DATA SCIENCE INTERVIEW PREPARATION (30 Days of Interview Preparation)

Day-18

Q1. What is Levenshtein Algorithm?

Answer:

Levenshtein distance is a string metric for measuring the difference between two sequences. The Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other.

By Mathematically, the Levenshtein distance between the two strings a, b (of length |a| and |b| respectively) is given by the leva, b(|a| , |b|) where :

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Where, 1 (ai≠bi): This is the indicator function equal to zero when ai≠bi and equal to 1 otherwise, and leva, b(i,j) is the distance between the first i characters of a and the first j characters of b.

Example:

The Levenshtein distance between "HONDA" and "HYUNDAI" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

insertion
substitution
deletion

H		O	N	D	A	
H	Y	U	N	D	A	I

H	O		N	D	A	
H	Y	U	N	D	A	I

Q2. What is Soundex?

Answer:

Soundex attempts to find similar names or homophones using phonetic notation. The program retains letters according to detailed equations, to match individual titles for purposes of ample volume research.

Soundex phonetic algorithm: Its indexes strings depend on their English pronunciation. The algorithm is used to describe homophones, words that are pronounced the same, but spelt differently.

Suppose we have the following sourceDF.

```
+-----+-----+
|word1|word2|
+-----+-----+
|   to|  two|
|brake|break|
| here| hear|
| tree| free|
+-----+-----+
```

Let's run below code and see how the soundex algorithm encodes the above words.

```
val actualDF = sourceDF.withColumn(
  "w1_soundex",
  soundex(col("word1"))
).withColumn(
  "w2_soundex",
  soundex(col("word2"))
)

actualDF.show()
```

```
+-----+-----+----------+----------+
|word1|word2|w1_soundex|w2_soundex|
+-----+-----+----------+----------+
|   to|  two|      T000|      T000|
|brake|break|      B620|      B620|
| here| hear|      H600|      H600|
| tree| free|      T600|      F600|
+-----+-----+----------+----------+
```

Let's summarize the above results:

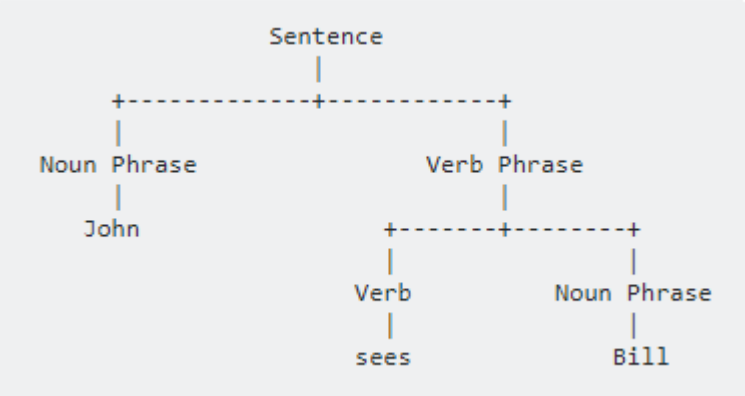
- "two" and "to" both are encoded as T000
- "break" and "brake" both are encoded as B620
- "hear" and "here" both are encoded as H600
- "free" is encoded as F600 and "tree" is encoded as T600: Encodings are similar, but word is different

The Soundex algorithm was often used to compare first names that were spelt differently.

Q3. What is Constituency parse?

Answer:

A constituency parse tree breaks a text into sub-phrases. Non-terminals in the tree are types of phrases, the terminals are the words in the sentence, and the edges are unlabeled. For a simple sentence, "John sees Bill", a constituency parse would be:



Above approaches convert the parse tree into a sequence following a depth-first traversal to be able to apply sequence-to-sequence models to it. The linearized version of the above parse tree looks as follows: (S (N) (VP V N)).

Q4. What is LDA(Latent Dirichlet Allocation)?

Answer:

LDA: It is used to classify text in the document to a specific topic. LDA builds a topic per document model and words per topic model, modelled as Dirichlet distributions.

