


Short-text Semantic Similarity using GloVe word embedding

IRJET Journal

Related papers

[Download a PDF Pack](#) of the best related papers 

Short-text Semantic Similarity using GloVe word embedding

E. Hindocha¹, V. Yazhiny², A. Arunkumar³, P. Boobalan⁴

^{1,2,3}Final Year B.Tech, Dept of Information Technology, Pondicherry Engineering College, Puducherry, India

⁴Associate Professor, Dept of Information Technology, Pondicherry Engineering College, Puducherry, India

Abstract - Word embedding are word representations in the form of vectors that allow to maintain certain semantic information of the words. There exist different ways of taking profit of the semantic information the words have, as there exist different ways of generating the word vectors that represent those words (e.g. Word2Vec model vs. GloVe model). By using the semantic information the word embedding capture, we can build approximations to compare semantic information between phrases or even documents instead of words. In this project, we propose the use of the GloVe tool, presented by Stanford University, to train word embedding, use them to compare semantic differences between phrases and compare the accuracy of the system with previous leads to that alternative models were used, for instance, Word2Vec.

Key Words: GloVe word embedding; Bag of words; Semantic Similarity; Co-occurrence matrix; Value Stream Mapping(VSM).

1. INTRODUCTION

Computational Linguistics (CL) is the field of Computer Science that aims to model the human natural language in order to automatically treat digital text to accomplish an innumerable amount of tasks that are related to text processing, analysis, synthesis, and calculations of text characteristics that are involved in other problems, such as semantic parsing, text similarity, answer selection, etc. Generally, every problem is related to some sort of linguistic representation, ideally, digital text from the view of CL. Atomic pieces make up a text, known as words. Practically, a word is the linguistic unit, that generally possesses an intrinsic meaning (i.e. it expresses something) that grouped in conjunction with other words by following a set of grammatical rules, makes a sentence, building a more complex idea. Technically, in text, a word is a set of characters delimited by a blank space or a punctuation mark. As in every other Computer Science field, all data to be processed must first become encoded in order to be understandable by the computer. In the case of CL, there are several ways of encoding words, depending on several factors, that have relative advantages and disadvantages. For example, we can represent a text with a Bag of Words (BoW)

model: in this case, the words take the shape of a bit, that can have the value of 0 or 1 depending on the presence of that word in the text. Nevertheless, one of the potential downsides of the BoW model is that it does not capture the meaning of the words since it only takes into account if that word appears in a given text. Representation of words known as word embedding is the another example, more related to this project. Word embedding are n-dimensional vectors of real values that are built with a large corpus of plain text. One of the main characteristics of word embedding that make it special is the capability of keeping a relative meaning of the words, and that has opened up a whole world in text representation and processing. There are way more than two methods for estimating continuous representations of words; a couple of well-known classic examples are Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA). However, the most drawback of this ways is that the high process value, since they need to be calculable from an oversized corpus, and therefore the operations created so as to estimate the vectors are not scalable enough. That is the reason for the popularity of Word2Vec models. In this paper of 2013, Mikolov et al. presented a novel efficient way of calculating continuous word vector representations and developed software that implemented the models that are explained in their paper, giving to the scientific community a valuable tool to keep investigating on word embeddings. From that time, other efficient methods of unsupervisedly estimating word vectors have grown in the scientific community, and one of the most famous is GloVe, that stands for Global Vectors.

2. LITERATURE SURVEY

2.1 Survey on LSTM encoder for finding short text similarity

Lin Yao proposed an LSTM encoder for finding text similarity. The proposed work conduct experiments on 2 dataset: MSR Paraphrase Corpus dataset and Quora QR dataset. They give different lengths of short text. In training process, autoencoder uses inception module to get lot of features from multiple dimensions and it improves LSTM cell to know about the word sequence information of short texts

and choose to forget the information gained before. The vectors of short texts is the output of encoder. In evaluating stage cosine distance is used to calculate short text similarity. The proposed system requires only unlabelled short text data.

2.2 Survey on Semantic text similarity for English and Spanish text

Eneko Agirre proposed a system that includes two snippets of text. Semantic textual similarity (STS) captures the notion that some text are more similar than others by measuring the texts degree of semantic equivalence. STS aims to create a unified framework for combining several semantic components. The System includes three subtask i.e English, Spanish and interpretable subtask. The English subtask dataset consists of pairs of sentences from new headlines (HDL), image descriptions (Images), and a pair from a committed belief dataset. The Spanish subtask follows the same method like English subtask except that the similarity scores were adapted to fit a range from zero to four. This task introduces explanatory layer for finding the similarity. It contains a pair of sentences, system aligns the chunks across both the sentences and also for every alignment, it classify the type of relation and gives the corresponding similarity score.

