KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science

Second cycle, 30 credits

A comparison of different methods in their ability to compare semantic similarity between articles and press releases

JULIUS ANDERSSON

Stockholm, Sweden, 2022

A comparison of different methods in their ability to compare semantic similarity between articles and press releases

JULIUS ANDERSSON

Master's Programme, Computer Science, 120 credits
Date: September 29, 2022

Supervisors: Yulia Sidorova, Peter Braroe
Examiner: Olov Engwall
School of Electrical Engineering and Computer Science
Host company: NewsMachine AB
Swedish title: En jämförelse av olika metoder i deras förmåga att jämföra semantisk likhet mellan artiklar och pressmeddelanden

Abstract

The goal of a press release is to have the information spread as widely as possible. A suitable approach to distribute the information is to target journalists who are likely to distribute the information further. Deciding which journalists to target has traditionally been performed manually without intelligent digital assistance and therefore has been a time consuming task. Machine learning can be used to assist the user by predicting a ranking of journalists based on their most semantically similar written article to the press release. The purpose of this thesis was to compare different methods in their ability to compare semantic similarity between articles and press releases when used for the task of ranking journalists. Three methods were chosen for comparison: (1.) TF-IDF together with cosine similarity, (2.) TF-IDF together with soft-cosine similarity and (3.) sentence mover's distance (SMD) together with SBERT. Based on the proposed heuristic success metric, both TF-IDF methods outperformed the SMD method. The best performing method was TF-IDF with soft-cosine similarity.

Keywords

Semantic similarity, TF-IDF, SBERT, Cosine similarity, Soft-cosine similarity, Sentence mover's distance

Sammanfattning

Målet med ett pressmeddelande är att få informationen att spriddas till så många som möjligt. Ett lämpligt tillvägagångssätt för att sprida informationen är att rikta in sig på journalister som sannolikt kommer att sprida informationen vidare. Beslutet om vilka journalister man ska rikta sig till har traditionellt utförts manuellt utan intelligent digital assistans och har därför varit en tidskrävande uppgift. Maskininlärning kan användas för att hjälpa användaren genom att förutsäga en rankning av journalister baserat på deras mest semantiskt liknande skrivna artikel till pressmeddelandet. Syftet med denna uppsats var att jämföra olika metoder i deras förmåga att jämföra semantisk likhet mellan artiklar och pressmeddelanden när de används för att rangordna journalister. Tre metoder valdes för jämförelse: (1.) TF-IDF tillsammans med cosinus likhet, (2.) TF-IDF tillsammans med mjuk-cosinus likhet och (3.) sentence mover's distance (SMD) tillsammans med SBERT. Baserat på det föreslagna heuristiska framgångsmåttet överträffade båda TF-IDF-metoderna SMD-metoden. Den bäst presterande metoden var TF-IDF med mjuk-cosinus likhet.

Nyckelord

Semantisk likhet, TF-IDF, SBERT, Cosinus likhet, Mjuk-cosinus likhet, Sentence mover's distance

Acknowledgments

I would like to thank everyone in NewsMachine AB and especially Peter Braroe, my supervisor at NewsMachine for his guidance and support. I would also want to thank my supervisor at KTH, Yulia Sidorova for her support and suggestions.

Stockholm, September 2022
Julius Andersson

Contents

Chapter 1

Introduction

The goal of a press release is to distribute its information to as many as possible. This can be done in many ways. One way to do this is to simply send the press release to everyone by email, for example. However, if everyone used this approach to distribute their press releases, the messages would likely be considered spam. Another way could be to use ads to distribute the press release. However, ads are often expensive to use. A more cost-effective approach is to target journalists who are likely to distribute the information further. By sending the press release to interested journalists, they will hopefully write articles about it. Therefore, the information contained in the press release will be distributed cost-effectively. However, it is not possible to send the press release to all existing journalists as this will face the same problem already mentioned. Therefore, the difficulty with this approach lies in deciding which journalists to target. This has traditionally been performed manually but it is a time consuming task. If the person who is planning to distribute a press release could be assisted somehow in their search for relevant journalists, the time required could be reduced. Machine learning could be a way to assist the person by predicting a ranking of the journalists according to their interest in the press release.

1.1 Background

The problem of ranking journalists by their interest in the press release is a difficult one for machine learning algorithms because of the lack of labeled data. The ideal data would be, to have for each press release, the journalists' interest in it. This data is however difficult to collect and therefore other approaches must be found that do not need the problem-specific labeled data.

NewsMachine AB is a Swedish company that specializes in media coverage, publicity and analysis. NewsMachine has been collecting news articles and the journalists who wrote them from the web for many years. By using this data, it is possible to instead rank the journalists by their most semantically similar written article to the press release. It is expected that this ranking is correlated to the ranking of journalists based on their interest in the press release. Ranking the journalists by their most semantically similar written article to the press release could therefore be used as an approximation. The question is then which method to use when comparing the semantic similarity between an article and a press release. This thesis aims to test whether a newer method based on the BERT architecture can outperform simpler traditional methods in this regard.

