The NLP Canvas and AQuA System for Documents

Speakers:
Abhishek Parikh, Dhara Kotecha
Infocusp Innovations Pvt. Ltd.

At ICDAR 2019

infocusp
Innovations Pvt. Ltd.

©InFoCusp Innovations Pvt. Ltd. 2019

About US!

- We are computer engineers, working and researching at INFOCUSP INNOVATIONS PVT. LTD.
  - As a company, we work in the domain of Machine Learning, Natural Language Processing, Computer Vision and others under the larger umbrella of Artificial Intelligence.
- We work with:
  - Cerebellum Capital: Financial Modelling
  - LegalSifter: Natural Language Processing and Document Analysis
  - Bryte: Digital Signal Processing and Smart BEDS!
  - Agnetix: Smart Greenhouse Systems
  - RecruitmentSmart (Past Client): Natural Language Processing and Document Analysis
  - and many more...

About US!

- Website: http://infocusp.in/
- Reach out to the organization:
  - Mr. Nisarg Vyas, Principal Founder and CEO: nisarg@infocusp.in
  - Mr. Urvik Patel, CTO: urvik@infocusp.in

About US!

- Abhishek Parikh:
  - I graduated from DA-IICT with B.Tech Honors.
  - Working with INFOCUSP since January 2017 in different projects with different top-notch clients.
  - Domain of Work: Natural Language Processing, Machine Learning, Software Engineering
  - You can reach out to me on:
    - Via Email: abhishek@infocusp.in, parikhabhi007@gmail.com
    - Via LinkedIn: www.linkedin.com/in/parikhabhi007

infocusp
Innovations Pvt. Ltd.

About US!

- Dhara Kotecha:
    - I graduated with Masters in Technology from DA-IICT.
    - My dissertation was in the field of Document Image Binarization and GANs for facial expression transfer.
    - Working with INFOCUSP since July 2019 in different projects with different top-notch clients.
    - Domain of Work: Natural Language Processing, Computer Vision, Object Detection, Video Processing
    - You can reach out to me:
        - Via Email: dhara@infocusp.in, kotechadhara6@gmail.com
        - Via LinkedIn: www.linkedin.com/in/dhara-kotecha

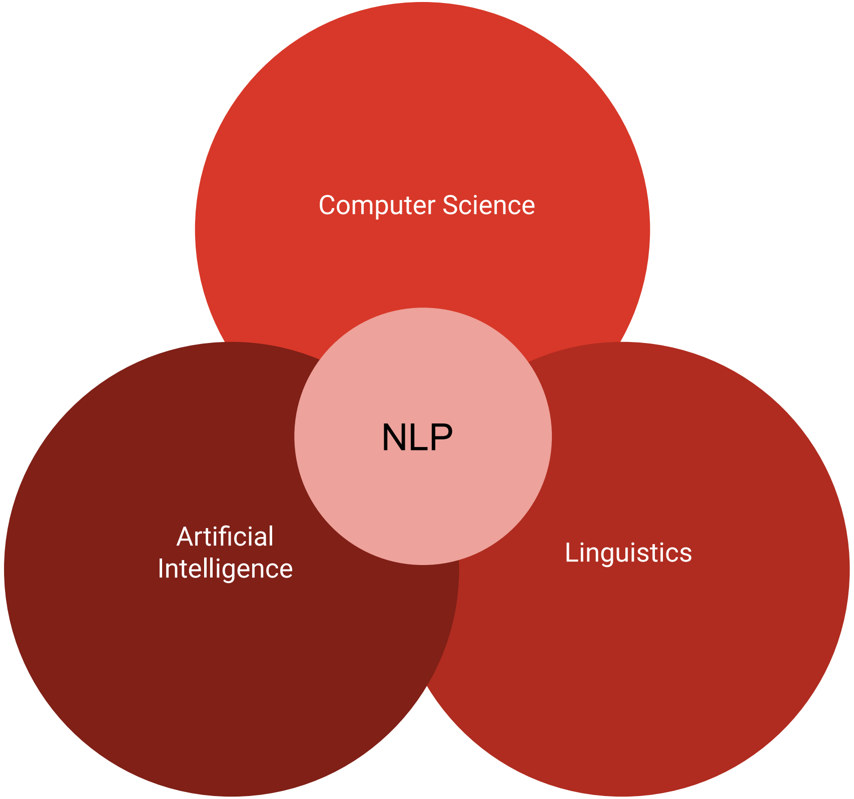
LET’S START!

- Natural Language Processing and its building blocks!
- Advance Topics in NLP
- Code/Demo:
  - BERT based Automatic Question Answering System
  - Basic NLP (if time-permits)

Natural Language Processing

- Natural Language Processing (NLP) is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages.
- In particular how to program computers to process and analyze large amounts of natural language data.

Natural Language Processing



©InFoCusp Innovations Pvt. Ltd. 2019

Natural Language

- Natural Language refers to the way we, humans, communicate with each other.
- Types:
    - Text
    - Speech

Natural Language

- Natural language is primarily hard because it is messy.
- There are few rules.
- And yet we can easily understand each other most of the time.

- Natural Language is highly ambiguous.
- Example:
  - Woman, without her man, is helpless.
  - Woman! Without her, man is helpless!
  - Homophones and Homographs

Applications of Natural Language Processing

- Text Recognition
- Speech Recognition
- Natural Language Generation
- Translation

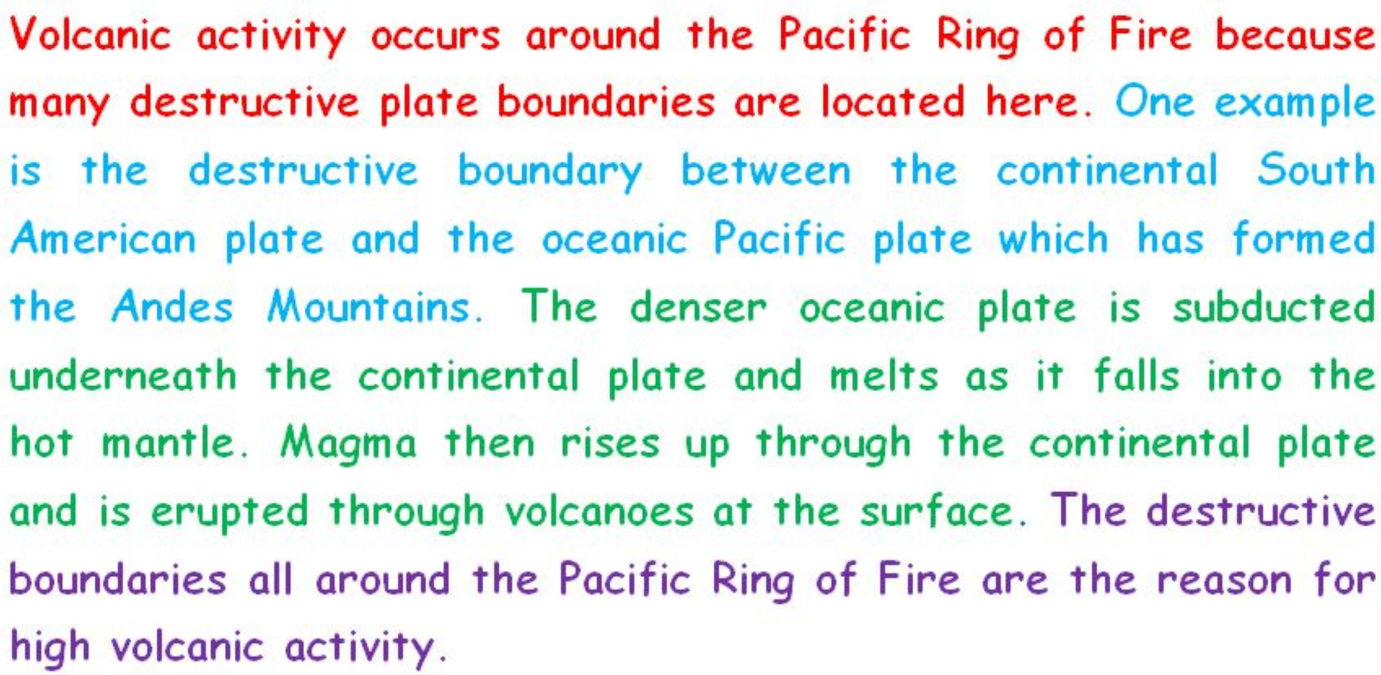
- In the context of the conference we will mainly focus on the **Document Analysis** part.

From Natural Human Language to Linguistics

- What do you mean by Linguistics?
  - **Linguistics** is the scientific study of language, including its **syntax**, grammar, **semantics**, and **phonetics**.
  - Classical Linguistics (Rules-based approach)
  - Computational Linguistics (Statistical approach)

©InFoCusp Innovations Pvt. Ltd. 2019

Let us understand the process!



Volcanic activity occurs around the Pacific Ring of Fire because many destructive plate boundaries are located here. One example is the destructive boundary between the continental South American plate and the oceanic Pacific plate which has formed the Andes Mountains. The denser oceanic plate is subducted underneath the continental plate and melts as it falls into the hot mantle. Magma then rises up through the continental plate and is erupted through volcanoes at the surface. The destructive boundaries all around the Pacific Ring of Fire are the reason for high volcanic activity.

Understanding the process!

- Cleaning the data (Only keeping the required text).
- Case-sensitivity.
- Breaking documents into paragraphs.
- Breaking paragraphs into sentences.
- Breaking sentences into words.
- Grammar
- Focusing on words - their meanings, synonyms and their usages.
- Lemmatization and Stemming
- Part of Speech (Verb, Noun, Adjective, Adverb, Number etc.)
- Named Entity Relation
- Keyword recognition

Sentence Boundary Detection

- Sentence Boundary Detection is a problem in itself.
- **There. Are various way! To detect Sentence Boundary.**
- Why do we do Sentence Boundary Detection?
    - So that we have to look at a smaller sentence and decide the next steps!
    - Long sentences are difficult for a normal human being to read. And so for a machine to understand!
- Erroneous Sentence Boundary might lead to loss of important data semantics.

