Lecture 3 : Natural Language Processing 🌱

> **Class**: NLP
>
> **Date**: 10.00 AM, Yesterday
>
> **Author**: Lasal Hettiarachchi

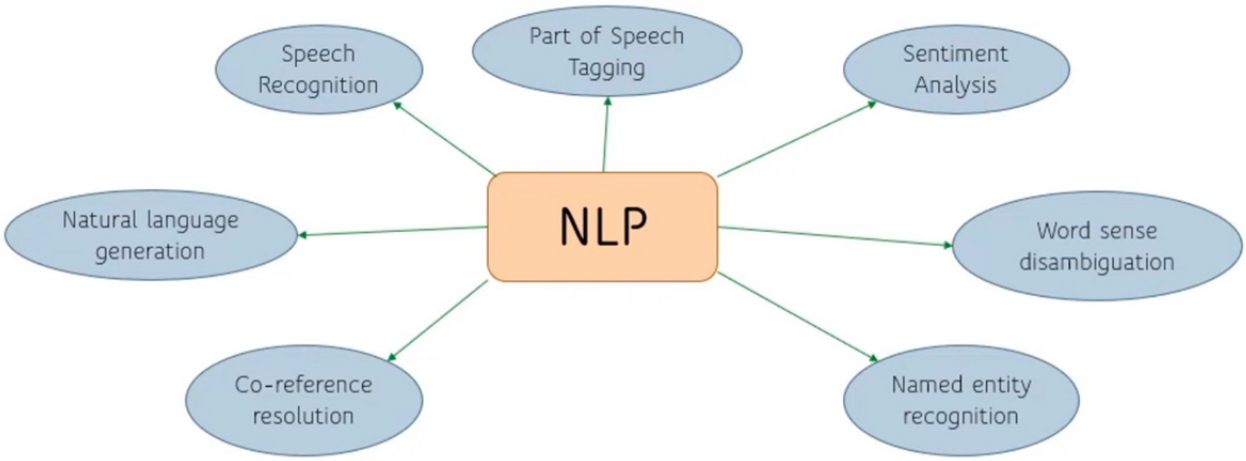
Key learnings:

- RNN architecture

Natural language processing (NLP) refers to the branch of computer science that combines computational linguistics-rule-based modeling of human lanquage-with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to 'understand' its full meaning,

Applications of NLP

- Spam detection
- Machine translation (seq-seq translators and transformers)
- Virtual chatbots
- Sentiment Analysis
- Text summerization (summarize large volumes of data and create synopses)



Key Words:

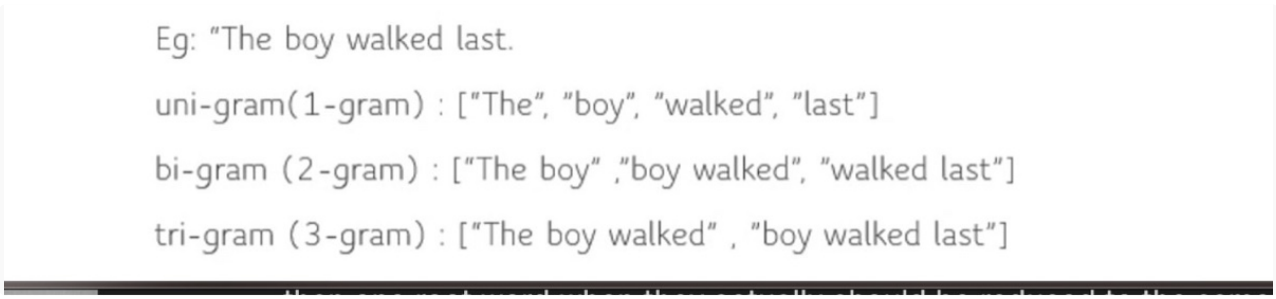
- Corpus : A corpus is a collection of authentic text or audio organized into datasets. Authentic here means text written or audio spoken by a native of the language or dialect.
- Document : A "document" is a distinct text. This generally means that each article, book, or so on is its own document. If you wanted, you could treat an individual paragraph or even sentence as a "document"

