Article Recommendation system with NLP

Conrad Tingström
School of Engineering
Jönköping University
Jönköping, Sweden
tico15op@student.ju.se

Kai Qin Elvis Sng
School of Engineering
Jönköping University
Jönköping, Sweden
snka22ty@student.ju.se

Arthur Vinot
School of Engineering
Jönköping University
Jönköping, Sweden
viar22yu@student.ju.se

Johan Åkerblom Svensson
School of Engineering
Jönköping University
Jönköping, Sweden
akjo18id@student.ju.se

***Abstract*—This report presents three different approaches for creating an article recommendation system. This project was done as a collaboration between SävsjöAppen and Jönköping University. Three approaches were tested, LDA, TF-IDF vectorization and cosine-similarity, and sentence embedding. All the developed models showed promising results and should be further investigated. Due to the short timeframe of this project and labeled data being delivered late in the project, limited evaluation was performed. Some evaluation methods were developed and tested, such as comparing the results of different approaching to find whether they were similar.**

Keywords—NLP, Document similarity, LDA, Cosine similarity, Sentence embedding

I. Introduction

The goal of this project was to find and test different approaches to be able to recommend related articles based on one article, meaning that a user reading an article should be recommended additional articles related to the subject of the one being read. Having that in mind, it is noted that the implementation of the approaches should be relatively short in terms of calculating time. The user should not need to wait for a prolonged period before receiving new article recommendations. Knowing also that the calculation time is heavily reliant on the number of articles, but also on the way each approach is implemented. For example, relatively high computing time when initializing to set up the model and pre-process all the data will not affect the user experience because this process is done at an earlier time, with the results stored. The desired outcome is that requesting related articles does not take too long and the computation time when adding a new article stays within acceptable limits. To achieve the task of finding recommending articles, techniques for topic modeling and scoring similarities between articles were researched. The three techniques selected are using Term Frequency–Inverse Document Frequency (TF-IDF), Latent Dirichlet Allocation (LDA), and sentence embeddings. These three methods suppose that the data is cleaned and pre-processed beforehand, so part of this report focuses on the pre-processing of the data the remainder is about the three approaches used and their results.

II. Background

A. Pre-processing

In NLP, preprocessing and data cleaning are important parts of the process. The data consisted of a list of articles with their titles.
The first step was to clean the data. The formats of the articles were different and some of them had HTML tags. So, the first step was to remove them using regex. Then, remove characters not recognized due to encoding issues, such as “;” and “&”. Any numbers were also removed because finding loose number matches between articles might not be significant in finding relevant articles. Worth noting is that while this removes some noise, and thereby reduces the dimensionality of the data, it also removes information that might be potentially relevant in finding related articles, e.g., dates and times.

With the data cleaned, the data is then pre-processed by removing stop words and lemmatizing the text. Stop words are common words in a designed language. Stop-words were obtained from the NLTK library to remove common Swedish words that do not add much meaning to a sentence. Having the stop-words present in the data will cause the results to be inaccurate given the frequent use of these stop-words in every text. If stop words are not removed, articles become too similar only because they are written in the same language and thus have the same recurrent stop words.

Lemmatization is the process of identifying the “root word” for any given word. For example, if we have a verb, we want to remove any conjugation and have the verb in its base form. For nouns and adjectives, we want to remove any plural forms, etc.… So, if you apply that to our articles, we get better results because words that didn’t match because of one being in the plural form or because of the conjugation, now match. To implement lemmatization there are multiple solutions, we use the *spaCy* library, very frequently used for NLP purposes.

B. Latent Dirichlet Allocation (LDA)

The Latent Dirichlet Allocation is an algorithm used for topic modeling which considers a document as a mix of topics and each topic as a mix of words. So, the algorithm scores each word for each topic, meaning the algorithm says for each word how probable it is that it belongs to each topic. We then can represent each topic with the top X words. The problem with this method is that each document is viewed as a collection of words, so the grammatical order of the words

does not matter. We must make sure that the stop words are well removed otherwise it will bias the results. One of the other problems is that we must know how many different topics we want because it's a parameter of the model. [1]

C. TF-IDF and cosine similarity

The next step we have is tokenization and sentence embedding. Because we can't pass words into a model we must use numbers. This step task is translating words into numbers. The basic solution to solve that issue is to one-hot encode every word of each article, so you'll end up with a list of numbers corresponding to the mapping of each word. The downside of this method is that any relation between words is omitted as well as the disposition of each word. To counter this downside, we have another method called word embeddings. This method can be implemented using a pre-trained model like Word2Vec developed by Google. It maps each word to a vector of numbers so that each word that a like between each other like have somehow a quite similar vector. For example, if we map the name of two countries like "Sweden" and "Denmark" then both word vectors should be close to each other. This solution solves the problem of the relation between words. But it still doesn't consider the whole sentence because it processes each sentence individually.

