Sentence Extraction for Machine Comprehension

Hemavati Sabu, Meghana Nagori

Abstract*: *Machine comprehension is a broad research area from Natural Language Processing domain, which deals with making a computerised system understand the given natural language text. Question answering system is one such variant used to find the correct 'answer' for a 'query' using the supplied 'context'. Using a sentence instead of the whole context paragraph to determine the 'answer' is quite useful in terms of computation as well as accuracy. Sentence selection can, therefore, be considered as a first step to get the answer. This work devises a method for sentence selection that uses cosine similarity and common word count between each sentence of context and question. This removes the extensive training overhead associated with other available approaches, while still giving comparable results. The SQuAD dataset is used for accuracy based performance comparison.

Index Terms*: *Machine Comprehension, cosine similarity, word embedding, NLP, SQuAD.

I. INTRODUCTION

With the wave of digitization, more and more digital equipments are reaching the market. However, many people around the world are not able to use technology to the fullest. This is because to make the systems run at our command, the user needs to be familiar with the language the machine can understand. Achieving this level of technical soundness is difficult, considering the literacy level all around the world. This situation has led to a research trend that aims at making digital machines understand natural language, the language used by us, the humans for communication. Machine comprehension is one such research area which will give the system an ability to understand user's query in natural language and provide a correct response based on the data it has been supplied with. Machine comprehension is gaining importance for use in a variety of applications like search engines, chatbots, humanoids, consumer query system, etc.

Research in this area ranges from simple answer extraction to neural networks based systems for spotting answers in the given content. Earlier systems used to work on predicting the correct option for MCQ (Multiple Choice Questions) type questions using TF-IDF (Term Frequency - Inverse Document Frequency) scores of words in question and paragraph [1]. This was a very simple approach and worked well for simple questions. But, in reality, paragraphs and questions do not necessarily share common words and word orders. This scenario needs one to know the meaning of words which might not be directly accessible from the paragraph. Several deep learning strategies are being employed for the purpose. Using a machine learning based approach along with proper word embedding method is therefore needed to get correct answers. Recursive Neural Network (RNN) has been considered the best machine learning paradigm for natural language processing with algorithms like Long short-term memory (LSTM) model, bidirectional LSTM (BiLSTM) model, attention pointer model, and their combinations due to its ability to consider neighbouring sequences [2].

Now Convolutional Neural Network (CNN) is also being encouraged in this area. CNN has been preferred mostly for computer vision applications, but not much for machine comprehension. Lisa Yan came up with convolutional approach in 2017 where she compared CNN and CNN-LSTM for SQuAD dataset [3]. In 2018, Yuan et al. proposed CNN-LSTM method using commonsense knowledge for this purpose [4].

Instead of searching the whole paragraph for the answer, it can be helpful to search it in a specific sentence only [5]. This means the smaller the context, the better the chances of getting accurate answer. If the sentence containing the answer is selected precisely, then the overhead of processing the whole paragraph and embedding whole with question can be minimized.

We can treat this as a classification problem, where we classify sentences in the context based on whether they include the answer or not. In 2014, Yu et al. proposed deep learning for predicting sentence containing answer [5] and obtained 71% accuracy for the same. Here, they used a neural network based approach. Two models were constructed for sentence selection- bag-of-words model and bigram model. Bigram model uses CNN. They trained the model on so that the model learns the "semantic" similarity between question answer pairs. Pre-trained word embedding is used for better learning. In 2016, Rajpurkar et al. used multinomial logistic regression and with 180 million features to get the sentence detection accuracy of 79% [6]. This work tries to avoid the computational complexity involved with these approaches while getting comparable accuracy.

Revised Manuscript Received on 30 July 2019.

* Correspondence Author
Hemavati Sabu*, Computer Science and Engineering, Government College of Engineering, Aurangabad, India.
Dr. Meghana Nagori, Computer Science and Engineering, Government College of Engineering, Aurangabad, India.

The proposed algorithm for sentence prediction is described in the next section. This is followed by experiments and results discussed in sectionIII.

II. PROPOSED SYSTEM

To select the sentence as containing the answer, the sentence should have similar context as the question. For similarity finding, most common approach is to count the maximum number of common words between the documents or sentences. But, if document size is larger, many times, the irrelevant topics may get identified as relevant.

A. Word Embedding

In order to use the lexical content in mathematical models, we need to convert words into some numerical representation. The representation should be able to quantify the semantic and syntactic relation of the word with other words in the context.

