

Project Report:

Extract main topics of the document^{*}

Universität Düsseldorf

Dinh Viet Dung
vidin101@hhu.de

October 4, 2023

Abstract

When it comes to analyzing huge amounts of text data or trying to find relevant documents for research, it is too big of a task to do it manually. It is time-consuming and also expensive. So to help quickly grab the right documents or understand the content which further aids in decision-making and analysis. It is needed for automatically sorting documents instead of doing it manually. There are many models developed for this task e.g. LSA, Bert, LDA[1] but for this research, I focus on the basic approach to find the main topic by filtering the most common words and grouping paragraphs with the similarity, and those paragraphs share the same topic.

1 Introduction

The goal of this project is to get a better understanding of the basic technique used in topic extraction to help better understand other complex machine learning methods like e.g. LSA, LDA. Through this research, I want to know which processes are needed and how to implement them in Python along with which tools are important.

The research focuses on answering the question if we have a document that contains many small different texts, can we categorize those paragraphs by topic and how well it can be? For this reason, the document used is a document that contains many emails. After going through the processes, all the emails with similar content should belong to one topic group.

^{*}<https://github.com/shockorjoke/getTopic>

2 Data and Resources

This approach does not use machine learning techniques but pure step-by-step techniques, a big document is not needed. However, the document should be big enough to test the similarities of paragraphs since it can be the case that no emails share the same information.

The document is extracted from sklearn¹ which contain 18846 sample emails. For this project, only 100 emails are used for evaluation, each email is one paragraph. Python packages needed are *numpy*, *scikit-learn* and *spacy*, and the language package 'en_core_web_sm' is also needed for tokenized and lemmatized text.

3 Method

In this section, the complete process will be presented along with some graphic outputs to describe what the functions are supposed to do.

3.1 Prepare input for topic extraction and paragraph grouping

3.1.1 Tokenize and lemmatize text

All the stopwords have to be removed since they do not represent any topic, it goes the same for non-alphabetic words or special symbols e.g. '@,.,!'. This can be done quickly with *Doc*² from spacy. After this process, we got a set of all unique alphabetic non-stopwords that exist in the document in lemmatized form and a new modified document that contains only lemmatized paragraphs without stopwords and non-alphabetic words. For example Table 1:

Input Para.	Output Para. Para.	Output token set
I am a little confused on all of the models of the 88-89 bonnevilles. I have heard of the LE SE LSE SSE SSEI. Could someone tell me the differences are far as features or performance. I am also curious to know what the book value is for prefereably the 89 model. And how much less than book value can you usually get them for. In other words how much are they in demand this time of year. I have heard that the mid-spring early summer is the best time to buy.	little confused model - bonneville . hear le se lse sse ssei . tell difference far feature performance . curious know book value prefereably model . book value usually . word demand time year . hear mid - spring early summer good time buy .	{'make', 'brandish', 'marash', 'ma', 'prompt', 'rich', 'status', 'later', 'celp', 'second', 'gasoline', 'react', 'formation', 'hewlett', 'forgett'

Table 1: Output paragraph and token set

¹sklearn dataset

²Doc document

3.1.2 Count the frequencies of the remaining lemmatized words for each paragraph

In this step, I use Counter³ from the collection module to count the frequency of all remaining lemmatized words in each paragraph. Each paragraph is a dictionary where the keys present the words and the values represent their frequency in the current paragraph. In case the word does not appear in the paragraph, its frequency is 0. After this step, we get a list of paragraph dictionaries, the elements look as Table 2:

lemmatized Para.

```
little confused model - bonneville .  
hear le se lse sse ssei . tell  
difference far feature performance . curious  
know book value prefereably model .  
book value usually . word  
demand time year . hear mid - spring  
early summer good time buy .
```

Output frequency Dict.

```
'chest': 0,  
'dos': 0,  
'cambridge': 0,  
'lb': 0,  
'hear': 2,  
'assessment': 0,  
'bds': 0,  
'value': 2,  
'ohio': 0,  
'reinstall': 0,  
'hockey': 0,  
.....
```

Table 2: Words frequency in a paragraph

3.2 Extract main topic & grouping paragraph

3.2.1 Extract main topic

The main topic of paragraphs is extracted with a simple method, namely the most common words in the paragraph. This means the topic can be interpreted through a list of the most frequent words in the paragraph. In this project, I use the top 8 words since the lengths of emails in this document are relatively short. The function outputs a dictionary where the keys represent paragraph numbers and the values represent the top word along with its frequency, like in Figure 1:

```
{0: {'hear': 2,  
    'value': 2,  
    'time': 2,  
    'bonneville': 2,  
    'model': 2,  
    'book': 2,  
    'software': 1,  
    'need': 1},  
 ... }
```

Figure 1: Caption

This step is not only needed to extract the main topic of the emails but also take a role later to check if similar paragraphs also share the same topic. We will come back with this later in the last section.

³Counter

3.2.2 Grouping paragraph

The last step is grouping paragraphs to reduce the number of topics. Until now, each paragraph has its own topic. But it could be the case that some paragraphs present the same topic. So by grouping those similar paragraphs and assigning them to one topic, we can reduce the number of topics and categorize them.

