CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Good News AI

Investigating feasibility of categorizing positive sentiment in general news

Master's thesis in Computer science and engineering

Klas Ludvigsson
Magnus Andersson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Good News AI

Investigating feasibility of categorizing positive sentiment in general news

Klas Ludvigsson
Magnus Andersson

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Good News AI
Investigating feasibility of categorizing positive sentiment in general news
Klas Ludvigsson
Magnus Andersson

Supervisor: Marwa Naili, Department of Computer Science and Engineering.
Advisor: Ignacio Mancha & Annika von Hofsten, i3tex AB
Examiner: Jennifer Horkoff, Department of Computer Science and Engineering.

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Good News AI
Investigating feasibility of categorizing positive sentiment in general news
Klas Ludvigsson
Magnus Andersson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In today's society we are constantly fed information about catastrophic or sad events through media. While it is important to know about these events, it should be equally important to also see all the good things that are happening in our world. Therefore, this thesis proposes two algorithms for classifying full-length news articles to remove the non-positive articles. Traditionally these types of algorithms require a large amount of labelled data, but this thesis explores possibilities for sentiment classification with a limited amount of labelled data. The best performing algorithm presented is this thesis achieves a precision percentage of 98% with only 40 articles used for training.

Keywords: Computer, science, computer science, thesis, sentiment classification, clustering.

Acknowledgements

We would like to extend our thanks to our academic supervisor Marwa Naili, and our company supervisors Ignacio Mancha & Annika von Hofsten at i3tex for the continued guidance and support throughout the thesis. And a special thanks to our examiner Jennifer Horkoff and our opponent Jan Jürgen Eisenmenger for their valuable feedback.

Klas Ludvigsson & Magnus Andersson, Gothenburg, June 2020

Contents

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**
1.1 Background . . . 1
1.2 Problem Statement . . . 2
1.3 Aim . . . 3
1.4 Limitations . . . 3
1.5 Road map . . . 3

2 Theory **5**
2.1 Word Embeddings . . . 5
2.1.1 Word2vec . . . 5
2.1.2 Doc2vec . . . 7
2.2 Classification . . . 8
2.2.1 K-means . . . 8
2.2.2 Supervised K-means . . . 9
2.3 Definition of Good . . . 9
2.3.1 Utilitarianism . . . 9
2.3.2 Right-based ethics . . . 10
2.4 Sentiment Classification . . . 10
2.5 Related Works . . . 11

3 Methods **13**
3.1 Defining "Good" . . . 13
3.2 Collecting data . . . 14
3.2.1 News sources . . . 14
3.2.2 Unlabelled Data . . . 14
3.2.3 Labelled Data . . . 14
3.3 Pipeline . . . 15
3.3.1 Preprocessing . . . 16
3.4 Classification algorithms . . . 16
3.4.1 Seeding . . . 17
3.4.2 K-means . . . 17
3.4.3 Tree Clusters . . . 17
3.4.4 Fixed Point Classifier . . . 19

3.4.5 Exploring the Labelled Data 20
3.5 Evaluation 21
3.6 Visualization of results 21

4 Results and Discussion **23**
4.1 Datasets 23
4.1.1 Labelled 23
4.1.2 Unlabelled 24
4.2 Word Embedding Parameters 24
4.3 K-means Evaluation 25
4.3.1 Results 25
4.3.2 Discussion 26
4.4 Tree Clusters Evaluation 26
4.4.1 Results 26
4.4.2 Discussion 26
4.5 Fixed Point Classifier Evaluation 28
4.5.1 Results 28
4.5.2 Discussion 29
4.6 Overall results 30
4.7 Visualization of website 31

5 Conclusion **33**
5.1 Conclusion 33
5.2 Future work 34

36

A Appendix 1 **I**
A.1 Complete result tables I

List of Figures

2.1 Distributed Memory Model of Paragraph Vectors[11] 8
2.2 Bag of Words version of Paragraph Vector[11] 8

3.1 Visualization of the data flow through the modules in the trainer . . . 15
3.2 Visualization of the data flow through the modules in the application. 16
3.3 Visual of a single tree 18
3.4 Single type versus Double type article comparison 18

4.1 Depth impact with type Single. The regression line was calculated using least squares. 28
4.2 Depth impact with type Double. The regression line was calculated using least squares. 28
4.3 Correlation between labelled data size and precision using least squares 30
4.4 Current look of website 32

List of Tables

4.1 Labelled dataset statistics 23
4.2 Unlabelled dataset statistics 24
4.3 K-means evaluation results 25
4.4 Tree Clusters evaluation results 27
4.5 Tree Clusters average precision for depth 27
4.6 Fixed Point Classifier evaluation results 29
4.7 Best evaluation results for each model 30
4.8 Best FPC model compared to related works 31

A.1 Fixed Point Classifier evaluation results II
A.2 Tree Clusters evaluation results III

1
Introduction

In recent years the task of algorithmically reading and interpreting text has been constantly evolving. With the advancement in speed and availability of computer hardware, these new algorithms are able to process far more data then what was previously possible. At the same time, large datasets could be created by scanning and downloading text from the internet. The remaining task was then, and still is today, to find the best way to convert this text data into something a computer can analyze and draw conclusions from.

The conclusions that can be drawn depends on the problem that one attempts to solve. One of these possibilities is to try to comprehend the emotional intent of a text, which falls within the field of sentiment classification. Sentiment classification, a form of text classification, is a heavily researched area in the field of machine learning. The currently used algorithms for this can, if not understand, then at least extract information from text intended for the human eye. Traditionally these algorithms require large amounts of manually labelled data to be able to successfully classify text.

1.1 Background

The task of text classification usually entails classifying documents into topics, e.g. politics and sports. When classifying as such, topic-related words are the key features. For example the words "team" or "football" are related to sport articles. However, in sentiment classification, the goal is to extract the positive or negative sentiment of a text[13].

Sentiment is a inherently subjective thing, a text one person interprets as positive might be interpreted as negative by another. Positive words or sentences have a positive sentiment attached to them. For example, sentences about happiness, kindness and success are generally classified as positive, and sentences about hate, violence and sadness are generally classified as negative. Humans are usually very good at judging sentiment given a context, but for a machine it is very challenging.

Starting from the new century, the field of sentiment classification began to grow as opinion based texts became widely available on the world wide web[13]. However this research was, and still is today, limited in some regards. These limitations in-

clude: Use of sentiment data from an unverified third party that have applied their own definition of what is positive sentiment[12], or having the scope of texts be limited to one or a few topics. These limitations point towards unexplored possibilities within the field of sentiment classification.

1.2 Problem Statement

There are still gaps in the field of sentiment classification, questions to which there are no satisfactory answers. Due to the ever-changing nature of language in general, and on the internet in particular, previous research might not be applicable today. For example previous research focused on headlines will not apply to modern articles, since in today's internet landscape a misleading headline is a common occurrence[24]. Up-to-date research is needed in a field as adaptive as sentiment classification.

Another problem is the absence of a large labelled dataset. This is because labelling data is costly and time-consuming. A common approach is to use predefined sentiment generated through lexicon-based techniques[26]. However, this introduces another problem in that the predefined sentiment carries bias. Predefined sentiment, as mentioned before, is subjective and will therefore differ from context to context[7]. Addressing all these shortcomings will push sentiment classification forward.

This thesis aims to address these issues by answering the following question: Can positive sentiment be categorized in full-length news articles, according to some specific definition of good, using a limited amount of labelled data? To answer this, two sub-questions will be answered.

How much data will be required? The research question asks for limited amount of data, but how does different sizes of data impact the result.

Are there any current methods that can solve this problem or do new algorithms need to be developed?

This work is done at the behest of i3tex AB, because they wish to contribute to the field of artificial intelligence, and do not have access to any large labelled datasets which makes this thesis a fitting choice. i3tex is trying to improve the world around them in a multitude of ways, one of which is establishing a standard of what constitutes a better world, and strive towards this in all their business dealings.

1.3 Aim

This thesis aims to develop an application capable of classifying news articles as positive news, without the requirement of large labelled datasets. The main goals of this thesis are as follows:

- Develop an application capable of collecting, classifying and displaying positive news articles
- Create new algorithms for sentiment classification of news articles
- Explore how fewer labelled data points affect the classification results
- Explore how changing the parameters in the algorithms affect the classification results
- Compare the new methods to previous work in the field

1.4 Limitations

The labelled dataset will be collected manually during a limited time period. Therefore, the width of the collected news articles will be restricted to what is available. If a particular category of news never appears as positive in the time-period then there will be no positive entry of it in our labelled dataset.

Additionally, since we will only have access to our limited labelled dataset, a comparison against using a larger labelled dataset with the same methods will not be made.

The thesis is not interested in overall performance of the model, but rather a specific evaluation parameter (precision). Most of the previous research in the field have diverging focus and is therefore incompatible for comparison. A large-scale comparison against these models would not improve the validity of the results presented in this thesis and is therefore excluded.

1.5 Road map

The thesis is structured as follows: Chapter 2 provides the theory behind established techniques used in this thesis. Chapter 3 describes the final system as well as the methods used for developing it. Chapter 4 presents and discuss the results of each individual method, but also the best overall results. Chapter 5 concludes the thesis with a summary regarding the used methods and results, together with a discussion regarding future work.

