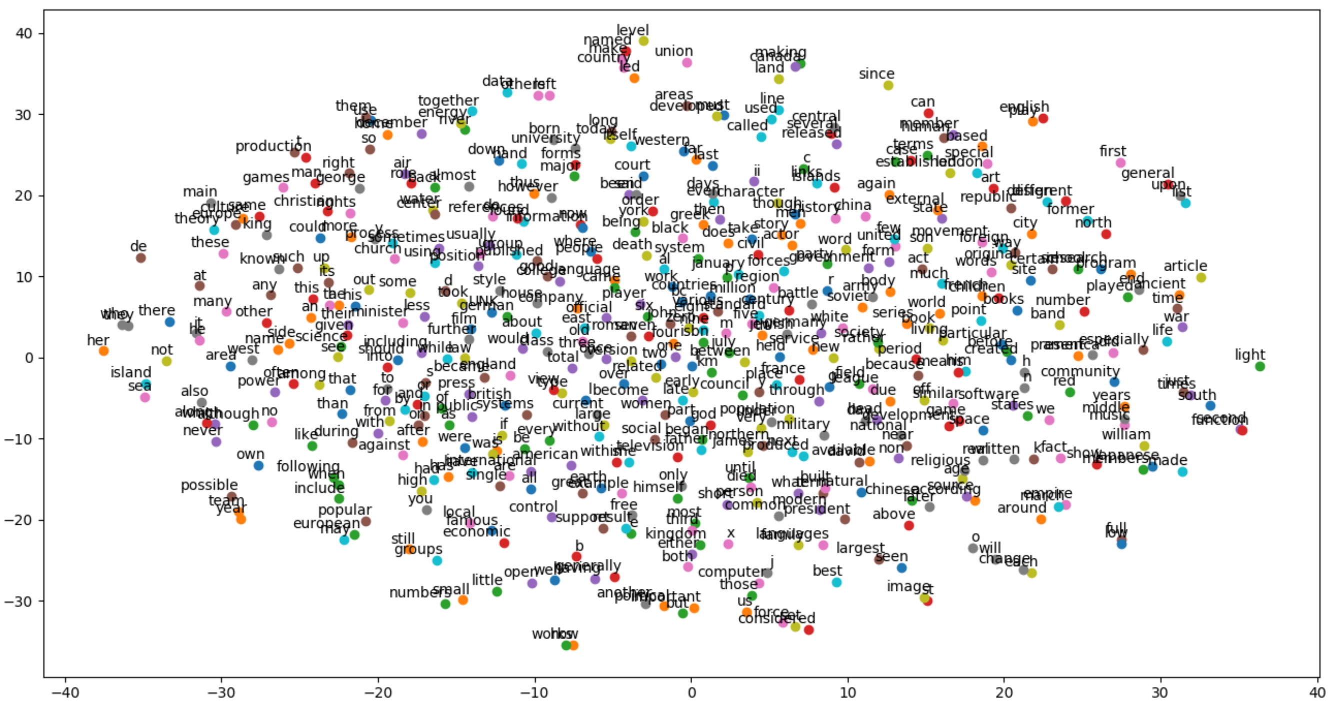
Kashyap Kathrani

9 Followers About Follow

All about Embeddings

Kashyap Kathrani May 18, 2020 · 8 min read



The computers can't understand the text the way we do, but they are good at processing numerical data. So, we need a model which can convert the textual information into numerical form. Hence, Text Embeddings are building blocks for any Natural Language Processing based system.

Embedding)

Word Embedding

First let's begin with word embedding, which is n-dimensional vector space representation of words such that semantically similar words (for instance, "boat" — "ship") or semantically related words (for instance, "boat" — "water") are closer in the vector space depending on the training data.

For capturing the semantic relatedness, the documents are used as context and while for capturing semantic similarity, the words are used as context.

The most widely used for word embedding models are word2vec and GloVe both of which are based on unsupervised learning.