- Tokenization: Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation. Here is an example of tokenization:

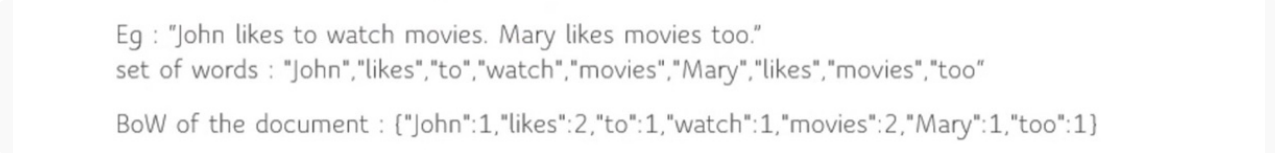
The major question of the tokenization phase is what are the correct tokens to use? In this example, it looks fairly trivial: you chop on whitespace and throw away punctuation characters. This is a starting point, but even for English there are a number of tricky cases. For example, what do you do about the various uses of the apostrophe for possession and contractions?

- Tokens and Tokenization: tokenization is, generally, an early step in the NLP process, a step which splits longer strings of text into smaller pieces, or tokens. Larger chunks of text can be tokenized into sentences, sentences can be tokenized into words, etc. Further processing is generally performed after a piece of text has been appropriately tokenized.
- Normalization: Is the process of brining the text in the corpus to a common ground. This is an essential text preprocessing step.
    - Eg: "Home" → "home", (lowercasing all the text)
    - When we normalize text, we attempt to reduce its randomness, bringing it closer to a predefined "standard". This helps us to reduce the amount of different information that the computer has to deal with, and therefore improves efficiency. The goal of normalization techniques like stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.
- Stemming: Stemming is the process of eliminating affixes (suffixed, prefixes, infixes, circumfixes) from a word in order to obtain a word stem. (rule based methods)
    - Eg: Running → Run
    - Stemming is the process of reducing the words to their word stem or root form. The objective of stemming is to reduce related words to the same stem even if the stem is not a dictionary word. For example, connection, connected, connecting word reduce to a common word "connect"
    - Porters Algorithm is used
    - Over-stemming : where a much larger part of a word is chopped off than what is required, which in turn leads to words being reduced to the same root word or stem incorrectly when they should have been reduced to more stem words. For example, the words "university" and "universe" that get reduced to "univers".

    - Under-stemming: occurs when two or more words could be wrongly reduced to more than one root word when they actually should be reduced to the same root word. For example, the words "data" and "latum" that get reduced to "dat" and "datu" respectively (instead of the same stem "dat").
- Lammatization: Lemmatization is related to stemming, but how it differs from stemming is that it is able to capture canonical forms based on a word's Lemma
    - Eg: better → good
- Stop words : Stop words are words which contribute less for the overall meaning. They are essential to maintain a structure of a sentence but won't add value for the NLP task at hand.
    - In the sentence: "The cat sneaked to the behind door.Here words like "the" and "to" are considered stop words.
- N-grams: is one type of representation model, where we take N sequences of text to form a unit of text.



- Bag of words: Is one of the simplest representation in NLP. In this model a document (sentences, articles etc) is represented by a set of words, disregarding grammar or the order of the words, like taking the words and putting it into a bag.



- Vocabulary : The entire set of terms used in a body of text.
- Word Embedding : Each token is embedded as a vector before it can be passed to a machine learning model. While generally referred to as word embeddings, embeddings can be created on the character or phrase level as well. Following the techniques section is an entire section on different types of embeddings

NLP modeling can be catagorized into 2

- Statistical Modeling
    - Count Vectorization (Using BOW)
    - tf-IDF
- Machine learning methods
    - Word2Vec
    - RNN
    - LSTM

- Transformers

What is Representational learning

- Representation learning is a class of machine learning approaches that allow a system to discover the representations required for feature detection or classification from raw data. The requirement for manual feature engineering is reduced by allowing a machine to learn the features and apply them to a given activity.
- In representation learning, data is sent into the machine, and it learns the representation on its own. It is a way of determining a data representation of the features, the distance function, and the similarity function that determines how the predictive model will perform. Representation learning works by reducing high-dimensional data to low-dimensional data, making it easier to discover patterns and anomalies while also providing a better understanding of the data's overall behaviour.
- Basically, Machine learning tasks such as classification frequently demand input that is mathematically and computationally convenient to process, which motivates representation learning. Real-world data, such as photos, video, and sensor data, has resisted attempts to define certain qualities algorithmically. An approach is to examine the data for such traits or representations rather than depending on explicit techniques.