1.2 Objective of this thesis

The objectives of this thesis are: (1.) to compare traditional methods against a method based on the BERT architecture in their ability to compare the semantic similarity between articles and press releases when used for the task of ranking journalists and (2.) to compare the methods' time and memory consumption.

1.3 Delimitations

This thesis will only deal with articles and press releases written in the Swedish language.

1.4 Ethical and societal aspects

When using machine learning models to solve problems there are always computational resources needed. Training large neural networks can consume a large amount of energy. Instead of training a sentence BERT model, a pre-trained model was used to try to follow the Sustainable Development goals of the United Nations [1]. However, this thesis did not implement all methods as efficiently as possible due to time constraints. Of course, once a choice of method has been made, the method should be more optimized to work efficiently.

Another ethical aspect is how the journalists are chosen. For example, there have been discussions about how Google's search engine is suggesting

its results. It is natural to ask whether the selection of pages is fair. The same question can be asked about the ranked journalists of a press release. Are the results from the method fair? Because the ranking of journalists is based on their most semantically similar article to the press release, each user can easier check if the results are fair.

Lastly, the constructed ranking of journalists is currently only intended to assist users in their search for relevant journalists. However, in the future, a system could be developed to fully automate this task, which could lead to people losing their jobs. One of the reasons why this task is easier to automate is that there are no major consequences if the press release is sent to an incorrect journalist from time to time. Nevertheless, to maintain a system and further improve it, there will be a need for people with this kind of knowledge. This is just a natural consequence of technological progress, many jobs from one hundred years ago do not exist today but new one have emerged.

Chapter 2

Background

2.1 Text pre-processing

To prepare the data for further processing, a text pre-processing step is often necessary. This step can also include removing noise and to simplify the data. Some possible approaches will be mentioned below [2].

2.1.1 Tokenization

Tokenization is the task of splitting text into tokens. One obvious way to do this is by splitting the text into words. There are however more complex ways of doing it, e.g. WordPiece which is used in BERT [3].

2.1.2 Stop words

Words that do not contain much information can be removed to both simplify and remove noise from the data. Examples of these words are ”a”, ”the” and ”is”.

2.1.3 Lemmatization

Words can have different forms. Lemmatization is the process of transforming each word to its base form. For example, playing, plays and played will be transformed to play.

2.2 Embeddings

2.2.1 Word embeddings

2.2.1.1 One hot encoding

The easiest way of representing words as vectors is one hot encoding [2]. Here, one simply uses a vector with the size of the vocabulary. Each word is assigned an index and the vector representation is having all the vector components set to zero except the component at the index. This component will have its value set to one. In this way, it is very easy to assign a vector for each word. However, there exist problems with this approach. The vector space has a dimension equal to the vocabulary which can be very large. Also, the Euclidean distance between two words is independent of the words. In many cases, it is desirable to have a vector representation where similar words are closer to each other in the vector space.

2.2.1.2 Word2vec

Word2vec is an approach that addresses the problems previously described. The word2vec model is a shallow neural network with 2 layers. There exist two different types of this model, the continuous bag of words (CBOW) model and the skip-gram model which both are trained on a corpus [2]. The CBOW model is trained by giving the surrounding words, try predicting the center word and the skip-gram model is trained by giving the center word, try predicting the surrounding words. The weights of the model can then be used to get the vector representation of each word. The vectors for similar words will lie closer in the vector space. A common value of the dimension of the space is around 100-300 and we, therefore, typically, have a large reduction in dimensions compared to the one hot encoding. However, each word will be given its vector representation without any consideration of the context in which the word is used. A common example, to highlight this, is the two phrases ”We are going to the river bank” and ”We are going to the bank to deposit money”. Here the word bank has two different meanings but will be given the same vector representation.

2.2.1.3 BERT

After the paper ”Attention is all you need” [4] was published, much focus has been spent on transformer networks. The BERT (Bidirectional Encoder

Representations from Transformers) model was released in 2018 and uses an encoder architecture. It was pre-trained on masked language modeling and next sentence prediction [5]. Because of its architecture, BERT can represent words by taking into consideration their context. In the previously mentioned example, the word bank will have a different embedding in the two different cases. This is an improvement from the word2vec model. BERT has however mainly been used by fine-tuning it for other NLP (natural language processing) downstream tasks. In their paper, they reported state of the art performance in many NLP tasks. Because of the release of their pre-trained model, it is possible to fine-tune it on other tasks with small computational resources compared to those used to pre-train it.

2.2.2 Text Embeddings

2.2.2.1 Term frequency

By having a fixed vocabulary, a document is transformed into a vector representation by summing over every word's one hot encoding [2]. Words not in the vocabulary are ignored. This representation is a simple one and much information is lost, such as the relationship between words, etc.