From Sentences to Words

- Breaking the stone into minute pieces - cleaning, experimenting and polishing might lead to diamonds.

- English is much easier language because we can split sentences on white-spaces to get the words.
- But it is a challenge for many different languages.

Stemming and Lemmatization

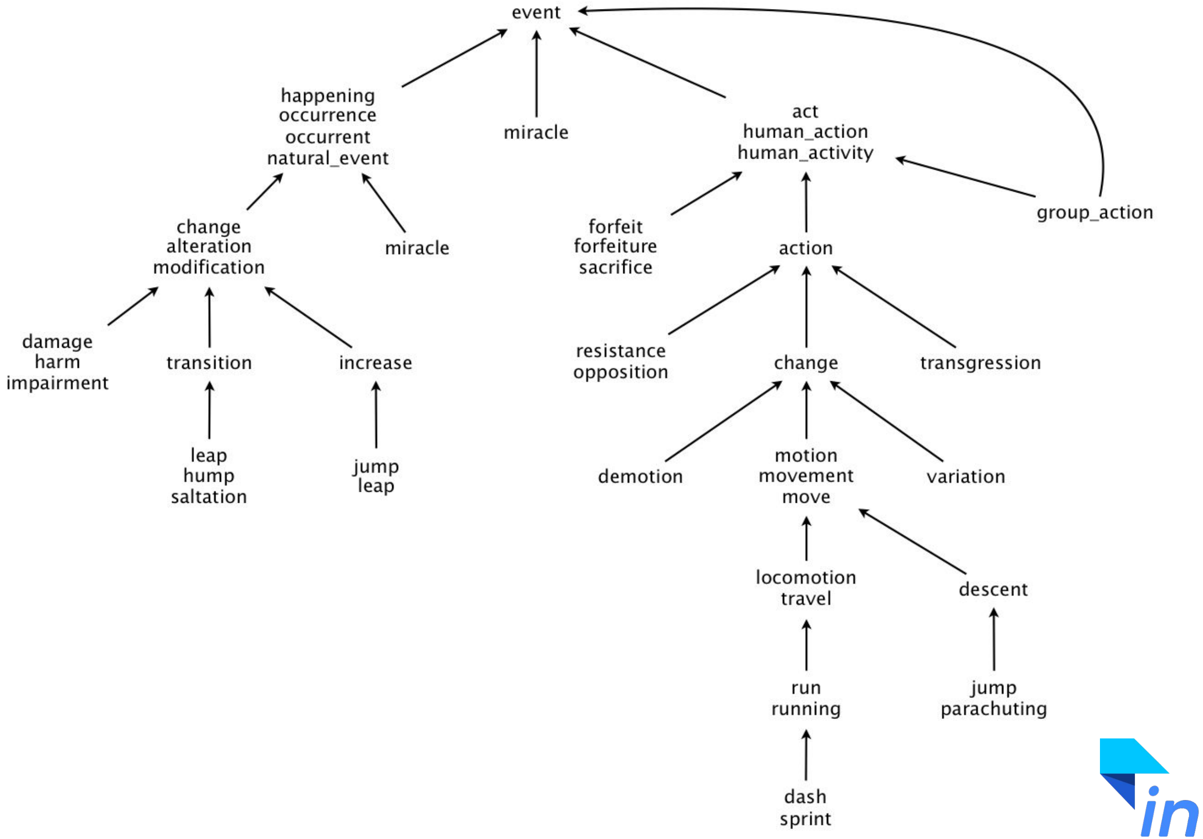
- **“Word normalization.”**
- The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.
- **Stemming**: Stemming is the process of converting the words of a sentence to its non-changing portions.
  - Example : The stem of amusing, amusement, and amused would be “amus”.
  - Different types of Stemmers, used commonly: Lancaster Stemmer, Porter Stemmer. Snowball Stemmer etc.
- **Lemmatization**: Lemmatization is the process of converting the words of a sentence to its dictionary form.
  - Example: Lemma for amusement, amusing, and amused will be “amuse”.
  - Lemma for “women” will be “woman”

WordNet and Synsets

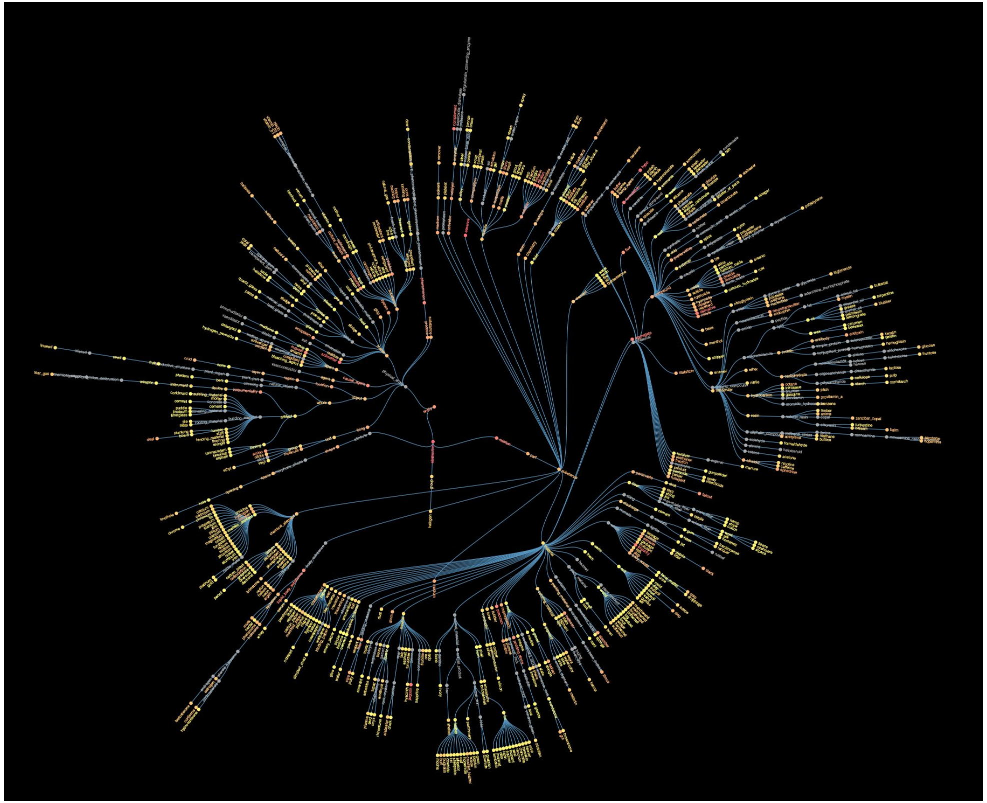
- WordNet is a lexical database for the English language.
- Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.
- Synsets are interlinked by means of conceptual-semantic and lexical relations.
- WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings.
- Synsets: Synonyms--words that denote the same concept and are interchangeable in many contexts--are grouped into unordered sets.

For more details go to: https://wordnet.princeton.edu/

WordNet and Synsets



WordNet and Synsets



Stanford Core NLP

- Stanford CoreNLP provides a set of human language technology tools.
- It can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, mark up the structure of sentences in terms of phrases and syntactic dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract particular or open-class relations between entity mentions, get the quotes people said, etc.
- Stanford CoreNLP integrates many tools including the part-of-speech (POS) tagger, the named entity recognizer (NER), the parser, the coreference resolution system, sentiment analysis, bootstrapped pattern learning, and the open information extraction tools.
- More details can be found at: https://stanfordnlp.github.io/CoreNLP/

Stanford Core NLP

- Tokenization
- Sentence Splitting
- Lemmatization
- Parts of Speech
- Named Entity Recognition
- Dependency Parsing
- Coreference Resolution
- Natural Logic
- Open Information Extraction
- Sentiment

Stanford Core NLP

- **Part-of-Speech (POS)**:
  - A category or a class to which a word is assigned in accordance with its syntactic functions.
  - In English the main parts of speech are noun, pronoun, adjective, determiner, verb, adverb, preposition, conjunction, and interjection.
- **Name-Entity-Relation (NER)**:
  - For English, by default, this annotator recognizes named (PERSON, LOCATION, ORGANIZATION, MISC), numerical (MONEY, NUMBER, ORDINAL, PERCENT), and temporal (DATE, TIME, DURATION, SET) entities.
  - 12 classes by default.

Stanford Core NLP visualisation at: http://nlp.stanford.edu:8080/corenlp/

Dependency Parsing

- Relations among the words are illustrated above the sentence with directed labeled arcs, from heads to dependents. This is called Dependency Structure.
- When this Dependency Structure is parsed it form a tree like structure called Dependency Tree.
- It always has a root node that marks the root of the tree and head of the entire structure.
- The relationship between nodes directly encode important information that is often buried in complex sentences.

NLTK

- Natural Language Toolkit (NLTK) is a platform for building Python programs to work with human language data.
- It provides easy-to-use interface to over 50 corpora and lexical resources.
- It provides text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

spaCy

- spaCy is an open-source software library for advanced Natural Language Processing, written in the Python programming language.
- It offers the fastest syntactic parser in the world. (as they claim)
- spaCy handles large-scale extraction tasks

For more information go to: https://spacy.io/

spaCy

- Features
  - Tokenization
  - Named entity recognition
  - Support for 28+ languages
  - 13 statistical models for 8 languages
  - Pre-trained word vectors
  - Easy deep learning integration
  - Part-of-speech tagging
  - Labelled dependency parsing
  - State-of-the-art speed
  - Robust, rigorously evaluated accuracy

infocusp
Innovations Pvt. Ltd.

What next?

- So, we saw different libraries and methods to normalize words, find linguistic and semantic relations between them but what do we do with it?
- Or the real question is how to convert the human language to machine understandable language.

- That’s where Vectors comes into play!