Word2Vec

Word2Vec is basically a predictive embedding model. It mainly uses two types of architecture to produce vector representation of words

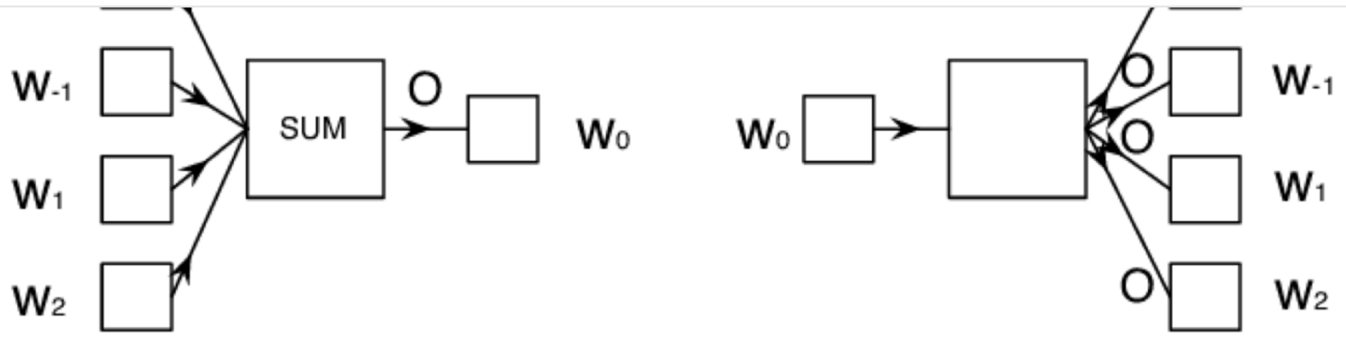
1. Continuous Bag-of-Words (CBOW)

In this architecture, the model predicts which is the most likely word in the given context. So, the words which have equal likelihood of appearing are considered as the similar and hence occur closer in the dimension space.

Suppose in a sentence we replace 'boat' with 'ship', then the model predicts the probability for both and if turns out to be similar then we can consider that the words are similar.

2. Skip-gram

This architecture is similar to that of CBOW, but instead the model works the other way around. The model predicts the context using the given word.



