Report on Similarity Analyzing Approach

The computational complexity of cosine similarity is linear in the number of unique words in the documents, making it well-suited for short documents. However, its performance may not be optimal when dealing with large documents. The paper titled "Document Similarity for Texts of Varying Lengths via Hidden Topics" (https://aclanthology.org/P18-1218) proposes a solution for accurately measuring the similarity between long documents and short summaries. This method is designed to tackle the challenges posed by vocabulary and context mismatch. The approach is to compare the texts in a common space of hidden topics, enabling a multi-view generalization of the papers. Another paper, "An improved semantic similarity measure for document clustering based on topic maps," (https://arxiv.org/ftp/arxiv/papers/1303/1303.4087.pdf), proposes a new semantic similarity measure based on topic maps to overcome the limitations of traditional vector-based models. This measure uses a lexical database and semantic class hierarchy to calculate similarity. These proposed methods can be applied to enhance our current model.

In the paper, 'Text Similarity based on English Morphological Analyzer Approach' (https://www.iasj.net/iasj/download/f52acc684b6d24e3), the developed method is summarized by applying the English Morphological Analysis on the text. Then it applies statistics and linguistics approaches such as; synonym, word position, and Part-Of-Speech (POS), then it computes word frequencies. For short texts, frequency is enough to compare two texts and compute the ratio of similarity between them. For long texts, the method uses an additional linguistic approach. It applies keywords extraction then computes the similarity. While for very long texts, it applies text summarization then computes the similarity.

The used algorithms for short texts and long texts are stated below:

Algorithm for calculating similarity ratio between two short texts

Input: all facts stored on working memory, (facts of source-text (ST), and facts of tested text (TT)).

Output: similarity ratio.

Begin

Step1: create: similarity-negation-counter (sn), similarity-counter (sc), and different-noun-counter (dnc). sn=1, sc=1, dnc=1.

Step2: for each two sentences if equal negation flag then sn=sn+1.

Step3: compare all words frequencies in two texts as following:

3.1: if they are equal then sc=sc+freq.

3.2: if they are not equal then add minimum (sc=sc+freqMin), and check POS for the missing word, if it is noun then dnc=dnc+missing.

Step4: make sure that nc=negation counter of ST.

Step5: if sc=frequencies of ST then similarity- ratio(sr)=111, goto End.

Algorithm for calculating similarity ratio between two long texts

Input: all facts stored on working memory, (facts of source-text (ST), and facts of tested text (TT)).

Output: similarity ratio.

Begin

Step1: compute similarity-negation (sn) using negation counters (nc) of ST and TT:

If ncTT>=ncST then sn=29 else sn=29-(abs(ncST-ncTT))*(29/ncST)

Step2: create: source-frequency-counter (freq1), tested-frequency-counter (freq2), and different-noun-counter (dnc). freq1=1, freq2=1, dnc=1.

Step3: for each keyword at ST, add its frequency (sf) to counter, freq1=freq1+sf. Then do:

3.1: if found the keyword at TT with the same frequency (tf) or more than sf; sf<=tf, then add to counter, freq2=freq2+sf.

3.2: else if tf<sf then check POS for the keyword. If it is noun then add the different to dnc, dnc=dnc+(sf-tf).

Step4: if tf<sf and subject pronouns counter of ST (c1) less than subject pronounscounter of

The cosine similarity analyzer uses the cosine similarity metric to measure the similarity between two documents or texts represented as vectors.
As for improving the cosine similarity analyzer, the paper "Similarity Analyzer for Semantic Interoperability of Electronic Health Records Using Artificial Intelligence Techniques: Initial Experiments"suggests the following:

1. Utilizing advanced Natural Language Processing (NLP) techniques such as lemmatization, stemming, and stop-word removal to preprocess the texts and improve their vector representation.
2. Incorporating external knowledge sources, such as medical ontologies, to further improve the vector representation and the cosine similarity scores.
3. Using deep learning models, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), to learn more sophisticated representations of the texts and improve the cosine similarity scores.
4. Using the transformer architecture, such as BERT, to encode the texts and perform semantic similarity analysis.

The paper provides initial experiments and results to demonstrate the effectiveness of these approaches. However, further experimentation and validation is needed to fully assess the potential of these approaches and their impact on the performance of the cosine similarity analyzer.