Bag of words

A bag of words is a representation of text that describes the occurrence of words within a document. We just keep track of word counts and disregard the grammatical details and the word order. It is called a "bag" of words because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document

A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

1. A vocabulary of known words.
2. A measure of the presence of known words.

Example :

Below is a snippet of the first few lines of text from the book "A Tale of Two Cities" by Charles Dickens, taken from Project Gutenberg.

> *It was the best of times,*
> *it was the worst of times,*
> *it was the age of wisdom,*
> *it was the age of foolishness,*

Now we can make a list of all of the words in our model vocabulary. The unique words here (ignoring case and punctuation) are:

- "it"
- "was"
- "the"

- "best"
- "of"
- "times"
- "worst"
- "age"
- "wisdom"
- "foolishness"

That is a vocabulary of 10 words from a corpus containing 24 words.

Then the document vector is created.The objective is to turn each document of free text into a vector that we can use as input or output for a machine learning model. The simplest scoring method is to mark the presence of words as a boolean value, 0 for absent, 1 for present.

```
"it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
"it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
"it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]
```

Things such as n-grams are used to reduce the number of words

Next is to score the words.

In the worked example, we have already seen one very simple approach to scoring: a binary scoring of the presence or absence of words.

Some additional simple scoring methods include:

- **Counts**. Count the number of times each word appears in a document.
- **Frequencies**. Calculate the frequency that each word appears in a document out of all the words in the document.