We tested the TF-IDF method, it goes through all the documents and creates a matrix corresponding to each word in each document. It assigns a weight to each word in every document. The weight increases if this word appears often in the document and it decreases if the frequency of the word among all documents is high. We used this method in two ways. The first way was to vectorize each document as-was after pre-processing. The second way was to extract the named entities of each document and then vectorize the resulting sequences. The named entity recognition model we use is in the *spaCy* library with one of their pre-trained Swedish language models, namely sv_core_news_sm, and we pass it our text.

We use sentence embedding to be more accurate in conversion from text to numbers and to get all the relations between words in a sentence. There are multiple methods to embed a sentence. We are only going to talk about the one we tested. Unlike word embeddings, sentence embeddings give a unique vector based on the sentence. For example, with these two sentences: "Eleven players are on the field." & "He is the best player in the team", even though the player has the same meaning in this case they will not have the same representation. This pre-trained model can detect the function of each word in the sentence, to tell whether it's a noun, an adjective, a verb… And it also can tell the words related to it in the sentence. All this information can provide us with a vector of numbers representing the whole sentence. Using that we can see which document looks like another one by using cosine similarity because we have an embedded vector for each article. The cosine similarity ranges from 0 to 1. The higher the cosine similarity is the higher the similarity between each article is [2].

Due to the relative simplicity of the TF-IDF approach, we were able to perform some evaluation using the labeled data received towards the end of the project. The labeled data was a list of 11 manually constructed topics, each consisting of several articles. The articles had been manually reviewed and labeled with the most relevant topic. In total, the labeled data was comprised of around 70 articles. For evaluation, we first randomly selected a subset of the full dataset (excluding the articles present in the labeled data), containing 10% of the total amount of articles, then added the articles in the labeled data to this subset, constructing the test set. We then computed the similarity of each article in every topic in the labeled data, with every article in the test set, thus creating a similarity ranking for the entire test set, for each article present in the labeled data. Then, for each article in the labeled data we extracted the similarity ranks for all articles within the same topic.

D. Sentence Embedding

This approach explores the utility of pre-trained models. Given relatively little data, using pre-trained models might provide better and more efficient results as large amounts of data were trained in the model. The pre-trained model used for this approach is the bert-base-swedish-cased pre-trained model because Swedish articles are provided. Using the cleaned and pre-processed data, word embeddings are obtained from the pre-trained model followed by sentence embeddings. The sentence embeddings are calculated by averaging the word embeddings in each article. It should be noted that lemmatization was not performed for this approach as it causes the removal of context in each article. As the pre-trained models already contain different word forms, lemmatization can be omitted. Cosine similarity scores are also used to score the similarities between two sentence embeddings representing the original articles. To evaluate the accuracy of the model, given links data is used. The desired outcome is that tagged articles should have a high cosine similarity score between each other, as the tagging was done to group similar articles together. If the tagged articles should have a high cosine similarity score in the current model, it would show that the model is working as intended.

The approach to evaluating the accuracy of the model is as follows:

1. Sort the tags by frequency in the given links data
2. For each tag:
    a. Choose the first article found having the same tag
    b. Average the cosine similarity scores for articles with the same tag using the first article as the basis of comparison
        i. Articles with the same tag should have a high cosine similarity score. This means that the percentile of the cosine similarity scores should be high as well
    c. Calculate the percentile of averaged cosine similarity score
3. Average the percentiles obtained for each tag

The resultant percentile obtained is 93.33. This means cosine similarity scores between tagged articles are in the 93rd percentile on average, implying the cosine similarity scores between tagged articles are relatively much higher than the remaining articles that do not have the same tag. As such, it can be concluded that cosine similarity scores from the sentence embeddings can be used to rank and recommend similar articles to a large extent. [3]