2
Theory

This chapter will describe the different techniques that will be used in the project, as well as previous work that aims to achieve similar goals. The first section will describe how word embeddings work in general and how the two word embedding libraries used in this thesis, Doc2vec and Word2vec, works in particular. The second section will describe different methods for classification that are used in the thesis. In the third section the notion of "good" will be discussed from different angles. Finally, the concept of sentiment classification together with previous work similar to this will be described.

2.1 Word Embeddings

When working with large quantities of text data you need to structure it in a way that allows for efficient processing. One way to do this is to map the words into a vector space, which then allows the words to be subject to mathematical operations. This technique is called word embeddings, and it includes many techniques for dimension reduction of text data.

In this thesis we will use Doc2vec[11] to represent the articles for its ability to take into account the context of an entire document, not only a sentence. This functionality is highly relevant because this thesis is focused on classifying full-length articles. Doc2vec is an extension of the word embedding library Word2vec which will be described first.

2.1.1 Word2vec

In 2013 Mikolov et al.[14] presented two new models for word embeddings called continuous bag-of-words model (CBOW) and continuous skip-gram model. CBOW uses the surrounding words to predict the current word from the neighbouring words. The skip-gram model instead uses the current word to predict the surrounding words. The two models are used to find similarity between words and allows for relations between words to be extracted with arithmetical operations. For example the resulting vector of subtracting *man* from *king* is very close to the word vector for the word *queen*. These algorithms were then released as a python package called Word2vec and is primarily used for topic modelling.

Training of the Word2vec model involves two steps:

- Building vocabulary
- Training word vectors based on their context

Building of the vocabulary is done by assigning a word index to all of the words that will be used in the model. This is done to allow each word to be represented with 1-of-V encoding. 1-of-V encoding is a simple scheme that converts a word in a vocabulary of size V into a vector of length V, where the index of the word is set to 1 and all other indexes are 0. For example if our vocabulary is made up of the words *good*, *news* and *AI*, we can give the words the indexes 0,1, and 2 respectively. Then the word *good* can be represented with 1-of-V encoding as the vector [1,0,0], and *news* as [0,1,0] etc.

Word2vec then uses the 1-of-V encodings as input and output for a fully connected neural network with one hidden layer. The input and the output layer have the same amount of nodes as there are words in the vocabulary, and the hidden layer is the size of the resulting vector representation of the word, specified by the user. Initialization values for the weights of each edge is a random number between -1 and 1. To train the network, it is given a context as input vector and a target vector as a "goal" for what the output vector should be. The error vector is calculated by subtracting the output vector from the target vector, and is then used for the back-propagation. Using the example mentioned earlier together with CBOW to predict the word "AI" from "good news", the input vectors are [1,0,0] and [0,1,0], and the target vector is [0,0,1]. The input vectors are then multiplied with the weights and the output vector is the average of those vectors. The error is then calculated and the back-propagation can begin.

To learn the output vectors, Word2vec provides two options: Hierarchical Softmax and Negative Sampling. These two algorithms also have, according to Mikolov et al.[14], their own advantages and disadvantages. Hierarchical Softmax is said to work better with infrequent words and Negative Sampling with frequent ones.

The input and target vectors are defined differently depending on whether CBOW or skip-gram is used:

The CBOW model, as mentioned previously, uses the surrounding words to predict the current word. The model is given a window size that corresponds to the interval length of surrounding words in the context. The words in the window are averaged with the task to maximize the probability of each word in the vocabulary appearing as output given the context of the word. This probability is given by Equation 2.1, where N is the number of words in the vocabulary, $w(t)$ is the t:th word in the context and c is the window size.

$$\frac{1}{N}\sum_{t=1}^{N}\log p(w(t)|w(t-\frac{c}{2}),...,w(t+\frac{c}{2})) \tag{2.1}$$

The skip-gram model, almost as an inverse to the CBOW model, uses the cur-

rent word as context to predict the surrounding words. The error is here computed by comparing the output to each of the word vectors in the window interval. Maximization is then done on the probability given by Equation 2.2, where N is the number of words in the vocabulary, $w(t)$ is the t:th word in the context and c is the window size.

$$\frac{1}{N}\sum_{t=1}^{N}\sum_{j=t-c,j\neq t}^{t+c}\log p(w(j)|w(t)) \tag{2.2}$$

The two models both have, as with Hierarchical Softmax and Negative Sampling, their own advantages and disadvantages. The CBOW model performs better with frequent words and is faster, while skip-gram handles infrequent words better and works well with smaller datasets, according to Mikolov et al.[14].

When the model have finished training, the similarity between two word vectors a and b can be calculated as the cosine similarity between the vectors, as can be seen in Equation 2.3.

$$\cos(\mathbf{a},\mathbf{b}) = \frac{\mathbf{ab}}{\|\mathbf{a}\|\|\mathbf{b}\|} = \frac{\sum_{i=1}^{n}\mathbf{a}_i\mathbf{b}_i}{\sqrt{\sum_{i=1}^{n}(\mathbf{a}_i)^2}\sqrt{\sum_{i=1}^{n}(\mathbf{b}_i)^2}} \tag{2.3}$$

2.1.2 Doc2vec

Le and Mikolov[11] continued to develop an extension to word vectors, called paragraph vectors. Paragraph vectors address some key weaknesses of bag-of-words models. Previous techniques only work on surrounding words in a fixed interval, but are unable to differentiate one document from another. Paragraph vectors do not suffer this problem and can produce one vector for a document consisting of multiple sentences. This allows for queries of the type "find the document most similar to this other document", which is a useful functionality for this thesis. This updated version of Word2Vec was released under the name Doc2Vec.

The model work similarly to Word2vec with the addition of a document index, equivalent to the word indexes in Word2vec, to identify from which document a sentence comes from. Doc2vec uses two techniques for training the paragraph vectors: Distributed Memory Model of Paragraph Vectors (PV-DM) and Distributed Bag of Words version of Paragraph Vector (PV-DBOW), and they are essentially paragraph vector versions of the techniques used in Word2vec.

PV-DM uses a sequence of words, sampled from the document, together with the paragraph id to predict the next word of the sequence, as shown in figure 2.1 where the input vector is the word sequence "the cat sat" together with the paragraph id D, and the target vector contains only the word "on".

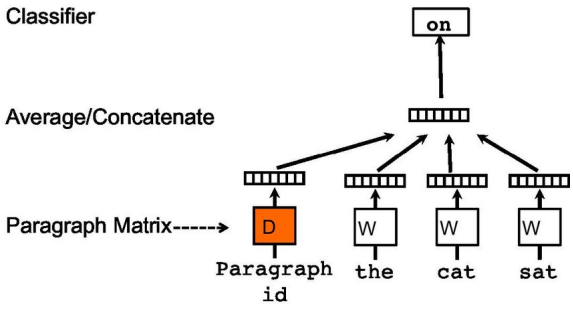

Figure 2.1: Distributed Memory Model of Paragraph Vectors[11]

PV-DBOW instead uses only the paragraph id to predict randomly sampled sequences from the document, as shown in figure 2.2 where the input vector is only the paragraph id *D*, and the target vector contains the sentence "the cat sat on".

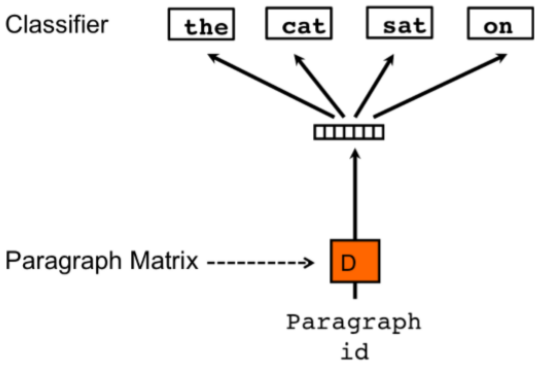

Figure 2.2: Bag of Words version of Paragraph Vector[11]

For calculating similarities Doc2Vec uses the same method as Word2Vec, i.e cosine similarity, as can be seen in Equation 2.3.

2.2 Classification

Classification methods are the core of this thesis. The task of classification is to predict a label for some previously unseen data. In this thesis the focus will be on binary classification; either the data is positive or it is not. In this section are the existing classification methods used in this thesis explained.

2.2.1 K-means

K-means clustering technique aims to partition a dataset into k subgroups, also called clusters. Each cluster is defined by a centroid, which is calculated as the average of all the points in the cluster. The data points are assigned to clusters by comparing their distances to the centroid of each cluster, and choosing the closest.

This is an iterative process of calculating the centroid and finding clusters until the clusters move less than a threshold or a set amount of iterations are reached. Different kernels, that defines the preferred distance metric, can be used for comparing distances. The results are clusters of data points close to each other that might share similarities, for example positive and non-positive clusters[10].

2.2.2 Supervised K-means

Supervised k-means is standard k-means with some constraints. These constraints are considered hard constraints, meaning that they must be satisfied. There are commonly two types of constraints, one of which will be used in this thesis, the *cannot-link* constraint[23]. The *cannot-link* constraint prevents the introduced points from being assigned to the same cluster.

