DATA SCIENCE INTERVIEW PREPARATION (30 Days of Interview Preparation)

DAY 19

Q1. What is LSI(Latent Semantic Indexing)?

Answer:

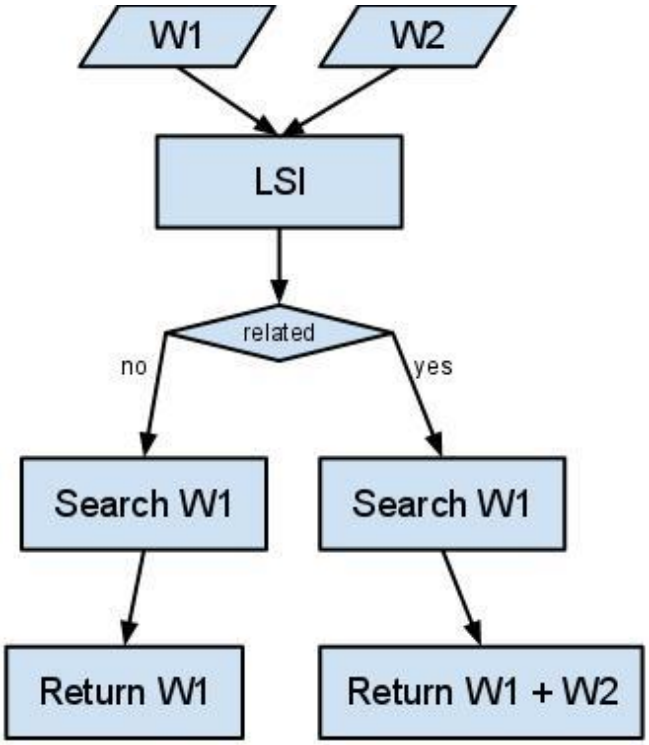
Latent Semantic Indexing (LSI): It is an indexing and retrieval method that uses a mathematical technique called SVD(Singular value decomposition) to find patterns in relationships between terms and concepts contained in an unstructured collection of text. It is based on the principle that words that are used in the same contexts tend to have similar meanings.

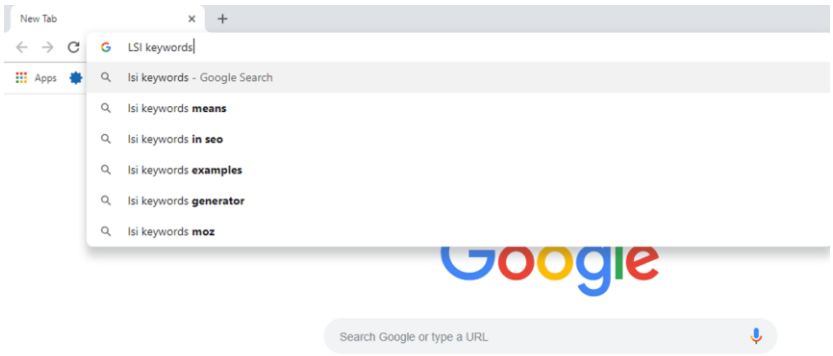
For example, Tiger and Woods are associated with men instead of an animal, and a Wood, Parris, and Hilton are associated with the singer.

Example:

If you use LSI to index a collection of articles and the words “fan” and “regulator” appear together frequently enough, the search algorithm would notice that the two terms are semantically close. A search for “fan” will, therefore, return a set of items containing that phrase, but also items that contain just the word “regulator”. It doesn't understand word distance, but by examining a sufficient number of documents, it only knows the two terms are interrelated. It then uses that information to provide an expanded set of results with better recall than an understandable keyword search.

The diagram below describes the effect between LSI and keyword searches. W stands for a document.





Q2. What is Named Entity Recognition? And tell some use cases of NER?

Answer:

Named-entity recognition (NER): It is also known as entity extraction, and entity identification is a subtask of information extraction that explore to locate and classify atomic elements in text into predefined categories like the names of persons, organizations, places, expressions of times, quantities, monetary values, percentages and more.