Word embedding [7][8] is a technique to map each word as vector of real numbers which are n-dimensional vector. The vector is a semantic description of how that word is used in context. The basic idea of word embedding is that words of similar meaning are grouped nearby forming vector clusters, while antonyms are placed far apart in the n-dimensional vector space.

There are different word embedding approaches, the three most popular ones being - Word2vec[9],[10], GloVe[11][12], and fastText[13][14]. These three are considered as silver bullets of word embedding in NLP.

Word2vec method developed by Google involves high semantic nature in obtaining dependency of words. Under this approach, the distance between two words matters more than intrinsic properties of corpus. GloVe by Stanford University research group is a model for word representations. It is a unsupervised learning algorithm. It is created using Wikipedia articles. FastText is a word embedding method devised by Facebook's AI Research (FAIR) lab. FastText uses a neural network for word embedding. The model helps to create unsupervised learning or supervised learning algorithm for obtaining word vector representations. It is considered to be better for syntactic tasks. Pre-trained models for 294 languages are made available by Facebook through library 'FastText'.

Word embedding gives the syntactic and semantic score of words in the given context. We compared the performance of the system for three different word embedding methods.

B. Similarity Metric

While selecting a sentence, we assume that the sentence containing the 'answer' should have context similar to that of the question. For estimating the similarity, a commonly used approach is counting the maximum number of common words between the two entities being compared. However, if the document size is larger, there can be more than one instance of the same number of matching words.

This may lead to wrong sentence getting identified as appropriate. Such ambiguous cases can further be discriminated based on word similarities. With each word being represented in the form of a vector through word-embedding, we can use vector distance to estimate the word similarity. Two such distance metrics are enumerated below.

a) Euclidean Distance

It refers to the straightline distance between two vectors, here word vectors. The Euclidean distance function is as given in Eq. 1:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad \ldots(1)$$

where a and b are two n-dimensional vectors.

b) Cosine Similarity

Cosine similarity is a measure of the angular distance between two vector points. It is computed as the cosine of the angle between two vectors when the vectors are projected in a multi-dimensional space. The mathematical formulation is as given in Eq. 2:

$$cos\theta = \frac{\vec{a}.\vec{b}}{\left\|\vec{a}\right\| \left\|\vec{b}\right\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\sqrt{\sum_1^n b_i^2}} \quad \ldots(2)$$

where a and b are two vectors.

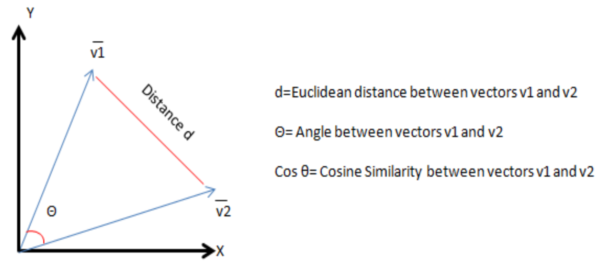
Y
X
v1
v2
Distance d
Θ
d=Euclidean distance between vectors v1 and v2
Θ= Angle between vectors v1 and v2
Cos Θ= Cosine Similarity between vectors v1 and v2

Fig. 1. Cosine Similarity and Euclidean Distance

Fig. 1 illustrates the concept of Euclidean distance and cosine similarity more clearly.

These two metrics give a measure of degree of similarity between the two words, given the word embedding vectors of the two. The cosine similarity has the advantage that even if the two words are far apart by the Euclidean distance, they could still have a smaller angle between them yielding higher cosine similarity.

Suppose we have three words -
word1 = 'college', word2 = 'employee', word3 = 'student'
Then we will calculate the 100-dimensional vector for each of these words. Cosine similarity and Euclidean distance for the same are then given as follows -
Euclidean distance ('college', 'employee') = 6.78762
Euclidean distance ('college', 'student') = 4.61085
Cosine similarity ('college', 'employee') = 0.341881
Cosine similarity ('college', 'student') = 0.724388
The smaller the Euclidean distance, the more similar the words are; while the larger the cosine similarity value, the smaller the angle between word vectors, and the more similar the words are. According to this, the words 'college' and 'student' are more similar compared to the words 'college' and 'employee', which is also intuitive.

III. EXPERIMENT AND RESULTS

In the experiments, we have used SQuAD dataset [6], a dataset made available for question answering research purpose by a research group from Stanford. The SQuAD dataset involves 87599 <context, question, answer> pairs for training, and 12727 pairs of <context, question> for testing.

