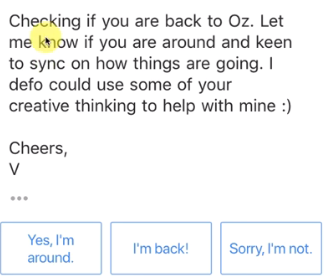
Chapter 1: Part 2

**Bag of Words Model**

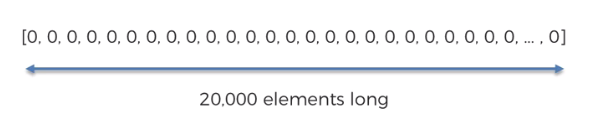
**1.2.1 Bag of words**

Look at following email:



* We're going to be looking at how we can apply ***NLP*** to this ***email*** in the next couple sections and it will help us work with a real life example.
* Notice those three *suggestions* (blue colored) from the Gmail app. Those are also from NLP (we'll come back to this later).
* We're going to ***create a model***, that will give us an a **Yes/No** response by analyzing that email. (That's a first step into NLP and then we can expand that and more.)

**1.2.2 Array of words**



* Notice this array of 20,000 elements. Because we will work with 20,000 words in our NLP.
* If we search in Google: "How many words in English language?", Then we have several results:

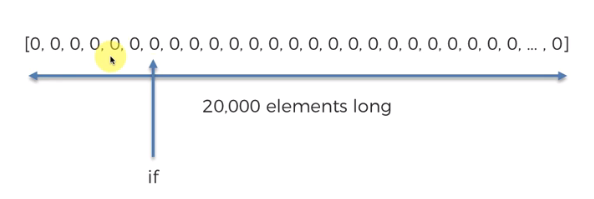
|  |  |
| --- | --- |
|  |  |

1. So there are 171,476 total English words in Oxford dictionary.
2. But an average person knows 20,000 to 30,000 words.
3. In every-day use or chatting, people use approximately 3000 words. Just 3000 words provides coverage for around 95 % of common texts.

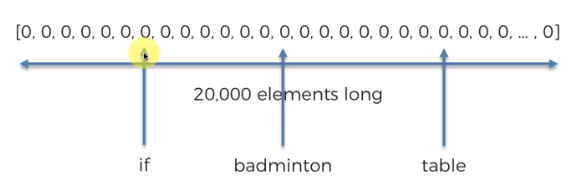
* Hence our array of 20,000 words is enough to build an NLP app. Basically every word in the English (mostly used) language has a position somewhere on this vector.
* Interesting fact is even when we search in google it also apply NLP to show us related results from our query:

|  |  |
| --- | --- |
|  |  |

* For example the word "**if**" could have this position.



* Say it is in the seventh position in our custom made vector, it is always going to be on that position.
* For instance the word "**badminton**" is always in 200th position, the word "**table**" is going to be 55th position etc.



And this is like how this bag of words model works.

* SOS, EOS and last entry: Note that, the *first two entry* are going to be reserved for *SOS* (start of sentence) and *EOS* (end of sentence)
* The *last entry* will be reserved for *special words* (those other 151,474 words in 171,476 that doesn’t fall into 20,000 words). If they come up we're going to just throw them all in this last entry.
* Example: Let's go back to our e-mail text.

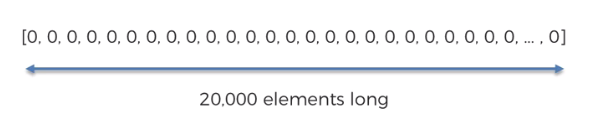
Hello KK,

Checking if you're back to Oz. Let me know if you are around. . . . . etc. etc..

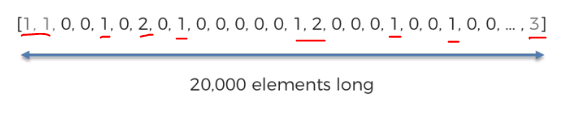
Cheers V.

Let's see how this can be put into our bag of words.

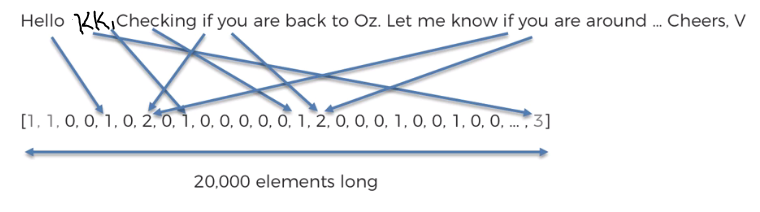
* The array of ***20,000 zeros*** is the ***Bag-of-Ward*** that we are constructing here.