The main goal of this step is to find similar paragraphs with cosSim^4 similarity and group them. A simple explanation of this algorithm can be found in the footnote. In general, this method scores the similarity between documents based on their represented vector. In the following, I will present a small example:

The table represents the frequency of words in the phrases, in this case, both *hello* and *world* occurs 1 time in phrase *hello world* while only *hello* occurs in phrase *hello*. We have the vector for *hello word* = (1,1), and vector for *hello* = (1,0). The diagram shows the vector representing both phrases.

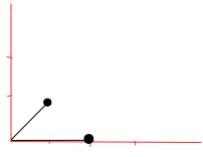
word frequency			vector represent
	hello	world	
hello world	1	1	
hello	1	0	

Table 3: Small cosSim example

Apply the cosSim formula we have the cosSim similarities of the two phrases:

$$\begin{aligned}\cos(\theta) &= \frac{A \cdot B}{\|A\|_2 \|B\|_2} \\ &= \frac{(1 \cdot 1) + (1 \cdot 0)}{\sqrt{1+1}\sqrt{1+0}} = 0.71\end{aligned}$$

Now inputs for cosSim have to be vectors. A simple way to do this is to create a dictionary that contains paragraph numbers and its word frequency list. After that, paragraphs are compared pairwise, and the paragraphs with higher cosSim scores are put together.

In order to narrow the number of topics, I put an additional transitive step to find more paragraphs with potential similarities. I call it a transitive step since it works exactly like that. When paragraph 1 is similar to paragraph 2 and paragraph 2 is similar to paragraph 3, then I will put those 3 paragraphs together. This step is to cover cases like e.g.:

Para 1: [computer, gamming, esport, player, steam]

Para 2: [computer, gamming, esport, gammer, epic]

Para 3: [esport, gammer, epic]

⁴CosSim simple explain

All 3 paragraphs share the same topic about gaming but if we calculate cosSim similarity for (1,3), the result will be 0. But from (1,2) and (2,3) we can set (1,2,3) in the same category.

4 Results and Discussion

The following is one example result of the similar paragraphs group with cosSim threshold = [0.4, 0.25, 0, 1], Table 4:

similar	common words
23 _{0.4}	{'array': 3, 'hst': 2, 'communications': 1, 'servicing': 1, 'online': 1, 'use': 1, 'lines': 1, 'host': 1}
46 _{0.4}	{'design': 4, 'array': 4, 'reboost': 3, 'hst': 3, 'servicing': 2, 'rendering': 2, 'spacecraft': 2, 'orbit': 2}
89 _{0.25}	{'guy': 2, 'disk': 2, 'scsi': 2, 'ide': 2, 'use': 1, 'lines': 1, 'time': 1, 'point': 1}
1 _{0.2}	{'x': 5, 'face': 5, 'header': 3, 'get': 2, 'swamp': 2, 'handle': 1, 'hope': 1, 'world': 1}

Table 4: Similar paragraph with threshold 0.4, 0.25, and 0.2

As we can see, when the threshold for cosSim score is set lower, more paragraphs can be put in the same categories. But the common word lists also have fewer intersections. With the cosSim score set = 0.4, both paragraphs (23, 46) have more intersection between their common word lists. And they both seem to talk about *space service topic* which is also true because both talk about *HST Servicing Mission Scheduled* in the raw document. With this threshold, we reduce the number of topic to 95/100.

If we reduce the cosSim score to 0.25 then we get triple (23, 46, 89) in the same category and reduce the number of topics to 64. Look at the common word, the topic still looks like it belongs to *technology*. And the raw subject of the email is about *Hosed HD* which also still belongs to *technology* but not about *space* anymore. Now it is based on how well/broadly we define a topic we can accept this similarity or not. If we set threshold = 0.2 then paragraph 1 also comes in the group but the common words really do not seem to fit with other paragraphs.

5 Challenges and open issues

This project is very basic and still lacks some attributes needed for a complete topic extraction. There are two steps that could be added to future projects which can help to extract the topic better. One could add a synonym tool to increase the similarities since this basic method only covers exact words. Another could be to apply machine learning techniques to actually predict the topics from a given document. Of course, there are already tools like LDA, LSA, Bert, Top2Vec but It would be better if I could implement the

algorithm myself instead of using the already available model. This will help me better understand the concept and develop my own model. Since my approach only gets the topic from common words, not actually the topic name, I do not make a comparison with LDA or Bert. But in the future work when those two steps are added. It would be more reasonable to make a comparison.

6 Summery and conclusion

This project has shown that the basic approach toward topic extraction is on the right track. Even though the method is still not able to predict the topic, it has partially successfully categorized the dataset in the document with the same topic based on cosSim and most common words. As mentioned at the beginning, this is a basic approach toward topic extraction and still lacks the ability to become a topic extraction model. Future work including both additional steps is needed for the model to be completed.

References

- [1] MonkeyLearn Inc. In: *Topic Analysis: The Ultimate Guide* (2023). URL: <https://monkeylearn.com/topic-analysis/>.