- Each document is modeled as a distribution of topics, and each topic is modelled as multinomial distribution of words.
- LDA assumes that every chunk of text we feed into it will contain words that are somehow related. Therefore choosing the right corpus of data is crucial.

- It also assumes documents are produced from a mixture of topics. Those topics then generate words based on their probability distribution.

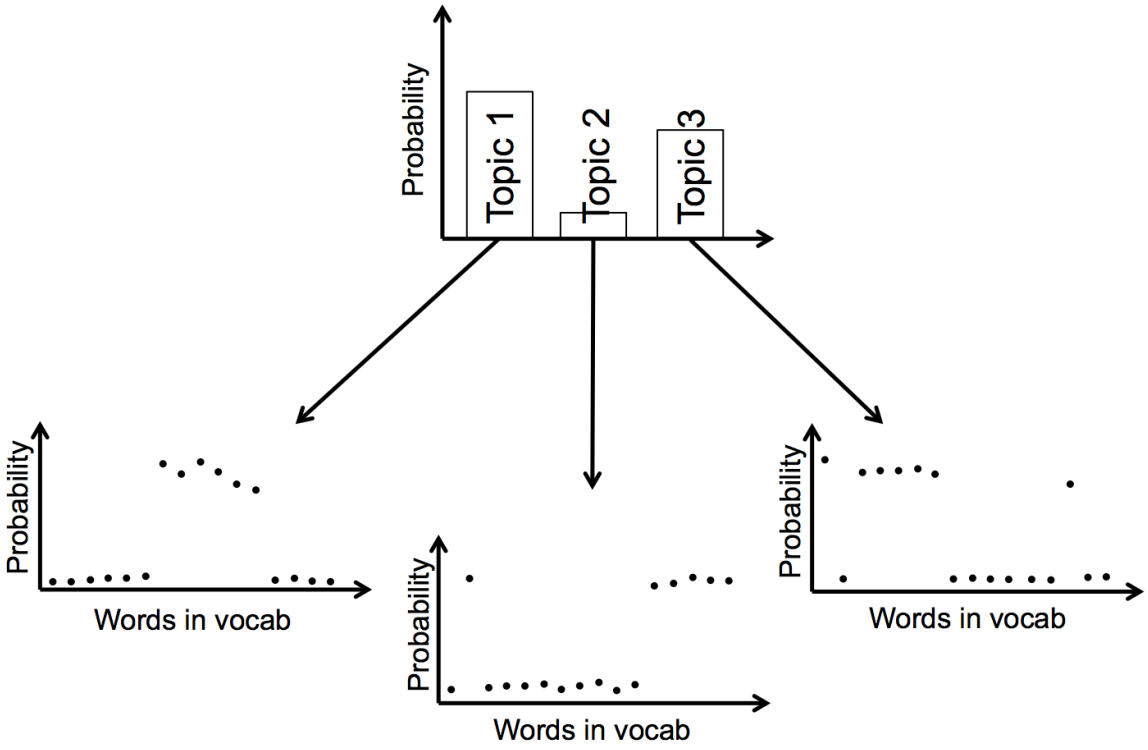
The Bayesian version of PLSA is LDA. It uses Dirichlet priors for the word-topic and document-topic distributions, lending itself to better generalization.

What LDA give us?

It is a probabilistic method. For every document, the results give us a mixture of topics that make up the document. To be precise, we can get probability distribution over the k topics for every document. Every word in the document is attributed to the particular topic with probability given by distribution.

These topics themselves were defined as probability distributions over vocabulary. Our results are two sets of probability distributions:

- The collection of distributions of topics for each document
- The collection of distributions of words for each topic.



Q5.What is LSA?

Answer:

Latent Semantic Analysis (LSA): It is a theory and the method for extract and represents the contextual usage meaning of words by statistical computation applied to large corpus of texts.

