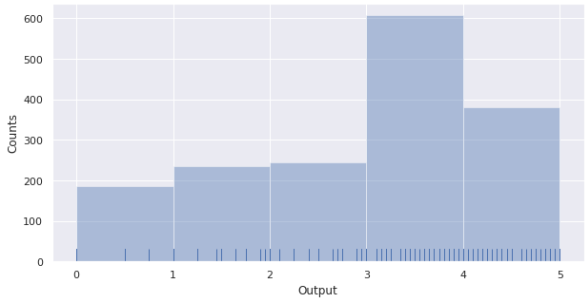
BMI 598, project, final report (Semantic Text Similarity problem; Klim Drobnykh and Jitesh Pabla)

Introduction

The task is about finding an automated method to measure the degree of semantic equivalence between two samples of biomedical texts. In our case, the biomedical texts contain just one sentence from clinical notes. The similarity is measured by real numbers from 0 (completely dissimilar) to 5 (complete semantic equivalence). Performance of the method is measured by the Pearson correlation coefficient between the predicted similarity scores and reference human judgments (two clinical experts were asked to manually annotate every text pair).

Methods

The provided dataset contains 1654 biomedical sentence pairs, along with the similarity score for each pair. The similarity score is a real number ranging from 0 to 5, which is distributed in a skewed fashion. The figure on the right there is a histogram that visualises this distribution for the training set.



As for the preprocessing step, let's assume we have the following sentence:

Chlorthalidone 50 mg tablet 1 tablet by mouth one time daily.

We decided to follow the standard preprocessing procedure:

- convert all the letters to lowercase (**chlorthalidone 50 mg tablet 1 tablet by mouth one time daily.**)
- convert digits to words (**chlorthalidone fifty mg tablet one tablet by mouth one time daily.**);
- remove the punctuation (**chlorthalidone fifty mg tablet one tablet by mouth one time daily**);
- tokenize ([‘**chlorthalidone’, ‘fifty’, ‘mg’, ‘tablet’, ‘one’, ‘tablet’, ‘by’, ‘mouth’, ‘one’, ‘time’, ‘daily’**]);
- remove the stop-words ([‘**chlorthalidone’, ‘fifty’, ‘mg’, ‘tablet’, ‘one’, ‘tablet’, ‘mouth’, ‘one’, ‘time’, ‘daily’**]).

The token vectors were used to extract the following features for each sentence:

- **Token-based features [TB]:** We used several set similarity measures to capture the similarity between the two sets of tokens. These measures include: Jaccard, overlap, cosine, dice, Tversky index, 2 Q-gram, 3 Q-gram, 4 Q-gram, Monge-Elkan and bag similarities.
- **Embedding based features [EB]:** Each token in the set of tokens for a sentence is looked up in a set of pre-trained word embeddings, and then we used averaging, maximizing and minimizing to create a single vector (for a sentence). We wanted to use something more sophisticated (like bidirectional LSTM), but we read from several papers that mean pooling performs better than LSTM. Similarity measures like cosine, euclidean, manhattan, Pearson correlation and word mover's distance were calculated for these vectors as extra features. Initially, Google's word2vec was used, but the sentences in the dataset contain a lot of biomedical domain-specific terminology that is not found in general English word corpora. Therefore there was a need to use certain corpora like PubMed etc. to capture the semantic meaning of each sentence in order to measure their similarity. So we used BioWordVec embeddings (fastText-based) and Word2Vec models trained on PubMed and PMC datasets. We also wanted to try BioSentVec, but the model is 21 Gb and we don't have that much RAM available. We thought about word correction, because for general domain Word2Vec there were 245 words that are not in the dictionary, but after using BioWordVec we had only 10 words that were not in the dictionary [',ä¢', 'currentand', 'odegarden', 'dextroamph', 'errin', 'plexion', 'locationemergency', 'didnaa', 'educationready', 'locaation'].

Also, we tried BERT models, because they showed the best results for several NLP tasks during the last year. Different pretrained BERT models were used to extract the word-piece embeddings (and the CLS vector) for each sentence one at a time. The embeddings were pooled (average, minimum and maximum pooling) into one vector for each sentence. The similarity measures used in the EB features were used here too on the pooled vectors (from both sentences) and the CLS vectors to extract the final features for each BERT model.

We also tried to input both sentences at once and use the CLS vector with regression, but it did not perform as well as extracting embeddings one sentence at a time. (Principal component analysis was used to reduce the 768 dimension CLS vector to 20 dimensions).

Some selected BERT models (BERT base, BioBERT, RoBERTa) were also fine-tuned on semantic text similarity (STS) benchmark and the given biomedical STS dataset. The models were fine-tuned in the following manner: both CLS and embeddings were extracted for 2 sentences and cosine similarity was used as a loss function. That way, the BERT

model is trained for a pair of sentences at a time. We tried adding dense layers on the BERT output before extracting the similarity features, but the performance dropped severely.

The final step was machine learning. In particular, regression algorithms. For that, we just used a simple gradient boosting trees algorithm. For performance evaluation on the training set, we split the training data to 80% actual training data and 20% validation data. We used 80% to tune the hyperparameters (using cross-validation) and 20% for validation. Gradient boosting trees were used because we had a lot of features with different scales and sources. The hyperparameters we tuned include: different number of estimators, depth, minimal samples required to split and subsampling rate.