Machine friendly representation of documents

- Machines are not able to interpret language in the way humans do.
- Machines understand numbers and hence arises a need to convert words, sentences and documents into machine interpretable numerical form.

Why and What is Vectorization?

- The ability to process natural human language is one of the things that makes machine learning algorithms so powerful.
- However, when doing natural language processing, words must be converted into vectors that machine learning algorithms can make use of.
- If a word can be translated into a single number, documents can be translated to a vector.
- Due to varsity of words, these vectors tend to be high-dimensional.
- This whole process of representing text data into numbers is called “**Word Vectorization**” or “**One-Hot Representation**” or “**Word Embedding**”.

Bag-of-Words (BoW)

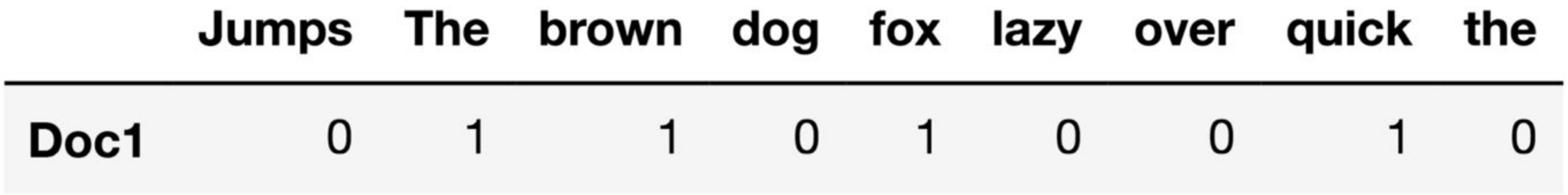
- What is the best representation of documents in numerical form?
- Concept of Vocabulary

©InFoCusp Innovations Pvt. Ltd. 2019

Count Vectorizer

- Simplest model or algorithm which counts the frequency of words in a document to create a document vector.
- Count the number of times each term in the vocabulary appears in the document.
- The vector formed by this model is nothing but a frequency vector of vocabulary words.

Sample Document: “The quick brown fox. Jumps over the lazy dog!”

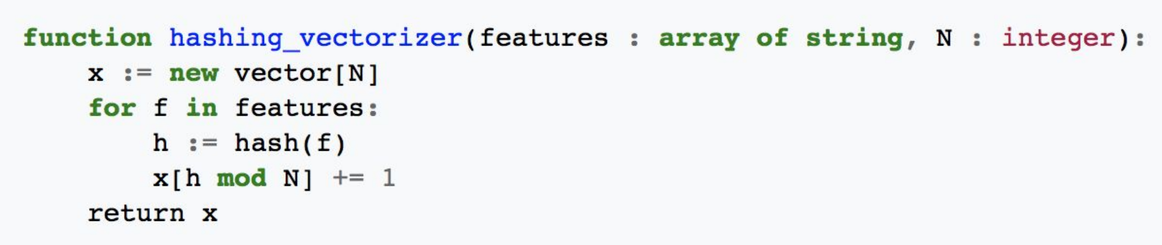


	Jumps	The	brown	dog	fox	lazy	over	quick	the
Doc1	0	1	1	0	1	0	0	1	0

©InFoCusp Innovations Pvt. Ltd. 2019

Hashing Vectorizer

- Instead of building a vocabulary and using that as feature, we define a hashing function which will help us generate a vector of a pre-defined length by hashing the original features.
- Example:

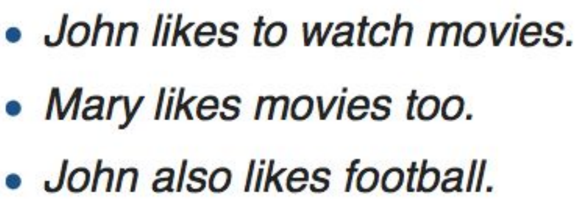


```
function hashing_vectorizer(features : array of string, N : integer):
    x := new vector[N]
    for f in features:
        h := hash(f)
        x[h mod N] += 1
    return x
```

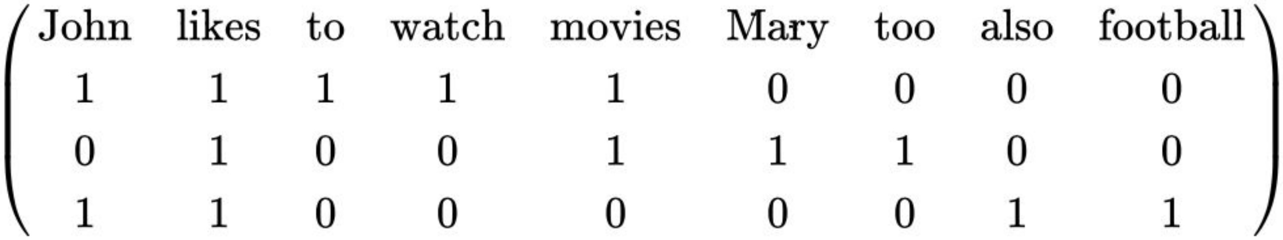
- Suppose, our feature vector is ["cat","dog","cat"] and hash function is: **hash("cat") = 2** and **hash("dog")=1**.
- Suppose, output feature vector dimension (N) to be 4. Then output x will be [0,2,1,0].

Hashing Vectorizer

- When we create vectors by using vocabulary, it utilizes a lot of memory.
- Let’s take the example of three documents: (Vocab size = 9)



- *John likes to watch movies.*
- *Mary likes movies too.*
- *John also likes football.*

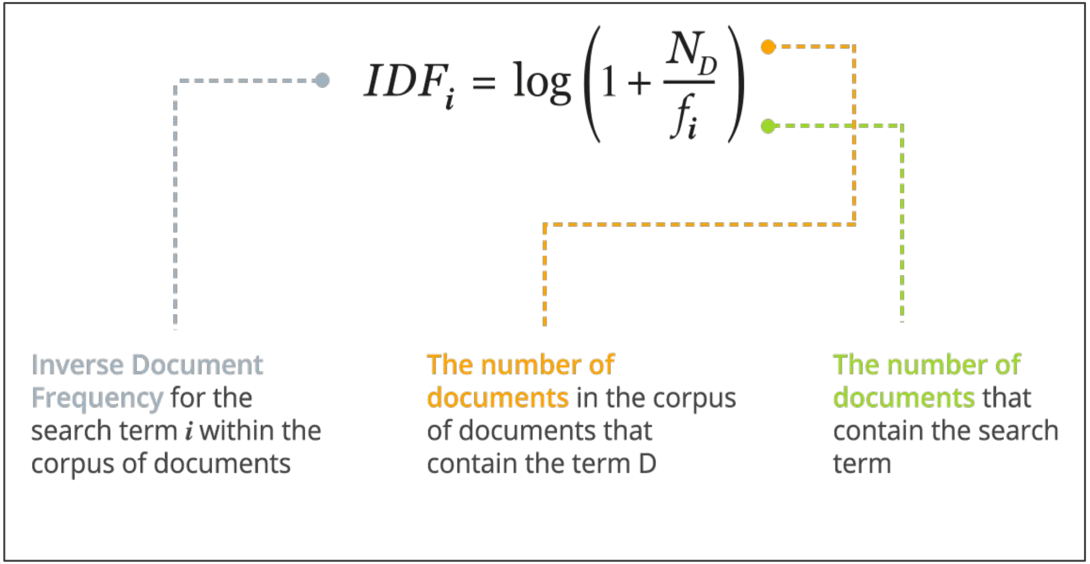


$$\left(\begin{array}{ccccccccc} \text{John} & \text{likes} & \text{to} & \text{watch} & \text{movies} & \text{Mary} & \text{too} & \text{also} & \text{football} \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

©InFoCusp Innovations Pvt. Ltd. 2019

Term Frequency - Inverse Document Frequency (TF-IDF)

$TF_{i,D}$ = (Number of times the term i appears in the document D) / (Total number of words in the document D)



$$IDF_i = \log\left(1 + \frac{N_D}{f_i}\right)$$

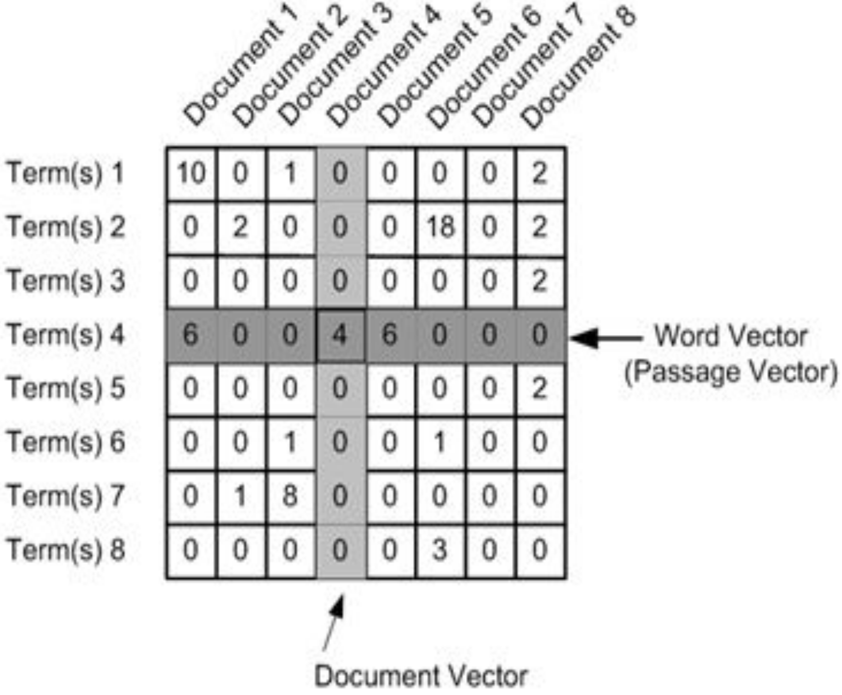
Inverse Document Frequency for the search term *i* within the corpus of documents