(https://www.researchgate.net/publication/224683313_Enhancing_Text_Clustering_Using_Concept-based_Mining_Model) proposes a concept-based analysis of the importance of words or phrases within a document.The frequency of a term is analyzed statistically to assess its significance at the sentence and document levels.The study then explores the impact of individual term on the sentence semantics. Here conceptual term frequency (ctf) is proposed to analyze individual concept at the sentence-level and term frequency (tf) is proposed to analyze frequency of individual concept at the document-level .
here the weight of each concept is calculated as $tf\ weight$ at the document-level and the weight of each concept is calculated as $ctf\ weight$ at the sentence-level.

$$tf\ weight_{i1} = \frac{tf_{ij1}}{\sqrt{\sum_{j=1}^{cn1} (tf_{ij1})^2}}$$

$$ctf\ weight_{i1} = \frac{ctf_{ij1}}{\sqrt{\sum_{j=1}^{cn1} (ctf_{ij1})^2}}$$

Algorithm: Concept-based Term Analyzer

Step 1: Initialize a new document d_{doci} .

Step 2 :Initialize an empty list L to store the matching concepts.

Step 3:For each sentence s in the document d_{doci}.

3.1: Initialize a new concept c_i in sentence s.

3.2: For each concept c_i in the set of concepts {c1, c2, ..., cn} in sentence s

3.2.1:Compute the term frequency tf_i of concept c_i in document d_{doci}

3.2.2 Compute the sentence term frequency cf_i of concept c_i in sentence s.

Step 4:For each document d_k, where k = {0, 1, .., d_{doci} - 1}, , c_i exist do

4.1For each concept c_j in the set of concepts {c1, c2, ..., cm} in sentence s.

4.2. If $c_i = c_j$

4.2.1 Compute the average of tf_i and tf_j as $tf\ weight$.

4.2.3. Compute the average of ctf_i and ctf_j as $ctf\ weight$.

4.2.3. Add the new concept matches to list "L".

Step 5:Output the matched concepts list L.

In our model we didn't focus on the importance of each term at document and sentence level .Using the term weight mentioned above , the model would have a better chance of accurately analyzing the similarity between articles.

This article (https://medium.datadriveninvestor.com/tf-idf-for-similarity-scores-391c3c8788e8) explains the concept of using TF-IDF for measuring similarity scores.Main idea is that "The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus (data-set)."

Computing the Term Frequency(tf):

Term Frequency indicates the number of occurences of a particular term *t* in document *d*.

tf(t, d) = N(t, d)

wherein tf(t, d) = term frequency for a term *t* in document *d*.

N(t, d) = number of times a term *t* occurs in document *d*

Since term frequency rely on the occurrence counts, thus, longer documents will be favored more. To avoid this, normalize the ***term frequency***

tf(t, d) = N(t, d) / ||D||

wherein, ||D|| = Total number of term in the document

Computing the Inverse Document Frequency(idf):

It typically measures how important a term is. TF considers all terms equally important. However, it is known that certain terms, such as auxiliary verbs, prepositions, conjunctions may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scaling up the rare ones.

First find the document frequency-

$$df(t) = N(t)$$

where-

$df(t)$ = Document frequency of a term t

$N(t)$ = Number of documents containing the term t

The idf of a term is the number of documents in the corpus divided by the document frequency of a term.

$$idf(t) = N/ df(t) = N/N(t)$$

But the factor (most probably integers) seems too harsh so, the logarithm is taken.

$$idf(t) = log(N/ df(t))$$

tf-idf Scoring:

Now that both tf and idf are defined, these two are combined to produce the ultimate score of a term t in document d.

$$tf\text{-}idf(t, d) = tf(t, d)* idf(t, d)$$

Our model can be enhanced by initially computing the term frequency-inverse document frequency (tf-idf) and subsequently utilizing cosine similarity. This approach allows the model to place greater emphasis on the most significant words, thereby resulting in a more precise outcome.

Our model may face challenges in finding the similarity in papers translated into a language other than English. We can implement a step of initial translation of the transmitted document into English to improve the model's performance. This way, the model will have a better chance of accurately analyzing the document's similarity with other papers.