New idea (dissimilar words)

When we implemented the methods discussed above, we decided to analyze our results. So, we sorted all the training samples by absolute magnitude of residuals (difference between true and predicted score). We used cross-validation to predict similarity scores for all the training samples. Among the most different samples, we found a lot of cases like below:

Sent1: "loratadine 10 mg tablet 1 tablet by mouth one time daily.", Sent2: "Pravachol 20 mg tablet 1 tablet by mouth one time daily.", Similarity = 1.0, Predicted = 3.5987742299232144;

Sent1: "ibuprofen tablet 2 tablets by mouth as needed.", Sent2: "prednisone 20 mg tablet 2 tablets by mouth one time daily.", Similarity = 1.0, Predicted = 3.102518178110601.

And we came up with the idea to extract only the words that are different in the sentences. We wanted to use a variation of the Levenshtein distance, but on tokens instead of characters. In that way, we can find the minimal number of words that should be added, removed or modified to convert one sentence to another. Since the remaining words are not really connected in the form of a complete sentence, we use word2vec rather than BERT. Since in that case, we do not care about the positions of the words anymore (with word2vec, we use the pooling methods), so we can treat 2 sentences as 2 sets of tokens. For sets of tokens, our idea can be implemented in an easy way: we can compute the common set of stems between 2 sentences. Next, we can remove all the tokens whose stems are not in that set, such that, we have only the dissimilar words left and for them we can compute our token and embedding-based features.

Experimental results

For the preprocessing and embedding-based features we used one Jupyter Notebook, which was run on one of our desktop computers. For BERT-related models, we used separate notebooks which were run in Google Colab.
In the below table, you can see the results of our experiments:

Method	**Pearson correlation score on test [train] datasets**	
Levenshtein [character-based]	0.03 [0.105]	
Token-based [stanfordnlp]	0.618 [0.812]	
Token-based [nltk]	0.578 [0.796]	
Token-based [nltk + digits]	0.62 [0.810]	
Embedding-based:	Features	+ token-based
● Word2Vec	0.66 [0.829]	0.70 [0.845]
● BioWordVec	0.664 [0.753]	0.693 [0.806]
● Word2Vec on PMC	0.575 [0.789]	0.654 [0.828]
● Word2Vec on Pubmed&PMC	0.624 [0.795]	0.672 [0.824]
● Word2Vec on Pubmed	0.641 [0.783]	0.687 [0.816]
● Word2Vec on Wikipedia + Pubmed&PMC	0.586 [0.782]	0.663 [0.824]
BERT-based:		
● NLI pretrained + STSB finetuned	0.663 [0.76]	

• BERT fine-tuned on our dataset	0.762 [0.887]
• BERT	0.471 [0.695]
• BioBERT	0.63 [0.77]
• BioBERT fine-tuned on our dataset	0.809 [0.92]
• ClinicalBERT	0.652 [0.751]
• RoBERTa fine-tuned on our dataset	0.778 [0.888]
• Combined BERT features	0.837
Combined [different BERT + word2vec-based]	0.845

As we can see from the table above, embedding-based models work well on the testing subset of our train data, but on test data they do not work well at all. We expected BERT models to outperform word2vec, but not that much. We think that the reason here is that train and test datasets are different.

Also, we can see that token-based features can improve embedding-based methods, and embedding-based methods can improve BERT. We can also see that because of our choice of the regressor (Gradient Boosting Trees), the score is not dropping as we add more and more features.

	Dissimilar words	NLTK tokenizer + Digits	Combined (NLTK + Digits + Dissimilar)
Token-based (TB)	0.622	0.797	0.807
Embedding-based (EB)	0.673	0.824	0.833
Combined (TB + EB)	0.678	0.843	0.842

The results using only 'dissimilar words' were worse (due to less information). We added 'dissimilar words' features to embedding-based features, but the result did not change significantly (you can see that it improved in the first 2 rows, but did not change much for the third one). A manual analysis showed that this was due to the embeddings used. Even though most of the dissimilar words for the cases we found were biomedical terms, we saw that BioWordVec cannot be used to measure the similarity between different biomedical terms (specifically drugs): we know that Loratadine and Claritin are similar drugs [allergy], but they are more similar to Tramadol [narcotic]. The cosine similarity between Claritin and Loratadine is 0.675, but between Loratadine and Tramadol is 0.728. It looks like even in BioWordVec, all the drugs just have similar vectors.

Conclusions

- Biomedical terms seem to play a huge role in the biomedical STS task. We examined the sentences and even though the meaning seemed to be the same, they had low scores just because the biomedical terms were not related (e.g., different diseases or symptoms);
- It's difficult to capture the semantics, because the same meaning can be expressed in completely different words. This is one of the main reasons why BERT-based approach works better as compared to both token-based and embedding-based methods;
- When we combined all the features [TB+EB+BB], the results improved in comparison to BB features only;
- BERT approaches performs well on unseen test data (compared to Token and Word2vec based approaches);
- Capturing semantics in a domain specific NLP task (like biomedical STS) using general domain approaches is a challenging task (Google Word2Vec, regular BERT, dissimilar words idea);
- Our methods don't outperform the state-of-the-art. The primary reason for this is the fact that our BERT models are not fine tuned on other biomedical tasks such as named entity recognition (NER) etc.

Contribution

The contributions of both members are equal for the project, from idea formulation to writing code and presenting the results in the report.