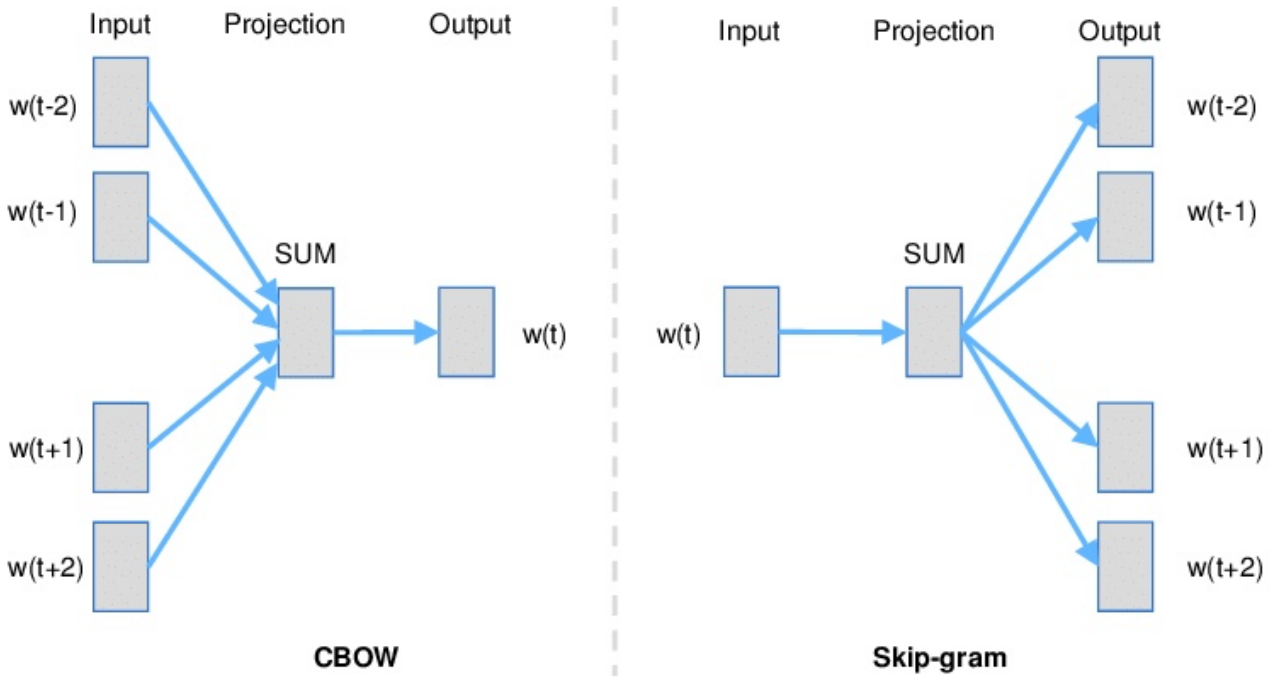
limitations

- **Vocabulary**: The vocabulary requires careful design, most specifically in order to manage the size, which impacts the sparsity of the document representations.
- **Sparsity**: Sparse representations are harder to model both for computational reasons (space and time complexity) and also for information reasons, where the challenge is for the models to harness so little information in such a large representational space.
- **Meaning**: Discarding word order ignores the context, and in turn meaning of words in the document (semantics). Context and meaning can offer a lot to the model, that if modeled could tell the difference between the same words differently arranged (“this is interesting” vs “is this interesting”), synonyms (“old bike” vs “used bike”), and much more.

Word2Vec Models

This is also a type of an auto encoder (auto encoders are usually used to reduce noice and shrink the input feature set and expand it again to the same size)

- The conecpt of know your neighbours is applied here.
- Order of the word is taken into account (context matters)



We take a sentence made of words and performs one hot encoding

"King Ruled England"

King [1,0,0]

Ruled [0,1,0]

England [0,0,1]

Then we concaternate all the values and give as inputs to the architecture.

- We concaternate King and england together and the architecture will provide the one hot encoding of ruled
- 100001 → 010

Word2Vec Extended

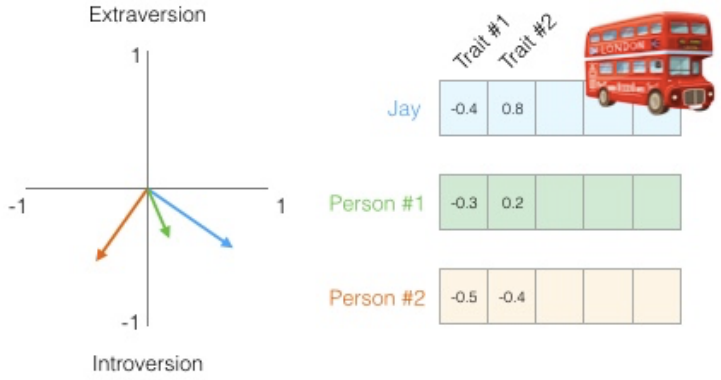
The mechanics of generating embeddings with word2vec.

Personality embedding...

On a scale of 0 to 100, how introverted/extraverted are you (where 0 is the most introverted, and 100 is the most extraverted)? Have you ever taken a personality test like MBTI – or even better, the Big Five Personality Traits test? If you haven't, these are tests that ask you a list of questions, then score you on a number of axes, introversion/extraversion being one of them.

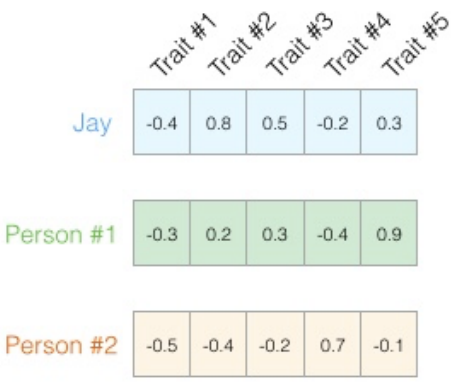
Openness to experience	79 out of 100
Agreeableness	75 out of 100
Conscientiousness	42 out of 100
Negative emotionality	50 out of 100
Extraversion	58 out of 100