2.2.2.2 Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) is an extension of the term frequency representation [2]. First, the term frequency is calculated as before. However, some words might be used in a lot of documents and therefore do not carry much information. To lessen these words' impact, the inverse document frequency gives less weight to words that occur in many of the documents in the corpus. The weight for each word w in a document d is calculated by,

$$TF\_IDF(w, d, c) = Freq(w, d) * \log(\frac{|c|}{df(w)}), \tag{2.1}$$

where Freq(w,d) is how many times the word occurs in the document, $|c|$ is the size of the corpus and lastly, $df(w)$ is the number of documents that contain the word.

2.2.2.3 Latent semantic analysis

Latent semantic analysis (LSA) is a topic model which uses matrix factorization to find a lower dimensional document representation [6]. A document-term matrix X is a matrix where each column is a term frequency representation of a document. Truncated singular value decomposition (SVD) can be applied to lower the dimensions of X,

$$X_{m,n} \approx U_{m,k}\Sigma_{k,k}V^T_{n,k}. \tag{2.2}$$

The approximation is due to only using the first k singular values in the Σ matrix. Document i, can be represented as a lower dimensional vector $\Sigma_{k,k}\hat{v_i}$, where $\hat{v_i}$ is the ith column in V^T. The algorithm for truncated SVD can be performed efficiently and new documents can be added without redoing all the calculations.

2.2.2.4 Latent Dirichlet allocation

Latent Dirichlet allocation (LDA) was introduced by Andrew Y. Ng et al. in 2003 [7] and it is a generative probabilistic model. Like LSA it is also a topic model. Each document is represented as a distribution over latent topics [8]. The number of possible latent topics is a parameter that can be chosen freely. To estimate the LDA model's parameters, Gibbs sampling or the Expectation-Maximization method can be performed.

2.2.2.5 Text embeddings by using word embeddings

It is possible to get a text embedding by simply taking the average of all word embeddings in the text. Reimers and Gurevych [9] compared several methods in various textual similarity tasks (STS). They found that using the average of GloVe embeddings outperformed the average of BERT embeddings. Li et al. looked further into the embedding space of BERT [10]. One of their findings was that using the two last layers of BERT did outperform Glove by a small amount. Ethayarajh [11] found that the word embeddings in BERT, only occupy a narrow cone in the vector space. This is an undesired property because the cosine similarity between any two words will have a high value. Furthermore, Li et al. found that this is also the case for sentences [10]. They did also find that the word frequency biases the embedding space. Low-frequency words in the training data were, on average, further away from the origin than high-frequency words.

2.2.2.6 Sentence BERT

Fine-tuning BERT for sentence semantic similarity tasks has shown state of the art performance [9]. However, when comparing two sentences, both have to be fed into the network at the same time. If you would like to find the most similar sentence pair from a collection of sentences, you are forced to compare all possible pairs which are very computationally expensive. With this in mind, Reimers and Gurevych [9] created the model Sentence BERT (SBERT) to embed sentences. To compare the semantic similarity between two sentences, it is possible to just use the cosine similarity on their embeddings. The task of finding the most similar sentence pair from a collection of 10 000 sentences took around 65 hours using BERT while only around 5 seconds using SBERT.

SBERT starts with a pre-trained BERT model. It then adds a pooling operation on top of BERT to get a fixed size embedding for a sentence. The chosen pooling operation was to take the average of all the output vectors from BERT. In one of their experiments, the network was fine-tuned using a siamese network strategy. The training data used were the SNLI and multi-genre NLI datasets. Together, they consist of around 1 million sentence pairs with the labels neutral, contradiction and entailment. First, they did send each sentence through the BERT and pooling layer to get the embeddings v and u. Next, they concatenate the embeddings u and v together with the difference $|u - v|$. This was fed into a linear layer and then a softmax layer. Lastly, a cross-entropy loss was used. The performance was tested on the SemEval STS tasks 2012-2016, the STS benchmark and the SICK-Relatedness datasets. The data of these datasets consist of sentence pairs where the label is the semantic relatedness of the pair with a score from 0 to 5. SBERT showed state of the art performance on these tests in regard to other sentence embedding methods.

2.3 Distance and similarity measures

To compare two vectors or distributions a measurement of distance or similarity is needed. Several alternatives are mention below.

2.3.1 Euclidean distance

The Euclidean distance is perhaps the first distance measure a human would think of. It is the length of a straight line connecting two points.

Mathematically, it is described by

$$EucDist(A, B) = \sum_i \sqrt{(A_i - B_i)^2}, \tag{2.3}$$

where A and B are vectors.

