A Survey on Refined Preprocessing method for Intent Detection

Abhiraj Chaudhary, Hitesh Goyal
VIT Chennai, Tamil Nadu, India 600127
Email: abhiraj.chaudhary2019@vitstudent.ac.in,
hitesh.goyal2019@vitstudent.ac.in

Abstract—**In our previous effort on developing a large-scale Intent Detection Model we developed a promising neural network which allowed us to detect the Intent of context on an accurate scale. Later, we realised that there exist multiple word embedding methodologies. Here we try to analyse the different word embeddings and how they affect our Neural Network. Finally concluding which one would be the most effective of them all.**

1. Introduction

Word Embeddings in Natural Language Processing (NLP) are relatively new fields. Although there has been quite a bit of progress, since the development of GloVe by Jeffrey Pennington in August 2014, focused research is still required to further the field of NLP. Hence, in this research paper we are looking at different Word Embeddings and in what ways they affect the neural network pipeline and furthermore the accuracy.

2. Related Work

In the papers, there is a clear focus on figuring out the methodology of Word Embeddings. Herein we can see how GloVe and Word2Vec are clearly more advanced than TF-IDF. Focusing on developing pipelines with GloVe and Word2Vec in mind, it is possible for us to come up with a high accuracy model for Intent Detection. Hence, discovering this area is important as it will be the base for our latter approaches.

2.1. Using TF-IDF to Determine Word Relevance in Document Queries

This is comparatively an old method. It was developed to search for words in a large scale folder containing multiple documents. Methodology involves frequency distribution of each word in the overall document. This gives us a relevance ranking table which allows for the determination of relevance of each document in the document pool.

2.2. Efficient Estimation of Word Representations in Vector Space

In this Word2Vec paper, The Word2Vec model is used to extract the notion of relatedness across words or products such as semantic relatedness, synonym detection, concept categorization, selectional preferences, and analogy. A Word2Vec model learns meaningful relations and encodes the relatedness into vector similarity. This allows us to focus on the relations of specific words rather than the overall number of words to frequency distribution. Word2Vec is capable of finding similar meaning replaceable words to outcome interchangeable words.

2.3. GloVe: Global Vectors for Word Representation

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. The good part about this model is the amount of work done on GloVe. Stanford University has done an excellent job of providing us with large-scale vocabulary pre-trained which efficiently cuts down our model processing.

3. Method

3.1. Datasets

The chosen dataset is "Sarcasm detection in News Headlines" which can be found on Kaggle. The dataset contains news headlines and their classification as sarcastic or not and even contains a link to the articles. The dataset has been chosen due to its relatively clean sentences, and easier processing. The labels attained are reliable and this makes for a good, reliable dataset for research and comparative purposes.

3.2. Preprocessing methods

3.2.1. TFIDF. The full form of TF-IDF is Term Frequency-Inverse Document Frequency is a product of 2 factors, TF and IDF. TF is the number of times a word is repeated in a document. IDF is derived from the number of documents in which the word is repeated in all the documents. Intuition behind TF-IDF is that the weight assigned to each word not only depends on a word's frequency, but also how frequent that particular word is in the entire corpus/corpora.

3.2.2. Word2Vec. Word2Vec is used to extract the notion of relatedness across words or products such as semantic relatedness, synonym detection, concept categorization, selectional preferences, and analogy. This allows us to compare vectors and be able to find similar replaceable words in a sentence. Allowing us to extract the contextual meaning of each word.

3.2.3. GloVe. It is an unsupervised learning algorithm developed by Stanford for generating word embeddings by aggregating a global word-word co-occurrence matrix from a corpus. The benefit of GloVe over any other embedding is the vast vocabulary trained and provided by Stanford for public use.

3.3. Using the Preprocessing methods

3.3.1. Vector Sum. The concept is as simple as the name itself. We simply add the individual vectors of the words in the document getting the *sum* of all of their individual *meanings*.

3.3.2. Ordered Vector. In case of ordered vectors, we make a list of all the word vectors following one another as if it were a sentence in vector form. To be able to pass this into the model, we pad the list with null vectors making it one of standard shape.

3.4. Model Architectures

3.4.1. Simple Neural Network. The simple Neural network has an input layer, one hidden layer with 128 nodes and batch normalization and then the output layer.

3.4.2. LSTM for Vector Sum. The LSTM neural network has the same input layer as the simple neural network. It is followed by the hidden, LSTM layer which contains 8 LSTM units and then the same output layer.