2.3 Survey on open source framework for text similarity

Daniel Bar proposed a system that is designed a framework designed to streamline to the development of text similarity. For text processing, UIMA based pipeline is used in which the words are tokenized by using corpus reader and the data combined in different ways and is preprocessed in which the stop words and symbols are removed and abbreviation are expanded. Then the text similarity is computed among the given input texts and similarity score is given for each test. This is done by using various models such as greedy string tiling, Double metaphonic and explicit semantic analysis. With the help of the estimated similarity score, the post processing of the scores is done and the text similarity is detected.

2.4 Survey on Novel word embedding and translation-based language modeling for extractive speech summarization

Kuan -yu chen proposed a system that formulate a translation based language modeling framework for extractive speech summarization. Two sets of word

representations namely desired word representations and separate context word representations is used for embedding the words. The language modelling for extractive speech summarization for which each sentence of a spoken document, the sentences are selected based on corresponding generative probability. In this way the words from two different representations are embedded by using TBLM method.

2.5 Survey on Extensive feature extraction from word alignments for semantic textual similarity

Christian Hanig proposed an extensive feature extraction method for semantic textual similarity which focuses on categories of alignments such as named entities, temporal expressions, measurement expressions and dedicated negation handling. Here the word tokenization is done which is the splitting of words and word lemmatization is performed. After that elongation of words is replaced and the clitics are replaced. In named entities, the main name and surname are combined and aligned. Normalized temporal expressions aligns words with same time interval. Measurement expressions aligns words that express same absolute value. Arbitrary token sequence ignores case information, punctuations and symbols. Negations aligns the two words that has straight negative tokens each other and the remaining content words are grouped together. Then the features of the words are extracted by using defined score based on alignment process. 40 features were extracted and then it is divided into non- alignment features and alignment features. Then the extracted features is trained and tested against the English and Spanish data.

2.6 Survey on Probabilistic FastText for multisense word embeddings

Piotr Bojanowski proposed a system to enrich word vectors which is a morphological word representations. Given a word vocabulary of size W where a word is identified by its index w . here the morphology is modeled by considering subword units and the words are represented as the sum of its character n -grams. For a word at a particular position, both positive and negative samples are randomly selected. But for a context position, only one possible choice is given. The skipgram model ignores the internal structure of words by using a distinct vector representation of each word. A special boundary symbols $<$ and $>$ is included to distinguish prefixes and suffixes. Thus this model allows the representation of the words which also allows to represent rare words.

3. OVERVIEW OF THE PROPOSED SYSTEM

3.1 Problem Definition

The objective of this project is to determine and prove whether a system using word embeddings generated with GloVe can perform better than state-of-the-art systems that use the collection of models Word2Vec to build the word vector representations for their final use in the field of text similarity. We compare these two methods (GloVe and Word2Vec) in many ways in order to find which aspects of the word embeddings are totally different for the task of semantic text similarity. After analyzing the results, we also aim to use the currently generated word embeddings with GloVe in several different ways to improve the performance of our model.

3.2 System Model

Pre-trained word embeddings are a unit a staple in deep learning for NLP. The Renowned word2vec is the pioneer of word embeddings in mainstream deep learning. GloVe is another commonly used method of obtaining pre-trained embeddings. Unfortunately, there are very few practitioners that appears to truly perceive GloVe; several simply think about it "another word2vec". In reality, GloVe is a much more principled approach to word embeddings that provides deep insights into word embeddings in general.

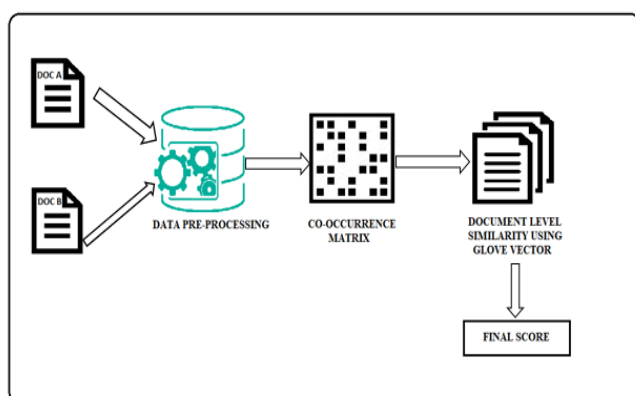


Fig- 1 Architecture Diagram for Short-text Semantic Similarity using GloVe word embedding

3.3 Modules Description

3.3.1 Text Pre-Processing

The 'Bag of Words' model was a very important insight that created NLP thrive. This model consists on receiving a listing of labelled text corpora, creating a word count calculate every

corpus and determining with what quantity frequency every word (or morpheme to be more precise) appears for every given label. After that, the Bayes' Theorem is applied on an unlabeled corpus to check that label (a sentiment analysis that labels between positive and negative, perhaps) it has a greater probability of belonging to, based on morpheme frequencies.

