**Index**

1. **Word Embeddings**
2. **Neural network Architecture (Bi-LSTMs / GRU / etc)**
3. **Regularization**
4. **Mean Teachers**
5. **Virtual Adversarial Training**
6. **Optimization**
7. **Basic Model Development ( until 30th April ).**

**Development**

1. **BERT | GPT2 | EIMO**
2. **Bad Examples via GANs**

**To Do**

1. **Gensim custom model training for the word vectors.**
2. **Dataset pre processing**
3. **Save the embedding layer.**

**Word embedding** has a core idea to provide mathematical representations of words and we can apply some linear function on them.

Applications:

1. Similarity between words/document.
2. Semantic Relationship
3. Recommendation

**Pre requisites:**

1. Bag of Words : Frequency count distribution.
2. TF-IDF : logarithmic scaled inverse fraction of documents containing the term.
   1. tf(t,d) -> count of t in d | idf -> how common is t across all d
3. SIF : Weighted average of WE -> a/(a+p(w)) | a=parameter ; p(w)= frequency of word in corpus
4. Word2vec : is a shallow NN with one hidden layer (d dimension) and optimization functions as Negative Sampling and Hierarchical Softmax.
5. GloVe : aggregates the global word-word co-occurrence matrix from a given corpus. Sometimes gives better results than skip-grams. Selection depends on the user.

Note : Although the NN hidden layer’s learnt features for a word is still a mystery, they do not need to be known as long as the information is embedded consistently by the same network. However, they are somewhat of similar fashion to traditional approaches.

**BOW**: Since words/tokens are discrete or symbolic representations. The semantic relationship between them is not easy to compute with earlier approaches. Due to its basic idea of using a histogram of token frequencies and then normalizing.

**TF-IDF**: This approach kind of provides a weighting to above count scores, but still fails to capture the semantic relationship. Examples to & from occur many times but no information is provided relating to them.

**Word2Vec**: has two unsupervised model variants/algorithms

1. CBoW : Predict a target word based on context window words.
2. Skip Grams (Exact Reverse) : Predict the context based on a target word. This approach has proven to give better results.

The main principle behind word2vec is distributional hypothesis -> Words that occur in the same context tend to have similar meanings. https://arxiv.org/pdf/1310.4546.pdf

Now to represent a sentence we will take the mean of all the word embeddings from the model.

Limitations:

* Words not present are usually ignored.
* Uni-gram natural behaviour.

**Comparison between Glove and W2V** :

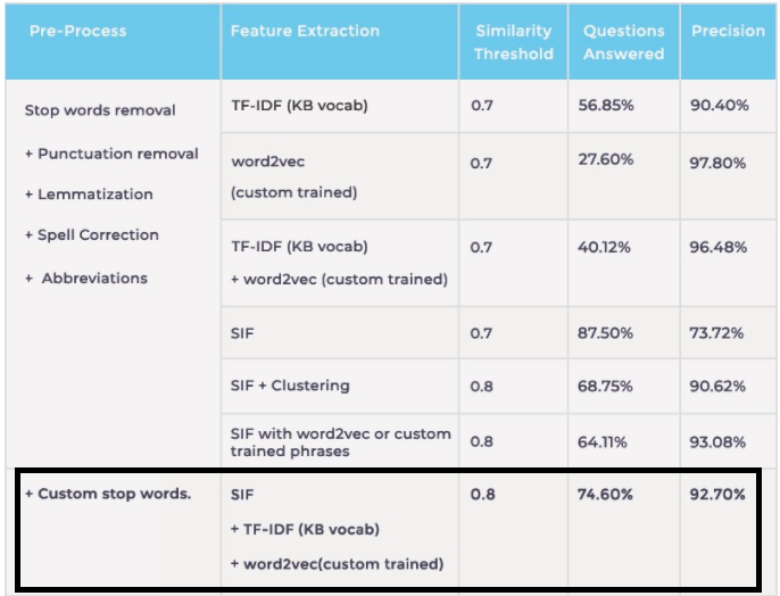
1. Neural Networks:

Glove doesn’t use neural networks and the loss function is the difference between the product of word embeddings and log of the probability of co-occurrence. We try to minimize this and use SGD but solve it as we solve a linear regression.

While in w2v, a neural net is used and we train the word on its context(skip-grams) or train the context on the word(CBoW) using one hidden layer.

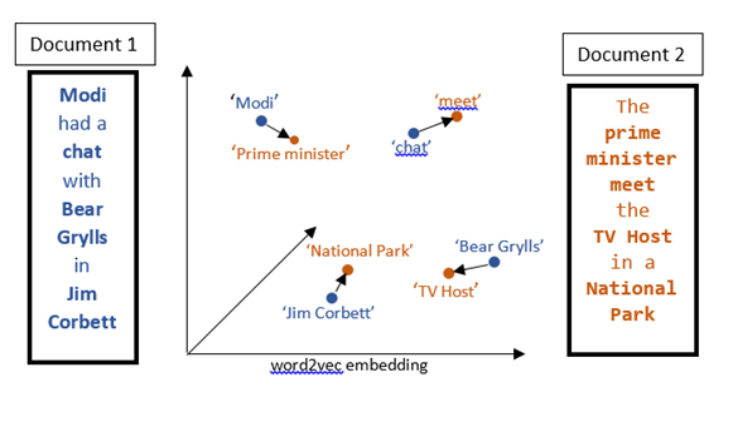
1. Global Information : w2v has no explicit information by default.

However, GloVe creates a global co-occurrence matrix by estimating the probability a given word will co-occur with other words. This fact makes it work better. However, in practical sense, performance is found similar.



**Statistical Measures :**

* Cosine : dot products of two vectors
* Euclidian
* Jaccard
* Word Mover’s Distance : WMD incorporates the knowledge encoded in w2v/Glive space and leads to high retrieval accuracy. Try avoiding if word ordering is very frequent. https://towardsdatascience.com/word-movers-distance-for-text-similarity-7492aeca71b0



Gensim : Popular library for word embedding operations.

Pre-processing:

* Stop words
* Non ASCII characters
* HTML Tags
* Punctuations
* Lemmatization : morphological variants of root/base word aka lemma.
* Spell Correction
* Abbreviations