Architectures of Word2Vec

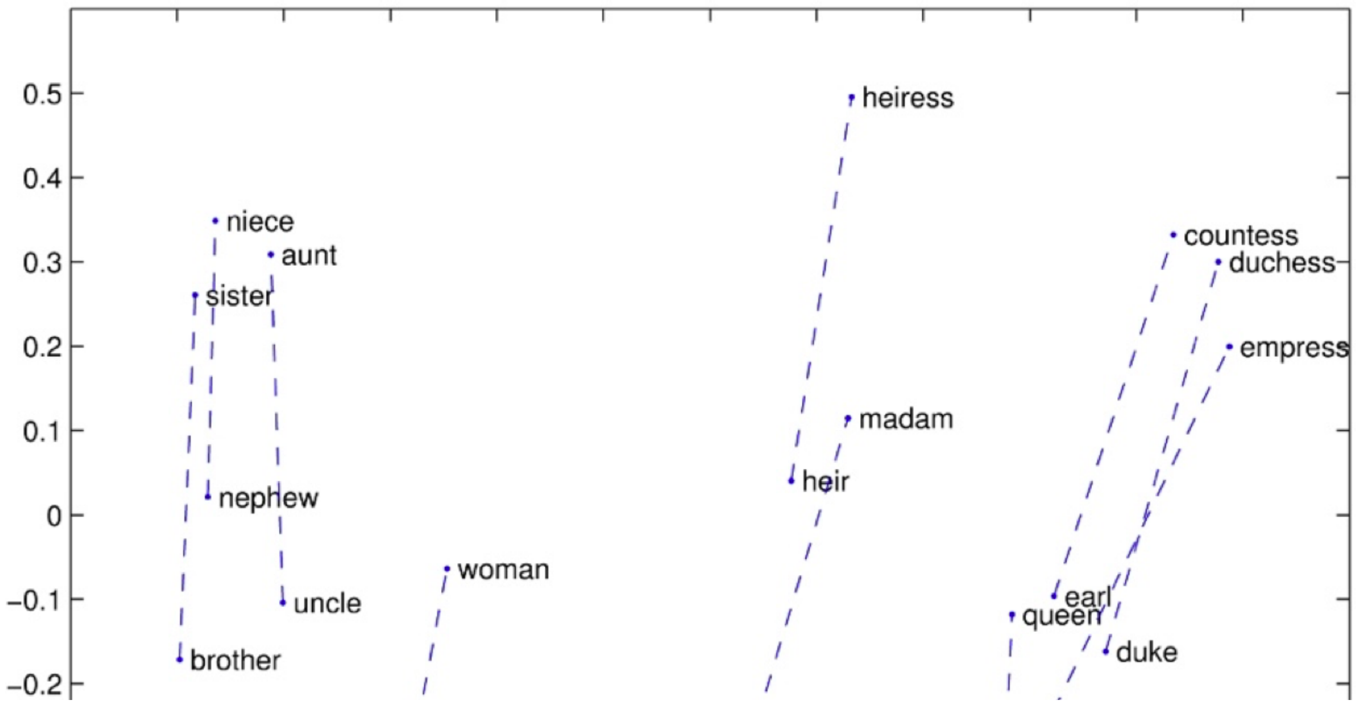
Research Paper

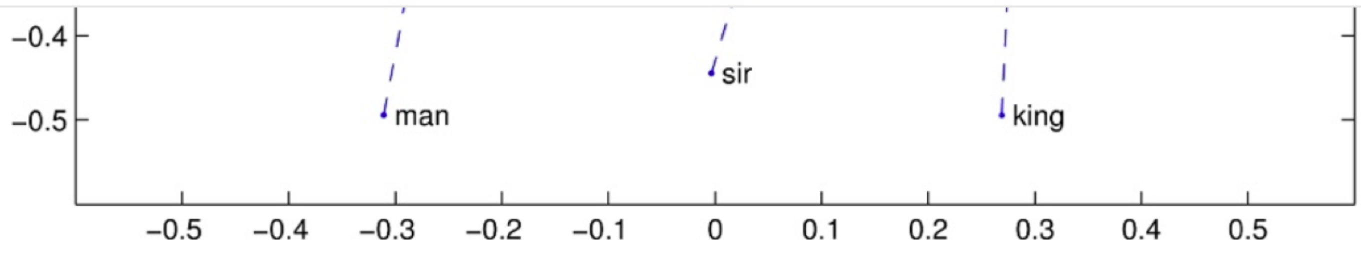
Implementation

GloVe

Both the architecture of the Word2Vec are the predictive ones and also ignores the fact that some context words occurs more often than others and also they only take into consideration the local context and hence failing to capture the global context.

The GloVe model is trained on aggregated global word to word co-occurrence matrix from a given text collection of text documents. This co-occurrence matrix is decomposed to form denser and expressive vector representation.





Vector Representation of GloVe

The above figure represents some of words after embedding in the vector space. Here, we can see the pairs formed of man and woman, queen and king, uncle and aunt and other pairs. So, it's able to differentiate the concept of sex or gender.

Research Paper

Implementation

FastText

One of the main disadvantages of Word2Vec and GloVe embedding is that they are unable to encode unknown or out-of-vocabulary words.

So, to deal with this problem Facebook proposed a model FastText. It is an extension to Word2Vec and follows the same Skip-gram and CBOW model. but unlike Word2Vec which feeds whole words into the neural network, FastText first breaks the words into several sub-words (or n-grams) and then feed them into the neural network.

For example, if the value for n is 3 and the word is '*apple*' then tri-gram will be ['<ap', 'app', 'ppl', 'ple', 'le>'] and its word embedding will be sum of vector representation of these tri-grams. Here, the hyper-parameters '*minn*' and '*maxn*' are considered as 3 and the characters '<' and '>' represents start and end of the word.

So, using this methodology unknown words can be represented in vector form as it has high probability that its n-grams are also present in other words.

Research Paper

Implementation

the approach is not accurate enough. As, they don't take into consideration the order of words in which they appear which leads to loss of syntactic and semantic understanding of the sentence.

For example, "You are going there to teach not play." And "You are going there to play not teach." Both of these sentences will have same representation in the vector space but they don't mean the same.

Also, the word embedding model cannot give satisfactory results on large amount of text data, as same word may different meaning in different sentence depending on the context of the sentence.

For example, "I have scuba diving in my **bucket** list." And "There is a **bucket** filled with drinking water." In both the sentences, the word "bucket" has different meanings.

So, we require a kind of representation which can retain the contextual meaning of the word present in a sentence.

Sentence Embedding

Sentence embedding are similar to the word embedding but instead of words, they encode whole sentence into vector representation. The obtained vector representation retains good properties by inheriting these features from underlying word embedding.

A simple way of obtaining sentence embedding is by averaging the word embeddings of the all the words present in the sentence. But they are not accurate enough.