2.3.2 Cosine similarity

Given two vectors, it is possible to measure the angle θ between them. The similarity is then given by taking the cosine of the angle. The similarity is given by [12]

$$CosSim(A, B) = cos(\theta) = \frac{A^T B}{\sqrt{A^T A}\sqrt{B^T B}}. \tag{2.4}$$

Using cosine similarity together with, for example, a term frequency representation of a text, one important problem can be seen. Two texts with no words in common will have a cosine similarity of zero even though many words might have a similar meaning. For example, consider the two sentences ”Biden speaks in New York” and ”The President talks in Manhattan”. These sentences are similar but would have a low similarity score using term frequency with cosine similarity.

2.3.3 Soft-cosine similarity

Soft-cosine similarity aims to solve the previous problem by introducing a relation matrix M where each element $m_{i,j}$ describes the relation between the word i and j. The similarity is then defined as [12],

$$SoftcosSim(A, B) = \frac{A^T M B}{\sqrt{A^T M A}\sqrt{B^T M B}}. \tag{2.5}$$

If each word is only related to itself then M is the identity matrix and the soft-cosine similarity is equal to the cosine similarity. The matrix M can be constructed in many ways but Charlet et al. [12] did use word2vec to find the relation between words. They found that the following equation worked well for constructing M,

$$m_{i,j} = max(0, CosSim(v_i, v_j))^2, \tag{2.6}$$

where v_i and v_j is the word2vec representation of word i and j.

2.3.4 Word and sentence mover's distance

The paper "From Word Embeddings To Document Distances" [13] presented a new distance between documents. By having a word embedding, it is possible to measure the distance between two documents by measuring the minimum distance the embedded words have to travel to transform one document into the other. If having two documents d_1 and d_2, they did use TF-IDF to get two vectors v_1, v_2 representing the documents. Introduce a flow matrix T where $T_{i,j}$ represent how much word i in d_1 is travelling to word j in d_2. The flow has to be non-negative $T_{i,j} \geq 0$. If d_1 is to be transformed entirely into d_2 there are two conditions on T that has to be fulfilled. First, the flow out from each word i in d_1 has to be equal to its weight,

$$\sum_j T_{i,j} = v_1^i. \tag{2.7}$$

Second, the incoming flow for each word in d_2 has to be equal to its weight,

$$\sum_i T_{i,j} = v_2^j. \tag{2.8}$$

The cost of transforming d_1 to d_2 is then,

$$\sum_{i,j} T_{i,j} * c(i,j), \tag{2.9}$$

where $c(i,j)$ is the cost of travel from word i to j. In the paper, it was chosen to be the Euclidean distance in the embedding space [13]. The distance between two documents is then defined as the minimum cost of transforming one into the other which is a linear optimization problem. The average time complexity of this problem is $O(p^3 log(p))$, where p is the size of unique words in the documents. Therefore the distance can be slow to compute when p is large.

The word mover's distance is just a case of the more general earth mover's distance. There is therefore no restriction in only using word embeddings. For example, sentence mover's distance (SMD) has also been implemented [14].

2.3.5 Jensen-Shannon divergence

To measure similarity between two probability distributions, the method Jensen-Shannon (JS) divergence can be used. It is commonly used with LDA

to compare document similarity [15]. The formula for JS divergence is,

$$JS(P_1||P_2) = \frac{1}{2}KL(P_1||\frac{P_1+P_2}{2}) + \frac{1}{2}KL(P_2||\frac{P_1+P_2}{2}), \tag{2.10}$$

where P_1 and P_2 are two probability distributions and KL is the Kullback-Leibler divergence, also called relative entropy,

$$KL(P_1||P_2) = \sum_i P_1(x_i) \log \frac{P_1(x_i)}{P_2(x_i)}, \tag{2.11}$$

where x_i is the possible outcomes.

2.4 Sentence BERT from the National Library of Sweden

As our data consist of only Swedish articles, the amount of available pre-trained models will be small. Fortunately, the National Library of Sweden (KB) has trained several models and made them available to the public. One of these models is SBERT. Due to limited training data, KB used the approach of knowledge distillation [16], namely, transferring knowledge from one model to another. To make a model operating in one language able to handle multiple languages using knowledge distillation was introduced by Reimers and Gurevych in 2020 [17]. SBERT was trained on a large number of labeled sentence pairs in English. Such labeled data does not exist in many languages including Swedish. What is easier to come by is English sentences translated to the language of interest. Denote these as (E_i, S_i), where E_i is the English sentence and S_i is the corresponding Swedish sentence. The idea is that the translated sentence should lie in the same location in the embedding space as the corresponding English sentence. The trained English model M is called the teacher model. It is then possible to train a new student model $\hat{M}$ in both English and Swedish by getting the embedding for E_i by the teacher model and the embeddings for both E_i and S_i by the student model. For each sample, the training loss for the student model is

$$(M(E_i) - \hat{M}(E_i))^2 + (M(E_i) - \hat{M}(S_i))^2. \tag{2.12}$$

KB used paraphrase-mpnet-base-v2* as the teacher model and the Swedish pre-trained BERT model as the student†.