In each text document, particular terms represent specific entities that are more informative and have a different context. These entities are called named entities, which more accurately refer to conditions that represent real-world objects like people, places, organizations or institutions, and so on, which are often expressed by proper names. The naive approach could be to find these by having a look at the noun phrases in text documents. It also is known as entity chunking/extraction, which is a popular technique used in information extraction to analyze and segment the named entities and categorize or classify them under various predefined classes.

Named Entity Recognition use-case

- **Classifying content for news providers-**
  NER can automatically scan entire articles and reveal which are the significant people, organizations, and places discussed in them. Knowing the relevant tags for each item helps in automatically categorizing the articles in defined hierarchies and enable smooth content discovery.
- **Customer Support:**
  Let's say we are handling the customer support department of an electronics store with multiple branches worldwide; we go through a number mentions in our customers' feedback. Such as this for instance.

Now, if we pass it through the Named Entity Recognition API, it pulls out the entities Bangalore (location) and Fitbit (Product). This can be then used to categorize the complaint and assign it to the relevant department within the organization that should be handling this.

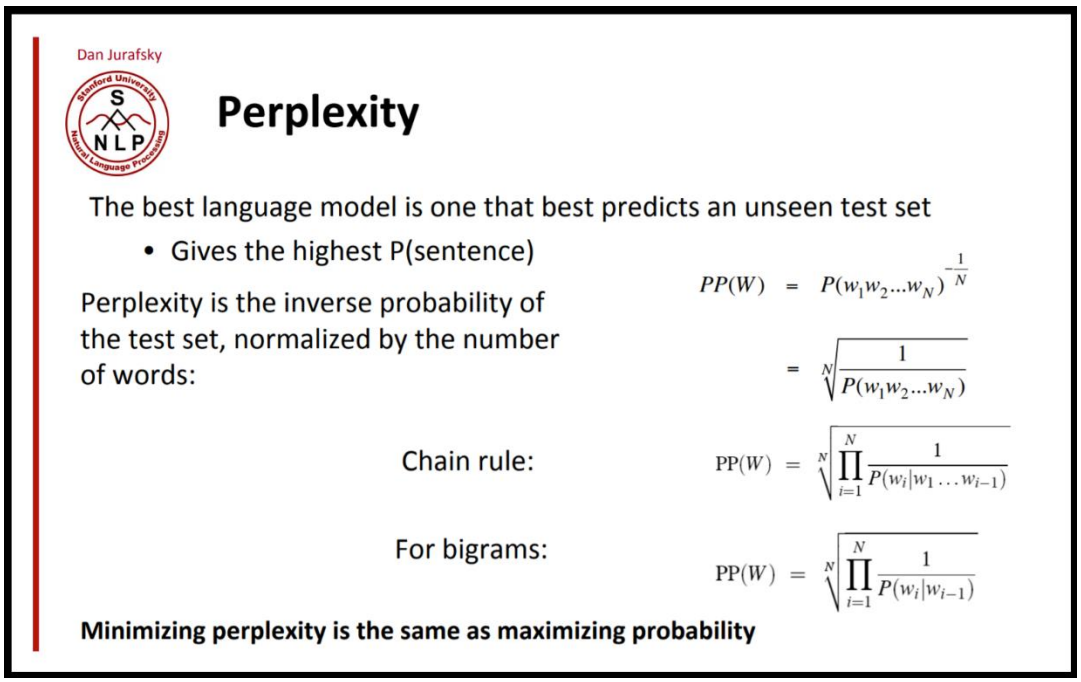
Figure 1: An example of NER application on an example text

Q3. What is perplexity?

Answer:

Perplexity: It is a measurement of how well a probability model predicts a sample. In the context of NLP, perplexity(Confusion) is one way to evaluate language models.

The term perplexity has three closely related meanings. It is a measure of how easy a probability distribution is to predict. It is a measure of how variable a prediction model is. And It is a measure of prediction error. The third meaning of perplexity is calculated slightly differently, but all three have the same fundamental idea.



Q4. What is the language model?

Answer:

Language Modelling (LM): It is one of the essential parts of modern NLP. There are many sorts of applications for Language Modelling, like Machine Translation, Spell Correction Speech Recognition, Summarization, Question Answering, Sentiment analysis, etc. Each of those tasks requires the use of the language model. The language model is needed to represent the text to a form understandable from the machine point of view.