Some of the state-of-the-art models for sentence embedding are ELMo InferSent and SBERT

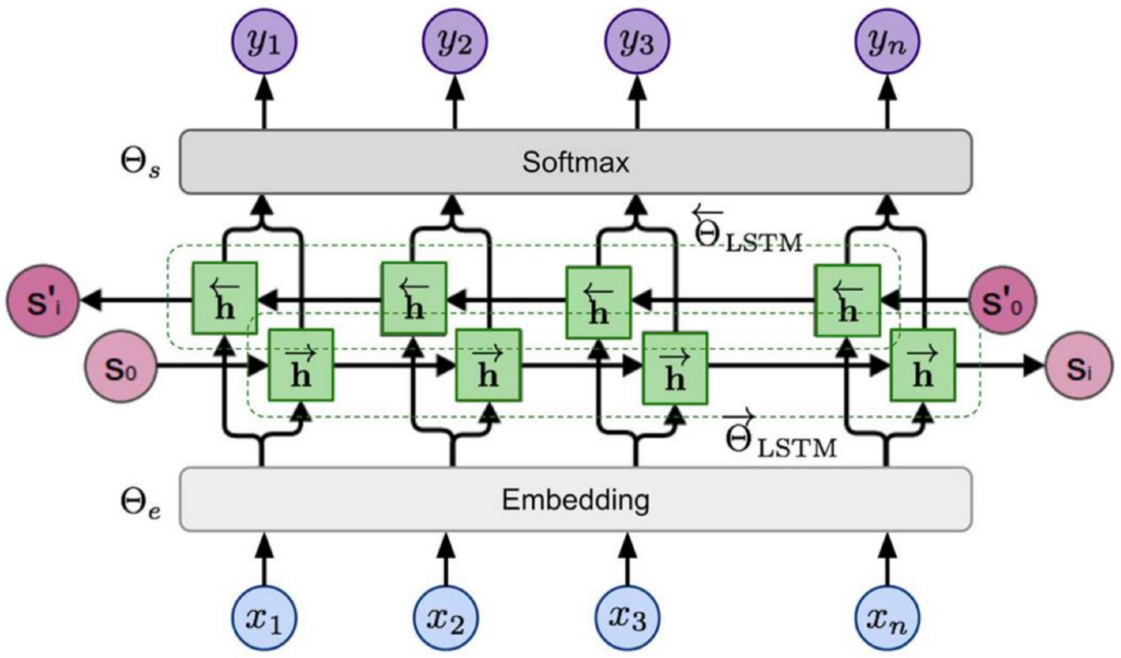
ELMo

ELMo which stands for Embedding from Language Model, is developed by ALLenNLP and it uses bi-directional deep LSTM network for producing vector representation. ELMo considers words within which context they have been used rather than creating dictionary of words with its vector form.

Like word2vec model, ELMo can also predict the next probable word in the sentence. And hence, if the model is trained on huge dataset set then it can be aware of the language pattern.

As the ELMo uses bidirectional LSTM it can get understanding of both next and previous word in the sentence.

The model consists of 2-layer of bi-directional LSTM and between both the layers, there is a residual connection which without any non-linear activation function allows gradient to traverse in the network. And this allows deep models to be trained effectively.



Architectures of ELMo

> *Though the ELMo model is built for sentence embedding but it can also produce vector representation of characters or words or paragraphs. All these types of embedding are created at runtime.*

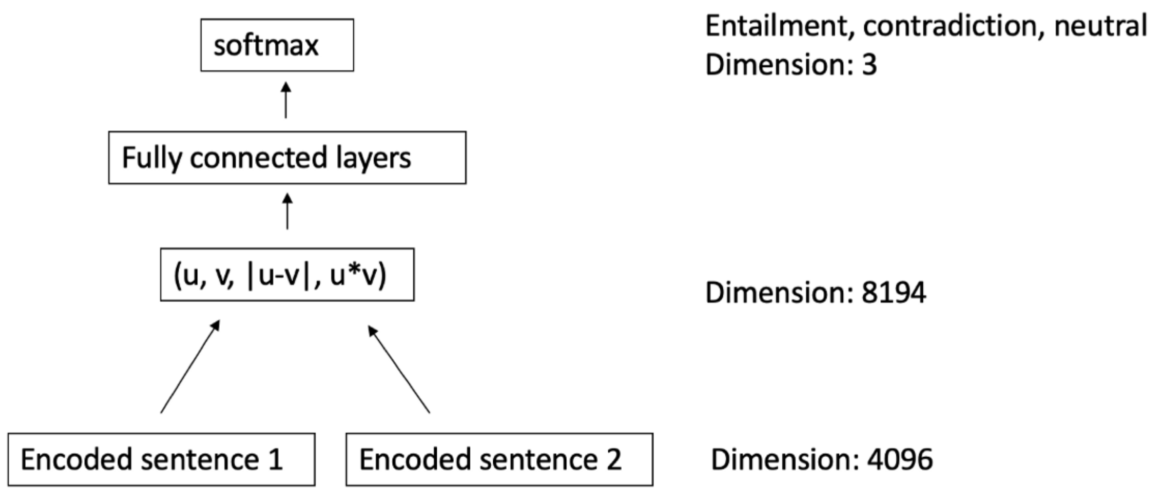
[Research Paper]

