POLITÉCNICA

UNIVERSIDAD
POLITÉCNICA
DE MADRID

Information retrieval, extraction and integration

-

Profile Based Retrieval Report

Bela Bönte, Laurin Kubelka, Dániel Marouf

Introduction **2**

Dataset **2**

About the articles 2

Importing the data 3

User profiles **3**

Profiles 3

Preprocessing **4**

Methods **5**

Method 1 5

TF-IDF Indexing 5

Term Frequency 5

Inverse Document Frequency 6

Query Vector Matrix 7

Comparison of query vector matrix and tf-idf index 7

Method 2 7

How it works 7

Spacy applied to the BBC Dataset 8

Preferences by user **8**

Evaluation **9**

First query : “Best tech gadgets for at home.” 9

Second query : “Best movies to watch while my husband reading about football” 10

Comparison with another group **11**

Introduction

Nowadays there is a flood of articles, which makes it more and more difficult for users to find the right information. Therefore, efficient search engines are necessary. This work shows an approach to solve this problem. The goal of the developed application is to input a text query and the output of the program is the ten most related articles in consideration of the user's interests.These interests are predefined and relate to all the possible topics in the given dataset. Depending on the query and the user profile, the ranking of the most related articles is going to change.

Dataset

The dataset is provided by University College Dublin[1], Machine Learning and Network Analysis Resources. The dataset contains BBC news from 2004 and 2005. The raw dataset was chosen. The collection contains 2225 articles from the BBC news website, which relate to articles in five different thematic categories. Which are the following:

- Business - 510 articles
- Entertainment - 386 articles
- Politics - 417 articles
- Sport - 511 articles
- Tech - 401 articles

About the articles

Each topic has a roughly similar amount of news. This is essential for this task, since all topics will be crawled by the model in a similar way and thus when running a query, all topics are on equal footing and only the user profile and the query will determine the outcome of the proposal. News is stored in different folders according to categories. Each article is stored in a separate txt file. The first line of the text is the title of the news, the next paragraph is the headline and the last one is the content. The structure of the file storage is important when importing data because the topic names are not included in the text file so the only way to import this information into the program is from the folder name.

[1] http://mlg.ucd.ie/datasets/bbc.html

Importing the data

The procedure for importing data into the program is as follows:

1. Navigate through the folders and save the folder name.
2. Enter a folder and scan the files inside.
3. The saved folder names are associated with the contents of the file.
4. All elements are appended to a list.

User profiles

The user profiles were created based on the topics that are given in the dataset since the structure of our program is based on the fact that users read content that present on the news platform. People's preferences are based on the most read topics of the articles. Based on these, six kinds of profiles have been created.

Profiles

- Peter profile: Politics and sport
- Lois profile: Entertainment, Tech
- Brian profile: Business, Politics
- Stewie profile: Tech
- Meg profile: Entertainment
- Chris profile: Sport

The goal is to get different results for the same query based on the preferences of the user. To illustrate with an example. In a query, all users type in "What's life like in America?". For the query, the program will examine all the articles and rank them based on the most similar news topics, then weight these categories based on user preferences and suggest different news to users based on their profile.

Preprocessing

As the dataset contains written articles, it is necessary to preprocess them, to extract useful and exclude meaningless information. Therefore, the next step after importing the dataset is to conduct different preprocessing tasks using the Regex Expressions from the Python “re” module.

```
def preprocess_articles(df):
  documents = []
  stemmer = WordNetLemmatizer()
  for sen in range(0, len(df.text)):
    # Remove all the special characters
    document = re.sub(r'\W', ' ', str(df.text[sen]))

    # remove all single characters
    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

    # Remove single characters from the start
    document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

    # Substituting multiple spaces with single space
    document = re.sub(r'\s+', ' ', document, flags=re.I)

    # Converting to Lowercase
    document = document.lower()

    # Lemmatization
    document = document.split()

    document = [stemmer.lemmatize(word) for word in document]
    document = ' '.join(document)

    documents.append(document)
  result = df.join(pd.DataFrame(documents, columns=["preprocessed_text"]))
  return result

df = preprocess_articles(df)
```

First, all non-word characters like special characters and numbers will be removed, as they do not give us any indications about the related topics. Additionally we are going to exclude any single characters. When the punctuation mark in "Ukraine's" is removed and replaced with a space, we obtain "Ukraine" and a single character "s" which does not present any useful information. To eliminate single characters with spaces on either side, a regular expression is used, which replaces all single characters with a single space. As only single characters that have spaces on either side have been removed, the ones at the start of a document still remain, and are therefore also excluded. Following that, multiple spaces that might occur are substituted with multiple spaces, and the data will be converted to lowercase only. The reason for that is to be able to treat the same words with different cases equally.