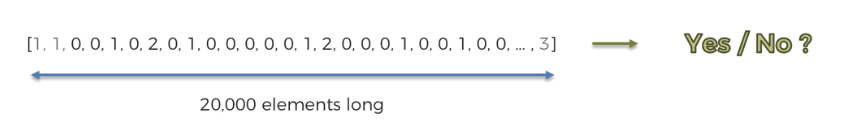
* We *throw* the *text* into this *bag of words* and those word in the sentence are *counted* and *0's will be updated*.



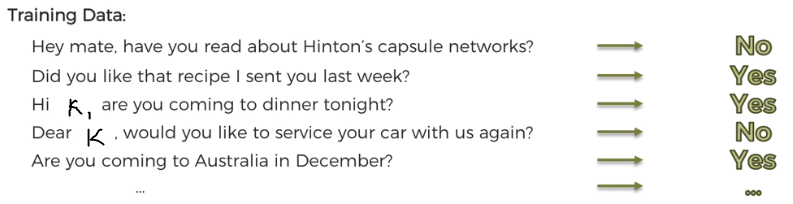
* So we through our text in the ***Bag-of-Ward*** and find and then increase the counter in each position of the associated words. The counters keep track the frecuency-of-words that appear in the sentence.
* For example: say "Hello" appear once so we put it in 5th position of the array and assign the value 1. "KK" is not an English word so we put it in the last entry/position.



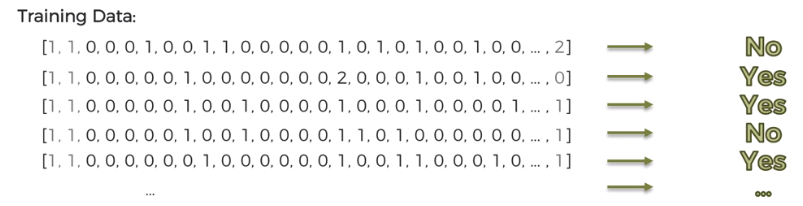
* Our goal is to come up with replies Yes/No to this e-mail which is now in the form of a vector.



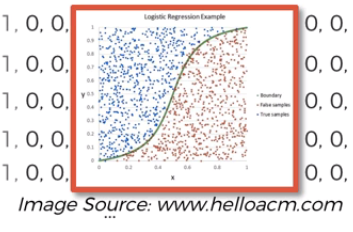
* Train a model with those data: We're going to do it through *training data*. We're going to look at all the *e-mails* that we have *replied* to.
* We're going to be *train the model* using *training data* (which are samples of sent emails) and we also *keep track* of the *responses* (i.e. Yes/No answers) of those emails.



* Then we *turn* these *emails &* corresponding *responses* to ***vectors***. Again each vector would be 20000 elements long. So lots and lots of vectors, and lots and lots of responses yes/no. Once we have all this data we're going to apply a NLP model.



* One of the algorithms we can apply to create a bag of words model is the ***logistic regression***.



* After the model is ***Trained***, i.e. it can ***Predict*** which emails respond will be ***Yes*** or ***No***, then we feed it to our test data (real email).
* Note that, we feed the emails in exactly the ***same*** ***format***, every time we train the data the independent variable vector always has the same ***length*** ***20000*** and always had the ***same*** ***format***.
* It's got the same number of features to get an answer.
* Deep-NLP: Now, using model based on ***logistic regression*** isn't a Deep-NLP. we're just using ***logistic regression***. It is just NLP using Bag-of-Word.
* However we can use Neural Networks. Since, we have a feature-vector, we could feed them into as an *input layer* like over *20000* *neurons* into our *neural network*. We can structure the NN by choosing correct amount of Hidden-Layers.
* Then we train this NN using millions of emails & responses. Through *forward-back propagation*, *sarcastic gradient descent*, updating the *weights* comparing with the response. And finally we can get an answer.
* So now it is a *Deep-NLP*. Of course it is still *"Bag-of-Word"* model, since we applied *Neural Network* to it now it becomes *Deep-NLP*.
* Therefore, in the first case we have ***NLP-Bag-of-Wards*** and in the second case we have ***Deep-NLP-Bag-of-Wards***.