The statistical language model is a probability distribution over a series of words. Given such a series, say of length m, it assigns a probability to the whole series.

It provides context to distinguish between phrases and words that sounds are similar. For example, in American English, the phrases " wreck a nice beach " and "recognize speech" sound alike but mean different things.

Data sparsity is a significant problem in building language models. Most possible word sequences are not noticed in training. One solution is to make the inference that the probability of a word only depends on the previous n words. This is called as an n-gram model or unigram model when $n = 1$. The unigram model is also known as the bag of words model.

How does this Language Model help in NLP Tasks?

The probabilities restoration by a language model is most useful to compare the likelihood that different sentences are "good sentences." This was useful in many practical tasks, for example:

Spell checking: You observe a word that is not identified as a known word as part of a sentence. Using the edit distance algorithm, we find the closest known words to the unknown words. These are the candidate corrections. For example, we observe the word "wurd" in the context of the sentence, "I like to write this wurd." The candidate corrections are ["word", "weird", "wind"]. How can we select among these candidates the most likely correction for the suspected error "weird"?

Automatic Speech Recognition: we receive as input a string of phonemes; a first model predicts for sub-sequences of the stream of phonemes candidate words; the language model helps in ranking the most likely sequence of words compatible with the candidate words produced by the acoustic model.

Machine Translation: each word from the source language is mapped to multiple candidate words in the target language; the language model in the target language can rank the most likely sequence of candidate target words.

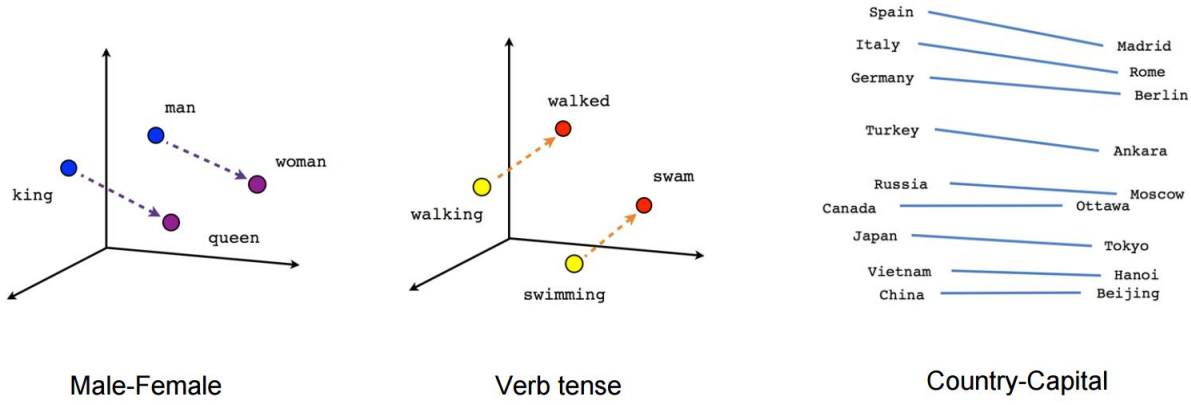
Q5. What is Word Embedding?

Answer:

A word embedding is a learned representation for text where words that have the same meaning have a similar observation.

It is basically a form of word representation that bridges the human understanding of language to that of a machine. Word embeddings divide representations of text in an n-dimensional space. These are essential for solving most NLP problems.

And the other point worth considering is how we obtain word embeddings as no two sets of word embeddings are similar. Word embeddings aren't random; they're developed by training the neural network. A recent powerful word embedding usage comes from Google named Word2Vec, which is trained by predicting several words that appear next to other words in a language. For example, the word "cat", the neural network would predict the words like "kitten" and "feline." This intuition of words comes out "near" each other allows us to place them in vector space.



Male-Female | Verb tense | Country-Capital

Q6. Do you have an idea about fastText?