The number of documents in the corpus of documents that contain the term D

The number of documents that contain the search term

*TF-IDF = TF * IDF*

TF-IDF Matrix representation of corpus of documents



	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8
Term(s) 1	10	0	1	0	0	0	0	2
Term(s) 2	0	2	0	0	0	18	0	2
Term(s) 3	0	0	0	0	0	0	0	2
Term(s) 4	6	0	0	4	6	0	0	0
Term(s) 5	0	0	0	0	0	0	0	2
Term(s) 6	0	0	1	0	0	1	0	0
Term(s) 7	0	1	8	0	0	0	0	0
Term(s) 8	0	0	0	0	0	3	0	0

Word Vector
(Passage Vector)
Document Vector

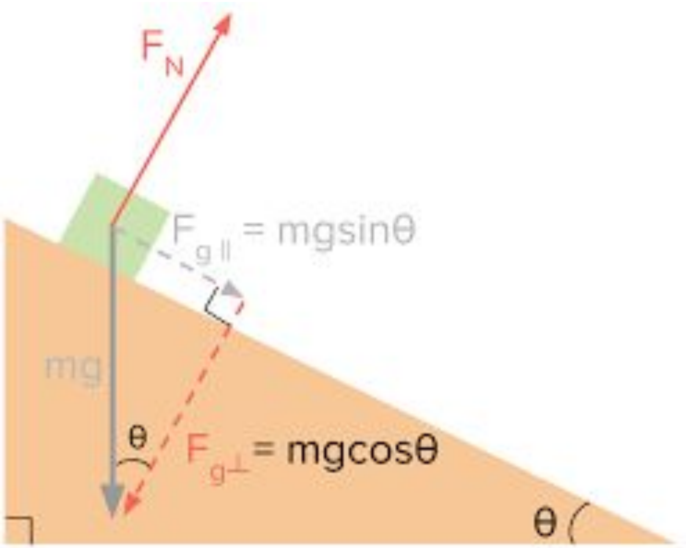
Recap

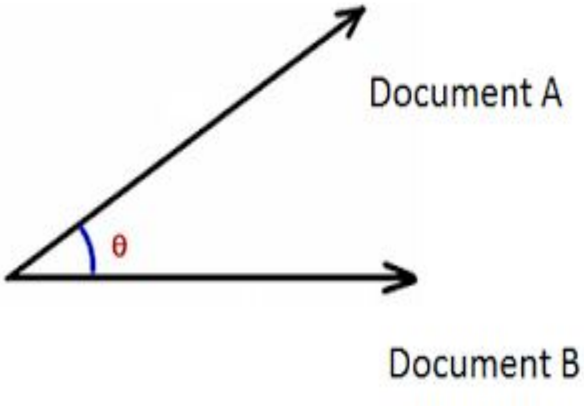
- What is NLP?
- Standard process and pipeline.
- Open-source/Freely available tools and libraries.
- Need of vectors.
- Bag of Words (BoW)
    - Count Vectorizer
    - Hashing Vectorizer
    - TF-IDF Vectorizer

Why vectors?

- One of the main focus of NLP is to compare different documents and look at the similarity between documents or similarity between the word-vectors.
- The vector representation of the documents makes the calculation of similarity very easy.
- Let’s discuss one of the commonly used similarity measures

Cosine Similarity





$$cosine\ similarity = cos(\theta) = \frac{A.B}{\|A\|\,.\,\|B\|}$$

n-gram Model

- Bag of Words Model considered each word independently. It is an unordered way of looking at the text without using the context.
- n-gram predicts the sequence of words.
- “n” in n-gram denotes the length of sequence. Based on varying “n” we have:
  - n=1 - monograms / unigrams
  - n=2 - bigrams / digrams
  - n=3 - trigrams and so on...

n-gram Model

- n-gram computes the probability of ith item given a sequence of previous (i-1) items.

$$P(x_i \mid x_{i-(n-1)}, \ldots, x_{i-1})$$

- For Example, we want the machine to learn that the words “Machine” & “Learning” are commonly used together. “Machine Learning” is a bigram.
- The probabilities are estimated using the **relative frequencies** of observed outcomes. This process is called **Maximum Likelihood Estimation (MLE) of n-gram models**.

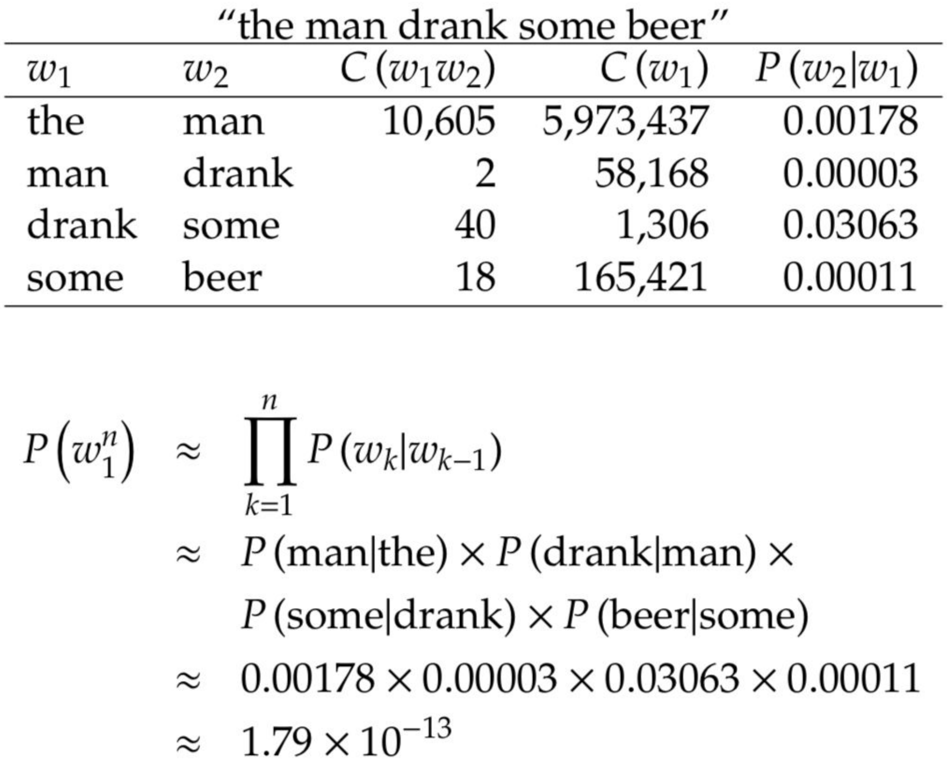
Example

- Compute probability of trigram: “I like bananas”

$$P\left(\text{bananas}|\text{i like}\right) = \frac{C\left(\text{i like bananas}\right)}{C\left(\text{i like}\right)}$$

Example: Estimate sequence using bigrams

Assumption: Probability of next word depends only on previous word.



“the man drank some beer”

w_1	w_2	$C(w_1 w_2)$	$C(w_1)$	$P(w_2 \mid w_1)$
the	man	10,605	5,973,437	0.00178
man	drank	2	58,168	0.00003
drank	some	40	1,306	0.03063
some	beer	18	165,421	0.00011

$$\begin{aligned} P\left(w_1^n\right) &\approx \prod_{k=1}^{n} P\left(w_k \mid w_{k-1}\right) \\ &\approx P(\text{man} \mid \text{the}) \times P(\text{drank} \mid \text{man}) \times \\ &\quad\ P(\text{some} \mid \text{drank}) \times P(\text{beer} \mid \text{some}) \\ &\approx 0.00178 \times 0.00003 \times 0.03063 \times 0.00011 \\ &\approx 1.79 \times 10^{-13} \end{aligned}$$

infocusp Innovations Pvt. Ltd.

k-skip-n-gram Model (Skip Grams)

- “Skip-grams" (or "k-skip-n-grams") are sequences of ordered but not-necessarily-adjacent (thus "skipped") units, where the gaps can be at most "k" units long.
- For example, in the sentence:

  "**The quick brown fox jumped over the lazy dog**"

  bigrams (2-grams): "The quick", "quick brown", "brown fox", fox jumped", "jumped over", "over the", "the lazy", and "lazy dog"

  1-skip-2-grams: all of the bigrams + "the, brown", "quick, fox", "brown, jumped", "fox, over", "jumped, the", "over, lazy", "the, dog".

- **Better but complex in terms of space and time complexity!**

Problems...

- The issue with BoW and n-grams models is the lack of context.
  - BoW could not capture any essences of the semantics while n-grams and its variant could only capture the backward semantics.
  - Space and Time Complexity
- What can be the solution?

Word2Vec

- This algorithm is used for learning vector representations of words, called “Word Embeddings”.
- These models are shallow, **two-layer neural networks** that are trained to reconstruct linguistic contexts of words.
- Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space.
- Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

One can use the **existing pre-trained word vectors** like GloVe, GoogleNews-vectors-negative300 etc. Standard words vectors are open-source and available freely.