3.4.3. LSTM for Ordered Vector. The LSTM neural network for the ordered vector has a different input layer as the other two networks and is followed by a similar, hidden, LSTM layer which contains 8 LSTM units and then the same output layer as the other two.

3.5. Implementation

To find out how well different preprocessing methods work, we developed an almost homogeneous pipeline. The key aspects of the pipeline were:

1) Loading/Creating the word vectorization function.
2) Converting text from the dataset to vector form.
3) Training and evaluating models

3.5.1. Vectorization. The aim of vectorization is to convert any word into the form of a numeric vector. Once done, this vector can be used in place of the word to train the model.

1) **TF-IDF** - TF-IDF vector is vocabulary and data specific. So, we load the data and create a TF-IDF vectorizing function using *sklearn* libraries.
2) **Word2Vec** - One of the most commonly used pre-built Word2Vec vectorization models is provided by *gensim*. The model is named *google-news-300*.
3) **GloVe** - The Natural Language Processing department of Stanford provides multiple pre-trained vectorization corpora. They can be downloaded and word-wise vector mapping can be done to convert the words to vectors.

3.5.2. Data conversion. The vectorization mechanism is word-level, i.e., the functions take the input as a word or token and provide its vector. When dealing with intent detection, it becomes important to make a document-level vector. For that, we use two approaches.

1) **Vector sum** - We take the element-wise sum of individual words to combine their meanings. So, the vector for a document would be the element-wise sum of all the vectors whose corresponding words occur in the document.
2) **Ordered Vector** - We understand that the previous approach may not be able to capture meaning too well due to its lack or order. So, we use another method in which we append the vectors one after the other and pass it into the model. The issue with this approach is that all models do not accept this type of an input due to which this model is only implemented on LSTMs.

3.5.3. Model training. A total of three models have been built for each of the approaches. The models have as similar an architecture as is possible. The data is passed into three models.

1) **Simple Neural Network** receives the vector sum and learns any features only through the vector sum itself.
2) **LSTMs for vector sum** is just like the simple neural network when it comes to input method and output method. The only difference is the learning model in between.
3) **LSTMs for ordered vectors** receive the ordered vectors and so, they get a *sentence vector* instead of a *final meaning vector*.

3.6. Results

When the pipeline is followed and the models are created, a total of 9 models are created which can be compared to one another.

	TFIDF	**Word2Vec**	**GloVe**
Simple NN	0.73	0.74	0.75
LSTM (VecSum)	0.74	0.75	0.72
LSTM (OrdVec)	0.73	0.78	0.76

3.6.1. Best Preprocessing method. We can see that the results for all the different methods are very similar. But, when seen in closer detail, it appears as though the best performance is seen in the case of the Word2Vec preprocessing method. In that, the ordered vector gives the best predictions.

3.6.2. Best model. The simple neural networks seem to perform almost as well as the LSTMs and even equal or surpass the results in some cases. But, overall, the best accuracies are seen in LSTMs trained on ordered vectors even if the margin is small.

3.6.3. Other observations. An important observation one cannot ignore is that fact that the vector sums work almost as well as the ordered vectors. This means that the vector sums are actually capable of capturing meaning to a great extent and that the order of the words does not affect the models as much as one would think it should.

4. Conclusion

Github: Refined Preprocessing Implementation

The identification of the most effective preprocessing method led us to discover that most preprocessing methods work almost equally well when it comes to a text classification task like intent detection.

It may be so that some perform slightly better than the others but that does not need to be the exact case. Even with an attempt at maintaining the homogeneity of the methods, there could be some slight internal changes which cause the observations to be the way they are.

5. Future Work

Finding out the results of more preprocessing (vectorization) methods like the usage of ELMO and BERT to see their effectiveness is an obvious next step. Moreover, to test the models' effectiveness, using other intent detection datasets like those for predicting formality and different sentiments can be useful. Finally, using more Deep Learning models like Graph Neural Networks, other Recurrent Neural Network variants, Convolutional Neural Networks, etc.

6. Acknowledgement

This research work was supported by Vellore Institute of Technology, Chennai under Dr. Shridevi S. We would like to thank her deeply for her support and motivation which helped us create this paper and perform research on this topic.

7. References

[1] Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. InProceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) 2014 Oct (pp. 1532-1543).

[2] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. 2013 Jan 16.

[3] Ramos J. Using tf-idf to determine word relevance in document queries. InProceedings of the first instructional conference on machine learning 2003 Dec 3 (Vol. 242, No. 1, pp. 29-48).

[4] NLP GloVe Dataset: nlp.stanford.edu/projects/glove/

[5] The Illustrated Word2Vec: jalammar.github.io/illustrated-word2vec/