III. Evaluation

We faced a big problem in evaluating our different approaches, and this issue applies to any approaches we could have used. Is the lack of labeled data. At first, the only tags we

had were very heterogenous, that is to say, many of them were irrelevant. Many articles had tags based on the author and we do not want our models to think articles are similar because it's the same journalist who wrote them. Because of this unlabeled data we had to come up with a method to evaluate our models. We decided to use the cosine similarity we talked about before. So, we take a sample of articles from different topics, and we evaluate their similarity with the other articles from the same topic. To do so we take the average cosine similarity for every article on the topic. The downside of this method is that we do not compare every article on the topic with one another. We only take a sample of the articles to evaluate the results so when we take a sample, we might have luck and take only articles that have good cosine similarity within their topic. The results for evaluating the sentence embedding using the approach described in II.D is shown in Figure . These results suggest that the approach is promising, as for every topic if we pick a random article, then its cosine similarity with 75% of the group is not as low as 0,5.

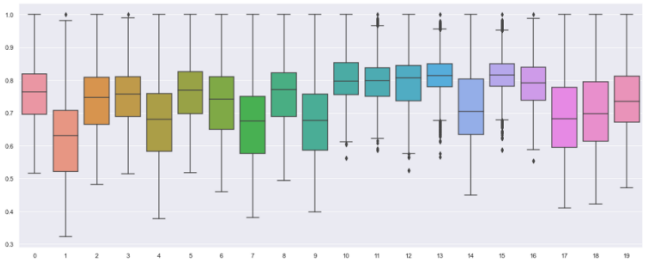

Figure 1. Evaluation results for the sentence embeddings.

A. TF-IDF

Following the evaluation approach described in II.C, the TF-IDF approach showed promising results as well. In Figure , the results of evaluating the full-text approach are shown. The boxes indicate how good the approach was at finding the articles within one of the topics, showing that regardless of which article in this topic was used, the approach roughly managed to rank all the remaining 10 articles of the topic in the top 20 similarity scores.

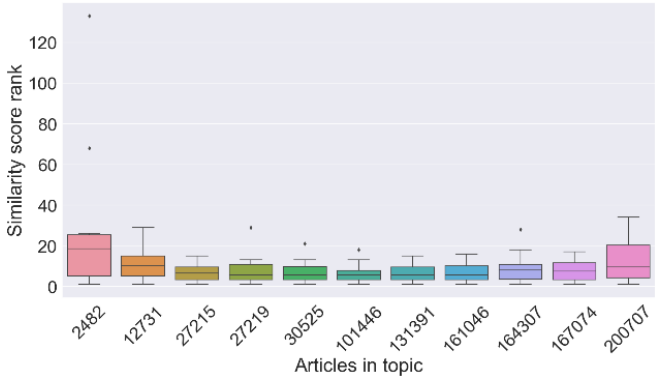

Figure 2. Evaluation results for the TF-IDF approach, using full-text articles.

Using the same approach, Figure 3 shows the results of evaluating the named entity approach. While this approach evidently did not perform on par for this topic, this was not the case for all topics, which leads us to believe that the issues might not lie in the approach itself, but rather in how the named entities were extracted.

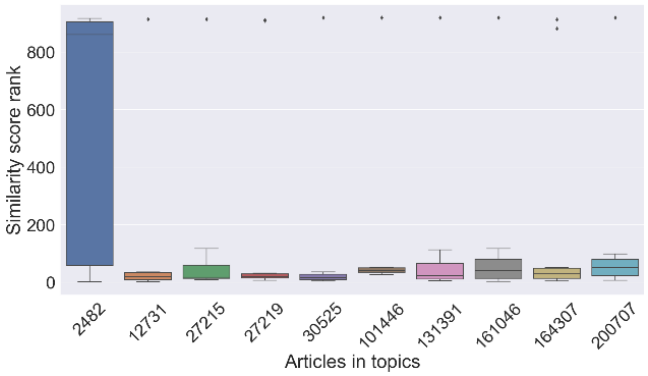

Figure 1. Evaluation results for the TF-IDF approach, using named entities.

B. LDA

LDA was used to create 16 different topics, each topics contains words that weigh differently on which topic a text should get. For example, topic 8 contains words such as, "polis", "skada", "räddningstjänst", articles containing these words have a higher possibility of being labeled as this topic. When labeling a text with a certain topic, a score is calculated from the words contained in the text. A text has a score for each topic and the one with the highest score gets set as the topic.