The algorithm start by adding N fixed points each time a centroid is to be calculated. These fixed points act like anchors, and each of them anchor their own cluster. This allows the clusters more movement than fixed centroids would, but still forces the clusters to remain within some general area. It also provides information about which of the clusters are categorized as positive and non-positive.

2.3 Definition of Good

What is good? In this thesis good will have to be defined, such that when the algorithm classifies a specific article as "good", a predefined system is available to judge if the predicted sentiment was correct. This work is not the first to try to deal with these intangible ideas. Different frameworks regarding ethical decision making have been presented throughout history. The most prevalent and diverse of ethics are the consequence-based group and the opposing action-based group.

This thesis was suggested by the company i3tex, with the intent to spread positive feelings around the offices. i3tex are working for a better world by pursuing the common good, and to do this they have developed a method called Better World Index (BWI)[8]. It consists of the following five criteria:

- Reduce environmental impact
- Reduce human suffering
- Make people's everyday life easier
- Increase people's security
- Facilitate communication and information flows

The BWI lays the groundwork for what is considered "good news" in this thesis.

2.3.1 Utilitarianism

According to the ethical theory of utilitarianism one should always make decisions based on what will increase the pleasure and decrease the pain for the largest group of people. In practice this implies that the needs of the many outweigh the needs of the few. This is often described as a consequentialist approach because it states

that all actions should be weighed only against the consequences it produces[6]. A utilitarian would for example consider it ethically correct to use humans as scientific test subjects against their will to find a cure for a disease that would save many more people, even if the test subject might loose their life.

2.3.2 Right-based ethics

In contrast to utilitarian in particular, and consequential ethics in general, are deontological ethics. The basic theory of deontology is that not all actions can be justified by their effects, no matter how morally good they might be. Right-based ethics is a branch of deontological ethics that focuses on people's rights. This practice forbids using another's body, labor or talent without consent[1]. In contrast to the example in the above section, a right-based ethical approach would consider the right against being used as a test subject without one's consent as an ethical violation, no matter how much good may come of it.

2.4 Sentiment Classification

There are three main approaches used to train models:

Supervised: Train a model to predict a label by feeding it prelabelled data.

Unsupervised: Train a model to predict a label by feeding it unlabelled data.

Semi-supervised: Train a model to predict a label by feeding it a small amount of prelabelled data and a large amount unlabelled data.

Supervised sentiment classification contains a finite amount of categories, and every category has some labelled training data associated with it. Then a model is trained using classification algorithms such as Naive Bayes and support vector machine (SVM). Another technique is to use a preexisting lexicon of sentiment items (word or n-grams)[13].

Unsupervised sentiment classification attempts to find the semantic orientation of a document without the use of labelled data. As an example of this, Turney[21] used the words "excellent" and "poor" respectively to represent positive and negative categories of user reviews in his text classification algorithm, explained further in Section 2.5.

Semi-supervised sentiment classification makes a compromise between the two. Clustering techniques using labelled data as starting seeds is one example of this. Semi-supervised Naive Bayes takes a small amount of labelled data and a large amount of unlabelled data, and using the small amount of labelled data and some initial assignment to the unlabelled data, reclassify the unlabelled data until a satisfactory model is achieved[18].

2.5 Related Works

Previous work exists in the field of sentiment classification. For example, Peter D. Turney had already in 2002 [21] developed a simple unsupervised learning algorithm that could classify text as positive or negative, with only the two words “excellent” and “poor” as training data. The algorithm classifies the reviews by calculating the semantic orientation of the phrases in the text. This semantic orientation comes from comparing which of the starting words the phrases are more closely related to. The algorithm achieved an accuracy of 74% and served as a proof of concept for the field in general.

Another example is Haribhakta and Doddi[26], who created a platform for classifying news as positive or negative. They did this with the supervised algorithm Support Vector Machine (SVM) and a labelled dataset collected from SentiWordNet. SentiWordNet is an extension of the lexical database WordNet and provides a sentiment score for a large number of words[3]. This was used to identify the sentiment scores of an article. Based on these sentiment scores and the specific combination of adjectives, adverbs and verbs they were able to calculate a score for an arbitrary news article.

Jang et al.[9] investigated how to use word embeddings together with convolutional neural networks to classify news articles and tweets. More specifically they intended to classify articles and tweets related to advertisement as negative data and articles with disease-related information as positive data. To accomplish this they collected news articles and tweets under a two-month period. They then used this data to create word embeddings using the word2vec library. Their results were an accuracy of 93% and concluded that classifying news articles can be done with a higher success rate than classifying tweets because of news articles uniform structure.

3
Methods

This chapter contains a definition of good used in the thesis, followed by a section about data handling and collection, an outline of the whole intended system, and a description of all algorithms used in the thesis.

3.1 Defining "Good"

This thesis' definition of good will be based on i3tex's BWI criteria, but it will be expanded upon with concepts from ethical frameworks to mitigate gray areas. The result is intended for a diverse group of people, and as such a rigorous and precise definition will filter out all but a small subset of acceptable articles.

The framework used to classify the articles will, on top of i3tex's BWI criteria, be a combination of the utilitarian and rights-based approach mentioned in section 2.3. To be considered as good news an article must address at least one of the five BWI criteria or adhere to both these two ethical requirements:

$\boldsymbol{E_1}$**:** Either increase the total sum of pleasure or decrease the total sum of pain
$\boldsymbol{E_2}$**:** Respect the rights of all parties involved

The resulting logical formula can be seen in Equation 3.1.

$$\mathbf{BWI} \vee (\boldsymbol{E_1} \wedge \boldsymbol{E_2}) \tag{3.1}$$

The intention of these additional requirements are to widen the scope for what good news entails. With the BWI criteria a news story about rescued animals would not necessarily be considered good, but it will fall within the extended scope as good news because it decreases the pain for the related parties and nobody is treated unjust.

3.2 Collecting data

The resulting application will be collecting data constantly to provide the user with the latest curated good news from the chosen news sources.

3.2.1 News sources

The news sources used in this thesis are:

- BBC[4]
- Reuters[16]
- ScienceDaily[17]
- The Guardian[19]
- The Independent[20]
- UN News[22]
- Wired[25]

These news sources were chosen because they cover a diverse set of news, and because we perceive them to hold a certain standard of credibility.

To be able to query all these different news sources constantly, the application will use Rich Site Summary (RSS) web feeds. These are standardized machine-readable web feeds used for this express purpose.

3.2.2 Unlabelled Data

Articles will be collected from the news sources to build an unlabelled dataset. Since thesis took place under a four month period, this dataset will be limited in size. To complement this a larger dataset will be used as well. The reasoning behind this is that the larger dataset will provide the general structure of articles to be comprehended by the word embeddings, while the collected data can provide a more up-to-date collection of words and phrases. For example Covid-19 will not be present in older articles, so all mentions of it will not affect the classification results because it is not present in the vocabulary. Note that since the larger unlabelled dataset will be used to provide a general structure for the word embeddings, diversity is the only important feature.

3.2.3 Labelled Data

The definition of good explained in 3.1 will be used to produce a labelled dataset. This labelling process will be done in two steps: first one party reads the full-length article and assigns either a positive or non-positive label, and then a second party verifies the correctness of the label. In total 400 articles will be labelled, 200 labelled as positive and 200 labelled as negative. Collection of labelled data will happen in two one month long iterations, where 200 articles are collected each iteration. This is done to ensure a wide variety of news is included. The data will then be used to train and evaluate the models.

3.3 Pipeline

The program will be divided into two distinct modules, the application and the trainer. The latter will train and evaluate all the models used to classify new articles, while the former will collect, classify, and display the articles on a website. Since the application needs the trained models to classify the articles, the trainer will need to execute before the application for the program to be functional. This separation was done because of the computational requirements needed to train the models are far more significant than the requirements for the rest of the application, and it would not be feasible to train new models each time the program is executed.

The trainer trains the models according to some algorithm-specific parameters. Its default behaviour is to train one model for each permutation of parameters, and after completing the training it evaluates every available model. The classifier is a combination of a word embedding model and one of the four classification algorithms seen in the Figure 3.1. The word embedding model is used to convert the documents into the vectors that are used for the classification. All the models are semi-supervised, so they will use some amount of the labelled data during training. When the training is done the models are saved for later use in the application. The rest of the labelled data is used for the evaluation, which is also performed each time the code base is updated to provide a progression log for the project. In Figure 3.1 the data flow of the trainer is visualized, where the classifier is the combination of a Doc2Vec-model and one of the 4 algorithms that are described later in this chapter.