Word2Vec

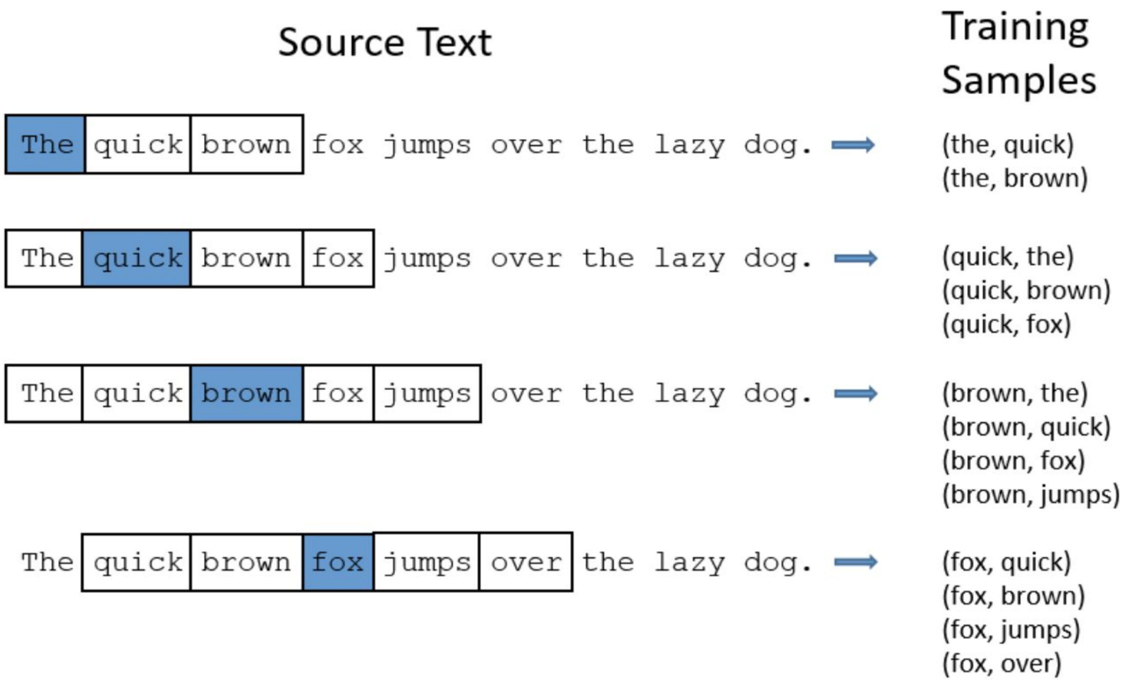
- The key benefit of the approach is that high-quality word embeddings can be learned efficiently (low space and time complexity), allowing larger embeddings to be learned (more dimensions) from much larger corpora of text (billions of words).

Word2Vec

- Two popular variants:
  - **Window** (We will use this variant during the upcoming slides.)
  - Full Document (leads to Topic Modeling)

- Two popular models:
  - Continuous Skip-Grams
  - Continuous Bag of Word (CBOW)

Generation of Training data for Word2Vec



Source Text	Training Samples
The quick brown fox jumps over the lazy dog. →	(the, quick) (the, brown)
The **quick** brown fox jumps over the lazy dog. →	(quick, the) (quick, brown) (quick, fox)
The quick **brown** fox jumps over the lazy dog. →	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown **fox** jumps over the lazy dog. →	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

Training Word2Vec

- We’ll train the neural network by feeding word pairs found in our training documents.

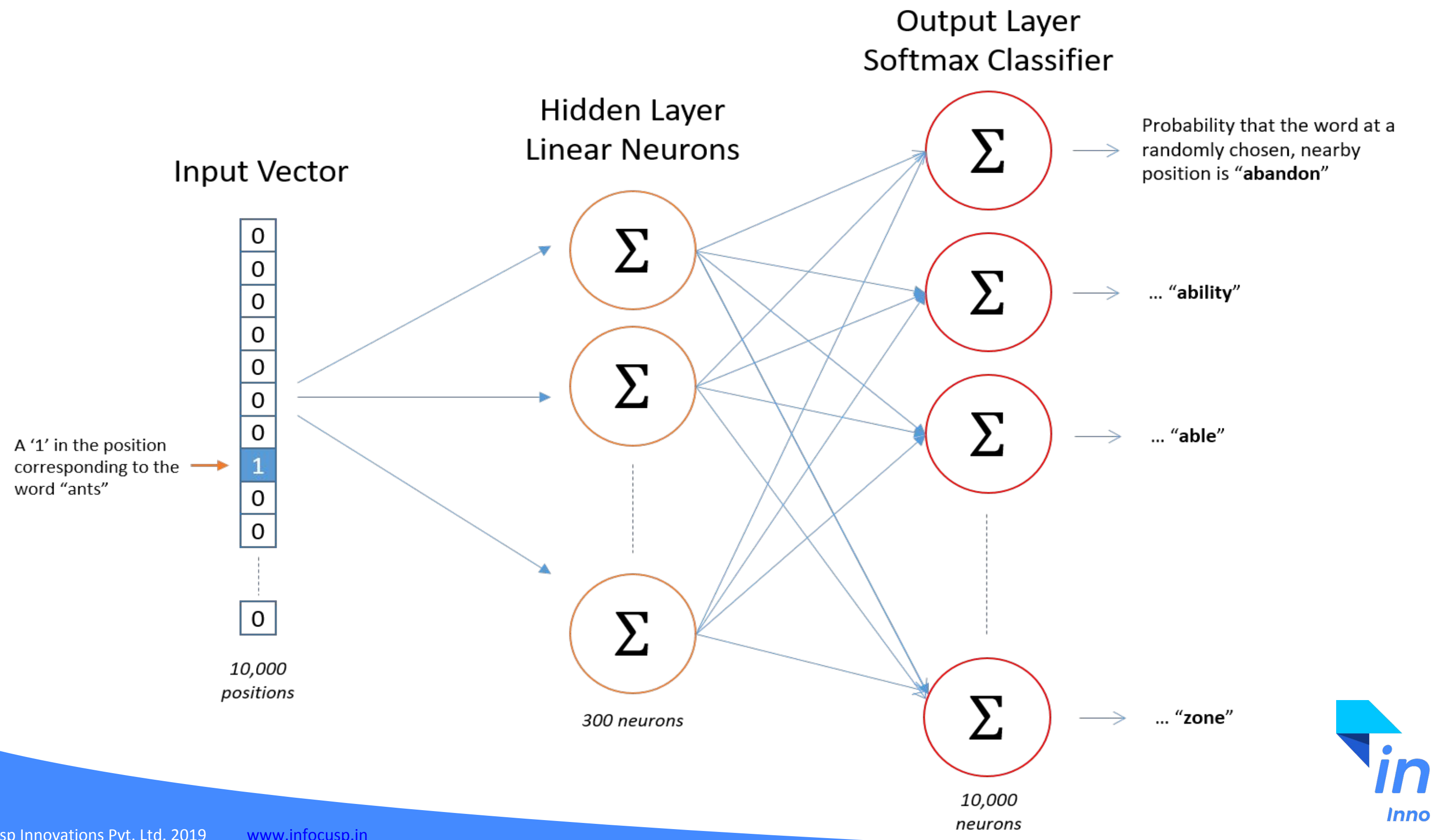
- The output probabilities are going to relate to how likely it is find each vocabulary word nearby our input word.

Continuous Skip-Grams Model

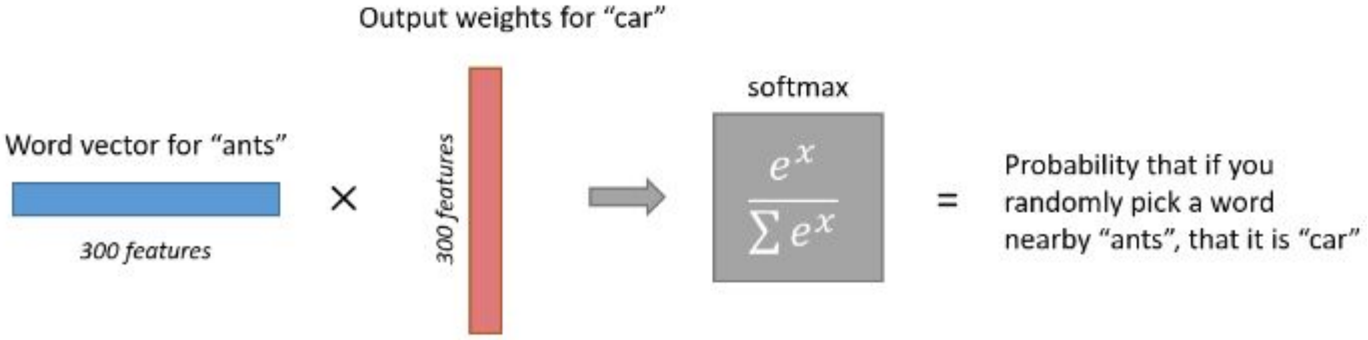
- Given a central word, it will predict its neighbouring words with different probabilities.

©InFoCusp Innovations Pvt. Ltd. 2019

Architecture of Continuous Skip-Grams Model



Interpretation of Skip-Grams Output



Continuous Bag of Words (CBOW) model

- Predict target word from the surrounding contextual words.