The texts/articles left from prepossessing had their LDA score calculated and got a topic assigned to them. These topics ranged from containing 30 to 821 articles. A method testing different numbers of topics was run on 16, 32, and 48 topics and out of these 16 were selected.

Since we did not have the evaluation data until very late in the project the LDA method was cross-evaluated with the cosine-similarity model. Cosine-similarity generated 10 articles with the highest similarity score from "Inte alltid sporten förbrödar" which is an article about a Christmas with some law enforcement involvement. All the articles recommended are labeled with topic 8 and contain some sort of law enforcement involvement. Figure 4 shows the recommended articles from cosine-similarity with the "idx" being the topic assigned to the article. Topic 8 is only 140 articles out of 6376. This result gives us some confidence that both LDA and cosine-similarity are viable approaches for creating a recommendation system for SävsjöAppen.

Text	Title	idx
juldag princip fridfull län bortsett hand pers...	Inte alltid sporten förbrödrar	NaN
juldag återvändarhelg festdag håll höglande fu...	Lugn juldag på Höglandet	8.0
misshandel olaga hot inträffa sjutid lördag kv...	Anhållen och frigiven efter misshandel	8.0
strax onsdagsmorgon polis samtal kvinna kvinna...	20-åring anhållen □ hot och misshandel av kvin...	8.0
man respektive årsålder greps fredag kväll mis...	Två greps efter misshandel	8.0
person inblanda polis rubricera försök grov mi...	Oklara omständigheter kring anmälan	8.0
kvinna årsålder hitta busshållpla halv onsdags...	Kvinna misshandlad och utkastad från en bil	8.0
seneftermiddag onsdag uppstå bråk gäng vetland...	Uppdaterat: Man efterlyst för dråpförsök	8.0
strax måndagskväll polis larm misstänkt grov m...	25-årig man skjuten i benet	8.0
halv tretid onsdagseftermiddag polis samtal po...	UPPDATERAD: Kvinna och flicka försvunna...	8.0
polis kalla adress ort kommun nyårsafton förmi...	En man misstänkt för ringa misshandel	8.0

Figure 3:Cosine-similarity & LDA cross-evaluation

IV. Future work

The next step that could really help us make a big step in the different approaches' evaluation, is succeeding in implementing a systemic test approach using the labeled data. That could help us have a quantitative approach regarding the

evaluation of our models. Another way to go forward would be to find a dataset of news articles written in Swedish already labeled by categories for example (“politics”, “sports” etc.…) and to train a model using those data and then evaluate the created model with the related data gave by Linus to see if this approach performs better than the ones we tested and implemented.

A. TF-IDF

At least two aspects of this approach need immediate attention moving forward; (1) the method used to extract the named entities from the articles and (2) the evaluation.

1) Named entity recognition

We expected to see better results using this approach than what is shown in Figure 3, and the results were not as bad in all the topics in the labeled data. In fact, there were topics were this approach performed on par with the full-text approach. Thus, moving forward we suggest looking into the method used for extracting the named entities. A possible entry-point would be to investigate using other language models than the one used in this project.

2) Evaluation

The evaluation approach itself was deemed to show promising results. Moving forward, however, it would be interesting to not only consider how well the related articles ranked, but also investigate what other articles scored high in similarity. Currently, the only means for doing this would be through manual evaluation (i.e., reading the articles to get a feel for how close they are to the topic). Thus, directly related to the evaluation would be to as soon as possible label a larger portion of the data to test against.

B. LDA

From the LDA perspective, when assigning a topic to a text, only the highest LDA score was selected as a topic. Something that could have been done instead is that the top two or three scores could be selected, or a limit could be calculated, and a text could get multiple topics as a result.

Another thing that could have been done differently is the number of topics selected. Our method for selected the number of topics is a reasonable approach. However, no attempt of discussing the number of topics with the company was made, doing this could have generated another result. But doing this should also be combined with the assigning multiple topics to articles.

V. References

[1] N. Seth, "Topic Modeling and Latent Dirichlet Allocation (LDA) using Gensim and Sklearn," *Data Science Blogathon*, 26 9 2021.

[2] W. Scott, "TF-IDF from scratch in python on a real-world dataset.," *Towards Data Science*, 15 02 2019.

[3] A. Pogaitzis, "NLP: Contextualized word embeddings from BERT," *Towards Data Science*, 20 03 2019.