One of the most important steps for cleaning the words is Lemmatization. It is used to reduce morphological variation. Lemmatization takes into account the context when converting a word to its meaningful basic form, known as a Lemma. So in other words it reduces the word-forms to linguistically valid lemmas (e.g., "builds", "building", or "built" become "build").

Methods

Two different methods have been chosen to assess their effectiveness and compare their performance.

Method 1

In the first approach a TF-IDF matrix and the query vectors are calculated without any existing model.Therefore, the initial step is to calculate the term frequency, that measures the number of appearances for each word in an article. It is also necessary to apply a normalization technique to the data, to put the importance of a word into perspective regardless of the length of a document and the absolute appearances of the word. Afterwards, the inverse document frequency is calculated, which determines how informative a term is.

TF-IDF Indexing

The indexing of data is crucial since the structure of news is constantly changing from scanning to processing to query execution. Indexing starts by saving all the news in a list and assigning an index to each item in the list. The processes are always iterated through according to the index. The matrices generated during the vectorization are assigned to the indexed news and compared with the matrix generated by the query and return to the user the indexed items that return the most similar news based on the user preferences and the query.

The TF-IDF score measures the importance of a word in a whole word corpus. The value increases if the word occurs more in one document and less in other documents. The score is composed of the Term Frequency (TF) and the Inverse Document Frequency. The two values are multiplied by each other.

Term Frequency

Mathematically it is calculated by dividing the frequency of the word in a document by the total number of words in the document.

$$TF_{i,j} = \frac{n_{i,j}}{\Sigma_k n_{i,j}}$$ [2]

The searched term is counted in the document and then divided by the total number of words. The calculated value is then returned.

[2] https://pahulpreet86.github.io/term-frequency-inverse-document-frequency/

Inverse Document Frequency

$$IDF\ (wt) = log(\frac{N}{df_i})$$ [3]

The Inverse Document Frequency is defined as the logarithm of the total number of documents divided by the documents containing the search term. In order to make sure that terms with no IDF score aren't suppressed, 1 is added to the idf score.

Technical Implementation[4]:

```
def termFrequency(term, document):
  return document.count(term.lower()) / float(len(document))

def inverseDocumentFrequency(term, documents):
  count = 0
  for doc in documents:
      if term.lower() in doc:
          count += 1
  if count > 0:
      return  1 + math.log(float(len(documents))/count)
  else:
      return 1.0
```

After TF and IDF scores are calculated, they are multiplied to form the TF-IDF score. Based on the words in the query, each word and document get assigned a TF-IDF score. So the resulting matrix would be (Words in the query) x (Documents) of TF-IDF vectors.

Technical Implementation[5]:

```
def generateVectors(query, documents):
    tf_idf_matrix = np.zeros((len(query.split()), len(documents)))
    for i, s in enumerate(query.lower().split()):
        idf = inverseDocumentFrequency(s, documents)
        for j,doc in enumerate(documents):
            tf_idf_matrix[i][j] = idf * termFrequency(s, doc)
    return tf_idf_matrix
```

[3] https://pahulpreet86.github.io/term-frequency-inverse-document-frequency/
[4] https://ted-mei.medium.com/demystify-tf-idf-in-indexing-and-ranking-5c3ae88c3fa0
[5] https://ted-mei.medium.com/demystify-tf-idf-in-indexing-and-ranking-5c3ae88c3fa0

Query Vector Matrix

To compare the TF-IDF matrix with the query, the query must also be converted into a vector. This vector is also formed with TF-IDF scores. The term frequency of the terms in the query is calculated depending on the occurrence in the query itself instead of the occurrence in the documents. The IDF score is calculated in the same way as explained above. The vector thus specifies which words are searched for and places particular emphasis on words that are repeated in the query. This creates a matrix of vectors, of the same size as the TF-IDF matrix.