Even though good (>90%) check scores may be achieved with this methodology, it's a pair of problems:

1. Syntactic and semantic accuracy isn't as high as it should be because of the fact that context is king. For example; 'Chicago' means one thing and 'Bulls' means another, but 'Chicago Bulls' means a completely different thing. Counting word-frequencies doesn't take this into account.
2. For more practical use cases, we need to understand that data in real-life tends to be unlabeled, therefore passing from a supervised to an unsupervised learning method yields a greater utility.

3.3.2 Co-occurrence matrix

To build a co-occurrence matrix, GloVe additionally takes native context into account by computing the co-occurrence matrix employing a fixed window size (words are deemed to co-occur once they seem together within a fixed window). For instance, the sentence "The cat sat on the mat" with a window size of two would be converted to the co-occurrence matrix. Notice however the matrix is symmetric: this can be as a result of once the word "cat" seems within the context of "sat", the opposite (the word "sat" appearing in the context of "cat") also happens.

	the	cat	sat	on	mat
the	2	1	2	1	1
cat	1	1	1	1	0
sat	2	1	1	1	0
on	1	1	1	1	1
mat	1	0	0	1	1

Fig- 2 Co-occurrence matrix table

The important principle behind GloVe can be defined as follows: the co-occurrence ratios between two words in a context are strongly connected to meaning. This sounds

tough however the thought is admittedly easy. Take the words “ice” and “steam”, for instance. Ice and steam dissent in their state however they are similar in this they’re both forms of water. Therefore, we would expect words related to water (like “water” and “wet”) to appear equally in the context of “ice” and “steam”. In distinction, words like “cold” and “solid” would in all probability seem close to “ice” but however wouldn’t seem close to “steam”. The following table shows the particular statistics that show this intuition well:

Probability and Ratio	$k = \text{solid}$	$k = \text{gas}$	$k = \text{water}$	$k = \text{fashion}$
$P(k \text{ice})$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k \text{steam})$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k \text{ice})/P(k \text{steam})$	8.9	8.5×10^{-2}	1.36	0.96

Fig- 3 Probability and ratio table

The probabilities shown are basically just counts of how frequently the word k seems when the words “ice” and “steam” are in the context, where k refers to the words “solid”, “gas”, “water”, and “fashion”. As you can see, words that are related to the nature of “ice” and “steam” (“solid” and “gas” respectively) occur far more often with their corresponding words than the non-corresponding word. In distinction, words like “water” and “fashion” which are not particularly related to either have a probability ratio near 1. Note that the probability ratios are computed simply by using the co-occurrence matrix.

From here on, we’ll need a small of mathematical notation to create the reason easier. We’ll use X to refer to the co-occurrence matrix and X_{ij} to refer to the i, j th element in X which is equal to the number of times word j appears in the context of word i . We’ll also define $X_i = \sum_l X_{il}$ to refer to the total number of words that have appeared in the context of i .

3.3.3 Document level similarity using GloVe vector

Document vector- In the VSM approach a document is delineated as a vector in word space. An element in the vector could be a measure (simple frequency count, normalized count, tf-idf, etc..) of the importance of the corresponding word for that document. We have gone over the mechanics of this in nice detail and therefore the made analysis that this approach makes attainable in our earlier posts – Stacks of Documents and Bags of Words and Reduced Order Models for Documents.

If 2 documents contain nearly identical distribution of words, the vectors they furnish rise to in the word space would be lot of parallel than otherwise. So the cosine between the 2 vectors would be nearer to one. Making the belief that 2 documents with a same distribution of words are similar (which we all know to be an inexact statement!) the VSM approach find their similarity by taking cosine of the document vectors as the measure. Clearly documents that don’t share several words would be thought about dissimilar during this model – once again an inexact conclusion to create. Let us consider few examples so we are clear about this.

