Drugs4Covid: Antón Aba Varela

Scope

On the notebook *soa.ipynb* presented in the course, we were shown different methodologies in order to produce text vectorization in order to capture document similarities. TF-IDF calculates the relative importance of a term for a document, based on the number of on its frequency in the document itself and the document frequency (number of documents in the corpus, which contain the term).

However, it lacks any understanding of the semantics or the context around these words. On the hand, the Document-to-Vector models attempt to create a N-dimensional space which serves as representation of the topics and semantics of words, sentences, documents, etc. They take into consideration the ordering and context around words within a narrow context, Context words are taken as words observed to surround a target word.

On the flip side, the words themselves and its importance is overlooked by these methods, which is really the basis of TF-IDF weighting. Therefore, I have thought that combining them in such a way that one fixes the weaknesses of the other, since each one has what the other one is lacking. Finally, it is in my best interest to keep this algorithm from being too complex, avoiding time-consuming lengthy executions.

Implementation

The notebook attached contains all the code necessary to run it full it, as well as small comments regarding some decisions taken. However, the proper explanation will be provided on this document below.

Preprocessing

On top of tokenizing the text, I appended the tokenized title next to it so that it can also be processed. I do not know why this was not done before because, although it is just one sentence, it is probably the most important one of them all to find similarities with other scientific documentation. As it can be seen in the results the most similar papers share one or two words in the title.

Secondly, I decided to incorporate a stemmer into the tokenization process. Why? Because, although a the Doc2Vec model will easily comprehend the common semantics, the TF-IDF method would understand think and thinks as 2 completely different words, and I would like to get rid of that disadvantage. Also, it will reduce the number of terms to be stored and processed when comparing documents

TF-IDF weighting

I have kept the same built as the one shown in the lecture since I do not feel proficient enough to make a more adequate version. It produces a dictionary where each word has its assigned weight.

Doc2Vec

I have kept the same built as in the *soa* notebook for the same reasons. The combined result uses generalization to create a low dimensionality vector which represents the content of the file.

Final model

In order to combine both models, I have decided the following approach, which will be illustrated with an example comparing 2 docs: doc1, doc2 with dictionaries dic1, dic2.

1st Compute **cosine similarity** between both vectors, representing the similarity in topics.

2nd Compute **term similarity** (initialized at 0):

for every word in doc1 that it is in also in doc2:

1. Vector = (dic1(word), dic2(word)).
2. **Term similarity** = term similarity + norm(vector)

Then, we obtain the final score for the comparison of both documents, by multiplying the **cosine similarity** times the **term similarity.** If both of them are large/small with respect to the other comparisons, then the score will be large/small too (respectively). On the other hand, if one of the similarities is large and the other small, the combination will balance out both values punishing in the cases in which one of the values is really low.

Results

Compared to the results from the Document Similarity notebook, it makes documents which are extremely similar have a higher score with respect to the not-so similar documents. In other words, if the previous implementation was linear-like this one is exponential-like. My model:

```
_____
Large-Scale Semantic Exploration of Scientific Literature using Topic-based Hashing Algorithms
Large-Scale Semantic Exploration of Scientific Literature using Topic-based Hashing Algorithms : 1.3267997932190807
Scalable Cross-lingual Document Similarity through Language-specific Concept Hierarchies : 0.45327704259779733
Efficient Clustering from Distributions over Topics : 0.43635095027416754
Drugs4Covid: Making drug information available from scientific publications : 0.376771414983554
An initial Analysis of Topic-based Similarity among Scientific Documents based on their Rhetorical Discourse Parts : 0.29294420808991983
Legal Documents Retrieval Across Languages: Topic Hierarchies based on synsets : 0.28492203427856083
Enhancing Public Procurement in the European Union through Constructing and Exploiting an Integrated Knowledge Graph : 0.2840423329059178
Cross-Evaluation of Term Extraction Tools by Measuring Terminological Saturation : 0.28125561243735125
Distributing Text Mining tasks with librAIry : 0.2721462851317256
Potentially inappropriate medications in older adults living with HIV : 0.2057106037772542
Semantic Saturation in Retrospective Text Document  Collections : 0.18904559866676657
```

Whereas before exact documents had a score of 1 and similar docs had a 0.99-0.96 score (or even narrower). Now the difference in scoring similarities is much wider. Soa's model:

```
Large-Scale Semantic Exploration of Scientific Literature using Topic-based Hashing Algorithms :
{'title': 'Large-Scale Semantic Exploration of Scientific Literature using Topic-based Hashing Algorithms', 'score': 1.0}
{'title': 'Efficient Clustering from Distributions over Topics', 'score': 0.9861946702003479}
{'title': 'Drugs4Covid: Making drug information available from scientific publications', 'score': 0.9829981923103333}
{'title': 'Scalable Cross-lingual Document Similarity through Language-specific Concept Hierarchies', 'score': 0.9792618155479431}
{'title': 'Enhancing Public Procurement in the European Union through Constructing and Exploiting an Integrated Knowledge Graph', 'score': 0.9790343642234802}
{'title': 'Legal Documents Retrieval Across Languages: Topic Hierarchies based on synsets', 'score': 0.9713021516799927}
{'title': 'Distributing Text Mining tasks with librAIry', 'score': 0.961406946182251}
{'title': 'An initial Analysis of Topic-based Similarity among Scientific Documents based on their Rhetorical Discourse Parts', 'score': 0.9529688954353333}
{'title': 'Potentially inappropriate medications in older adults living with HIV', 'score': 0.9393362402915955}
{'title': 'Semantic Saturation in Retrospective Text Document  Collections', 'score': 0.8930673003196716}
{'title': 'Cross-Evaluation of Term Extraction Tools by Measuring Terminological Saturation', 'score': 0.6838582158088684}
```

Conclusions

The results obtain show that my model is better suited for distinguishing between similar documents and identical documents, successfully reducing the problems initially perceived for Doc2Vec, which scored almost the same value in both cases. Also, I believe that it successfully relativizes the term importance using the document similarity, along with keeping the execution time almost.

If any improvement were to be done would be the deep study of different term similarity functions, which could be optimal for this task. I have tried multiple of them and the one that has given the better results is the one shown on this paper; however, it would be interesting to see what other people could come up with.