Comparison of query vector matrix and tf-idf index

To calculate the difference between two vectors, their cosine can be calculated. To calculate the difference between two vectors, their cosine can be calculated. Thus, for each document it is possible to match which document has the most words from the query, or the words that are not present in other documents. These vectors are quite similar to each other. So now the formed query vectors are matched with the TF-IDF index and the most suitable document is found.

Method 2

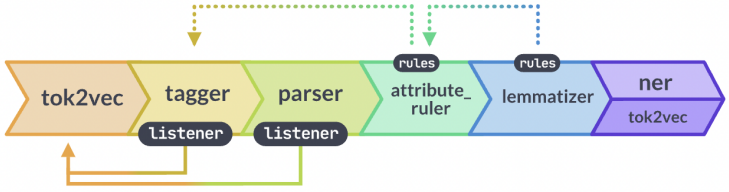
Method two is a black box approach. The underlying mathematical concept has already been worked out and the model can be easily inserted into the code via an import.

For this purpose the Python library Spacy was used.. SpaCy is a free, open-source library for advanced Natural Language Processing (NLP) in Python. [6] There are 64 trained pipelines for 19 languages. Of these, the en_core_web_sm pipeline was used. En_core_web_sm is a small English pipeline trained on written web text (blogs, news, comments), that includes vocabulary, syntax and entities.[7] This model was built to have a baseline for the evaluation.

How it works

When the tok2vec component is active, the tagger, morphologizer, and parser components all listen to it. If there is no morphologizer, the attribute ruler converts token.tag to token.pos and vice versa.

CNN/CPU pipeline design



[6] https://spacy.io/usage/facts-figures
[7] https://spacy.io/models

The attribute ruler ensures that whitespace is consistently tagged and transfers token.pos to token.tag if there is no tagger present in the code. When dependent parses from a parser are available, the attribute ruler may enhance its mapping from token.tag to token.pos, although the presence of a parser is not necessary in the case of English. Because it has its own internal tok2vec layer, the ner component is completely self-contained.

Spacy applied to the BBC Dataset

First the program goes through all the news and runs the built-in nlp function of spacy on the selected pipeline. It also runs the function separately for the query . It then uses the similarity function to match the query to all the news and returns a value between 0 and 1 where the closer the value is to 1 the more similar the query is to the given article. The function of similarity is the formula for determining cosine similarity, including the vectors generated by spacy.vector, which, according to the documentation, was trained using GloVe's w2v model .

Preferences by user

The results are weighted according to the profiles. This means that for each profile the hits are looked through and if the topic matches the user's preferred category, the original value is kept. However, if the preference is not the same then the result is squared and since the original value was between 0 and 1 the squaring will definitely make the article with a non-preferred category worse and these topics will be moved to the lower rank. The lower the similarity the more it will penalize but as the first 30 results are only scanned the order will only change here to ensure that the original search conditions are satisfied.

```
def results(user,results,df):
    resultsone=[]
    for i in range(len(results)):
      punish=True
      for pref in user:
        if pref==df.iloc[results[i][0]].topic:
          punish=False
      if punish==True:
        result_method_one=results[i][1]*results[i][1]
      else:
        result_method_one=results[i][1]
```

Evaluation

To demonstrate how the program works, two queries were tested. The first one is related to the query technology, in order to examine the difference between the two methods. In the second case, a more complex query was run with several topics in it to show differences in the order of articles by user preference through the better method from the first query.

First query : “Best tech gadgets for at home.”

```
Without user preferences First method
Username : Default

1. topic: tech, title : Millions buy MP3 players in US.
2. topic: tech, title : Millions buy MP3 players in US.
3. topic: tech, title : Digital UK driven by net and TV.
4. topic: tech, title : Sony PSP tipped as a 'must-have'.
5. topic: tech, title : Sony PSP tipped as a 'must-have'.
6. topic: tech, title : Gadgets galore on show at fair.
7. topic: tech, title : Gadget show heralds MP3 Christmas.
8. topic: tech, title : Gadget show heralds MP3 season.
9. topic: tech, title : Apple laptop is 'greatest gadget'.
10. topic: tech, title : Apple laptop is 'greatest gadget'.

Without user preferences Second method
Username : Default

1. topic: entertainment, title : DVD review: Spider-Man 2.
2. topic: politics, title : The memory driving Brown's mission.
3. topic: entertainment, title : Brits debate over 'urban' music.
4. topic: entertainment, title : DVD review: I, Robot.
5. topic: entertainment, title : Dance music not dead says Fatboy.
6. topic: tech, title : Honour for UK games maker.
7. topic: entertainment, title : German music in a 'zombie' state.
8. topic: tech, title : Poles play with GameBoy 'blip-pop'.
9. topic: politics, title : McConnell in 'drunk' remark row.
10. topic: entertainment, title : Uganda bans Vagina Monologues.
```

Listing the first ten results, it can be seen that the first method was almost all from the tech topic and the articles are more related to the query, while the second method has results from almost all categories and not really related to the original search subject. For this reason, we can say that our method gives better results than Spacy's built-in pipeline.

