicworks.cuny.edu/cgi/viewcontent.cgi?article=3473&context=gc_etds. [Online; Accessed 17/11-21]. 2017.

[27] ecmonsen, phseiff, and guy4261. *Gendered Words Dataset.* https://github.com/ecmonsen/gendered_words. [Online; Accessed 22/11-21]. 2021.

[28] Usman Malik. *Python for NLP: Movie Sentiment Analysis using Deep Learning in Keras.* https://stackabuse.com/python-for-nlp-movie-sentiment-analysis-using-deep-learning-in-keras/. [Online; Accessed 17/11-21].

[29] Sergio Virahonda. *An easy tutorial about Sentiment Analysis with Deep Learning and Keras.* https://towardsdatascience.com/an-easy-tutorial-about-sentiment-analysis-with-deep-learning-and-keras-2bf52b9cba91. [Online; Accessed 17/11-21].

[30] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, 2nd edition.* https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/. [Chapter 16: Sentiment Analysis]. 2019.

[31] Yasmeen Hitti et al. *A Dataset for Tackling Gender Bias in Text.* https://aiforsocialgood.github.io/2018/pdfs/track2/47_aisg_neurips2018.pdf. [Online; Accessed 15/11-21].

[32] Tony Sun et al. *They, Them, Theirs: Rewriting with Gender-Neutral English.* https://academicworks.cuny.edu/cgi/viewcontent.cgi?article=3473&context=gc_etds. [Online; Accessed 17/11-21]. 2017.

[33] Rishabh Misra. *News Category Dataset - Identify the type of news based on headlines and short descriptions - Version 2*. https://www.kaggle.com/rmisra/news-category-dataset. [Online; Accessed 17/11-21]. 2018.

[34] *spaCy, available trained pipelines for English*. https://spacy.io/models/en. [Online; Accessed 15/11-21].

[35] *Natural Language Toolkit*. https://www.nltk.org/. [Online; Accessed 13/09-21].

[36] *spacy.io*. https://spacy.io. [Online; Accessed 13/09-21].

[37] *spaCy 101: Everything you need to know*. https://spacy.io/usage/spacy-101. [Online; Accessed 13/09-21].

[38] Afshine Amidi and Shervine Amidi. *Recurrent Neural Networks cheatsheet*. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks. [Online; Accessed 22/11-21].

[39] Shipra Saxena. *Introduction to Long Short Term Memory (LSTM)*. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/. [Online; Accessed 22/11-21]. Mar. 2021.

[40] Xuan Hien Le et al. *Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting*. https://www.researchgate.net/figure/The-structure-of-the-Long-Short-Term-Memory-LSTM-neural-network-Reproduced-from-Yan_fig8_334268507. [Online; Accessed 22/11-21]. 2019.

[41] Christopher Olah. *Understanding LSTM Networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Online; Accessed 22/11-21]. Aug. 2015.

[42] Derrick Mwiti. *NLP Essential Guide: Convolutional Neural Network for Sentence Classification*. https://cnvrg.io/cnn-sentence-classification/. [Online; Accessed 22/11-21].

[43] Marc Moreno Lopez and Jugal Kalita. *Deep Learning applied to NLP*. https://arxiv.org/pdf/1703.03091.pdf. [Online; Accessed 22/11-21].

[44] Anja Bechmann and Geoffrey C Bowker. *Unsupervised by any other name: Hidden layers of knowledge production in artificial intelligence on social media*. https://journals.sagepub.com/doi/pdf/10.1177/2053951718819569. [Online; Accessed 13/12-21]. 2019.

[45] Harsha Bommana. *Deep NLP: Word Vectors with Word2Vec*. https://medium.com/deep-learning-demystified/deep-nlp-word-vectors-with-word2vec-d62cb29b40b3. [Online; Accessed 10/12-21].

[46] https://newbedev.com/. *What is the intuition of using tanh in LSTM*. https://newbedev.com/what-is-the-intuition-of-using-tanh-in-lstm. [Online; Accessed 22/11-21].

[47] Keras. *Conv1D layer*. https://keras.io/api/layers/convolution_layers/convolution1d/. [Online; Accessed 30/11-21].

[48] Adem Akdogan. *Word Embedding Techniques: Word2Vec and TF-IDF Explained*. https://towardsdatascience.com/word-embedding-techniques-word2vec-and-tf-idf-explained-c5d02e34d08. [Online; Accessed 30/11-21]. 2021.

[49] Akiko Aizawa. *An information-theoretic perspective of tf–idf measures*. https://www.sciencedirect.com/science/article/pii/S0306457302000213. [Online; Accessed 30/11-21; Chapter 2. A brief look at conventional statistical measures]. 2003.

[50] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*. https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf. [Online; Accessed 30/11-21; Chapter 6.2 Term frequency and weighting]. 2009.

[51] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*. https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf. [Online; Accessed 30/11-21; Chapter 6.2.1 Inverse document frequency]. 2009.

[52] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*. https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf. [Online; Accessed 30/11-21; Chapter 6.2.2 Tf-idf weighting]. 2009.

[53] Keras. *Keras Tokenizer source code*. https://github.com/keras-team/keras-preprocessing/blob/master/keras_preprocessing/text.py. [Online; Accessed 03/12-21]. Last update in 2020.

[54] Richard Sun. *Does punctuation matter in sentiment analysis?* https://voyagersun.wordpress.com/2018/07/09/does-punctuation-matter-in-sentiment-analysis/. [Online; Accessed 08/12-21]. 2018.

[55] Mansooreh Karami et al. *"Let's Eat Grandma": When Punctuation Matters in Sentence Representation for Sentiment Analysis*. https://arxiv.org/pdf/2101.03029.pdf. [Online; Accessed 08/12-21]. 2020.