Answer:

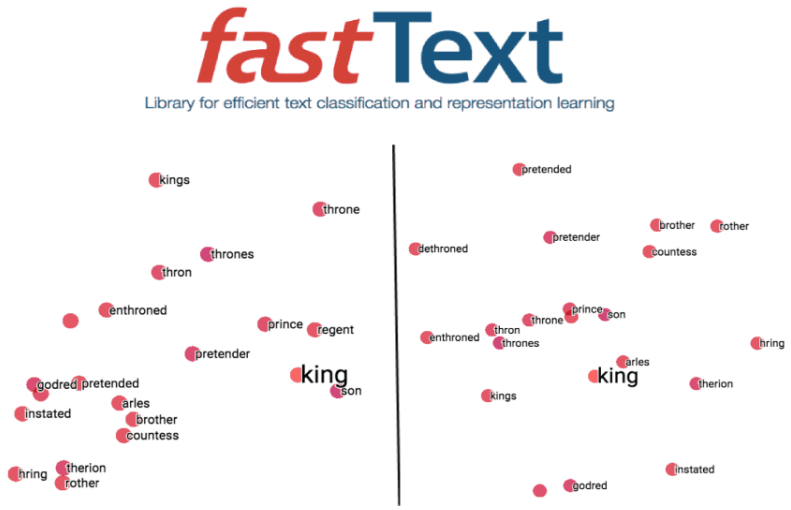
fastText: It is another word embedding method that is an extension of the word2vec model. Alternatively, learning vectors for words directly. It represents each word as an n-gram of characters. So, for example, take the word, “artificial” with n=3, the fastText representation of this word is <ar, art, rti, tif, ifi, fic, ici, ial, al>, where the angular brackets indicate the beginning and end of the word.

This helps to capture the meaning of shorter words and grant the embeddings to understand prefixes and suffixes. Once the word has been showed using character skip-grams, a n-gram model is trained to learn the embeddings. This model is acknowledged to be a bag of words model with a sliding

window over a word because no internal structure of the word is taken into account. As long as the characters are within this window, the order of the n-grams doesn't matter.

fastText works well with rare words. So even if a word wasn't seen during training, it can be broken down into n-grams to get its embeddings.

Word2vec and GloVe both fail to provide any vector representation for words that are not in the model dictionary. This is a huge advantage of this method.



Q7. What is GloVe?

Answer:

GloVe(global vectors) is for word representation. GloVe is an unsupervised learning algorithm developed by Stanford for achieving word embeddings by aggregating a global word-word co-occurrence matrix from a corpus. The resulting embeddings show interesting linear substructures of the word in vector space.

The GloVe model produces a vector space with meaningful substructure, as evidenced by its performance of 75% on a new word analogy task. It also outperforms related models on similarity tasks and named entity recognition.

How GloVe find meaning in statistics?

Produces a vector space with meaningful substructure, as evidenced by its performance of 75% on a new word analogy task. It also outperforms related models on similarity tasks and named entity recognition.

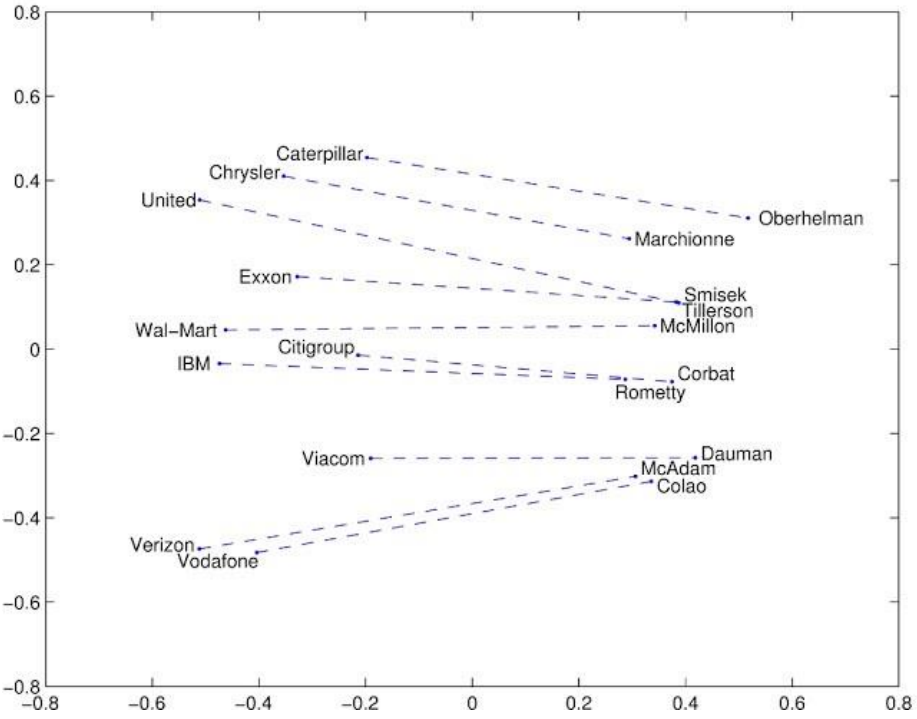
GloVe aims to achieve two goals:

