Document Similarity

Lakshmi Sahithi Yalamarthi
PSU ID – 907580033
yalam2@pdx.edu

Siri Chandana Koduru
PSU ID – 937066070
siri4@pdx.edu

Abstract

This document gives information on how to check the similarity between documents. Document similarity is a growing technique of Natural language processing. The main objective of this work is to get similarity between multiple documents. The documents are with different formats and needs to be cleaned and preprocessed before applying NLP techniques.In this paper we have outlined on how various algorithms have performed and showed the results which algorithm performs better. We evaluate the algorithms using baseline methods and all the results are presented in this paper.

1 Introduction

Document similarity is measured as the amount at which two documents are like each other. Document similarity detection is an interesting approach in the modern era, where tons of documents are available across websites on a particular topic. Detection of amount similarity helps in duplicate detection. It also helps in filtering unique ideas described by the author in each document. Generally, two documents are treated as similar if they discuss the same topic if they share the same authors or citations if they carry similar image content. All these differences between documents discuss the similarities on a high level. This paper discusses methods about finding similarity at a granular level. The source for this project is "Aspect Based Document Similarity for Research Papers" which is a paper published in COLANG 2020. In this project, we have used multiple Natural language processing techniques to find distance between two documents and calculated the amount of similarity between them. We have conducted experiments on large amounts of data, which is multi-label and multi-class, using multiple Classifier algorithms and drawn conclusions on the best performing one.

2 Related Work

We have referred the paper **Aspect Based Document Similarity for Research Papers** for the understanding of what Document Similarity is and what methods have they followed in the paper.The paper describes a model to detect similarity between two documents and the aspect in which they are similar. Most of the traditional document similarity models predict the percentage at which two documents are similar or dissimilar, but the paper discusses the amount of similarity between a document pair based on each aspect of paper.

2.1 Summary of the paper

The section title with citation acts as a label for the paper, the labels are preprocessed and normalized before labelling the datasets. Transformer based models like baseline LSTM, BERT models are used on multi label multi class pairwise documentation, to find text similarity between documents. For Aspect based classification, each document is divided into 11 label classes where each label indicates different sections of paper like Introduction, background, and conclusion, calculating F1-score, precision and recall on the individual labels of the paper shows in which aspects a document pair is similar. This paper performs negative sampling, by introducing a class name NONE which collects document pairs that are dissimilar when selected randomly. Several experiments with different transformer models are calculated on ACL Anthology and CORD-19 datasets. The experiments concluded that SciBERT has maximum prediction including the section of citation.

The major interesting aspect of this paper is negative sampling of input datasets, and four-fold classification between training and test datasets.

3 Methodology

To measure the Document similarity we have first chosen the dataset from 20 news groups dataset which contains around 18000 datasets, in the form of documents related to 20 different topics.We have picked 4 topics from the dataset on which we have done preprocessing of data using Natural Language Toolkit and Genism packages. Preprocessing has reduced the size of dataset to a large extent. After preprocessing the algorithms are applied onto the dataset after which a kfold cross validation is performed to check the accuracy of the prediction.

4 Experiments

4.1 Input Dataset

The dataset we chose is 20 newsgroups dataset which contains 18000 datasets, in the form of documents related to 20 different topics. Among these 20 topics, there are few dataset groups which have more similarities as they belong to the same topic. We have picked four categories (electronics, space, misc. for sale and religion) from different partitions, to evaluate differences and similarities between datasets.

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

4.2 Preprocessing

Before designing the algorithm, we picked a few input datasets with enough data in them. The data is preprocessed using Natural Language Toolkit and Genism packages. As part of preprocessing of data, each input document is divided into words and paragraphs, and data corpus is used to remove stop words and Lemmatization using the gensim dictionary package. Stop words do not add much value in calculation of similarity so they are removed. After the preprocessing stage, the volume of input datasets is reduced to a certain extent. A K-fold validation is performed on Input datasets, which separates data into training and test datasets.