From 87599, only 86304 pairs are usable. The remaining 1295 pairs are discarded due to problems of Unicode, answer span, conversion of character location to word location and actual answer retrieval using start and end index.

For the 86304 pairs, we determine the sentence containing the expected answer using the answer start index and the answer span details given in the dataset. The sentence number acts as ground truth for the sentence prediction task later.

A. Preprocessing

Before proceeding to sentence selection, preprocessing is applied to both the context and the question. The preprocessing consists of three steps - tokenization, lemmatization and stopword removal. The same are described in the following discussion, along with examples.

1) Tokenization: First, the punctuation marks are removed from the text. The whole text is converted to lowercase. Then, the context and question are broken into words.

For example, consider a statement-

“India has developed its own culture over centuries and holds a glorious history of the largest civilization.”

Tokenization then splits the sentence into a list of words as follows -

[‘india’, ‘has’, ‘developed’, ‘its’, ‘own’, ‘culture’, ‘over’, ‘centuries’, ‘and’, ‘holds’, ‘a’, ‘glorious’, ‘history’, ‘of’, ‘the’, ‘largest’, ‘civilization’]

2) Lemmatization: In English, we have different inflectional endings like ‘-ing’, ‘-s’, ‘-ed’ to represent the different verb forms and gender or number discrimination. These endings can be removed to get a base or dictionary form of the word, which will be more useful while matching the context and the question. This process of getting root words is called the Lemmatization and is performed on the list of words obtained after tokenization of context and question.

Lemmatizing for above example yields the following list - [‘india’, ‘have’, ‘develop’, ‘its’, ‘own’, ‘culture’, ‘over’, ‘century’, ‘and’, ‘hold’, ‘a’, ‘glorious’, ‘history’, ‘of’, ‘the’, ‘large’, ‘civilization’]

3) Remove Stopwords: Stopwords are words like articles and to be verbs which do not have any contribution in lexical understanding. These can, therefore, be neglected during the sentence selection task without any harm. Besides, we also remove the words that can never be similar in context and question. These usually include the Wh-words.

After removing stopwords in above example, we get-

[‘india’, ‘have’, ‘develop’, ‘its’, ‘own’, ‘culture’, ‘over’, ‘century’, ‘and’, ‘hold’, ‘glorious’, ‘history’, ‘of’, ‘large’, ‘civilization’]

B. Word embedding

Vector form for each word in context and question is obtained through word embedding. We use Google’s word vector file (300d) pre-trained on Wikipedia articles for word2vec embedding experiment. For glove embedding, the Glove Embedding vector files (100d) provided by Stanford has been used, where the global vector embedding obtained from Wikipedia articles. For FastText embedding, we used, FastText (300d), a word embedding method by Facebook, which uses word2vec and provides improved syntactic relationship between the words.

The words which have no vector in these pre-trained embedding models have been replaced with zero vectors. For a statement with 10 words and a question with 6 words, the word embedding step gives us vectors of size 10x300 or 10x100 and 6x300 or 6x100 as per the embedding method used (100d or 300d).

C. Steps for Sentence Selection

One approach for sentence selection is to take average of the word vectors in a sentence and compare it with the average of the word vectors of question. But the word information gets lost with this approach. Therefore, rather than using one vector per sentence, we consider vectors for each word in a sentence and find out similarity metric (cosine similarity/Euclidean distance) between each sentence word and question word. The corresponding steps are as follows -

1. For every word in the question, check the most similar word in the sentence using cosine similarity.
2. Add the similarities of these most similar words.
3. Sentence with the maximum sum is the most similar sentence.
4. This sentence is then the most probable place-holder for the answer.
5. If there is a tie, i.e., when two or more sentences have the same similarity score, the sentence which has maximum similar words with the question is considered as the most probable one.
6. If, still there is a tie, then arbitrarily, the first sentence is taken as the answer-holder.

First four steps in the above algorithm are shown through an example in the Table I below –

TABLE ISteps for Sentence Selection

Question / Sentence	~~what~~	Purpose	Is	Served	by	NGO
NGO		0.186	0.137	0.099	0.172	**1**
~~Serves~~ serf		-0.141	-0.090	-0.091	-0.008	0.041
To		0.552	0.628	**0.444**	**0.655**	0.121
Make		**0.557**	**0.689**	0.402	0.564	0.079
~~Children~~ child		0.398	0.543	0.311	0.461	0.261
Happy		0.374	0.534	0.236	0.354	0.036

Total score for the sentence

= sum of scores in bold
= 0.577 + 0.689 + 0.444 + 0.655 + 1
= 3.365

IV. RESULTS-

All the computations are performed on i3 processor with 8GB RAM. The implementation is done in python 3.7.3.

