NLP Final Project Report - Document Similarity Detection

Group 11.

Lakshmi Sahithi Yalamarthi

PSU ID – 907580033.

Siri Chandana Koduru

PSU ID - <>

Abstract:

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.

NLP Techniques for Similarity Detection:

Cosine Similarity:

It is the basic method of finding similarities between word embeddings. This project calculates 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.

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

Euclidean Distance:

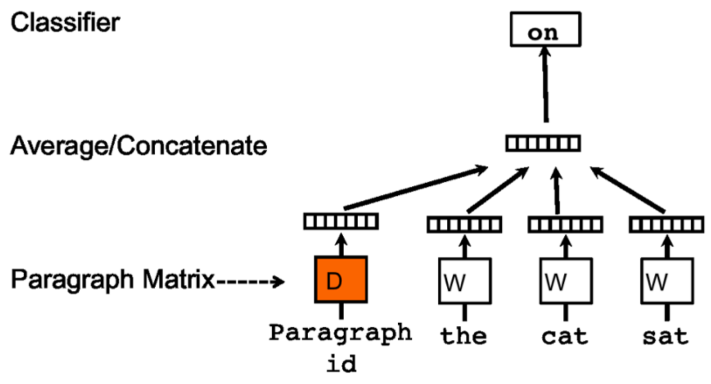
Euclidean distance is another form of

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.

BERT:

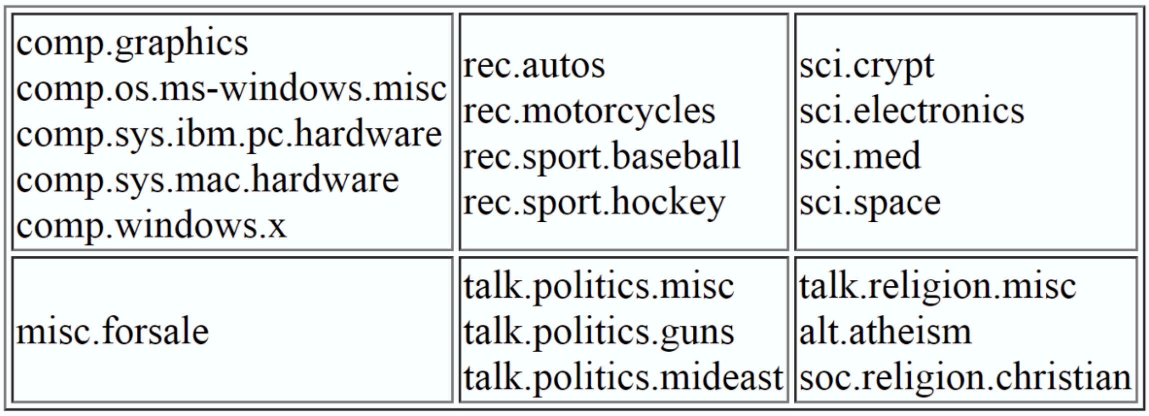
BERT is a dense vector model, it is used to obtain similarity metrics, by calculating the respective similarity between input sequences. BERT uses a transformer architecture, to get conclusions from word embeddings. In BERT text contains three types of embeddings, Token embeddings, segment embeddings 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.

Word to Vec:

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.

Input Datasets:

For purposes of testing and evaluating this model, 20 newsgroup dataset is used. It contains 18000 datasets, in the form of documents related to 20 different topics.



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

Among these 20 topics, there are few dataset groups which have more similarities as they belong to the same topic. Hence, This project has picked four categories (- electronics, space, misc. for sale and religion) from different partitions, to evaluate differences and similarities between datasets.

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.