4.3 Code flow

Words are tokenized to find the average number of words per sentence, word frequencies are calculated using the TF-IDF model. This model gives a normalized occurrence value of a word, by multiplying a local component (term Frequency) with a global component (inverse document frequency). TFIDF weighs down the words that most frequently occurred in the documents. After working with word frequencies, we work on paragraphs of the document. We have used an algorithm called, distributed memory model of paragraph vectors. This algorithm works on a unique id called paragraph id, it collects average value of word frequencies to predict words that are sampled in each paragraph. Outputs are passed into Transformers like BERT, as they have provided state of art performance. These transformers perform large scale semantic comparisons and cluster data. Document embeddings, cosine and Euclidean distances are calculated using Word2vec,Doc2vec, BERT, Glove, TF-IDF. Baseline methods are used to evaluate the model. Execution and performance of each transform used in model is evaluated and documents with most similarities are detected.

4.4 Techniques for Similarity Detection

We used Cosine Similarity and Eucledian distances on top of our algorithm to find out similarity.

Cosine Similarity: It is the basic method of finding similarities between word embeddings. Here we calculate pairwise similarities by performing cosine distance on document embeddings. If two words are pointing to the same direction in vector space then the angle between them is 0, and their similarity is 1 which is maximum. If the two words are placed orthogonal to each other then have a cosine similarity of 0. These values range from -1 to 1. Mathematically, it is a normalized dot product of two vectors. To calculate cosine similarity programmatically we use cosine-similarity package in sklearn metrics.

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

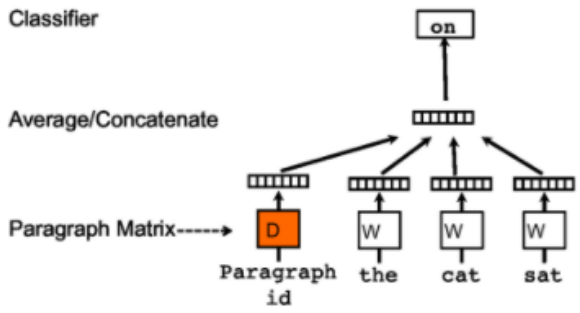
Eucledian Distance: The second method we used to find similarities is Eucledian

distance. Here we calculate pairwise wise similarities by performing Eucledian distance on document embeddings. It can be calculated as $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}$

4.5 Algorithms Used

TF-IDF: This assigns a number to every word in the document based on its frequency of occurrence, it is known as term frequency. Inverse document frequency factor is calculated by ignoring most repeated words in the documents, and considering frequency of the terms that occurs less frequently in the input corpus. In this project, TF-IDF is used to calculate the frequency of all documents with a particular word in it.

Distributed Memory Model of Paragraph vectors: In this model, each paragraph is mapped to a unique vector, represented as a column in matrix D, and each word is mapped to a unique vector in the matrix represented by W. The paragraph vectors are averaged to predict the next possible word in the paragraph. If there are N paragraphs in the corpus, M words if each paragraph is mapped to 'p' dimensions and each word is mapped to q dimensions then N x p + M x q dimensions in the model.



This figure shows that concatenation of three words to a paragraph matrix predicts the fourth word. Paragraph vector represents missing information from the context and acts as topic of the paragraph. This project calculates a bag of words and paragraph vectors to get unique data among all input datasets.

Word2Vec: Word to vector tokenized input data into bags of words and using algorithms like continuous bag of words and continuous skip grams it will find similarities between words. Each word is embedded to a vector, weighted average of the word embeddings are calculated and passed as input to cosine and euclidean distance to get pairwise similarity between documents.

GloVe: GloVe is an unsupervised learning algorithm that puts emphasis on the importance of word-word co-occurences to extract meaning rather than other techniques such as skip-gram or bag of words. The idea behind it is that a certain word generally co-occurs more often with one word than another. On top of running this algorithm the cosine and eucledian distances are run and compared.

BERT: BERT is a dense vector model, it is used to obtain similarity metrics, by calculating the respective similarity between input sequences. BERT uses a transformer architecture, to get conclusions from word embeddings. In BERT text contains three types of embeddings, Token embeddings, segment embeddings and position embeddings. All the word embeddings are compressed into sentence vectors. Sentence transformers library facilitates the implementation of BERT transformers. The output of BERT encoded document embeddings are used to calculate pairwise similarities in the input corpus.