[Implementation]

the best current universal sentence encoding methods. As the InferSent encodes whole sentence instead of encoding each word individually, preserving semantic understanding of the text, outperforming existing approaches such as Bag of Words and Word Embedding.

InferSent was developed by Facebook and it is supervised model trained on natural language inference task which also performs well on other NLP tasks. The sentences are encoded into 4096-dimension vector representation.

The training of model is done on Stanford Natural Language Inference (SNLI) dataset. This dataset is labelled and written by human for around 500K pairs of sentences. The relationship between each pair of sentence may be: contradiction, entailment, and neutral.



Architectures of InferSent

From the output of each pair of sentences, basic features are extracted and then passed on to fully connected layers. This is done because sentence encoding generated should not be dependent on the context and should apply to all the types of sentences.

The infersent model was tried on 7 different types of architecture for standard recurrent Encoders with:

2. GRU

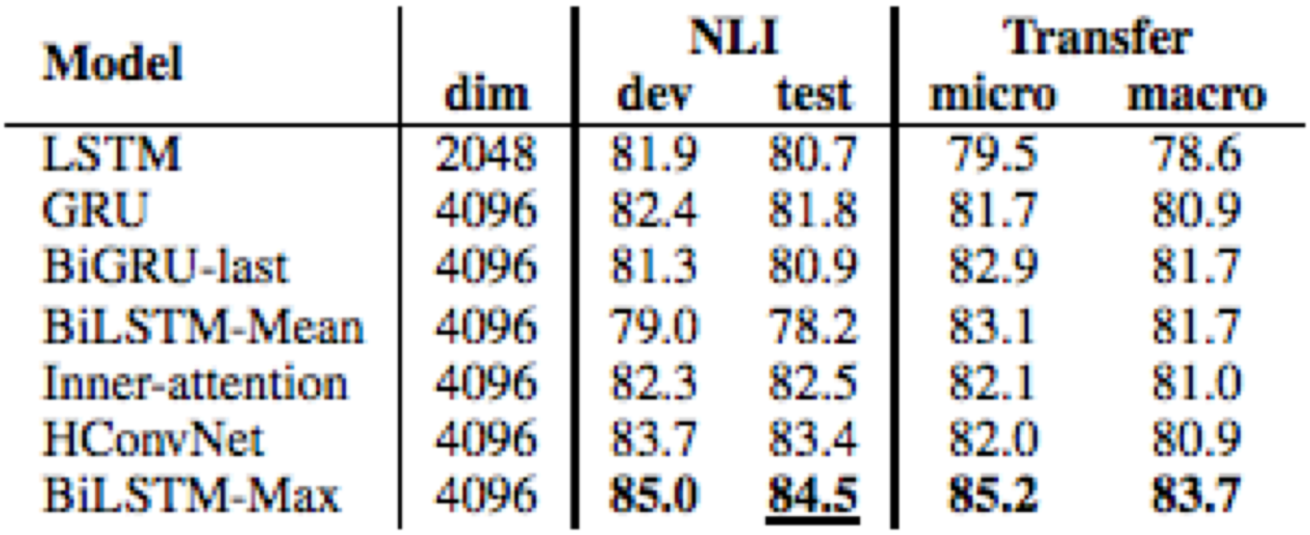
3. Bi-directional GRU (BiGRU)

4. Bi-directional LSTMs (BiLSTM) with mean pooling

5. Bi-directional LSTMs (BiLSTM) with max pooling

6. Self-attentive Network

7. Hierarchical convolutional networks



Model	dim	NLI dev	NLI test	Transfer micro	Transfer macro
LSTM	2048	81.9	80.7	79.5	78.6
GRU	4096	82.4	81.8	81.7	80.9
BiGRU-last	4096	81.3	80.9	82.9	81.7
BiLSTM-Mean	4096	79.0	78.2	83.1	81.7
Inner-attention	4096	82.3	82.5	82.1	81.0
HConvNet	4096	83.7	83.4	82.0	80.9
BiLSTM-Max	4096	**85.0**	**84.5**	**85.2**	**83.7**