Continuous Skip-Grams and CBOW in a nutshell

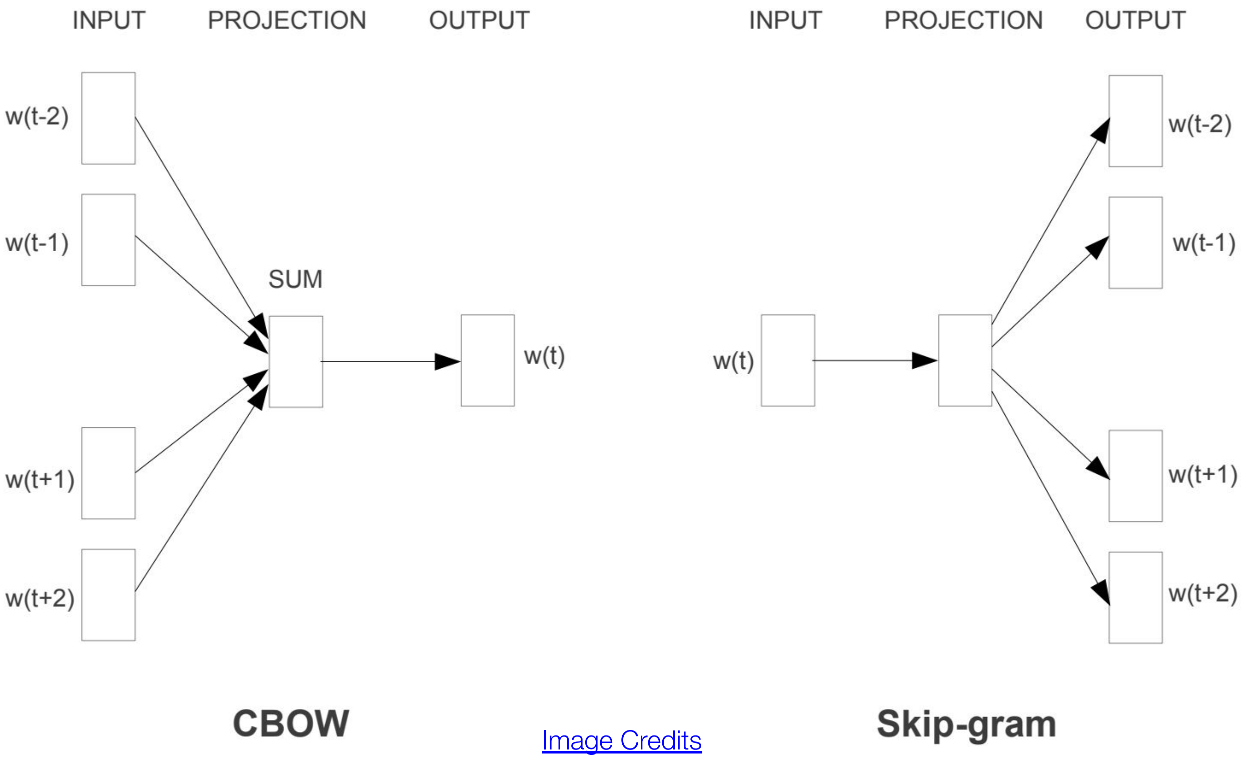
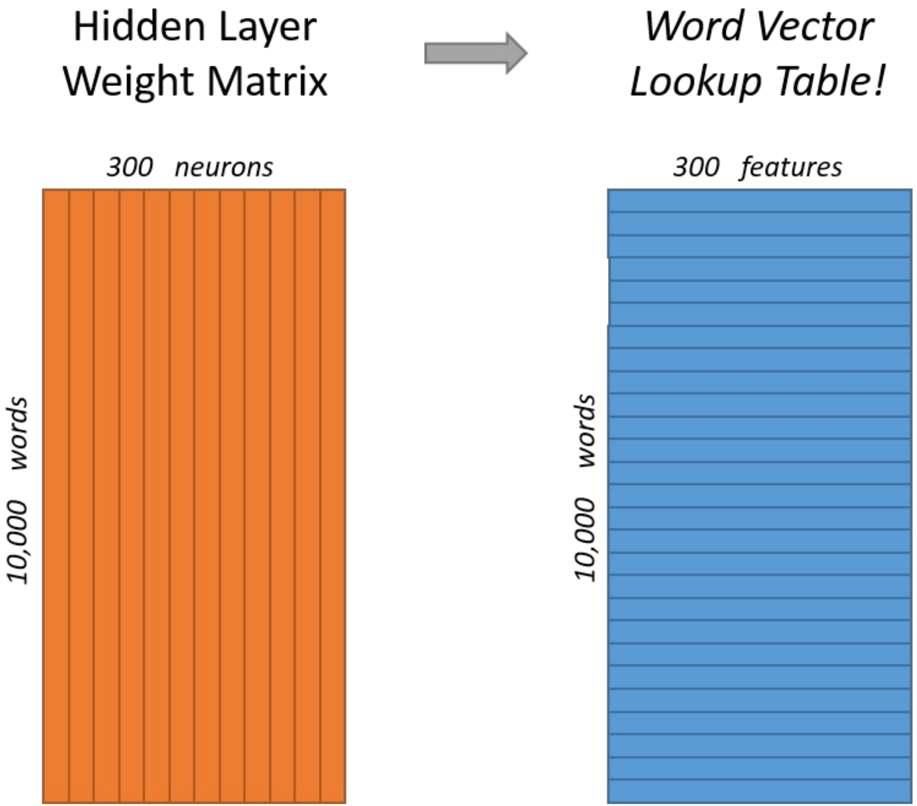


Image Credits

GloVe


Hidden Layer
Weight Matrix
Word Vector
Lookup Table!
300 neurons
300 features
10,000 words
10,000 words

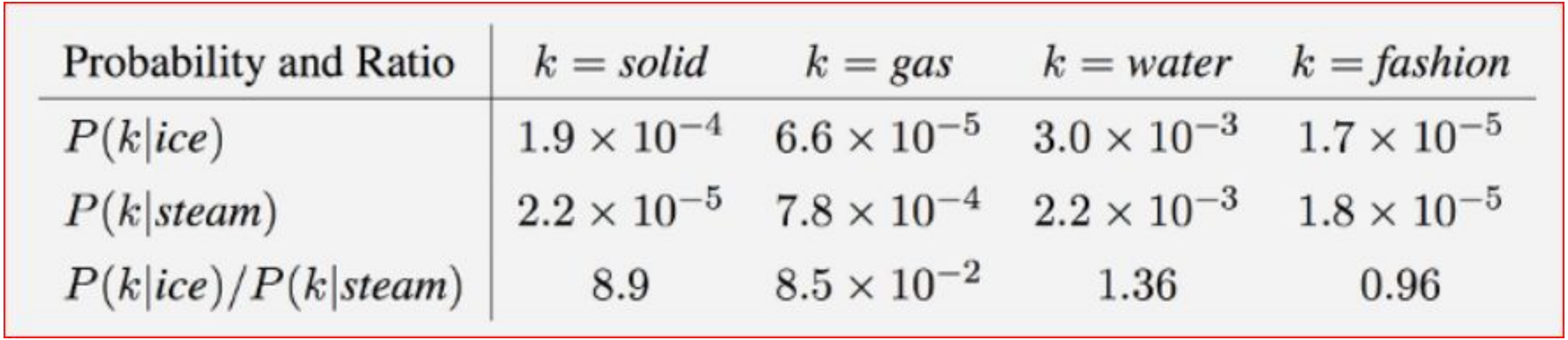
GloVe

- GloVe : Global Vectors for Word Representation
- GloVe is an **unsupervised learning** algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.
- Trained on extremely large data set, GloVe comes in a variant from 50D to 300D.
- The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning.

GloVe

- The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence.
- Owing to the fact that the logarithm of a ratio equals the difference of logarithms, this objective associates (the logarithm of) ratios of co-occurrence probabilities with vector differences in the word vector space.
- Because these ratios can encode some form of meaning, this information is encoded as vector differences as well.
- For this reason, the resulting word vectors perform very well on word analogy tasks, such as in the word2vec package.

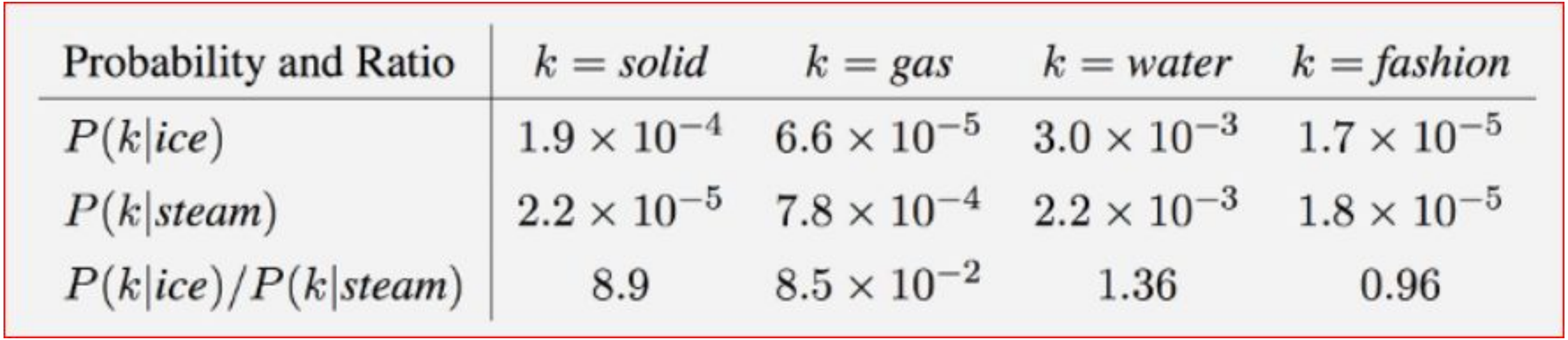
GloVe



Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

- As one might expect, ice co-occurs more frequently with solid than it does with gas, whereas steam co-occurs more frequently with gas than it does with solid.
- Both words co-occur with their shared property water frequently, and both co-occur with the unrelated word fashion infrequently.

GloVe



Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

- Only in the ratio of probabilities does noise from non-discriminative words like water and fashion cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam.
- In this way, the ratio of probabilities encodes some crude form of meaning associated with the abstract concept of thermodynamic phase.

NLP Tasks

- These different algorithms and models for Word Embeddings coupled with Machine Learning algorithms like SVM, Linear Regression, Logistic Regression etc. can be used to solve numerous NLP tasks.