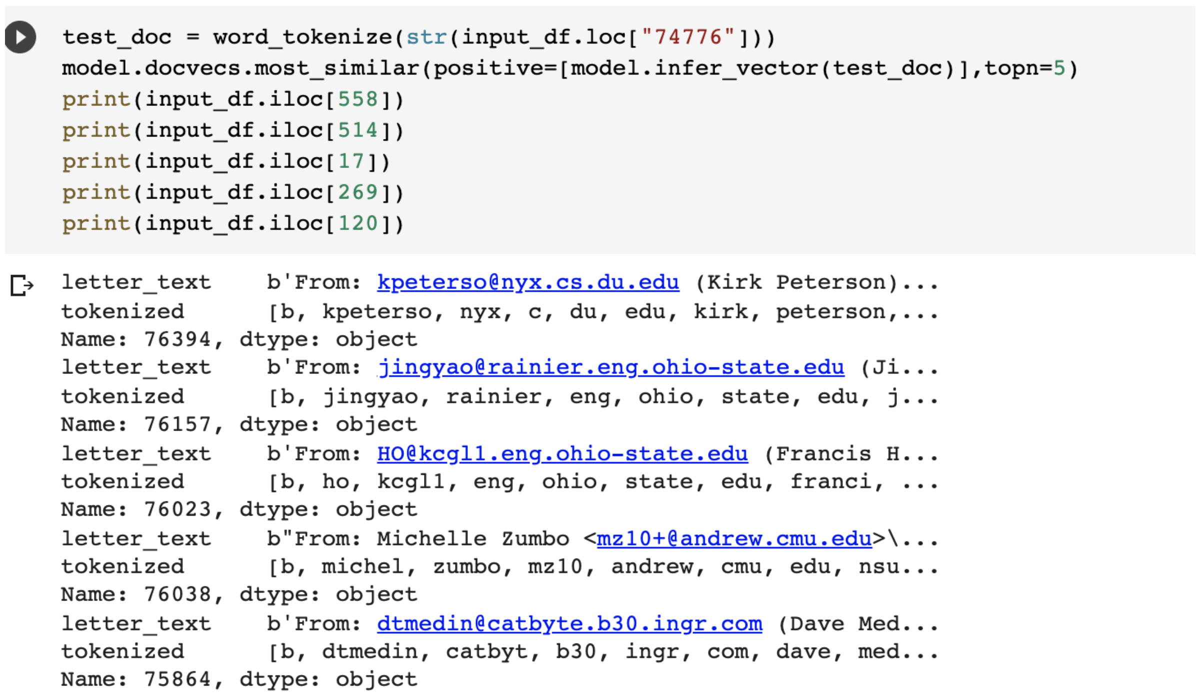
```
from nltk.stem import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
def tokenize(text):
  ###add utf encoding remove 'b' word
    text = str(text).lower()
    text = re.sub("[^a-zA-Z0-9]", " ", text)
    tokens = word_tokenize(text)
    tokens = [w for w in tokens if w not in stopwords.words(ENGLISH)]

    lemmatizer = WordNetLemmatizer()
    stemmer = PorterStemmer()

    clean_tokens_list = []
    for tok in tokens:
        lemmatizer_tok = lemmatizer.lemmatize(tok).strip()
        clean_tok = stemmer.stem(lemmatizer_tok)
        clean_tokens_list.append(clean_tok)

    return clean_tokens_list
```

Algorithm behind this project:

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.

Experiments:

This project involves calculating similarity between documents using baseline method evaluation. This 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
```

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.

$$Mean\ difference = \frac{\sum_{j=1, j\neq 3}^{4} (S(C_3, C_3) - S(C_3, C_j))}{3}$$

Results:

Documents in the same partition of input datasets are expected to be more similar when compared with different partitions. As the documents in the same partition show a pattern in their description.

All these screenshots show a similarity in documents as each document is describing a product to be sold, selling price and seller details. This result is aligned with the expected result as all three documents are related to the same partition of the input dataset.

Though the above documents share less similarity among them, as these are from different input partitions, mean similarity and average values have shown a similarity between these documents. They describe a common emotion. Our project has analyzed input datasets and found similarities as well as differences among multiple partitions.

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.

Code Link:

References:

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

https://dev.to/coderasha/compare-documents-similarity-using-python-nlp-4odp

https://arxiv.org/pdf/1908.10084.pdf

https://arxiv.org/pdf/1405.4053.pdf

http://text2vec.org/similarity.html#cosine_similarity_with_tf-idf

https://arxiv.org/pdf/1910.09129.pdf

https://towardsdatascience.com/bert-for-measuring-text-similarity-eec91c6bf9e1