* https://www.sbert.net/docs/pretrained_models.html#sentence-embedding-models
† https://huggingface.co/KB/bert-base-swedish-cased

Chapter 3

Method

3.1 Data

NewsMachine AB continuously scrapes the web for news articles. The data used in this thesis consisted of 78 163 data points of the form <text article, authors>. These articles had previously been extracted from some of the major news sites in Sweden (Aftonbladet, Svenska Dagbladet, SVT Nyheter, Expressen and Dagens Nyheter). Since the articles were automatically extracted, they may contain metadata and small errors such as spelling errors.

3.2 Methods

In this section, the chosen methods for comparing the semantic similarity between an article and a press release will be described. The first chosen method was a TF-IDF model used together with cosine similarity. This method is often used as a baseline [18][19] and although it is an old method it can still achieve good performance in some cases [6][19]. As described in the background section, cosine similarity does not take into consideration any relationship between different words. The second chosen method, therefore, used soft-cosine similarity instead of cosine similarity. The last chosen method used sentence mover's distance (SMD) together with SBERT. This method differs from the others in that it uses a bag of sentences instead of a bag of words. While information is lost by using a bag of sentences model it should be less than in using a bag of words model.

Two commonly used methods when comparing documents are the topic models LDA and LSA. Yet, they will not be tested in this thesis. The reason behind this is that the values for the hyper parameters must be set such as the

number of topics. It will be too expensive to perform a search for optimal values without having access to a labeled dataset.

3.2.1 Cosine TF-IDF

First, lemmatization was performed on all the articles. The Stanza package* was used for this. Next, the articles were embedded by using TF-IDF. The scikit-learn library† was used for this purpose. The vocabulary was built on the articles. The size of the vocabulary and therefore the dimension of the embeddings was 685 780. The press releases were first lemmatized and then embedded using TF-IDF with the IDF factors that were calculated from all the articles. Comparing the similarity between the embeddings of a press release and an article was done using cosine similarity (see equation 2.4).

3.2.2 Soft TF-IDF

To perform soft-cosine similarity, a word embedding must be chosen. The chosen embedding was the Swedish word2vec‡. The Gensim library§ was used to load the word2vec model. Equation 2.5 was used for computing the soft-cosine similarity, with M being calculated as in equation 2.6. However, if the vocabulary is of size m, the dimension of the similarity matrix is $m \times m$. This is a very large matrix if the size of the vocabulary is 685 780 as it was in cosine TF-IDF. If each entry in the matrix is represented by eight bytes the matrix will require more than three TB of memory. The word2vec model, however, only supports around 200 000 words. Therefore, many entries in the matrix will be zero and using a sparse matrix representation could decrease the memory consumption. Still, only using the supported words as the vocabulary would require around 320 GB. There was an attempt to only compute the matrix elements needed for each matrix multiplication. This drastically decreased the memory usage but instead slowed down the calculations to the degree it was unusable. The final solution was to limit the vocabulary. Only the 10 000 most frequent words in the articles were used as vocabulary.

3.2.3 SMDSBERT

To calculate the similarity between an article and a press release, SMD was used where each sentence was weighted equally. A pre-trained SBERT model

* https://stanfordnlp.github.io/stanza/lemma.html
† https://scikit-learn.org/
‡ https://www.ida.liu.se/divisions/hcs/nlplab/swectors/
§ https://radimrehurek.com/gensim/

from The National Library of Sweden was used for the sentence embeddings*. A python library was used for calculating the SMD†. This library uses the Euclidean distance as the cost of traveling from one sentence to another. The SBERT model was however trained to maximize the cosine similarity between semantically similar sentences. It was assumed that the SMD would be slow to compute and because the system should not be too slow, this highly optimized library was used instead of implementing a version that uses cosine similarity. By using the Euclidean distance, smart tricks can be used to lower the running time.

3.3 Experiment

The experiment aimed to compare the methods in their ability to compare the semantic similarity between articles and press releases when being used to rank journalists. When deciding which journalists to target for a press release, it is most likely, that only the highest ranked journalists will be the right target. Therefore, and because of time constraints, the experiment did only compare the methods' abilities in finding the five most semantically similar articles to a press release. To further simplify the experiment, a 3-point scale was used to score each of the predicted articles separately. The experiment is described in more detail below:

1. 100 random press releases were chosen.
2. For each press release, all the methods predicted their five most semantically similar articles to it from the set of all articles.
3. Each predicted article was manually assigned a score from a 3-point scale according to the same topic (2 points), related topic (1 point) or not at all related (0 points) to the press release.

3.3.1 Test environment

The processor used was a Ryzen 3200g from AMD and it has a total of four cores. The system had 16 GB of RAM and no external GPU was used.