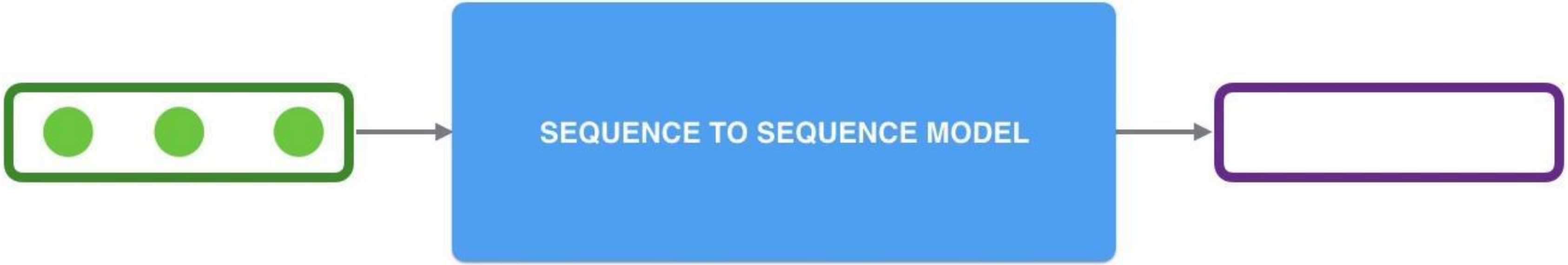
Advance Topics in Natural Language Processing

With the recent advances in Machine Learning and Deep Learning techniques,

let us present a gist of some of the advance topics of

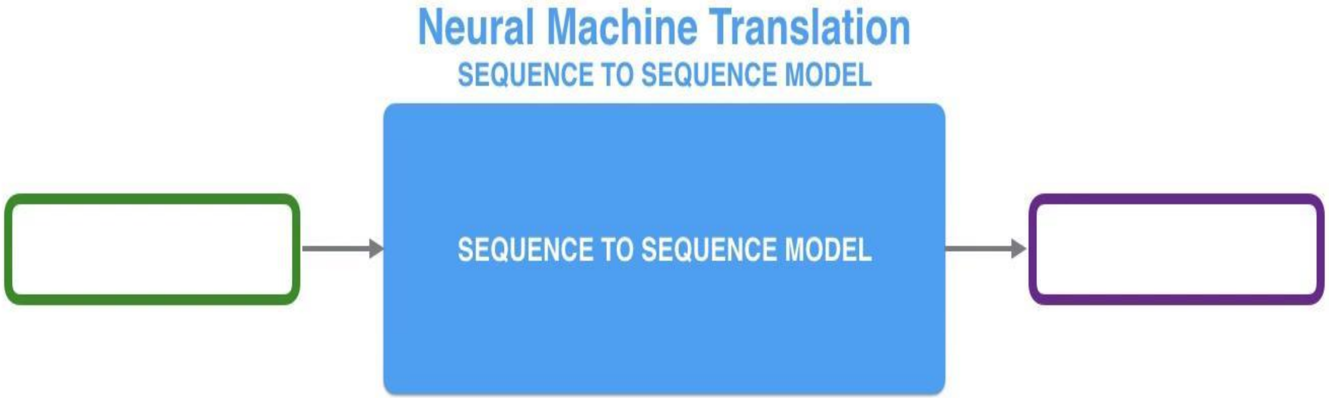
Natural Language Processing and with their application in Automatic Question Answering System (AQuA).

Processing sequence of items



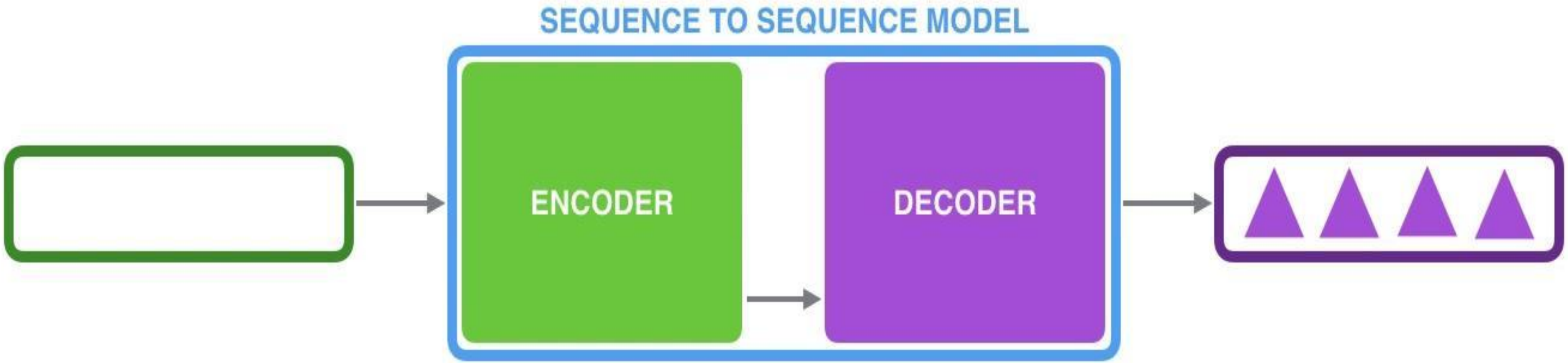
Credits: http://jalammar.github.io/illustrated-bert/

Sequence to Sequence example


Neural Machine Translation
SEQUENCE TO SEQUENCE MODEL
SEQUENCE TO SEQUENCE MODEL

Credits: http://jalammar.github.io/illustrated-bert/

Under the black box



Credits: http://jalammar.github.io/illustrated-bert/

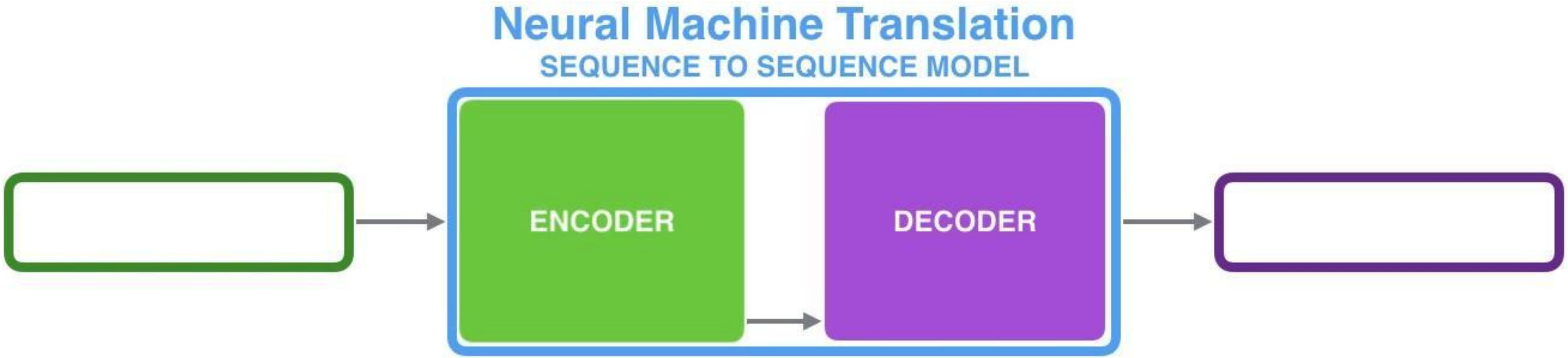
Example

What does this mean ?

“je suis étudiant”

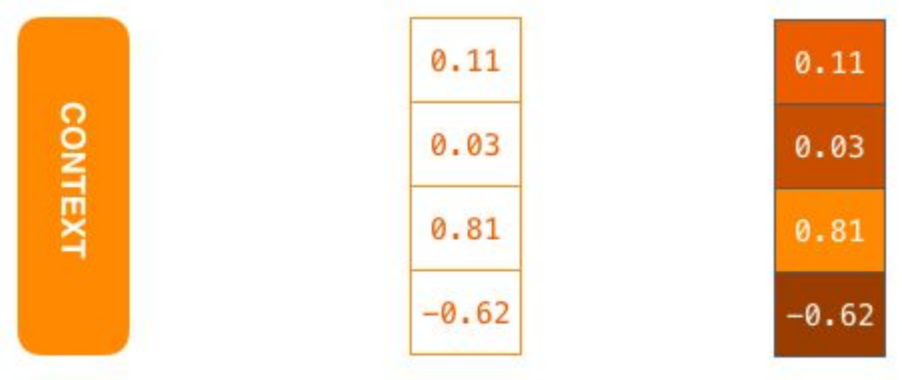
infocusp
Innovations Pvt. Ltd.

Example



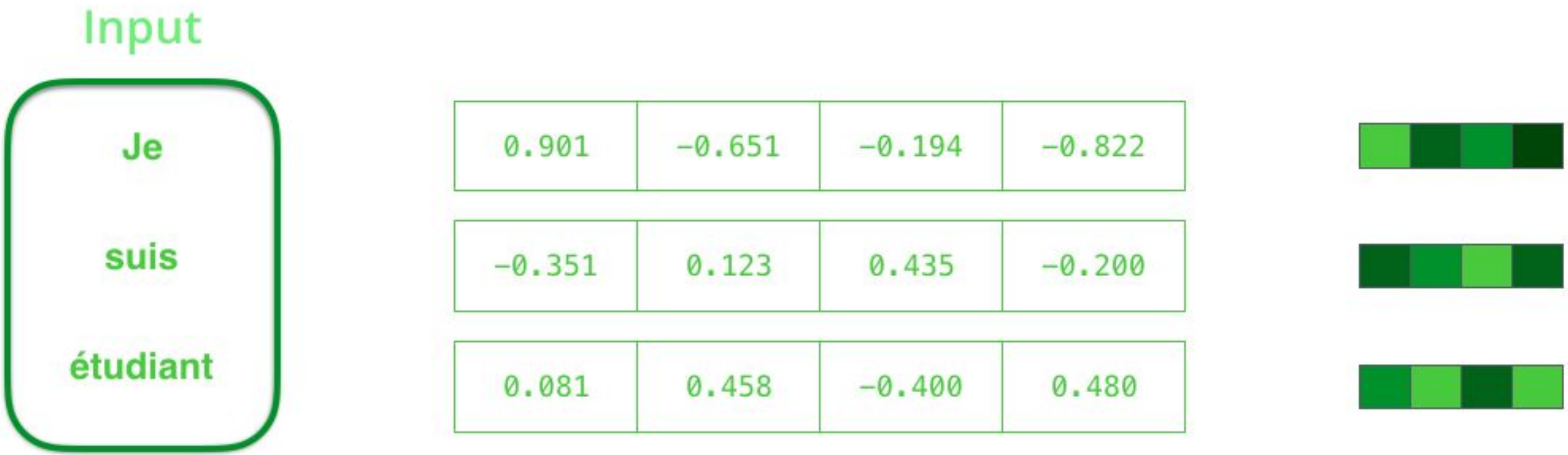
Credits: http://jalammar.github.io/illustrated-bert/