When we apply a simple approach of using cosine similarity between average sentence vector and average question vector, we get 67.55% accuracy. With Euclidean distance, the accuracy reduces to 66.78% accuracy. When the proposed approach is used, the accuracy obtained with different word embedding methods are listed in the Table

From the Table II, we see that the GloVe embedding proves to be the most efficient word embedding method. Further, we also observe that the glove vectors with 300 dimensional vector represent the words better than the 100d vectors. But, glove 100d provides speed in processing. The use of cosine similarity and Euclidean distance does not lead to a very significant change in accuracy with GloVe and both of them can be used.

TABLE IIAccuracy Result Table

Word Embedding Methods	Cosine Similarity		Euclidean Distance	
	Correct Selection	Accuracy	Correct Selection	Accuracy
Google Pretrained	67018	77.65	66629	77.26
FastText	67359	78.04	67473	78.18
Glove (300d)	67599	78.32	67577	78.30
Glove (100d)	67452	78.15	67367	78.05

We then tried using both the similarity metrics cosine similarity and Euclidean distance in sequence. GloVe embedding was used. When cosine similarity was followed by Euclidean distance accuracy of 78.03% was obtained, while if Euclidean distance was followed by the cosine similarity, the accuracy was 77.98%. However, the results are actually less than the ones reported in the table, where only a single feature was used.

Merits-

1) No need of training
2) Economy of Space and time
3) Simplicity
4) Comparable Accuracy

Demerits-

The method can be used where answers are in one sentence only, i.e. to give answers for factual questions

IV. CONCLUSION

The proposed approach is extremely efficient compared to the computationally extensive training involved in deep learning or computation of millions of features. It provides a much simpler and efficient way for selecting a sentence containing the answer for questions having answer words located in a single sentence. Glove embedding with cosine similarity proves better approach for sentence selection. Sentence selection task can be used in applications like knowledge base construction and information extraction. The approach has been tested on SQuAD dataset only. The future approach will be to increase accuracy while maintaining simplicity in the algorithm.

REFERENCES

1. M. Richardson, C. J. Burges, and E. Renshaw, "Mctest: A challenge dataset for the open domain machine comprehension of text," *in Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.
2. W. Shuohang and J. Jiang, "Machine comprehension using match-lstm and answer pointer," in *ICLR, 2017 International Conference on Learning Representations, Toulon, France, April 24-26: Proceedings. 1-15. Research Collection School of Information Systems*, 2017.
3. L. Yan, "A convolutional network approach to machine comprehension," *in Stanford University*, 2017.
4. H. Yuan, J. Wang, and X. Zhang, "Ynu-hpcc at semeval-2018 task 11: Using an attention-based cnn-lstm for machine comprehension using commonsense knowledge," *in Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), Association for Computational Linguistic*, 2018.
5. L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman, "Deep learning for answer sentence selection," *arXiv:1412.1632v1 [cs.CL]*, 2014.
6. P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv:1606.05250 [cs.CL]*, 2016.
7. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *in Proceedings of NIPS*, 2013.
8. T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *in Proceedings of Workshop at ICLR*, 2013.
9. Google, (2013) "Word2vec," [Online] Available: https://code.google.com/archive/p/word2vec/
10. Wikipedia, (2013) "Word2vec," [Online] Available: https://en.wikipedia.org/wiki/Word2vec.
11. J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," *in proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014.
12. J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," https://nlp.stanford.edu/projects/glove/ , 2014.
13. M. Tomas, G. Edouard, B. Piotr, P. Christian, and J. Armand, "Advances in pre-training distributed word representations," *in Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.
14. A. J. Piotr Bojanowski, Edouard Grave and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, p. 135-146, 2017.

AUTHORS PROFILE

Hemavati Sabu is ME student in Computer Science and Engineering Department of Government College of Engineering Aurangabad.

Dr. Meghana Nagori is a resourceful and dedicated Lecturer / Associate Professor / Data Science Specialist, with diverse experience and an exceptional record of sustained success; heading academic departments and driving forward best practice. A strategically focused leader and committed academic, passionate about ensuring the highest levels of teaching methodologies, who genuinely values opportunities to learn, improve and grow in pursuit of positive change. An accomplished manager, able to assess, define and execute the needs of teams; developing and training colleagues whilst heading academic projects and publishing research