Second query : “Best movies to watch while my husband reading about football”

User	Peter	Lois	Brian	Stewie	Meg	Chris
Preferences	politics, sport	entertainment, tech	business, politics	tech	entertainment	sport

Ranking	Default	Peter	Lois	Brian	Stewie	Meg	Chris
1	Hamm bows out for US	Hamm bows out for US	J-Lo and husband plan debut duet	Mrs Howard gets key election role	Gadget show heralds MP3 Christmas	J-Lo and husband plan debut duet	Hamm bows out for US
2	J-Lo and husband plan debut duet	Mrs Howard gets key election role	Connick Jr to lead Broadway show	Business confidence dips in Japan	Gadget show heralds MP3 season	Connick Jr to lead Broadway show	Thompson says Gerrard should stay
3	Connick Jr to lead Broadway show	Thompson says Gerrard should stay	Oscar host Rock to keep it clean	Hamm bows out for US	More movies head to Sony's PSP	Oscar host Rock to keep it clean	Can Smith work Scottish wonders?
4	Oscar host Rock to keep it clean	Can Smith work Scottish wonders?	Show over for MTV's The Osbournes	J-Lo and husband plan debut duet	Hamm bows out for US	Show over for MTV's The Osbournes	McClaren targets Champions League
5	Show over for MTV's The Osbournes	McClaren targets Champions League	Dame Julie pops in to see Poppins	Connick Jr to lead Broadway show	J-Lo and husband plan debut duet	Dame Julie pops in to see Poppins	Souness backs Smith for Scotland
6	Dame Julie pops in to see Poppins	Souness backs Smith for Scotland	Smith loses US box office crown	Oscar host Rock to keep it clean	Connick Jr to lead Broadway show	Smith loses US box office crown	McClaren hails Boro's Uefa spirit
7	Smith loses US box office crown	McClaren hails Boro's Uefa spirit	Oscar nominees lack pulling power	Show over for MTV's The Osbournes	Oscar host Rock to keep it clean	Oscar nominees lack pulling power	J-Lo and husband plan debut duet
8	Oscar nominees lack pulling power	J-Lo and husband plan debut duet	Pixies take on Reading and Leeds	Dame Julie pops in to see Poppins	Show over for MTV's The Osbournes	Pixies take on Reading and Leeds	Connick Jr to lead Broadway show
9	Pixies take on Reading and Leeds	Connick Jr to lead Broadway show	Dance music not dead says Fatboy	Smith loses US box office crown	Dame Julie pops in to see Poppins	Dance music not dead says Fatboy	Oscar host Rock to keep it clean
10	Dance music not dead says Fatboy	Oscar host Rock to keep it clean	Gadget show heralds MP3 Christmas	Oscar nominees lack pulling power	Smith loses US box office crown	'Comeback' show for Friends star	Show over for MTV's The Osbournes

Ranking	Default	Peter	Lois	Brian	Stewie	Meg	Chris
1	sport	sport	entertainment	politics	tech	entertainment	sport
2	entertainment	politics	entertainment	business	tech	entertainment	sport
3	entertainment	sport	entertainment	sport	tech	entertainment	sport
4	entertainment	sport	entertainment	entertainment	sport	entertainment	sport
5	entertainment	sport	entertainment	entertainment	entertainment	entertainment	sport
6	entertainment	sport	entertainment	entertainment	entertainment	entertainment	sport
7	entertainment	sport	entertainment	entertainment	entertainment	entertainment	entertainment
8	entertainment	entertainment	entertainment	entertainment	entertainment	entertainment	entertainment
9	entertainment	entertainment	entertainment	entertainment	entertainment	entertainment	entertainment
10	entertainment	entertainment	tech	entertainment	entertainment	entertainment	entertainment

As the results show, each user produced a different result. Users who had more than one preference received more types of articles, while users who had only one preference received most articles from there. Here it can also be seen that the articles are related to the query. Overall, it can be said that users got the majority of their results from their own topic.