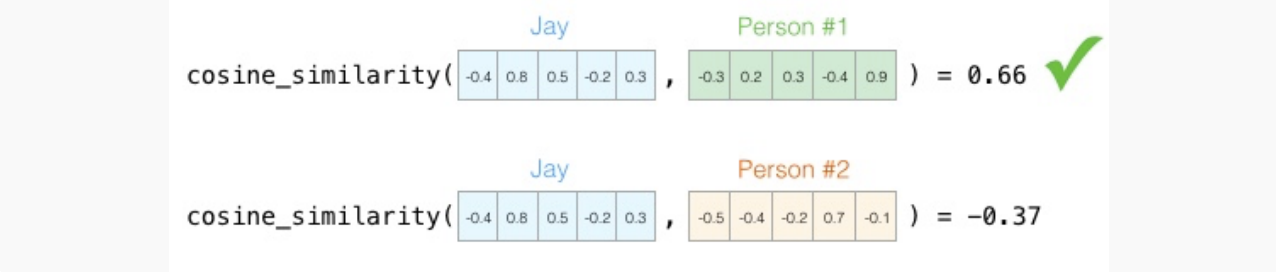
Lets add the traits as different dimensions and represent people as vectors



When dealing with vectors, a common way to calculate a similarity score is cosine_similarity:

Extending this to multiple traits





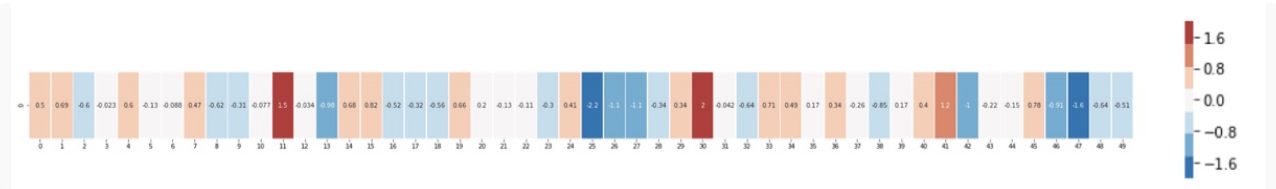
Word Embedding

Similar to how traits are represented as arrays, we can embed words to get a vector representation for its traits

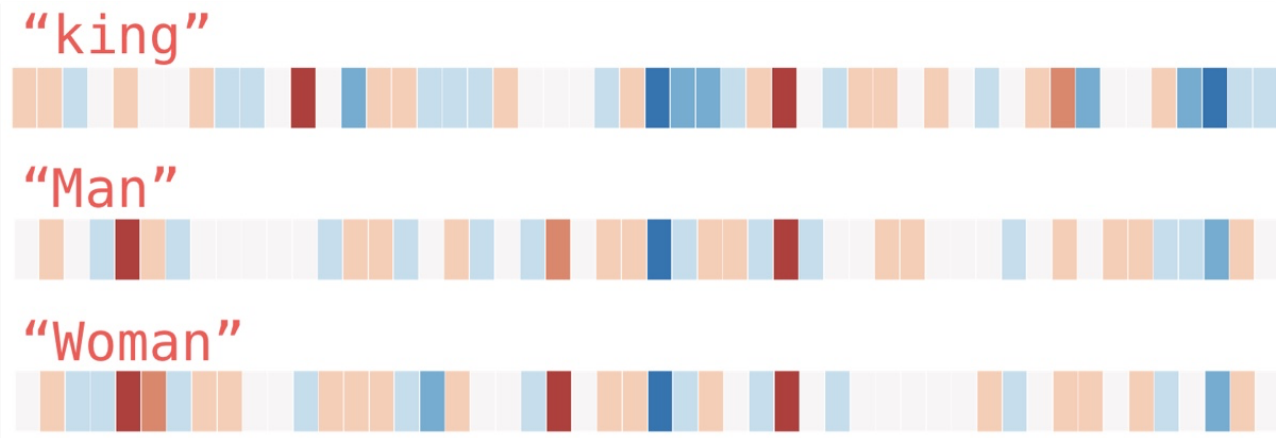
The GloVe representation of the word "King" is as follows,