Eucledian and Cosine distances from 5 algorithms we used

S.No	Algorithm	Cosine	Eucledian
1	TF-IDF	0.42	9.87
2	Word2vec	0.46	7.89
3	Doc2vec	0.55	7.69
4	GloVe	0.62	4.35
5	BERT	0.68	4.25

5 Evaluation and Baseline Methods

An ideal model of document similarity, must show high similarity between same partition documents and less similarity for cross-partition comparision. To evaluate this, a cartesian product of documents in a category a is compared with that of another category. results are obtained in the form of a matrix where all the values are summed up to calculate the average value of the cartesian product. To find similarities, the mean difference between the resulting average values is calculated. A higher mean difference shows the documents used in cartesian products are more distinct. As per this evaluation, the expected result will have high mean differences when compared across different topics of input partition.

Below are the mean-difference and same category average similarity for the 4 categories we have chosen for document similarity:

Category: misc.forsale
Mean difference: 0.2, Same-category average similarity: 0.
Category: sci.space
Mean difference: 0.089, Same-category average similarity: 0.33
Category: sci.electronics
Mean difference: 0.093, Same-category average similarity: 0.35
Category: soc.religion.christian
Mean difference: 0.28, Same-category average similarity: 0.45

6 Results obtained

We calculated similarity between documents using baseline method evaluation. The experiment is performed on high level granularity of input datasets. Besides this, to check similarity at granular level, between documents of a partition we have picked a category 'misc.forsale' and performed preprocessing, work tokenization of data to observe similarity using doc2vec gensim model. A document with name '74776' is picked as an 'test input document' and all other documents in the partition are evaluated to find the most similar document. This experiment resulted in 3 documents which share maximum similarity among all in misc.forsale category of input dataset.

```
test_doc = word_tokenize(str(input_df.loc["74776"]))
model.docvecs.most_similar(positive=[model.infer_vector(test_doc)],topn=5)
print(input_df.iloc[558])
print(input_df.iloc[514])
print(input_df.iloc[17])
print(input_df.iloc[269])
print(input_df.iloc[120])
```

```
letter_text    b'From: kpeterso@nyx.cs.du.edu (Kirk Peterson)...
tokenized      [b, kpeterso, nyx, c, du, edu, kirk, peterson,...
Name: 76394, dtype: object
letter_text    b'From: jingyao@rainier.eng.ohio-state.edu (Ji...
tokenized      [b, jingyao, rainier, eng, ohio, state, edu, j...
Name: 76157, dtype: object
letter_text    b'From: HO@kcgl1.eng.ohio-state.edu (Francis H...
tokenized      [b, ho, kcgl1, eng, ohio, state, edu, franci, ...
Name: 76023, dtype: object
letter_text    b"From: Michelle Zumbo <mz10+@andrew.cmu.edu>\...
tokenized      [b, michel, zumbo, mz10, andrew, cmu, edu, nsu...
Name: 76038, dtype: object
letter_text    b'From: dtmedin@catbyte.b30.ingr.com (Dave Med...
tokenized      [b, dtmedin, catbyt, b30, ingr, com, dave, med...
Name: 75864, dtype: object
```