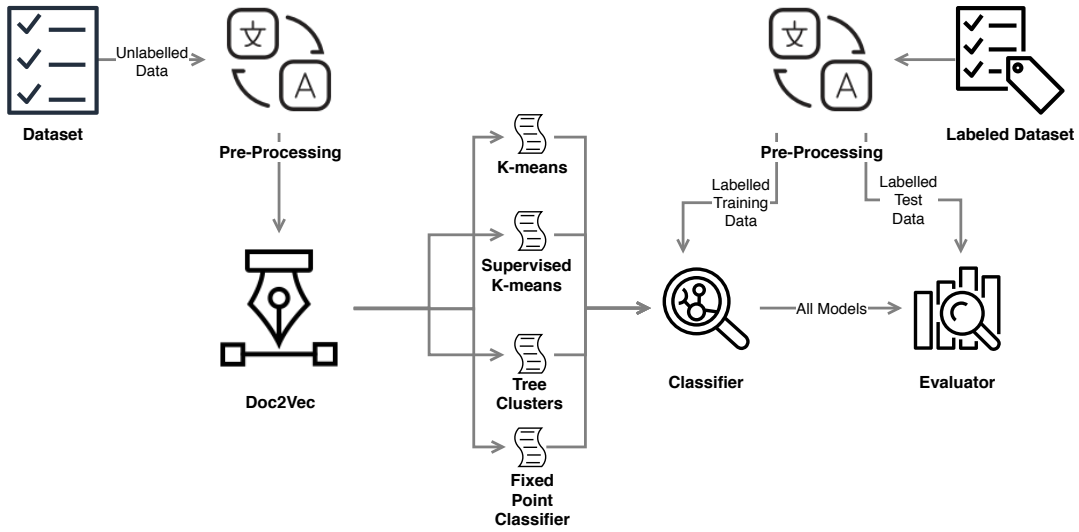

Figure 3.1: Visualization of the data flow through the modules in the trainer

The application receives a continuous stream of news articles from the scraper, which queries the chosen news sources every fourth hour. The aggregated articles are both saved to a database, and sent to be classified. News articles classified as positive are displayed on the designated website for the end users, while the negative articles are simply discarded. This whole process can be seen in Figure 3.2.

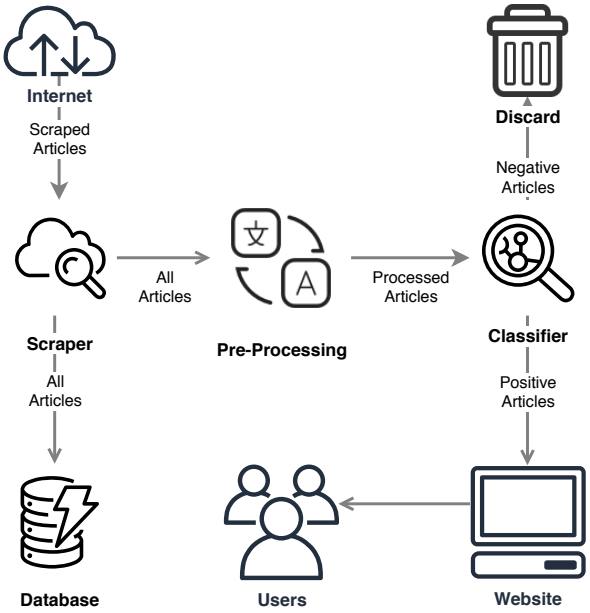

Figure 3.2: Visualization of the data flow through the modules in the application.

3.3.1 Preprocessing

The data both collected from the news sources and extracted from the dataset will be preprocessed the same way. The preprocess step includes: making all characters lowercase, removing words that have a length of two characters or less, removing non-alphabetic characters, and removing words that do not contribute to the sentiment of the text, such as "a", "and" or "be"(also known as stop-words).

Additionally, a stemmer will be used to reduce words to their root. Stemming is used to group words that mean the same thing, but are written on different forms. For example "happy" and "happiest" would be considered one word instead of two after this process.

3.4 Classification algorithms

In this section all classification algorithms used during this thesis are outlined, or further expanded upon if outlined in previous theory section. The different methods are all used to find articles similar to the labelled articles, and with this similarity classify incoming articles as positive or not positive.

Two custom algorithms, Tree Clusters and Fixed Point Classifier, are proposed because existing algorithms for sentiment classification are mainly focused on supervised learning and the current semi-supervised algorithms looked at do not fit the aims of this thesis.

3.4.1 Seeding

Every classification model in this thesis is using some form of semi-supervised learning, which involves using labelled data points for both training and evaluation. To avoid reusing training data as evaluation data, each model is assigned a seed and a size. The seed is a string of numbers used as input to a pseudorandom number generator to shuffle the labelled data in a specific and repeatable way. The size then corresponds to how much of the labelled data is used for training and how much is used for evaluation. This will allow for good variety in how the data is divided between the models and allow for pinpointing how specific changes impact the models. Seeding is used by every classification algorithm in this thesis to ensure all training data is randomly selected.

3.4.2 K-means

Standard k-means explained in section 2.2.1. Once the clusters have been generated, a subset of the test data is used to determine which sentiment each cluster of articles represent. Multiple clusters can be classified as positive or non-positive. The clusters with more positive than non-positive articles in them are chosen as positive clusters.

Regular k-means might have a hard time identifying positive and non-positive clusters, and lean more towards clusters of categories. To counter this, supervised k-means will be used, explained in section 2.2.2. Supervised k-means comes with inherent knowledge of which clusters are positive and non-positive, which can allow for more separation within categories. For example a fixed positive technology article and a fixed non-positive technology article could be used to avoid category clusters.

3.4.3 Tree Clusters

Tree clusters (TC) is an algorithm developed during this thesis. A Doc2Vec model, together with K starting points from the seeding process, are supplied to start the algorithm. For each starting point a paragraph vector is created, and from this vector a tree of the most similar paragraph vectors is explored. Similar paragraph vectors are found using the cosine similarity formula in Equation 2.3. The n most similar vectors are added as leaves. Then each vector in the list of leaves are run through the same process until the given depth, d, is reached.

The selection is depth-based, and is limited by d and n. Each time a new depth is reached, the n most similar vectors are added. This continues all the way until the max depth d is reached, and the tree is complete, as can be seen in Figure 3.3.

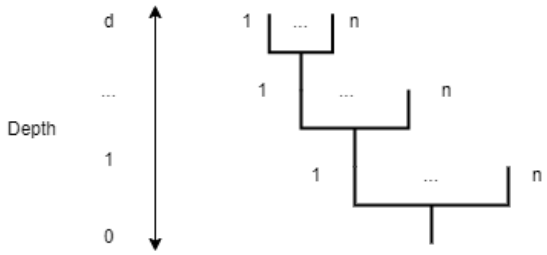

Figure 3.3: Visual of a single tree

There are two different types of TCs. The first type, called Single, generates only positive clusters, and compares the entirety of the dataset against those positive clusters. The second type, called Double, generates both positive and non-positive clusters, and makes the final classification based on the comparison of those clusters. Figure 3.4 shows how two articles would be classified by the different types of TC's.

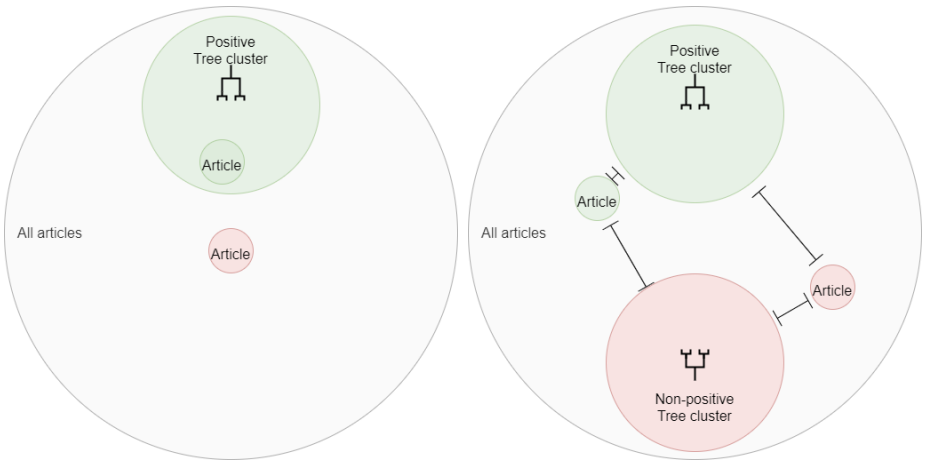

Figure 3.4: Single type versus Double type article comparison

This algorithm can produce clusters containing both positive and non-positive articles, even with the specific starting points. There are no requirements on the selected leaves, as long as they are within the n most similar. The advantage is that only things that are in the positive cluster will be classified as such, so it is very strict in that regard.

```
//Initialization of clusters
Get the n most similar vectors
//Add depth zero leaves
Do While there are vectors left
    If potential leaf exceeds selection-critera:
        Add potential leaf
    Endif;
EndDo;
//Iterate through leaves until stop-criteria is met
Do While depth not reached
    Do While there are still vectors to traverse
        If vector has not been traversed before
            Get the n most similar vectors
            //Note that vector has now been traversed
            Do While there are potential leaves left
                Add potential leaf
            EndDo;
        Endif;
    EndDo;
EndDo;
```

To classify a new article the n most similar articles, M, are used. All articles in positive clusters found in M are added together and compared against another sum, K. The max of the two sums determines the new label. In type Single, K is all other articles in M; in type Double, K is the sum of all articles in non-positive clusters found in M.