* https://huggingface.co/KBLab/sentence-bert-swedish-cased
† https://github.com/src-d/wmd-relax

3.4 Heuristic success metric

A metric was needed in order to do a comparison of the methods' abilities in finding the five most semantically similar articles to a press release. It was decided to use a heuristic success metric, namely, the mean score μ a method received,

$$\mu = \sum_i \frac{1}{n} s(i), \tag{3.1}$$

where $s(i)$ is the score awarded by the ith article and n is the total number of predicted articles. For example, if a method predicts 5 articles to a press release, the articles could have been assigned the following distribution of scores [0,2,1,2,1]. According to the metric, the method's score is $\frac{0+2+1+2+1}{5} = 1.2$. The metric score will lie between 0 and 2.

The metric is a heuristic and is expected to measure the general performance of a method. One of the reasons why it was chosen was that by using the mean, statistical tools can be used to analyze the results further.

3.4.1 Statistical analysis

The standard deviation can be calculated as,

$$\sigma = \sqrt{\frac{1}{n-1} \sum_i (s(i) - \mu)^2}. \tag{3.2}$$

When testing different methods, their estimated means and standard deviations will be different. For example, method 1 may have an estimated mean $\hat{\mu}_1$ and method 2 an estimated mean $\hat{\mu}_2$, where $\hat{\mu}_1 > \hat{\mu}_2$. Welch's t-test [20] can be used to estimate the probability for the hypothesis that the true mean μ_2 is instead greater than the true mean μ_1. However, Welch's t-test assumes that the underlying distributions are normally distributed which is not true in our case. Nevertheless, it is possible to prove that Welch's t-test still can be used for non-normal distributions [21] when using larger sample sizes.

Chapter 4

Results

4.1 Performance

Table 4.1, presents the sum of scores, the mean score and the standard deviation for each method. Figure 4.1 displays the distribution of received scores for each method.

The training time i.e. the time spent embedding all articles (and creating the similarity matrix in the case of soft TF-IDF) is shown for each method in Table 4.2. This Table also contains the average time required to predict the five most similar articles to the press release and the memory consumption for the embeddings (and the similarity matrix in the case of soft TF-IDF). Lastly, Table 4.3 shows the average number of characters in the predicted articles from each method. The number of unique articles predicted by the methods is also displayed there.

	Cosine TF-IDF	Soft TF-IDF	SMDSBERT
Sum of scores	472	487	353
Mean score	0.944	0.974	0.706
Standard deviation	0.894	0.805	0.805

Table 4.1: The sum of scores, the mean score and the standard deviation for each method. The highest mean score possible is 2.

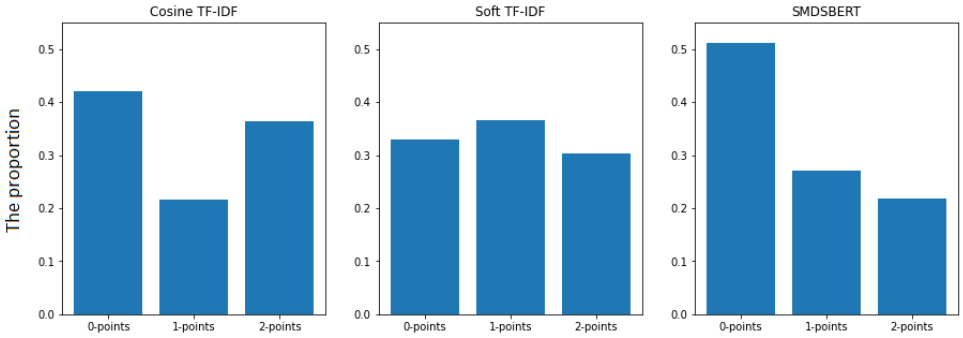

Figure 4.1: Distribution of received scores for each method.

	Cosine TF-IDF	Soft TF-IDF	SMDSBERT
Total training time (hours)	16.7	17.0	42.4
The average run time for one press release (minutes)	1.7	8.4	0.6
The memory consumption (GB)	0.19	0.66	6.4

Table 4.2: Running time and memory consumption.

	Cosine TF-IDF	Soft TF-IDF	SMDSBERT
Average number of characters	7464	8127	1494
Unique articles	400	405	273

Table 4.3: The average number of characters in the predicted articles and the number of unique articles for each method.

Chapter 5

Discussion

5.1 Ability to find semantically similar articles

It can be seen, from Table 4.1, that SMDSBERT obtained the lowest sum of scores and therefore also the lowest mean score. By using Welch's t-test it is highly unlikely (less than 0.1%) that SBERT's true mean score is to be higher than the TF-IDF methods. Soft TF-IDF had the highest sum of scores closely followed by cosine TF-IDF. However, it is not possible to draw the statistically significant conclusion that the true mean score is higher for soft TF-IDF than cosine TF-IDF.