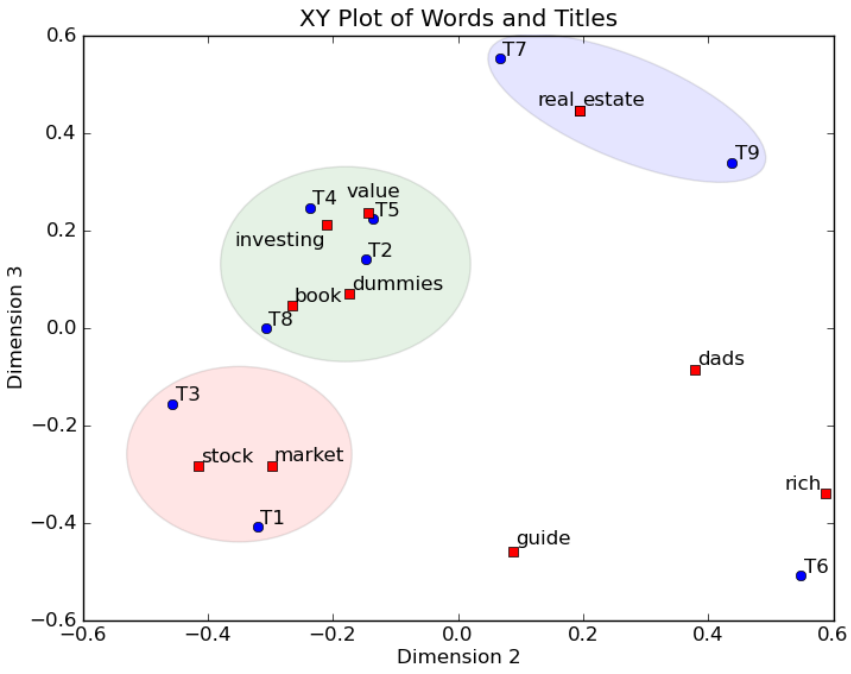
It is an information retrieval technique which analyzes and identifies the pattern in an unstructured collection of text and relationship between them.

Latent Semantic Analysis itself is an unsupervised way of uncovering synonyms in a collection of documents.

Why LSA(Latent Semantic Analysis)?

LSA is a technique for creating vector representation of the document. Having a vector representation of the document gives us a way to compare documents for their similarity by calculating the distance between vectors. In turn, means we can do handy things such as classify documents to find out which of a set knows topics they most likely reside to.

Classification implies we have some known topics that we want to group documents into, and that you have some labelled training data. If you're going to identify natural groupings of the documents without any labelled data, you can use clustering



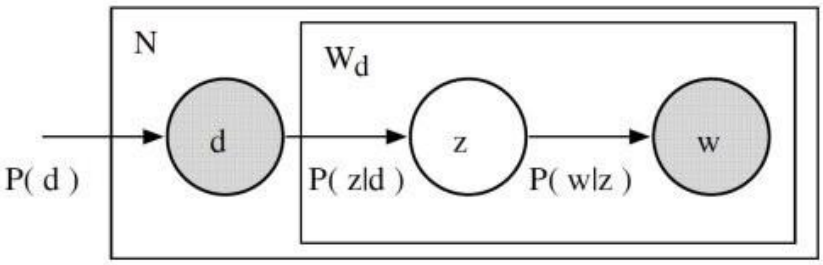
Q6. What is PLSA?

Answer:

PLSA stands for Probabilistic Latent Semantic Analysis, uses a probabilistic method instead of SVD to tackle problem. The main idea is to find the probabilistic model with latent topics that we can *generate* data we observe in our document term matrix. Specifically, we want a model P(D, W) such that for any document d and word w, P(d,w) corresponds to that entry in document-term matrix.

Each document is found in the mixture of topics, and each topic consists of the collection of words. PLSA adds the probabilistic spin to these assumptions:

- Given document d, topic z is available in that document with the probability P(z|d)
- Given the topic z, word w is drawn from z with probability P(w|z)



The joint probability of seeing the given document and word together is:

$$P(D,W) = P(D)\sum_{Z} P(Z|D)P(W|Z)$$

In the above case, P(D), P(Z|D), and P(W|Z) are the parameters of our models. P(D) can be determined directly from corpus. P(Z|D) and the P(W|Z) are modelled as multinomial distributions and can be trained using the expectation-maximisation algorithm (EM).