3.4.4 Fixed Point Classifier

The Fixed Point Classifier (FPC) is a classifier algorithm developed during this thesis. Despite not creating clusters, the classifier can be used to achieve a similar effect. k vectors are assigned as positive centroids, and k as non-positive centroids through the seeding process. Each vector represents a labelled article. A new unlabelled article is classified as positive if the resulting vector is closer to the positive centroids. To make the classifier more strict an offset can be added to the distances to the positive centroids, so that the classifier is weighted towards classifying articles as negative. The algorithm can use either euclidean distance or cosine distance to calculate the distances. This algorithm removes the process in the supervised k-means algorithm of finding the clusters, by using the fixed points as centroids, giving it a tremendous advantage in speed.

The algorithm also allows for multiple articles to be used for one centroid. In that case the centroid is defined as the average of all supplied articles' paragraph vectors.

```
//Initialization of centroids
Do for every set of articles
    Preprocess articles
    Generate doc2vec paragraph vectors for articles
    //Combine all articles for current centroid
    Calculate the average of paragraph vectors to be the centroid
Enddo;
```

```
//Classification of article
Preprocess article
Generate doc2vec paragraph vector for article
Do for every centroid
    Calculate distance to centroid
Compare average negative distance to positive distance
If positive distance + offset is larger
    Article is classified as positive
Otherwise
    Article is classifed as negative
```

The advantage of FPC over TC is that FPC only compares the incoming data point to previously labelled articles. The obvious drawback is that it relies heavily on the random labelled articles used as training data to produce the centroids.

3.4.5 Exploring the Labelled Data

There are several categories of articles in the set of "general news"-articles, like science, technology and sports. These categories will most likely share more similarities than two positive articles that are of different categories. For example, a positive sports article is likely more similar to a non-positive sports article than to a positive article about the environment. With the seeding system each model will use a pseudo-random subset of the labelled data for training and another for evaluation, but there might be some articles that yields better results than others. Finding these articles will give more insight into what the word embedding models interprets as 'most positive' and 'least positive'.

To calculate which points that are mathematically interpreted as 'most positive' and vice versa, the labelled articles with the furthest average distance from the articles with the opposite label can be calculated, as shown in Equation 3.2.

$$s_i = \frac{1}{N-1} \sum_{k=1,k\neq i}^{N} z_k \cdot d(v_i, v_k) + \frac{1}{N-1} \sum_{n=1,n\neq i}^{N} (z_n - 1) \cdot d(v_i, v_n) \tag{3.2}$$

where s_i is the final score, N is equal to the number of vectors in the training data, v_i is the i:th vector in the training data, $d(v, v)$ is a distance function, and z_i is given

by Equation 3.3.

$$z_i = \begin{cases} 1, & \text{if label of the i:th article is positive} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

The calculation is equivalent to the average distance from the given vector to the positive articles subtracted by the average distance to the non-positive articles. Therefore, a lower score indicates a more positive article and vice versa.

3.5 Evaluation

The evaluation of how well the algorithms perform is measured primarily by precision; how big percentage of the articles that have been classified as positive actually are positive, as shown in Equation 3.4. Additional parameters that are measured are recall, what fraction of the positive articles were classified as positive (Equation 3.5), and accuracy, how much of the data was classified correctly (Equation 3.6). Recall, and accuracy are not as important as precision, since the goal is to show only positive articles and not to find all positive articles.

$$P = \frac{A_p}{A_{cp}} \tag{3.4}$$

$$R = \frac{A_p}{A_{tp}} \tag{3.5}$$

$$A = \frac{A_p + A_{cn}}{A_p + A_{tn}} \tag{3.6}$$

where A_p is the amount of correctly classified positive articles, A_{cp} is the amount of articles classified as positive, A_{tp} is the amount of total actual positive articles, A_{cn} is the amount of correctly classified non-positive articles, A_{tn} is the amount of total non-positive articles.

To achieve a better evaluation for the algorithms they are cross validated. The cross validation includes evaluating multiple models with the same parameters but different seeds and calculating the average score across all models evaluated. Since this thesis explores how few labelled data points are required the majority of the labelled data will be used for testing, but the ratio of the training/testing data will differ from model to model.

3.6 Visualization of results

The platform to present the results of the final algorithm will be a website hosted on the company's internal web. Here all articles from the chosen RSS feeds will be processed, classified and displayed if deemed as positive news.

4
Results and Discussion

This chapter contains the results and discussion for all parts of this thesis. The first section describes the properties of the datasets. The second section discuss the word embedding model used for the algorithms. The following three sections present and discuss the results for each of the algorithms. After that a comparison between the best performing models is made. Finally, the last section discuss the resulting website.

4.1 Datasets

There are two datasets used in the evaluation: one labelled dataset used to train and evaluate the models, and one unlabelled dataset to train the word embeddings. In this section the characteristics of these datasets will be explained.

4.1.1 Labelled

400 articles have been collected and labelled, where 100 positive and 100 non-positive articles were collected during the period 2020-02-03 - 2020-02-18, and another 100 positive and 100 non-positive articles were collected during the period 2020-03-12 - 2020-04-14. The articles differ in length, so an average word count is provided for completeness. The articles were collected manually from the websites of the news sources listed in Table 4.1.

News source	Number of articles	Average word count
BBC	43	4731
Reuters	117	2094
ScienceDaily	54	4623
The Guardian	46	3944
The Independent	58	3208
UN News	35	2897
Wired	47	9235

Table 4.1: Labelled dataset statistics

When exploring the dataset, as described in Section 3.4.5, we found that the articles labelled as positive that were least similar to the non-positive articles were in the category of science news. Of the top 10 with the lowest scores, i.e the mathematically

"most positive", all were science related, even though a minority of the labelled articles were science related. This indicates that, of the news we collected, positive science articles differentiates themselves more from non-positive articles than other news categories.

4.1.2 Unlabelled

142570 unlabelled news articles were used in the creation of the word embedding model. The dataset was acquired on the website Kaggle, and was collected between 2000-05-15 and 2017-04-09, but 95% of the articles was collected from 2016 or 2017[2]. The news sources and the number of articles from each source are shown in Table 4.2. Note that, as mentioned in Section 3.2.2, the credibility of these news sources are as important because the articles are only used to provide structure for the word embeddings. More specifically, a larger dataset of similarly structures texts, in this case news articles, provides the Doc2Vec model with a broader understanding of the context, which in turn yields more accurate similarity queries.

News source	Number of articles	Average word count
Atlantic	7179	7909
Breitbart	23781	2985
Business Insider	6757	2656
BuzzFeed News	4853	4854
CNN	11488	4245
Fox News	4354	3148
National Review	6203	5679
New York Post	17493	2549
NPR	11992	4537
Reuters	10710	4068
Talking Points Memo	5214	2219
The Guardian	8681	5349
The New York Times	7803	6862
Vox	4948	7972
Washington Post	11114	6121

Table 4.2: Unlabelled dataset statistics

4.2 Word Embedding Parameters

A single word embedding model is used in the evaluation of all models. The parameters for this model were chosen by trying the different values recommended in the Doc2Vec documentation, then selecting the parameters that yielded the highest precision. The final parameters are:
Vector size: 300
Epochs: 20
Min count: 5

The **Vector size** corresponds to the size of the resulting paragraph vectors, **Epochs** corresponds to the number of iterations made over the training corpus and **Min count** corresponds to how many occurrences of a word are required for it to be included in the final vocabulary.

For all models distance is calculated using either euclidean or cosine distance. Cosine distance was chosen because it is what the word embeddings use to calculate the similarity between two paragraph vectors. Euclidean was chosen to have another metric to compare how well the cosine distance works.

4.3 K-means Evaluation

As is mentioned in Section 3.4.2, standard k-means is theorized to produce average results, with supervised k-means focusing the algorithm more on sentiment, and therefore, increasing precision. In this section are the results and discussion of the cross evaluation.

4.3.1 Results

Presented in Table 4.3 are the final results of the two k-means algorithms. All values are rounded to 4 decimals, each model have been evaluated using cross validation with 10 different seeds and all models make use of stemming in their preprocessing step. This is true for all models in this chapter.

The different parameters for the K-means models are size, type and kernel. Size corresponds to both K i.e the number of clusters and the number of labelled articles used when training the model. Size ranges from a minimum of 2, since there has to be at least 2 clusters, to a maximum of 20. The maximum of 20 was chosen because with larger sizes the models takes longer to train, without much effect on the results. Type is either "Standard" or "Supervised", and corresponds to which algorithm was used. Kernel is either "e", which implies that euclidean distance was used, or "c", which implies that cosine distance was used. The table is sorted after highest precision because this is the attribute considered to be most important in this thesis.

Type	Size	Kernel	Precision	Recall	Accuracy
Supervised	10	e	0.7361	0.5236	0.5810
Standard	2	c	0.7180	0.6606	0.7005
Supervised	2	c	0.7156	0.6563	0.6977
Standard	10	c	0.6750	0.9300	0.7411
Supervised	10	c	0.6582	0.7231	0.6733
Standard	10	e	0.6548	0.9058	0.7139
Standard	2	e	0.5047	0.9980	0.5093
Supervised	2	e	0.5007	0.7920	0.5019

Table 4.3: K-means evaluation results

4.3.2 Discussion

Poor precision was theorized at the start, because categories and themes were assumed to stand out more in news than sentiment. So that standard k-means performs well above 50% is a surprise; likewise, that the introduction of guidance in the form of supervised k-means barely improved precision. The theory beforehand was that categories of news would be the identifying factor of the clusters instead of sentiment. Therefore, a more finely-grained approach of 10 clusters could divide one category of news into different sentiment clusters and produce better results. This theory was not supported by the results, because the difference of 1,81% between the first and second models can be reasonably contributed to margin of error. The fact that the results are so similar points towards some overall similarity between the positive and negative articles across categories. These results are, however, still too imprecise. If this algorithm were to be used to classify everyday news, with the intuition that most of everyday news does not fall under the definition of good described in Section 3.1, the result would be a majority of non-positive news. For example: assuming a 9-to-1 split in favor of non-positive news, the best model would find on average 3.23 non-positive news for every "good news"-article.