To analyze the methods in greater detail, Figure 4.1 displays the distribution of different scores for each method. Soft TF-IDF had the lowest amount of articles that received zero points, which was given if the article was not related to the topic of the press release. Cosine TF-IDF had the highest amount of articles that received two points. To get two points, the article had to be on the same topic as the press release.

There could be several reasons for this result. There are two differences between cosine TF-IDF and soft TF-IDF. The first is the size of the vocabulary. Only the 10 000 most frequent words were used in the vocabulary for soft TF-IDF while the size of the vocabulary for cosine TF-IDF was 685 780. The cosine TF-IDF method has the advantage that it can compare the similarity between two documents with the help of very rare words. Because these words are rare, they are assigned a large IDF factor and in many cases, it was seen that these rare words were keywords. This may explain why the cosine TF-IDF method had a high amount of articles that received two points. However, rare words are not always important when deciding the topic of the article. It was

noted that for some press releases the cosine TF-IDF method predicted articles that contained rare words such as the city name which the press release also contained. The topics were although completely different. This could be one of the reasons why the method received a high amount of zero points. It would be interesting to see how the distribution of scores is affected by reducing the size of the vocabulary. The second difference is that soft TF-IDF can handle situations when the press release and article do not have many words in common but share words with similar meanings. This may be the reason for its low amount of zero points received. However, in some situations, it is, likely, that articles that contain many similar words to the press release get a higher similarity score than articles with a few but identical words. It would therefore be interesting to see how an increase in the importance of identical words in soft TF-IDF, affects the results. This can easily be done by adding a constant times the identity matrix to the similarity matrix.

As mentioned, SMDSBERT had the worst performance out of the three. One obvious reason for its worse performance could be that the SMD used the Euclidean distance as the cost of traveling from one sentence to another. The SBERT model was however trained to maximize the cosine similarity between semantically similar sentences. Table 4.3 might give some more insights into other reasons. The predicted articles by cosine TF-IDF and soft TF-IDF were on average much longer than the articles SMDSBERT predicted. The SMDSBERT method weight each sentence equally and for longer texts, it could be that many sentences are not relevant to the subject and therefore negatively affect the similarity score. Further research should evaluate how each sentence should be weighted. SMDSBERT did also more often predict the same article for different press releases. Its two most predicted articles were chosen 36 and 22 times. Both of these were very short articles, they consisted of only 220 and 526 characters. The article that was predicted 36 times was not related to any of the corresponding press releases while the article predicted 22 times was only related to two. Their embeddings seem to be close in SBERT space to many other press release embeddings of various topics. The reason for this is not clear and should be investigated further. Studying SBERT's embedding space and its properties may provide clues. For example, does the sentence frequency also bias the embedding space as the word frequency did bias the BERT space?

5.2 Time and memory consumption

From Table 4.2, it can be seen, that SMDSBERT took the most time to train, roughly two times longer than cosine TF-IDF and soft TFIDF. Cosine TF-IDF and soft TF-IDF only differ in the time it takes to construct the similarity matrix. The time difference between SMDSBERT and the other two methods was expected to be larger. However, the main contributor to the overall time was the lemmatization step performed in both cosine TF-IDF and soft TF-IDF. Using another lemmatization method can drastically decrease the total time spent. However, when the method is used to rank journalists the training step only has to be performed once on a large set of articles. After it has been performed once it only has to be incrementally updated with newly scrapped articles each day. More important is the time it takes to predict the five most similar articles to the press release. SMDSBERT was the fastest of them all with an average time of 0.6 minutes per press release. This was unexpected even though the library used for the SMD is highly optimized while the implementations of cosine similarity and soft-cosine similarity are not. Due to SMDSBERT being fast, it would be interesting to look if it is possible to instead use cosine similarity as the cost of traveling from one sentence to another without slowing down the method too much. It can also be seen that soft TF-IDF took significantly more time than cosine TF-IDF. The positive aspect of all these methods is, however, that they are embarrassingly parallel. Therefore running the methods on several processors can reduce the time required.

Cosine TF-IDF used the least amount of memory as expected. Soft TF-IDF used roughly three times the memory used by cosine TF-IDF and SMDSBERT used approximately 10 times more memory than Soft TF-IDF. Nevertheless, these methods' memory consumption grows linearly if the vocabulary is held fixed for Soft TF-IDF. However, as mentioned in Section 3.2.2, calculating the soft-cosine similarity requires a lot of memory as the vocabulary grows large. There have to be further investigations about whether it is possible to decrease the memory consumption by the similarity matrix.

5.3 Future Work

The list below summarizes interesting things to investigate further that have already been mentioned in the thesis.

- How is the results affected by using cosine TF-IDF with the same

vocabulary size as soft TF-IDF?

- Can a better score be achieved by using the cosine similarity instead of the Euclidean distance in the SMD? How much will it affect the time consumption?
- How should each sentence be weighted in SMD?
- Are there ways to reduce the memory consumption of the similarity matrix?
- Why did SMDSBERT in many cases predict the same article for different press releases?
- Can better results be obtained by adding a constant times the identity matrix to the similarity matrix?