Example:

➤ Take the following three documents in a 7-dimensional word space:

- Word Space: [‘avoid’, ‘capital’, ‘france’, ‘japan’, ‘letters’, ‘paris’, ‘tokyo’]
- Doc1: Tokyo is the capital of Japan $\Rightarrow d_1 = [0, 1, 0, 1, 0, 0, 1]$
- Doc2: Paris is the capital of France $\Rightarrow d_2 = [0, 1, 1, 0, 0, 1, 0]$
- Doc3: Avoid capital letters $\Rightarrow d_3 = [1, 1, 0, 0, 1, 0, 0]$

A simple count primarily based VSM vector illustration of every document is shown above as well. The cosine similarity between any pair of these vectors is equal to $(0 + 1*1 + 0 + 0 + 0 + 0 + 0) / (3^{0.5} * 3^{0.5}) = 1/3.0$. The math is all correct however we’d have liked to possess gotten higher similarity between Doc1 & Doc2 so that we could put them together in a geography bucket while placing the third somewhere else. But that’s the character of bag-of-words approach to documents.

A simple count primarily based VSM vector illustration of every document is shown above as well. The cosine similarity between any pair of these vectors is equal to $(0 + 1*1 + 0 + 0 + 0 + 0 + 0) / (3^{0.5} * 3^{0.5}) = 1/3.0$. The math is all correct however we’d have liked to possess gotten higher similarity between Doc1 & Doc2 so that we could put them together in a geography bucket while placing the third somewhere else. But that’s the character of bag-of-words approach to documents.

- Take an another triplet of documents spanned by a 6-dimensional word space:

- Word Space: ['decent', 'good', 'honest', 'lies', 'person', 'tell']
- Doc1: Be honest and decent => $d_1 = [1, 0, 1, 0, 0, 0]$
- Doc2: Be a good person => $d_2 = [0, 1, 0, 0, 1, 0]$
- Doc3: Tell lies => $d_3 = [0, 0, 0, 1, 0, 1]$

The documents do not share any words so their dot products are all zero, indicating that they are all dissimilar. But again we would have hoped to have got Doc1 & Doc2 to be more similar with either being more dissimilar to Doc3. The upshot from the examples here is that the bag-of-words based document vectors for assessing similarity and thus classification thereof can be misleading.

Document Vector with Word Embeddings-In general the quantity of distinctive words in an exceedingly given document may be a little, little fraction of the entire number of distinctive words within the corpus. So the document vectors are sparse with more zeros than non-zeros. This is a tangle particularly for neural nets where the quantity of input layer neurons is the size of the incoming vector. This is why the embedded word vectors have become common.

The length of the trained word vectors p is in generally abundant, abundant smaller than the size of the corpus word space. So the document vectors where the words are replaced with their lower dimensional vectors are very shorter – thus providing computational benefits. For example, the twenty-news document repository has over sixty thousand unique words. If we use 300-dimensional word vectors (i.e. $p = 300$) for every word, we will cut down the quantity of input neurons by a factor of two hundred, making it much more competitive to use them in large NLP tasks.

Sparse to Dense Document Vectors- If each word in a document has a known illustration in the same p -dimensional space, then the bag-of-words document vector is portrayed as a vector in that same p -dimensional space. We are merely combining the bag-of-words approach with word embeddings to come back up with lower dimensional, dense representations of documents. Let us take the instance 1.2 from earlier where we now also have a $px1$ vector representation for each of the six words 'decent', 'good', 'honest', 'lies', 'person', 'tell' making up the corpus vocabulary. The original 6-dimensional document vector d_i can be rewritten in p -space as a $px1$ vector d_i^*

$$\begin{aligned} \vec{d}_1^* &= 1 \cdot \vec{\text{decent}} + 0 \cdot \vec{\text{good}} + 1 \cdot \vec{\text{honest}} + 0 \cdot \vec{\text{lies}} + 0 \cdot \vec{\text{person}} + 0 \cdot \vec{\text{tell}} \\ &= \begin{bmatrix} \vec{\text{decent}} & \vec{\text{good}} & \vec{\text{honest}} & \vec{\text{lies}} & \vec{\text{person}} & \vec{\text{tell}} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ &\quad \text{p} \times 6 \text{ word-vector matrix } \underline{W} \quad \text{6} \times 1 \end{aligned}$$

Fig- 4 Vector representation

Generally, with ' n ' words and ' m ' documents we can directly extend the above. First we tend to get word vectors for every of these n words, so giving us the pxn word-vector matrix \underline{W} . The i^{th} document with a vector illustration d_i in the standard n -word space is remodelled to the vector d_i^* in the fake p -word-space with:

$$\vec{d}_i^* = \underline{W} \vec{d}_i \quad \forall i : 1 \text{ through } n$$

While it is a computational advantage to have shorter document vectors, we'd like to create positive that the documents haven't lost their meaning and relationships within the method

3.4 Input and Output

Input: Two short text documents.