4.4 Tree Clusters Evaluation

In this section are the results and the discussion of the cross evaluation.

4.4.1 Results

The different parameters for the Tree clusters models are type, size, depth and n. The type parameter indicates if the algorithm uses only positive clusters (Single) or if it uses both positive and negative clusters (Double), and that is why the size is doubled when using type Double. Size is the number of trees generated, and therefore equivalent with the number of labelled articles used since each tree is generated from one article. These articles comes from the seeding process described in Section 3.4.1, so the number of articles used for evaluation be slightly different depending on the Size parameter, more specifically it will be equal to the total amount of labelled articles subtracted by the Size. The largest size used is 40 articles because it is 10% of all labelled articles, so there is still enough data left for the evaluation. Depth and n are explained in Section 3.4.3. Depth ranges between 1 through 4, because higher values result in long compute times and less strict models. n is either 5 or 10 based on default values in the word embedding model. The cross validation results with precision above 0.8 can be seen in Table A.2. The full TC cross validation table can be found in Appendix A.

4.4.2 Discussion

From the results it can be seen that TC overall benefits from multiple small trees. The vast majority of the best TC's have 10 or more trees, with the best model of

Type	Size	Depth	n	Precision	Recall	Accuracy
Double	4	1	5	0.9147	0.0338	0.5154
Single	4	3	5	0.9074	0.0332	0.5143
Double	40	2	5	0.9064	0.1217	0.5547
Double	20	2	5	0.8965	0.0600	0.5271
Single	40	2	5	0.8952	0.2181	0.5959
Single	40	1	5	0.8841	0.0944	0.5413
Double	20	1	5	0.8808	0.1547	0.5668
Single	40	1	10	0.8751	0.2256	0.5966
Single	4	2	10	0.8718	0.0730	0.5321
Double	20	2	10	0.8703	0.1768	0.5753
Single	20	2	5	0.8623	0.0961	0.5406
Single	20	1	10	0.8546	0.0989	0.5408
Double	40	1	10	0.8522	0.1050	0.5433
Single	40	3	5	0.8418	0.3012	0.6222
Double	40	2	10	0.8368	0.2994	0.6206
Double	4	1	10	0.8345	0.0990	0.5412
Double	4	3	5	0.8341	0.1010	0.5396
Single	4	4	5	0.8337	0.0541	0.5212
Single	20	2	10	0.8233	0.3006	0.6172
Single	20	3	5	0.8224	0.1694	0.5664
Single	40	2	10	0.8187	0.4725	0.6834
Double	40	1	5	0.8167	0.2761	0.6081
Double	20	1	10	0.8151	0.3911	0.6508
Single	4	3	10	0.8143	0.1474	0.5571
Double	4	2	10	0.8120	0.0323	0.5145
Double	20	4	5	0.8091	0.1605	0.5618
Double	4	4	5	0.8036	0.0389	0.5165
Single	20	4	5	0.8033	0.2550	0.5961

Table 4.4: Tree Clusters evaluation results

both types being outliers.

It was theorized in Section 3.4.3 that too deep trees would create clusters with a mix of positive and non-positive articles, which would result in lower precision. This theory was confirmed by the results, as can be seen in Table 4.5.

Depth	Average precision
1	83%
2	85%
3	77%
4	77%

Table 4.5: Tree Clusters average precision for depth

The breath of the trees, n, also benefits from being smaller. The deepest trees, with

depth of 3 or 4, produce the best results when size and n are smaller.

An observable downside to this algorithm is that it suffers when it comes to recall. A solution to low recall would be to use bigger trees, since the best average recall is achieved when using a depth of 3 with 40% average recall. But this solution would suffer when it comes to precision because it allows for more non-positive articles to be included in the tree. The impact of different depth values have on precision and recall for the different types are shown in Figure 4.1 and Figure 4.2. It can be observed that greater depth values have a bigger impact on type Single TC's compared to the type Double TC's.

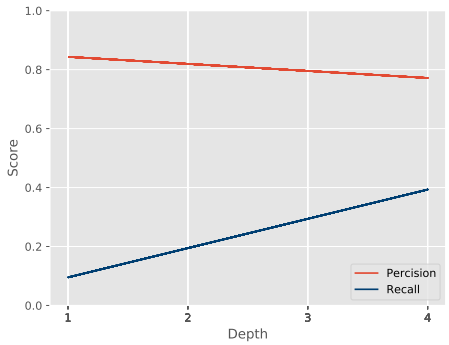

Figure 4.1: Depth impact with type Single. The regression line was calculated using least squares.

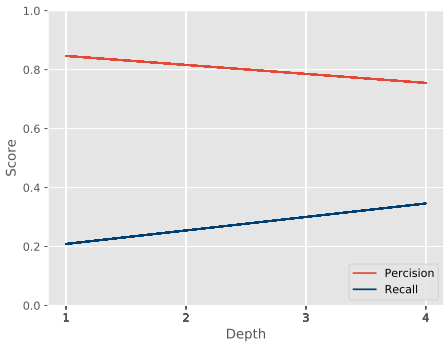

Figure 4.2: Depth impact with type Double. The regression line was calculated using least squares.

4.5 Fixed Point Classifier Evaluation

As mentioned in Section 3.4.4, FPC uses only labelled data to classify incoming articles, which was thought to reduce the number of false positives while decreasing the recall. In this section the results of the algorithm will be presented together with a discussion.

4.5.1 Results

The different parameters for the Fixed point classifier models are size, offset and kernel. Size is the number of fixed points used in the model, and is equivalent to the Size parameter in the Tree Cluster algorithm. Offset and kernel are explained in Section 3.4.4, with the caveat that the kernel is either "e" for euclidean distance or "c" for cosine distance. The values for the offset, [0, 0.2], are based on the evaluation during development, where offset greater than 0.2 showed a negative trend. The cross validation results with precision above 0.8 can be seen in Table 4.6. The full FPC cross validation table can be found in Appendix A.

Size	Offset	Kernel	Precision	Recall	Accuracy
40	0.1	c	0.9818	0.1044	0.5508
20	0.12	c	0.9671	0.0747	0.5366
40	0.08	c	0.9585	0.2183	0.6047
20	0.1	c	0.9569	0.1468	0.5705
20	0.08	c	0.9332	0.2637	0.6218
40	0.05	c	0.9059	0.4739	0.7122
40	0.12	c	0.9000	0.0383	0.5190
20	0.05	c	0.8701	0.4753	0.7016
4	0.15	c	0.8603	0.1020	0.5422
40	0.2	e	0.8533	0.3961	0.5894
40	0.15	e	0.8461	0.4356	0.6053
40	0.12	e	0.8439	0.4572	0.6128
40	0.08	e	0.8376	0.4678	0.6161
40	0.1	e	0.8335	0.4606	0.6131
40	0.05	e	0.8332	0.5056	0.6306
40	0	e	0.8165	0.5333	0.6369
4	0.12	c	0.8103	0.1803	0.5692
4	0.1	c	0.8012	0.2535	0.5947
20	0.15	c	0.8000	0.0179	0.5087

Table 4.6: Fixed Point Classifier evaluation results

4.5.2 Discussion

This model yields high precision while maintaining a recall above 10% using 40 articles, an offset of 0.1 and cosine distance as kernel. A balance of parameters can be noticed, where an identical model with an offset of 0.08 instead of 0.1 drop approximately 3 percent in precision while doubling the recall. On the other hand using the same model but with an offset of 0.12 it drops in both precision and recall.

One can also determine that cosine distance performed better than euclidean distance, with the former having an average precision of 85% and the latter 78%.

The correlation between size and precision seems to indicate that more data provides higher precision, as can be seen in Figure 4.3. The average precision is 75% for size 4, 82% for size 20 and 86% for size 40. This indicates that more data points provides a wider definition of what is considered positive.

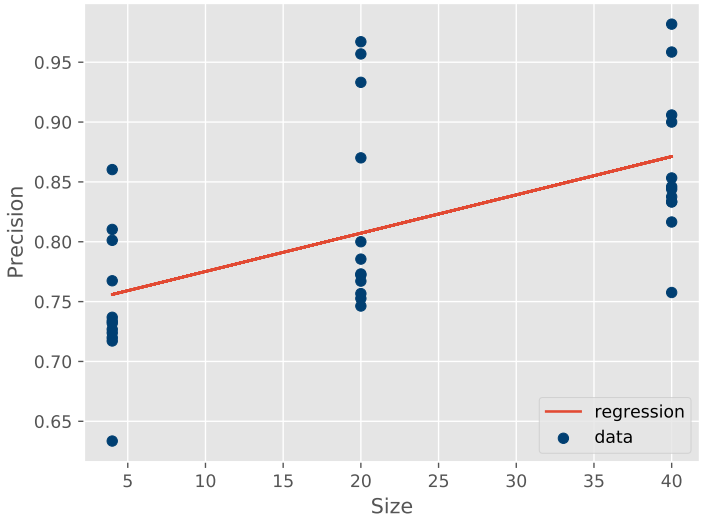

Figure 4.3: Correlation between labelled data size and precision using least squares

4.6 Overall results

A single model with the highest precision from each algorithm is displayed in Table 4.7. The best performing algorithm for this thesis is FPC, since it has the highest rated precision. The best performing TC got 6.7% lower precision with a tenth the amount of data. Standard k-means produced the best scores if all evaluation parameters would be of equal importance.