- (1) Create word vectors that **capture meaning in vector space**
- (2) Takes advantage of **global count statistics** instead of only local information

Unlike word2vec – which learns by streaming sentences – GloVe determines based on a **co-occurrence matrix** and trains word vectors, so their differences predict **co-occurrence ratios**

GloVe weights the loss based on word frequency.

Somewhat surprisingly, word2vec and GloVe turn out to be remarkably similar, despite starting off from entirely different starting points.



Q8. Explain Gensim?

Answer:

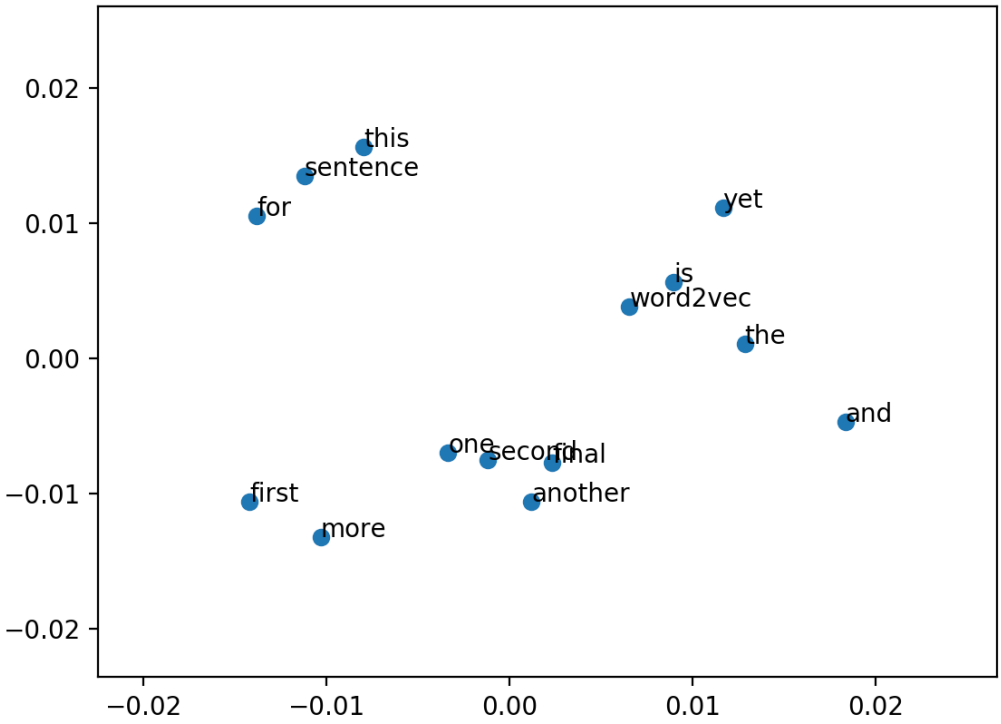
Gensim: It is billed as a Natural Language Processing package that does 'Topic Modeling for Humans'. But its practically much more than that.

If you are unfamiliar with topic modeling, it is a technique to extract the underlying topics from large volumes of text. Gensim provides algorithms like LDA and LSI (which we already seen in previous interview questions) and the necessary sophistication to built high-quality topic models.

It is an excellent library package for processing texts, working with word vector models (such as FastText, Word2Vec, etc) and for building the topic models. Another significant advantage with gensim is: it lets us handle large text files without having to load the entire file in memory.