Q7. What is LDA2Vec?

Answer:

It is inspired by LDA, word2vec model is expanded to simultaneously learn word, document, topic and paragraph topic vectors.

Lda2vec is obtained by modifying the skip-gram word2vec variant. In the original skip-gram method, the model is trained to predict context words based on a pivot word. In lda2vec, the pivot word vector and a document vector are added to obtain a context vector. This context vector is then used to predict context words.

At the document level, we know how to represent the text as mixtures of topics. At the word-level, we typically used something like word2vec to obtain vector representations. It is an extension of word2vec and LDA that jointly learns word, document, and topic vectors.

How does it work?

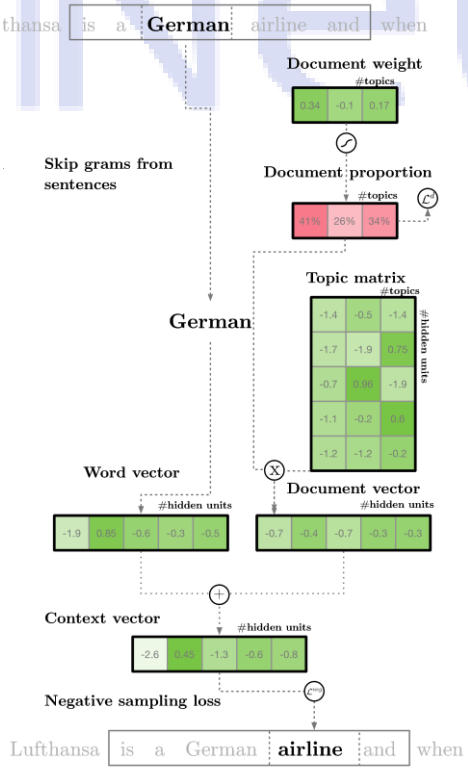
It correctly builds on top of the skip-gram model of word2vec to generate word vectors. Neural net that learns word embedding by trying to use input word to predict enclosing context words.

With Lda2vec, other than using the word vector directly to predict context words, you leverage a context vector to make the predictions. Context vector is created as the sum of two other vectors: the word vector and the document vector.

The same skip-gram word2vec model generates the word vector. The document vector is most impressive. It is a really weighted combination of two other components:

- the document weight vector, representing the "weights" of each topic in a document
- Topic matrix represents each topic and its corresponding vector embedding.

Together, a document vector and word vector generate "context" vectors for each word in a document. lda2vec power lies in the fact that it not only learns word embeddings for words; it simultaneously learns topic representations and document representations as well.



Q8. What is Expectation-Maximization Algorithm(EM)?

Answer:

The Expectation-Maximization Algorithm, in short, EM algorithm, is an approach for maximum likelihood estimation in the presence of latent variables.

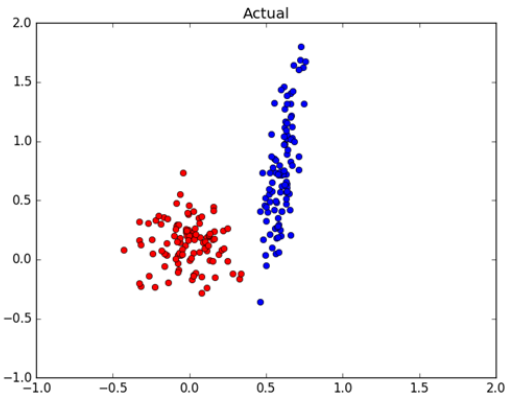
This algorithm is an iterative approach that cycles between two modes. The first mode attempts to predict the missing or latent variables called the estimation-step or E-step. The second mode attempts to optimise the parameters of the model to explain the data best called the maximization-step or M-step.

- **E-Step**. Estimate the missing variables in the dataset.
- **M-Step**. Maximize the parameters of the model in the presence of the data.

The EM algorithm can be applied quite widely, although it is perhaps most well known in machine learning for use in unsupervised learning problems, such as density estimation and clustering.