Below are the documents on which we obtained similarity when we have run the algorithm: The three screenshots below are of the documents that have similarity. The similarity detected here is that the three document texts are on how to sell a product.

```
o I am selling this toner because I recently
  bought a Brother HL-10V printer and the
  toner that I am selling.  I activated the
  toner, but ended up returning the printer.
  The store, however would not take back the
  toner.  This toner has been used to print
  only three pages and is in perfect condition.
  I will protect it for shipment so that no
  toner escapes.  It comes with all original
  packaging and manuals.  The toner is compatible
  with ANY laser printer that uses the model
  number of the toner I am selling; just look in
  your manual to see if it will work for you.
  I will not go below $60.00.  I will pay the
  shipping to anywhere in the continental
  United States.
```

```
From: HO@kcgl1.eng.ohio-state.edu (Francis Ho)
Subject: 286 Laptop
Nntp-Posting-Host: kcgl1.eng.ohio-state.edu
Organization: The Ohio State University
Lines: 18

MITSBISHI Laptop (MP 286L)

-286/12 (12,8,6 MHz switchable)
-2M RAM installed
-Backlit CGA (Ext. CGA, MGA)
-20M 3.5"HH HDD/1.44M 3.5" FDD
-2 COM/1 LPT ports
-complete manual set
-Built like a tank
-Excellent cosmetic cond.
-dark gray
-used very lightly

Problems:
(1)HDD stops working.
(2)LCD sometimes doesn't work (ext. CAG/MGA works).

Best Offer.
```

```
Subject: Pressure meter
Reply-To: dtmedin@catbyte.b30.ingr.com
Organization: Intergraph Corporation, Huntsville AL
Lines: 21

Heise model 710A pressure meter. This is a precision 4-1/2
digit meter measuring 0 - 15 PSI (absolute) in .001 psi
increments.  Case is in extremely good shape, and can be used
as a stand-alone meter or panel mounted. Brass fitting (looks
like standard 3/8") on back. Operates from 110 VAC.

I'd like $50 for it, or make an offer. It is a lot more useful to
a lab than as an ersatz barometer, which is what I've been using
it for.

--
-------------------------------------------------------------------------
      Dave Medin                              Phone:   (205) 730-3169 (w)
   SSD--Networking                                     (205) 837-1174 (h)
   Intergraph Corp.
      M/S GD3004                  Internet: dtmedin@catbyte.b30.ingr.com
 Huntsville, AL 35894             UUCP:  ...uunet!ingr!b30!catbyte!dtmedin

  ******* Everywhere You Look (at least around my office) *******

* The opinions expressed here are mine (or those of my machine)
```

7 Insights

When we tried to implement the model that we proposed on the input dataset using python we were initially not able to run the whole dataset as there was numerous amount of data and when loading too much of data at once there was a crash, the RAM of the system was not enough to accomodate that much amount of data. As every project to start with is a trial we have thought of implementing it in various ways. Initially we tried loading the data manually and not using the dataset available in scikit learn which resulted in a crash. So that now we know on how to use the dataset from scikit learn I would suggest this

method to take the dataset for purpose of coding in a lesser time. If we had much more time to do this we would have taken all 20 groups dataset instead of 4 for performing the document similarity.

8 Ethical Consideration

The data has been collected in a careful and thoughtful manner. We have in the dataset a wide variety of data which involves various categories and there are no limitations on the data used. The model works as described above and below is the github link attached for the code. Plagarism detection can be done using this model. When two documents are given as an input we can detect plagarism using the proposed algorithm.

9 Conclusion

Cosine and Euclidean distances calculated similarities at each stage of this model. BERT transformer performed well in detecting documents with similarities and obtained highest values of cosine and euclidean distance. At a modular level, documents with similarities are detected based on the emotion found in document. At granular level, documents are detected as similar if they describe homogeneous actions such as selling a product, describing a product, invitation to a conference or minutes of meeting. This project is a prototype to detect plagiarism in large input corpus. This can be further improved by using advanced transformations and with better processing of input data.

10 Future Directions

We are planning to improve it by better using of transformers for detecting granular level similarity. The next is to extend this algorithm perfectly for plagiarism detection. We also like to improve better processing of input data using emotion lexicons, POS tagging and use repeated punctuations.

11 Code Link

We have our notebook of the code in .ipynb format at `https://github.com/sirikoduru24/Document-similarity`. Just download and run the code using google colab and the results can be seen out there.

12 References

1.https://dev.to/coderasha/compare-documents-similarity-using-python-nlp-4odp
2.https://arxiv.org/pdf/1908.10084.pdf
3.https://arxiv.org/pdf/1405.4053.pdf
4. https://arxiv.org/pdf/1910.09129.pdf
5.https://towardsdatascience.com/bert-for-measuring-text-similarity
6.https://towardsdatascience.com/detecting-document-similarity-with-doc2vec-f8289a9a7db7
7.https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine$_similarity.html$