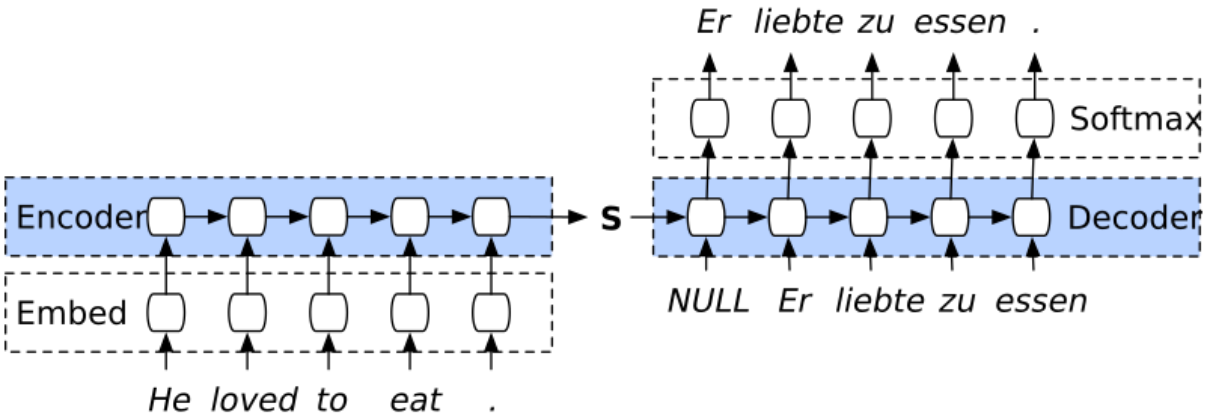
We can also tell as It is an open-source library for unsupervised topic modeling and natural language processing, using modern statistical machine learning.

Gensim is implemented in Python and Cython. Gensim is designed to handle extensive text collections using data streaming and incremental online algorithms, which differentiates it from most other machine learning software packages that target only in-memory processing.



Q9. What is Encoder-Decoder Architecture?

Answer:



The encoder-decoder architecture consists of two main parts :

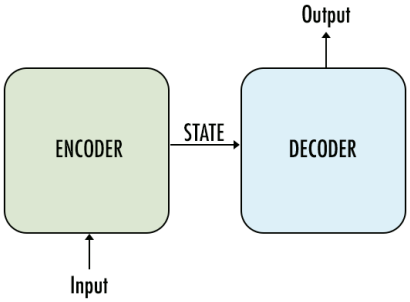
- **Encoder**:
  Encoder simply takes the input data, and trains on it, then it passes the final state of its recurrent layer as an initial state to the first recurrent layer of the decoder part.

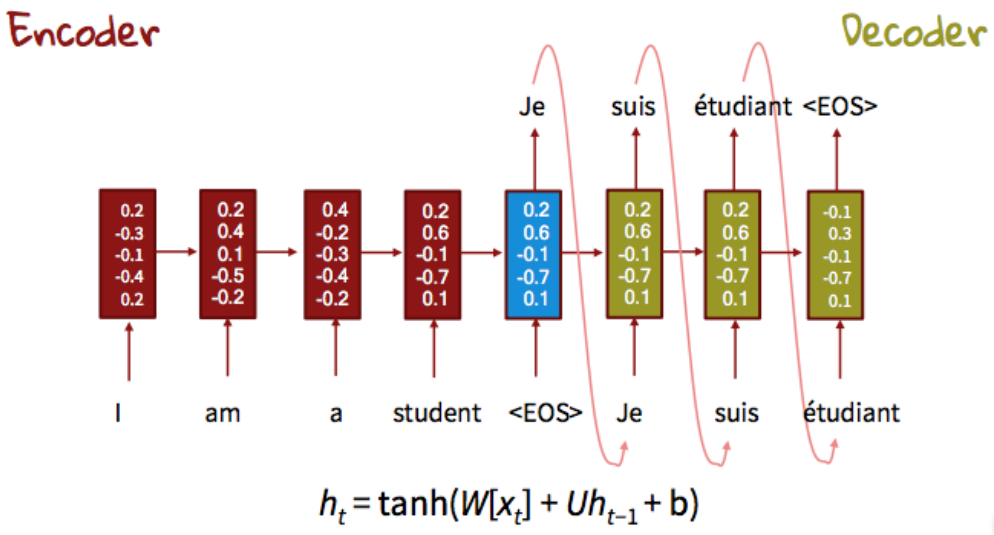
```
Encoder input : English sentences

Encoder initial state : It depends on the initializer we use
```