Algorithm	Size	Precision	Recall	Accuracy
Best FPC	40	0.9818	0.1044	0.5508
Best TC	4	0.9147	0.0338	0.5154
SVM[26]	480	0.8571	0.6667	0.6521
Best Supervised K-means	10	0.7361	0.5236	0.5810
Best Standard K-means	2	0.7180	0.6606	0.7005

Table 4.7: Best evaluation results for each model

Taking a look at the example with a majority of non-positive news from Section 4.3.2, with general news split 9-to-1 in favor of non-positive news, the best FPC model would yield 0.17 non-positive articles for every "good news"-article and the best TC model would yield 0.84 non-positive articles for every "good news"-article. So if we were to use the best FPC model on our website with this assumption we would get roughly 5 out of 6 positive news articles. On the other hand, if we were to use the best TC model it would be roughly an even split of positive and non-positive news articles showing up on the website.

Algorithm	Size	Precision	Recall	Accuracy
Best FPC	40	0.9818	0.1044	0.5508
SVM[26]	480	0.8571	0.6667	0.6521

Table 4.8: Best FPC model compared to related works

Table 4.8 shows a comparison between the best FPC model to the SVM model, described in Section 2.5. It can be observed that the precision of our model is significantly greater. For example, if we take the 9-to-1 scenario and apply it to the SVM model it yields 1.5 non-positive articles for every positive one. The results can be attributed to our focus on only precision, while the SVM model aimed for overall performance. Most of the related works does not even include the precision score at all.

4.7 Visualization of website

The resulting set of articles scraped from the news sources end up on a website if they are classified as positive, as can be seen in Figure 4.4. Each article have buttons for thumbs up and thumbs down, and the results of the ratings are saved in a database (but not currently used for anything else).

When observing what articles end up on the website when we use the FPC model with highest precision, we notice a majority of the articles being either science related. Out of 94 articles, 64 was science related, while the rest was scattered over a range of categories. These stats supports the observation that science news distinguish themselves the most from non-positive news articles, discussed while exploring the labelled dataset in Section 4.1.1.

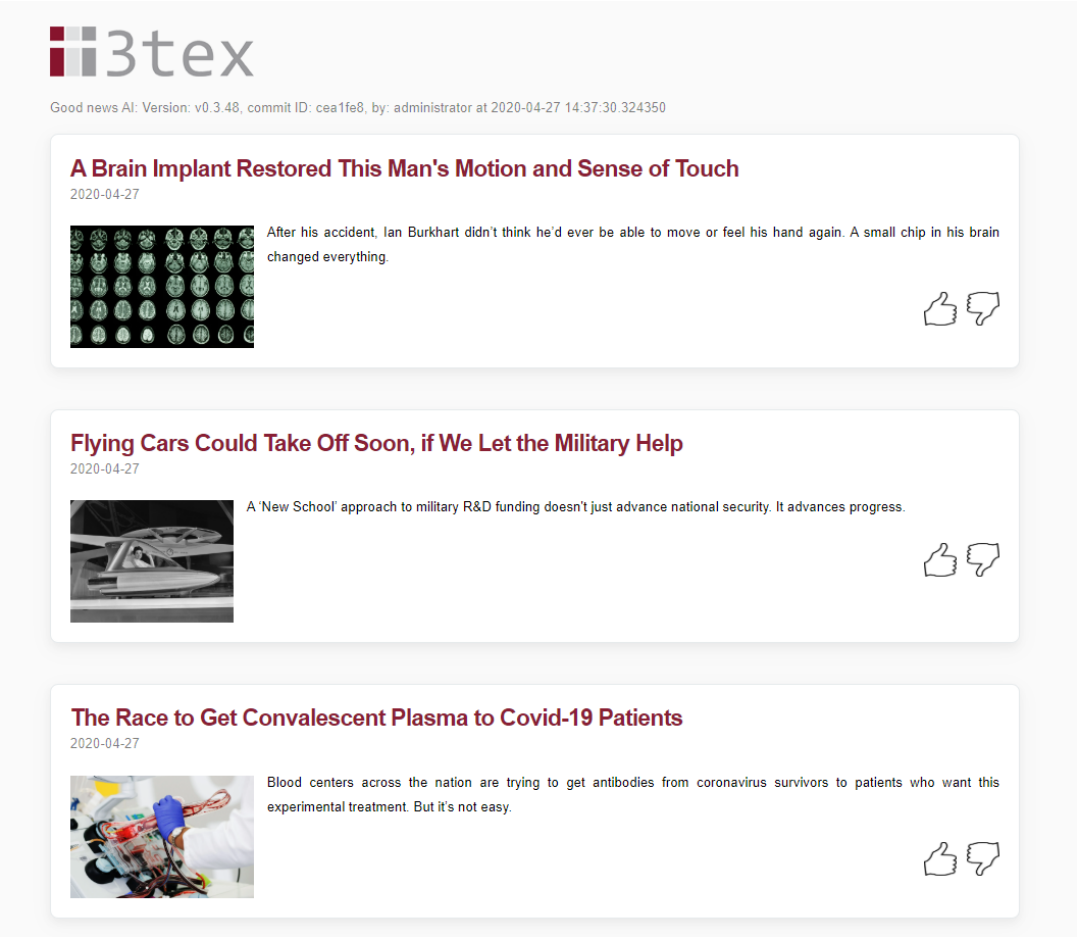

Figure 4.4: Current look of website

5

Conclusion

5.1 Conclusion

This thesis addressed the problem of classifying positive sentiment in full-length news articles, using a limited amount of labelled data. To solve this problem two sub-problems were explored: How much labelled data is needed and are there any existing methods that could solve this problem.

Existing algorithms were compared with two new algorithms with the focus being on avoiding false positives, and therefore the precision parameter was highly relevant. For existing algorithms, precision were not the main focus, so a new approach was necessary.

The first proposed algorithm, Tree Clusters, achieved a precision of 91.47%, which while above existing methods, is not high enough for the purpose of this thesis. The second proposed algorithm, Fixed Point Classifier, achieved a precision of 98.18%. That is a remarkably good result, which even in scenarios with a high imbalance of positive and non-positive news will produce a majority of positive news article. To display them, an application was developed for scraping, classifying and uploading news articles to a website.

The results show that is it possible to categorize positive sentiment from full-length news articles given a limited amount of labelled data. In this thesis 400 labelled articles were used, with at most 10 percent used as training data. The encouraging results of models with up to 40 data points support that small amounts of data can be used for precise results.

The methods proposed in this thesis performs as good or better than existing methods, while using a small amount of data, when solving the problem presented in this thesis. Previous work in general have focused on the accuracy of the models, which makes the results incompatible for comparison against the results presented in this thesis. Therefore, this thesis explores new avenues and is a step forward in the field of sentiment classification.

5.2 Future work

This thesis broaches on a wide variety of possible next steps. A limitation for this thesis was up-to-date data. Since the start of this thesis the news feeds of the world have changed remarkably, and one particular word previously unseen have been used extensively, Covid-19. For this thesis there were no aggregated datasets which included this word, which excluded the word from the models. The website currently aggregate news articles ad infinitum, and could make use of that to update the word embeddings' vocabulary so that it can interpret up-to-date news articles automatically.

The algorithms proposed could make use of continuous feedback while in use. Current functionality collects and creates a dataset based on feedback, but no further steps were taken. This data could be used to continuously train the models, which might compensate for the small amount of labelled data, and also decrease false positives.

Changing the Doc2Vec parameters or using another method for the word embeddings, such as FastText[5] and GloVe[15], is another area open for exploration.

Another possible avenue is to introduce a linguistic model, in combination with what is presented here, that focuses on identifying and filtering out key features in non-positive news articles, and thereby pushing general news toward a more balanced ratio of positive and non-positive.

From the data exploration we found some insight into what articles are considered "good news", but further work can be done to find the articles best suited to represent positive news, and therefore reduce the size of the training data even more.

One might also want to explore the possibility of using different training data for different categories of news, since some categories of news were under-represented for the best performing models.

Bibliography

[1] Larry Alexander and Michael Moore. "Deontological Ethics". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2016. Metaphysics Research Lab, Stanford University, 2016.

[2] *All the news*. https://www.kaggle.com/snapcrack/all-the-news. Accessed: 2020-04-23.

[3] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. "SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining". In: *LREC*. Ed. by Nicoletta Calzolari et al. European Language Resources Association, 2010. ISBN: 2-9517408-6-7. URL: http://nmis.isti.cnr.it/sebastiani/Publications/LREC10.pdf.