Comparison with another group

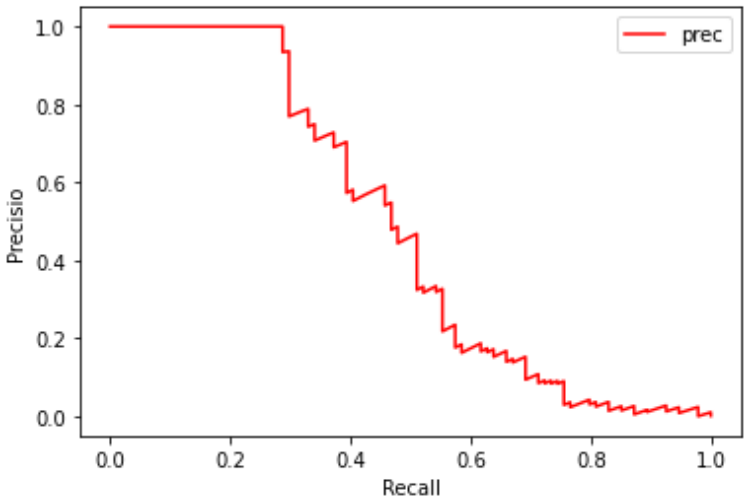
To be able to compare our results with the group of Azeez Abdikarim and Eleonora Renz. We are running the same set of test queries:

- "Best tech gadgets for at home"
- "Literature recommended based on book prize winners"
- "Financial advice for the stock market"
- "Council decision on new tax deals"
- "Club trading offers for football players"

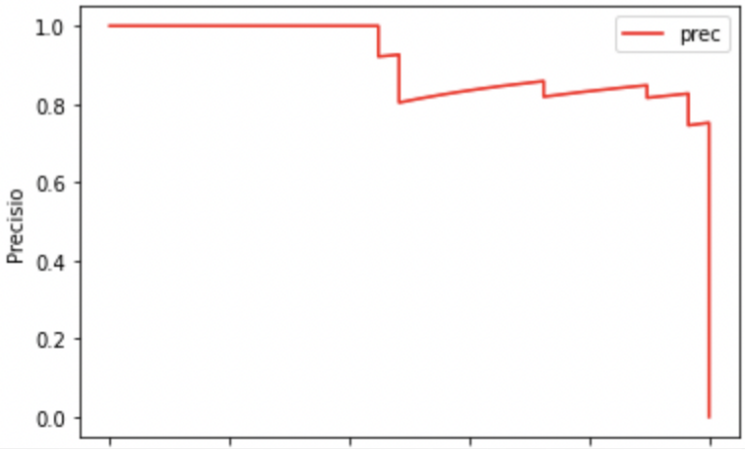
Even though we are using the same test queries, we decided to use a different method of evaluation. Since the user's interests are related to the possible topics, we wanted to include them in our evaluation process. Therefore, we measure the recall and precision based on whether the suggested articles are relevant or not.
In this table you can see an example for the first test query:

Rank	Relevant	Number retrieved	Relevant retrieved	Recall	Precision
1	yes	1	1	0.033	1
2	yes	2	2	0.067	1
3	yes	3	3	0.1	1
4	yes	4	4	0.133	1
5	yes	5	5	0.167	1
6	yes	6	6	0.2	1
7	yes	7	7	0.233	1
8	yes	8	8	0.267	1
9	yes	9	9	0.3	1
10	yes	10	10	0.333	1
11	yes	11	11	0.367	1
12	yes	12	12	0.4	1
13	no	13	12	0.433	0.928
14	no	14	12	0.433	0.867
15	no	15	12	0.433	0.812

This Precision-Recall-Curve shows whether the related articles are in the same topic as the input query as described in the previous paragraph:



Another approach that we wanted to try was to build the precision and recall scores based on whether the related articles are higher than a certain similarity score threshold:



This Precision-Recall-Curve of the other group shows that their method made more errors at the beginning of the sequence than our method, but in the second half of the test the other group also found the relevance of the content more often. The other group used a different method of assessing whether an article is relevant or not. They manually decided if an article is of relevance or not and then plotted their results.