For detail explanation of EM is, let us first consider this example. Say that we are in a school, and interested to learn the height distribution of female and male students in the school. The most sensible thing to do, as we probably would agree with me, is to randomly take a sample of N students of both genders, collect their height information and estimate the mean and standard deviation for male and female separately by way of maximum likelihood method.

Now say that you are not able to know the gender of student while we collect their height information, and so there are two things you have to guess/estimate: (1) whether the individual sample of height information belongs to a male or a female and (2) the parameters (μ, θ) for each gender which is now unobservable. This is tricky because only with the knowledge of who belongs to which group, can we make reasonable estimates of the group parameters separately. Similarly, only if we know the parameters that define the groups, can we assign a subject properly. How do you break out of this infinite loop? Well, EM algorithm just says to start with initial random guesses.



Q9.What is Text classification in NLP?

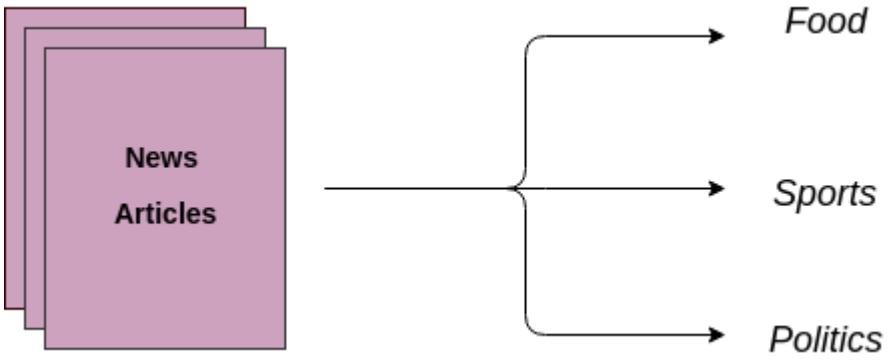
Answer:

Text classification is also known as text tagging or text categorization is a process of categorizing text into organized groups. By using NLP, text classification can automatically analyze text and then assign a set of pre-defined tags or categories based on content.

Unstructured text is everywhere on the internet, such as emails, chat conversations, websites, and the social media but it's hard to extract value from given data unless it's organized in a certain way. Doing so used to be a difficult and expensive process since it required spending time and resources to manually sort the data or creating handcrafted rules that are difficult to maintain. Text classifiers with NLP have proven to be a great alternative to structure textual data in a fast, cost-effective, and scalable way.

Text classification is becoming an increasingly important part of businesses as it allows us to get insights from data and automate business processes quickly. Some of the most common examples and the use cases for automatic text classification include the following:

- **Sentiment Analysis:** It is the process of understanding if a given text is talking positively or negatively about a given subject (e.g. for brand monitoring purposes).
- **Topic Detection:** In this, the task of identifying the theme or topic of a piece of text (e.g. know if a product review is about Ease of Use, Customer Support, or Pricing when analysing customer feedback).
- **Language Detection:** the procedure of detecting the language of a given text (e.g. know if an incoming support ticket is written in English or Spanish for automatically routing tickets to the appropriate team).



Q10. What is Word Sense Disambiguation (WSD)?

Answer:

WSD (Word Sense Disambiguation) is a solution to the ambiguity which arises due to different meaning of words in a different context.

In natural language processing, **word sense disambiguation** (WSD) is the problem of determining which "sense" (meaning) of a word is activated by the use of the word in a particular context, a process which appears to be mostly unconscious in people. WSD is the natural classification problem: Given a word and its possible senses, as defined by the dictionary, classify an occurrence of the word in the context into one or more of its sense classes. The features of the context (such as the neighbouring words) provide the evidence for classification.

For example, consider these two below sentences.

" The **bank** will not be accepting the cash on Saturdays. "

" The river overflowed the **bank** ."

The word " **bank "** in the given sentence refers to commercial (finance) banks, while in the second sentence, it refers to a riverbank. The uncertainty that arises, due to this is tough for the machine to detect and resolve. Detection of change is the first issue and fixing it and displaying the correct output is the second issue.