Output: It gives similarity score by comparing two short text documents, dataset used and the parameters used for evaluation.

4. EXPERIMENTAL RESULTS AND EVALUATION

4.1 Simulation Environment

The implementation of our proposed system was carried out using the Python 3.5 software running on a personal computer with a 2.07 GHz Intel (R) Core (TM) I3 CPU, 4 GB RAM and Windows 10 as the operating system.

4.2 Result Analysis

In this model, the system compare the GloVe word embedding with existing model of LSTM encoder and it proves it is better than the existing.

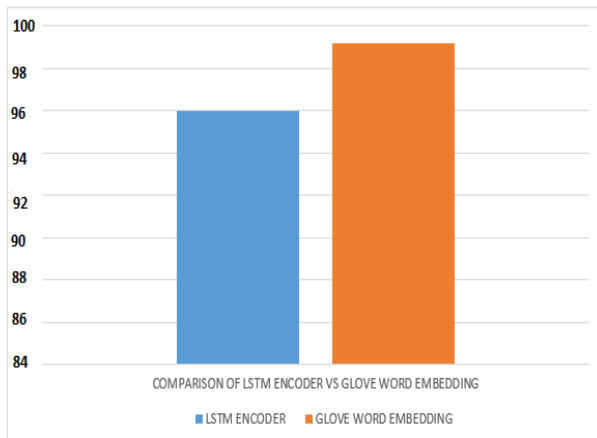


Fig- 5 Comparison of LSTM Encoder with GloVe word embedding

5. CONCLUSION AND FUTURE ENHANCEMENT

Terms of the loss measure between the particular ascertained frequencies and the expected frequencies. Whereas GloVe uses the log mean square error between the expected (unnormalized) probabilities and the actual observed (unnormalized) probabilities, word2vec uses the cross-entropy loss over the normalized probabilities. The GloVe paper argues that log mean squared error is best than cross-entropy because cross entropy tends to place an excessive amount of weight on the long tails. In reality, this is often most likely an issue that desires additional analysis. The essential thing to grasp is that word2vec and GloVe are literally virtually mathematically the same, despite having entirely totally different derivations.

In the future work, we are looking forward to try applying the proposed method on audio and video. Also, we are looking forward to enhance the proposed method to make the similarity rate higher than it while keeping the same GloVe word embedding or higher.

REFERENCES

- [1] Lin Yao, Zhengyu Pan, and Huansheng Ning, "Unlabeled Short Text Similarity With LSTM Encoder", IEEE Trans Special Section On Special Section On Advanced Big Data Analysis For Vehicular Social Networks, 2018.
- [2] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada

- Mihalcea, et al., Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability., SemEval@ NAACL-HLT, 2015, pp. 252-263.
- [3] Daniel Bär, Torsten Zesch, and Iryna Gurevych, Dkpro similarity: An open source framework for text similarity., ACL (Conference System Demonstrations), 2013, pp. 121-126.
- [4] Kuan-Yu Chen, Shih-Hung Liu, Berlin Chen, Hsin-MinWang, and Hsin-Hsi Chen, Novel word embedding and translation-based language modeling for extractive speech summarization, Proceedings of the 2016 ACM on Multimedia Conference, ACM, 2016, pp. 377-381.
- [5] Christian Hnig, Robert Remus, and Xose De La Puente, Exb themis: Extensive feature extraction from word alignments for semantic textual similarity., SemEval@ NAACL-HLT, 2015, pp. 264-268.
- [6] Athiwaratkun, B., Wilson, A., & Anandkumar, A. (2018). Probabilistic FastText for multisense word embeddings. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1-11. Association for Computational Linguistics.
- [7] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association of Computational Linguistics, 5 (1), 135-146.
- [8] Gamallo, P., & Pereira-Fari~na, M. (2017). Compositional semantics using feature-based models from wordnet. In Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications, pp. 1-11, Valencia, Spain. Association for Computational Linguistics.
- [9] Khodak, M., Risteski, A., Fellbaum, C., & Arora, S. (2017). Automated WordNet construction using word embeddings. In Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications, pp. 12-23.
- [10] Lee, G.-H., & Chen, Y.-N. (2017). Muse: Modularizing unsupervised sense embeddings. In Proceedings of EMNLP, Copenhagen, Denmark.