[0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 , -0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961 , -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 , -0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 , -1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042]

- Let's color code the cells based on their values (red if they're close to 2, white if they're close to 0, blue if they're close to -2):



We'll proceed by ignoring the numbers and only looking at the colors to indicate the values of the cells. Let's now contrast "King" against other words:

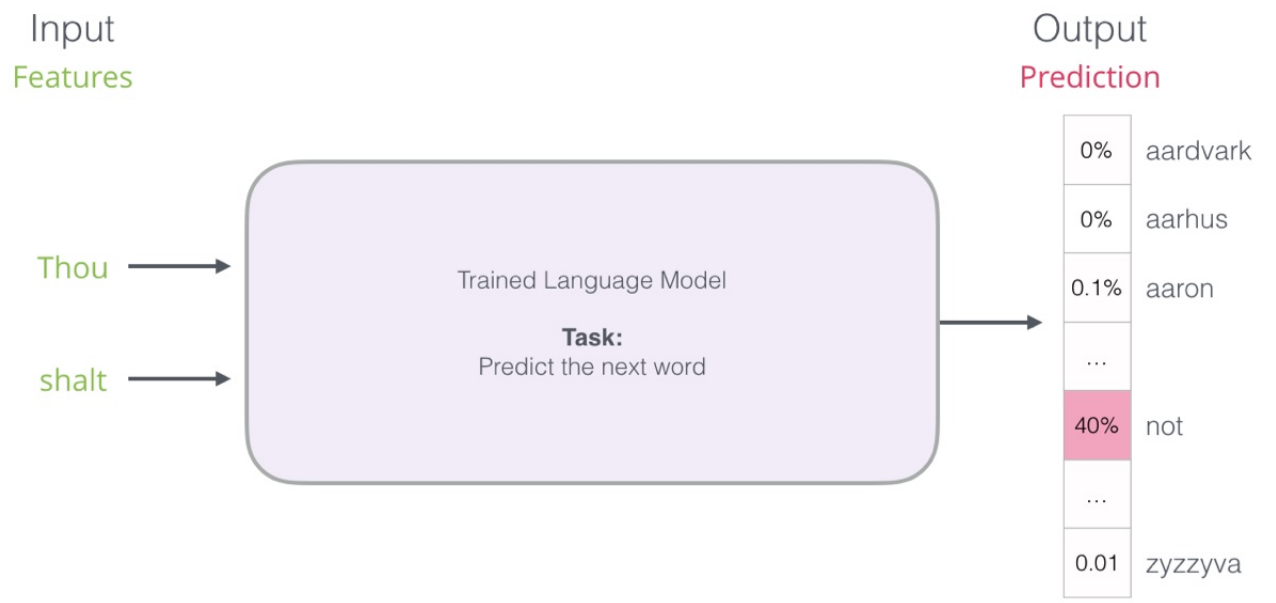


See how "Man" and "Woman" are much more similar to each other than either of them is to "king"? This tells you something. These vector representations capture quite a bit of the information/meaning/ associations of these words.

How is this applied in NLP ?

If one wanted to give an example of an NLP application, one of the best examples would be the next-word prediction feature of a smartphone keyboard. It's a feature that billions of people use hundreds of times every day.Next-word prediction is a task that can be addressed by a *language model*. A language model can take a list of words (let's say two words), and attempt to predict the word that follows them.

But in proper Word2Vec models, it outputs the probability of a word being the output label.



The Illustrated Word2vec
https://jalammar.github.io/illustrated-word2vec/
https://jalammar.github.io/illustrated-word2vec/

- The corpus is extremely important in this context. Since the embedding is created based on that.
- In the first example, we used one document as a vector
  - "it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
- But here we represent the embedding of one word as a vector

So how can this Word2Vec be used to predict something like spam / ham?

- we can add the vector embeddings of words to form the sentence vector representation

NLP models

Hugging Face – The AI community building the future.
"We're on a journey to advance and democratize artificial intelligence through open source and open science."
https://huggingface.co/