[4] *BBC*. https://www.bbc.com/news/10628494. Accessed: 2020-03-24.

[5] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *CoRR* abs/1607.04606 (2016). arXiv: 1607.04606. URL: http://arxiv.org/abs/1607.04606.

[6] Julia Driver. "The History of Utilitarianism". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2014. Metaphysics Research Lab, Stanford University, 2014.

[7] Pollyanna Gonçalves et al. "Comparing and Combining Sentiment Analysis Methods". In: *CoRR* abs/1406.0032 (2014). arXiv: 1406.0032. URL: http://arxiv.org/abs/1406.0032.

[8] i3tex. *i3tex - our vision*. https://www.i3tex.com/en/about-us/our-vision/. Accessed: 2020-02-04.

[9] Beakcheol Jang, Inhwan Kim, and Jong Wook Kim. "Word2vec convolutional neural networks for classification of news articles and tweets". In: *PLOS ONE* 14.8 (Aug. 2019), pp. 1–20. DOI: 10.1371/journal.pone.0220976. URL: https://doi.org/10.1371/journal.pone.0220976.

[10] Xin Jin and Jiawei Han. "K-Means Clustering". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 563–564. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_425. URL: https://doi.org/10.1007/978-0-387-30164-8_425.

[11] Quoc V. Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *CoRR* abs/1405.4053 (2014). arXiv: 1405.4053. URL: http://arxiv.org/abs/1405.4053.

[12] Kalev Leetaru. "Culturomics 2.0: Forecasting large-scale human behavior using global news media tone in time and space". In: *First Monday* 16.9 (Aug. 2011). DOI: 10.5210/fm.v16i9.3663. URL: https://journals.uic.edu/ojs/index.php/fm/article/view/3663.

[13] Bing Liu. "Sentiment analysis and opinion mining". In: *Morgan Claypool Publishers* (2012). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.244.9480&rep=rep1&type=pdf.

[14] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781 (2013). URL: https://arxiv.org/pdf/1301.3781.pdf.

[15] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[16] *Reuters*. http://feeds.reuters.com/reuters/topNews. Accessed: 2020-03-24.

[17] *Science Daily*. https://www.sciencedaily.com/. Accessed: 2020-04-15.

[18] Nagesh Bhattu Sristy and D.V.L.N Somayajulu. "Semi-supervised Learning of Naive Bayes Classifier with feature constraints". In: *Proceedings of the First International Workshop on Optimization Techniques for Human Language Technology*. Mumbai, India: The COLING 2012 Organizing Committee, Dec. 2012, pp. 65–78. URL: https://www.aclweb.org/anthology/W12-6105.

[19] *The Guardian*. https://www.theguardian.com/world/rss. Accessed: 2020-03-24.

[20] *The Independent*. http://www.independent.co.uk/news/world/rss. Accessed: 2020-03-24.

[21] Peter D. Turney. "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews". In: *CoRR* cs.LG/0212032 (2002). URL: http://arxiv.org/abs/cs/0212032.

[22] *UN News*. https://news.un.org/en/rss-feeds. Accessed: 2020-03-24.

[23] Kiri Wagstaff and Claire Cardie. "Clustering with Instance-level Constraints". In: *AAAI/IAAI* 1097 (2000), pp. 577–584. URL: https://www.aaai.org/Papers/AAAI/2000/AAAI00-180.pdf.

[24] Wei Wei and Xiaojun Wan. "Learning to Identify Ambiguous and Misleading News Headlines". In: *CoRR* abs/1705.06031 (2017). arXiv: 1705.06031. URL: http://arxiv.org/abs/1705.06031.

[25] *Wired*. https://www.wired.com/. Accessed: 2020-04-15.

[26] Kiran Shriniwas Doddi Yashodhara Haribhakta. "Categorization of News Articles using Sentiment Analysis". In: *International Journal of Scientific Research in Computer Science* 2 Issue 5 (September-October 2017), pp. 52–60. ISSN: 2456-3307.

A

Appendix 1

The stemmer used in this thesis was NLTK:s Snowball stemmer.

A.1 Complete result tables

The following are the complete result tables for the FPC, Table A.1, and TC, Table A.2, classifier evaluation.

Size	Offset	Kernel	Precision	Recall	Accuracy
40	0.1	c	0.9818	0.1044	0.5508
20	0.12	c	0.9671	0.0747	0.5366
40	0.08	c	0.9585	0.2183	0.6047
20	0.1	c	0.9569	0.1468	0.5705
20	0.08	c	0.9332	0.2637	0.6218
40	0.05	c	0.9059	0.4739	0.7122
40	0.12	c	0.9000	0.0383	0.5190
20	0.05	c	0.8701	0.4753	0.7016
4	0.15	c	0.8603	0.1020	0.5422
40	0.2	e	0.8533	0.3961	0.5894
40	0.15	e	0.8461	0.4356	0.6053
40	0.12	e	0.8439	0.4572	0.6128
40	0.08	e	0.8376	0.4678	0.6161
40	0.1	e	0.8335	0.4606	0.6131
40	0.05	e	0.8332	0.5056	0.6306
40	0	e	0.8165	0.5333	0.6369
4	0.12	c	0.8103	0.1803	0.5692
4	0.1	c	0.8012	0.2535	0.5947
20	0.15	c	0.8000	0.0179	0.5087
20	0.2	e	0.7855	0.5295	0.6000
20	0.15	e	0.7731	0.5595	0.6055
20	0.12	e	0.7722	0.5616	0.6055
4	0.08	c	0.7674	0.3217	0.6121
20	0.1	e	0.7671	0.5874	0.6111
40	0	c	0.7576	0.8872	0.8003
20	0.08	e	0.7567	0.5837	0.6032
20	0.05	e	0.7526	0.5974	0.6082
20	0	e	0.7463	0.6084	0.6074
4	0.12	e	0.7369	0.4672	0.5492
4	0.1	e	0.7338	0.4722	0.5477
4	0.15	e	0.7319	0.4606	0.5472
4	0.05	e	0.7269	0.4773	0.5523
4	0.08	e	0.7239	0.4828	0.5477
4	0.05	c	0.7199	0.4444	0.6374
4	0	e	0.7171	0.4859	0.5477
4	0	c	0.6335	0.6652	0.6402

Table A.1: Fixed Point Classifier evaluation results

Type	Size	Depth	n	Precision	Recall	Accuracy
Double	4	1	5	0.9147	0.0338	0.5154
Single	4	3	5	0.9074	0.0332	0.5143
Double	40	2	5	0.9064	0.1217	0.5547
Double	20	2	5	0.8965	0.0600	0.5271
Single	40	2	5	0.8952	0.2181	0.5959
Single	40	1	5	0.8841	0.0944	0.5413
Double	20	1	5	0.8808	0.1547	0.5668
Single	40	1	10	0.8751	0.2256	0.5966
Single	4	2	10	0.8718	0.0730	0.5321
Double	20	2	10	0.8703	0.1768	0.5753
Single	20	2	5	0.8623	0.0961	0.5406
Single	20	1	10	0.8546	0.0989	0.5408
Double	40	1	10	0.8522	0.1050	0.5433
Single	40	3	5	0.8418	0.3012	0.6222
Double	40	2	10	0.8368	0.2994	0.6206
Double	4	1	10	0.8345	0.0990	0.5412
Double	4	3	5	0.8341	0.1010	0.5396
Single	4	4	5	0.8337	0.0541	0.5212
Single	20	2	10	0.8233	0.3006	0.6172
Single	20	3	5	0.8224	0.1694	0.5664
Single	40	2	10	0.8187	0.4725	0.6834
Double	40	1	5	0.8167	0.2761	0.6081
Double	20	1	10	0.8151	0.3911	0.6508
Single	4	3	10	0.8143	0.1474	0.5571
Double	4	2	10	0.8120	0.0323	0.5145
Double	20	4	5	0.8091	0.1605	0.5618
Double	4	4	5	0.8036	0.0389	0.5165
Single	20	4	5	0.8033	0.2550	0.5961
Double	4	2	5	0.8000	0.0101	0.5048
Double	40	4	5	0.7967	0.2517	0.5933
Double	20	3	5	0.7905	0.3779	0.6384
Single	40	4	5	0.7897	0.3794	0.6387
Single	20	1	5	0.7739	0.0333	0.5133
Double	4	4	10	0.7674	0.1747	0.5634
Single	4	2	5	0.7667	0.0158	0.5073
Single	4	4	10	0.7649	0.2806	0.5959
Double	40	1	10	0.7588	0.5578	0.6908
Single	20	3	10	0.7557	0.4878	0.6647
Double	40	3	5	0.7531	0.5128	0.6725
Single	40	3	10	0.7269	0.6356	0.6984
Double	40	4	10	0.7197	0.5906	0.6800
Single	4	1	10	0.7167	0.0143	0.5064
Double	4	3	10	0.7096	0.3742	0.6136
Double	20	3	10	0.6842	0.7579	0.7037
Single	20	4	10	0.6791	0.6439	0.6692
Single	40	4	10	0.6556	0.7350	0.6744
Double	40	3	10	0.6458	0.8606	0.6936

Table A.2: Tree Clusters evaluation results