Performance of InferSent using various models

After testing performance of all the architectures on SNLI and transfer tasks, it was found out that the Bi-directional LSTMs (BiLSTM) with max pooling which uses GloVe vector as base features. The forward and backward LSTM runs on GloVe vectors and applies max-pooling after concatenating the hidden state of each word.

Research Paper

Implementation

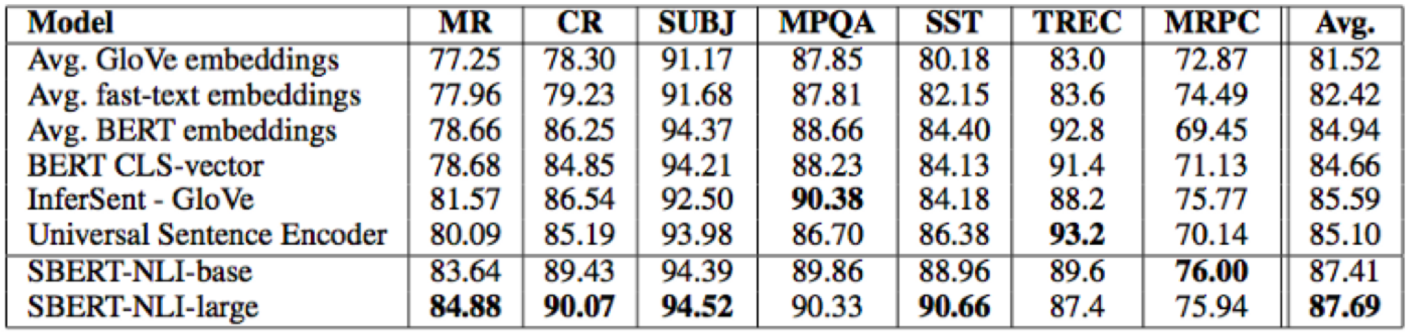
Sentence-BERT

10,000 sentences) as it requires that both sentences are fed into the network and this increases the computation by huge factor.

So, Sentence-BERT is modification of the BERT model which uses siamese and triplet network structures and adds a pooling operation to the output of BERT to obtain fix-sized semantically meaningful sentence embeddings. These generated sentence embedding can further be used for sentence similarity comparison (using cosine similarity), clustering and semantic search.

Research Paper

Implementation



Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	Avg.
Avg. GloVe embeddings	77.25	78.30	91.17	87.85	80.18	83.0	72.87	81.52
Avg. fast-text embeddings	77.96	79.23	91.68	87.81	82.15	83.6	74.49	82.42
Avg. BERT embeddings	78.66	86.25	94.37	88.66	84.40	92.8	69.45	84.94
BERT CLS-vector	78.68	84.85	94.21	88.23	84.13	91.4	71.13	84.66
InferSent - GloVe	81.57	86.54	92.50	**90.38**	84.18	88.2	75.77	85.59
Universal Sentence Encoder	80.09	85.19	93.98	86.70	86.38	**93.2**	70.14	85.10
SBERT-NLI-base	83.64	89.43	94.39	89.86	88.96	89.6	**76.00**	87.41
SBERT-NLI-large	**84.88**	**90.07**	**94.52**	90.33	**90.66**	87.4	75.94	**87.69**

Evaluation of embedding models using the SentEval toolkit | Source

The above table shows evaluation of different sentence embedding models using SentEval. SentEval is a tool-kit for evaluating the quality of sentence embedding created. It evaluates embedding based on 17 different downstream NLP tasks such as Movie Review (MR), Product Review (CR), sentiment analysis (SST), question-type classification (TREC).

Each embedding method has its own merits and demerits. Until now, there is no embedding model made which can perform well on every downstream NLP tasks. So, one should choose a embedding model keeping in mind the task to be performed using it.

for finding out embedding model suitable for your task.

· · ·

So, this was the gist of state-of-the-art embedding models. I hope that I have helped somebody understand these models. And if I've made any mistake, please let me know.

Thank you for reading!

Word Embeddings Sentence Embedding Infersent NLP Word2vec