As in many machine learning problems, different methods can be good in different aspects and combining several methods can yield better results than only using one. This is something that also should be tested. However, the current way of measuring performance is time consuming. Therefore, it is hard to test new methods. Further research should evaluate if it is possible to measure the performance more efficiently.

It is also desirable that the ranking of journalists is fast. Having to compare the press release against all articles is time consuming. Neural networks, for example, have shown good performance in news classification problems. One possible approach could be to only compare the press release against articles that was predicted by the neural network to belong to the same class. This is something that could be tried in the future to reduce the time required. The only obstacle is getting a news classifier model in Swedish. Using translation tools could hopefully help.

Chapter 6

Conclusions

6.1 Conclusions

According to the proposed success metric, soft TF-IDF had the best performance in finding the five most semantically similar articles to a press release. However, the difference in performance between soft TF-IDF and cosine TF-IDF was not large enough to be able to draw the statistically significant conclusion that soft TF-IDF is the best method. The only statistically significant conclusion that could be drawn was that SMDSBERT was the method with the worst performance.

Soft TF-IDF took, on average, almost five times longer to predict the five most semantically similar articles to a press release than cosine TF-IDF. The Soft TF-IDF's memory usage also scales in the worst case as m^2, where m is the size of the vocabulary. This can be a problem if it is important to use a large vocabulary.

References

[1] What are the sustainable development goals? Accessed: 2022-02-15. [Online]. Available: https://www.undp.org/sustainable-development-goals [Page 2.]

[2] U. Naseem, I. Razzak, S. K. Khan, and M. Prasad, "A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models," *CoRR*, vol. abs/2010.15036, 2020. [Pages 5, 6, and 7.]

[3] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *International Conference on Acoustics, Speech and Signal Processing*, 2012, pp. 5149–5152. [Page 5.]

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Page 6.]

[5] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Page 7.]

[6] O. Shahmirzadi, A. Lugowski, and K. Younge, "Text similarity in vector space models: A comparative study," *CoRR*, vol. abs/1810.00664, 2018. [Pages 8 and 15.]

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, no. null, p. 993–1022, mar 2003. [Page 8.]

[8] H. Jelodar, Y. Wang, C. Yuan, and X. Feng, "Latent dirichlet allocation (LDA) and topic modeling: models, applications, a survey," *CoRR*, vol. abs/1711.04305, 2017. [Page 8.]

[9] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *CoRR*, vol. abs/1908.10084, 2019. [Pages 8 and 9.]

[10] B. Li, H. Zhou, J. He, M. Wang, Y. Yang, and L. Li, "On the sentence embeddings from pre-trained language models," *CoRR*, vol. abs/2011.05864, 2020. [Page 8.]

[11] K. Ethayarajh, "How contextual are contextualized word representations? comparing the geometry of bert, elmo, and GPT-2 embeddings," *CoRR*, vol. abs/1909.00512, 2019. [Page 8.]

[12] D. Charlet and G. Damnati, "SimBow at SemEval-2017 task 3: Soft-cosine semantic similarity between questions for community question answering," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017. doi: 10.18653/v1/S17-2051 pp. 315–319. [Page 10.]

[13] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 957–966. [Page 11.]

[14] E. Clark, A. Celikyilmaz, and N. A. Smith, "Sentence mover's similarity: Automatic evaluation for multi-sentence texts," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019. doi: 10.18653/v1/P19-1264 pp. 2748–2760. [Page 11.]

[15] J. Wang and Y. Dong, "Measurement of text similarity: A survey," *Information*, vol. 11, p. 421, 08 2020. doi: 10.3390/info11090421 [Page 12.]

[16] F. Rekathati, "The kblab blog: Introducing a swedish sentence transformer," 2021. [Online]. Available: https://kb-labb.github.io/posts/2021-08-23-a-swedish-sentence-transformer/ [Page 12.]

[17] N. Reimers and I. Gurevych, "Making monolingual sentence embeddings multilingual using knowledge distillation," 01 2020. doi: 10.18653/v1/2020.emnlp-main.365 pp. 4512–4525. [Page 12.]

[18] L. Wang, C. Gao, J. Wei, W. Ma, R. Liu, and S. Vosoughi, "An empirical survey of unsupervised text representation methods on twitter data," *CoRR*, vol. abs/2012.03468, 2020. [Page 15.]

[19] J. Malmberg, "Evaluating semantic similarity using sentence embeddings," 2021. [Page 15.]

[20] S. Boslaugh, "Statistics in a nutshell," 2012. [Page 18.]

[21] D. R. Hunter, "Notes for a graduate-level course in asymptotics for statisticians," 2014. [Page 18.]

TRITA-EECS-EX-2022:734