- **Decoder :**
  The decoder takes the final state of encoder's final recurrent layer and uses it as an initial state to its initial, recurrent layer, the input of the decoder is sequences that we want to get French sentences.

```
Decoder input : French sentences

Decoder initial state : The last state of encoder's last recurrent
layer
```

Some more example for better understanding:



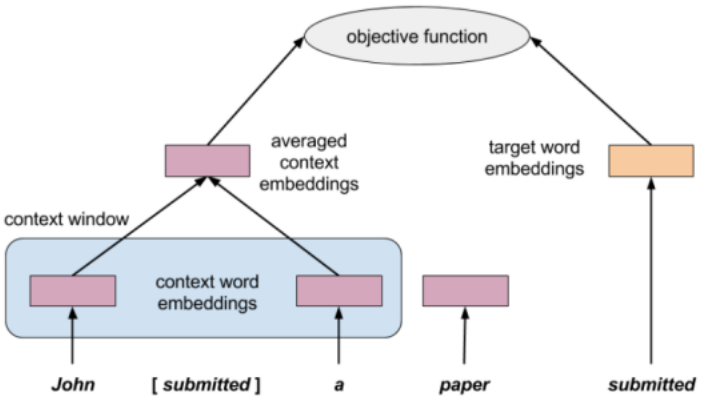


Q10. What is Context2Vec?

Answer:

Assume a case where you have a sentence like. I can't find May. Word May maybe refers to a month's name or a person's name. You use the words surround it (context) to help yourself to determine the best suitable option. Actually, this problem refers to the Word Sense Disambiguation task, on which you investigate the actual semantics of the word based on several semantic and linguistic techniques. The Context2Vec idea is taken from the original CBOW Word2Vec model, but instead of relying on averaging the embedding of the words, it relies on a much more complex parametric model that is based on one layer of Bi-LSTM. Figure1 shows the architecture of the CBOW model.

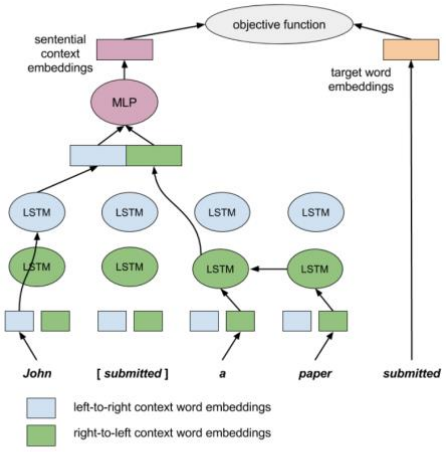
Figure1



(a) word2vec *CBOW*

Context2Vec applied the same concept of windowing, but instead of using a simple average function, it uses 3 stages to learn complex parametric networks.

- A Bi-LSTM layer that takes left-to-right and right-to-left representations
- A feedforward network that takes the concatenated hidden representation and produces a hidden representation through learning the network parameters.
- Finally, we apply the objective function to the network output.



(b) *context2vec*

Figure 1: *word2vec* and *context2vec* architectures.

We used the Word2Vec negative sampling idea to get better performance while calculating the loss value.

The following are some samples of the closest words to a given context.

Sentential Context	Closest target words
This [ ] is due	item, fact-sheet, offer, pack, card
This [ ] is due not just to mere luck	offer, suggestion, announcement, item, prize
This [ ] is due not just to mere luck, but to outstanding work and dedication	award, prize, turnabout, offer, gift
[ ] is due not just to mere luck, but to outstanding work and dedication	it, success, this, victory, prize-money

Table 1: Closest target words to various sentential contexts, illustrating *context2vec*'s sensitivity to long range dependencies, and both sides of the target word.