The Context Vector



Credits: http://jalammar.github.io/illustrated-bert/

Convert input words to vectors

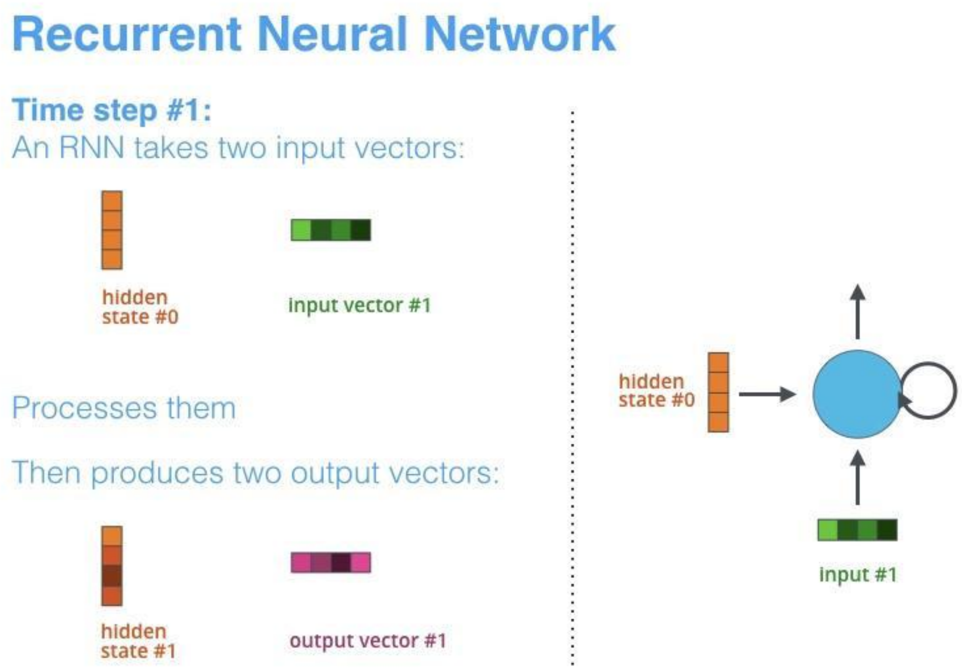


Input

Je	0.901	-0.651	-0.194	-0.822
suis	-0.351	0.123	0.435	-0.200
étudiant	0.081	0.458	-0.400	0.480

Credits: http://jalammar.github.io/illustrated-bert/

Encoder and Decoder structure - RNN



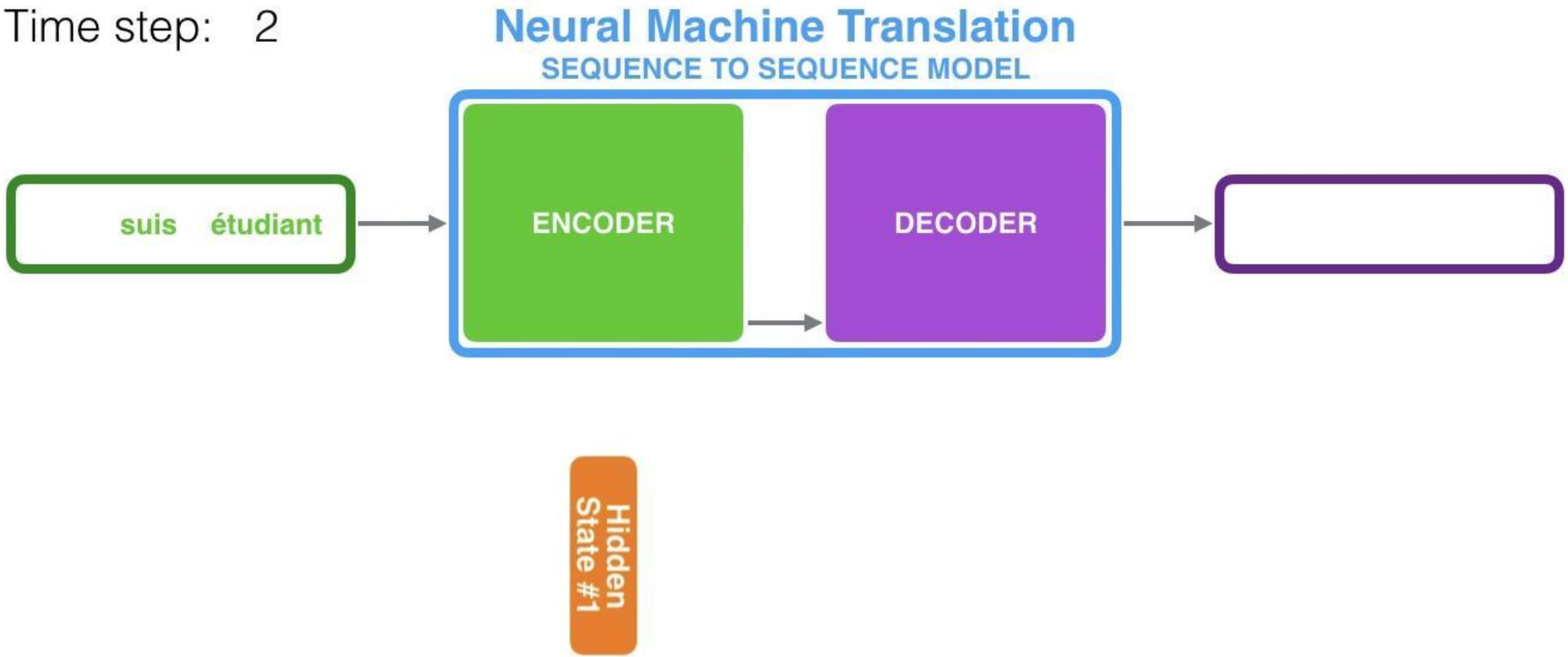
Recurrent Neural Network

Time step #1:
An RNN takes two input vectors:

Processes them

Then produces two output vectors:

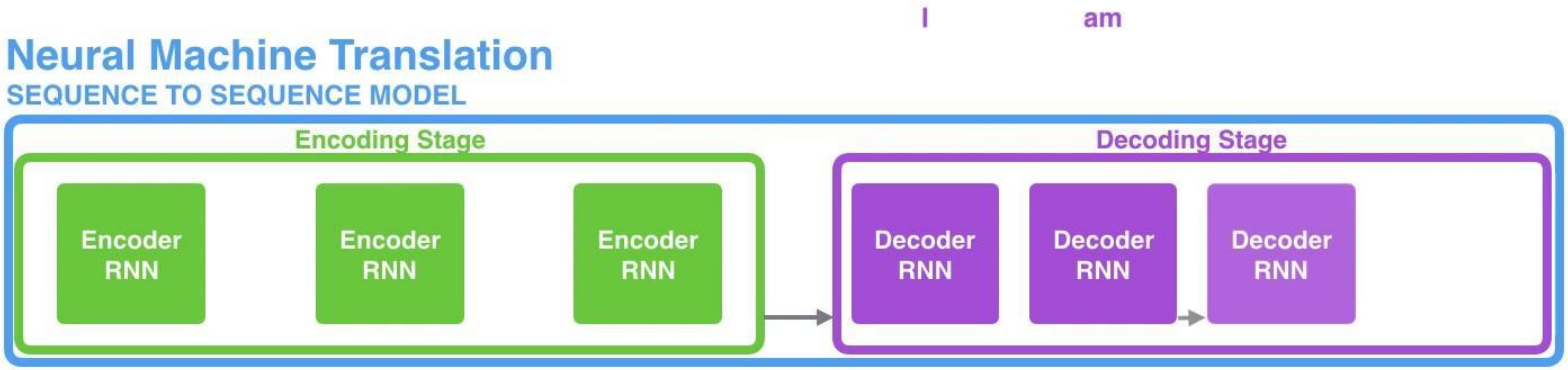
Credits: http://jalammar.github.io/illustrated-bert/



Credits: http://jalammar.github.io/illustrated-bert/

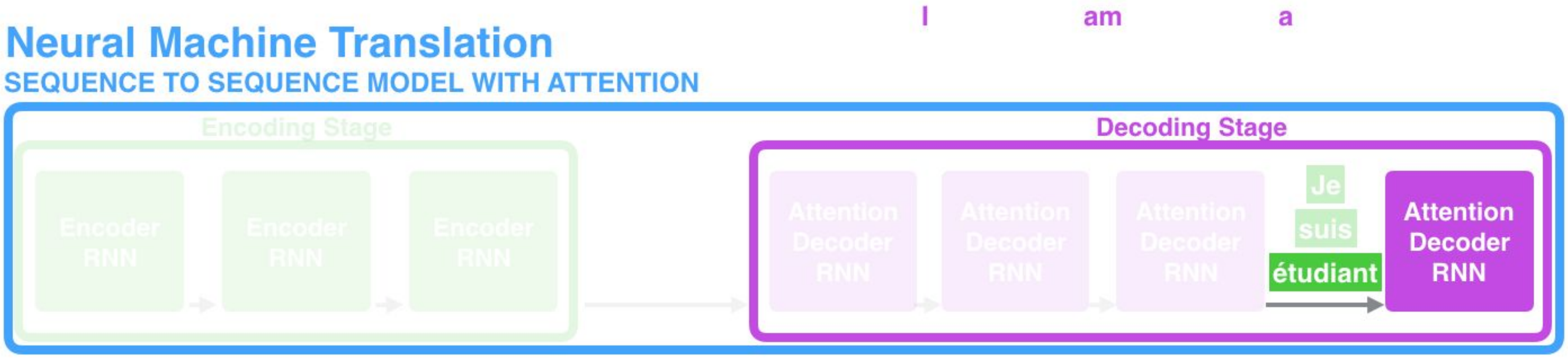
Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL

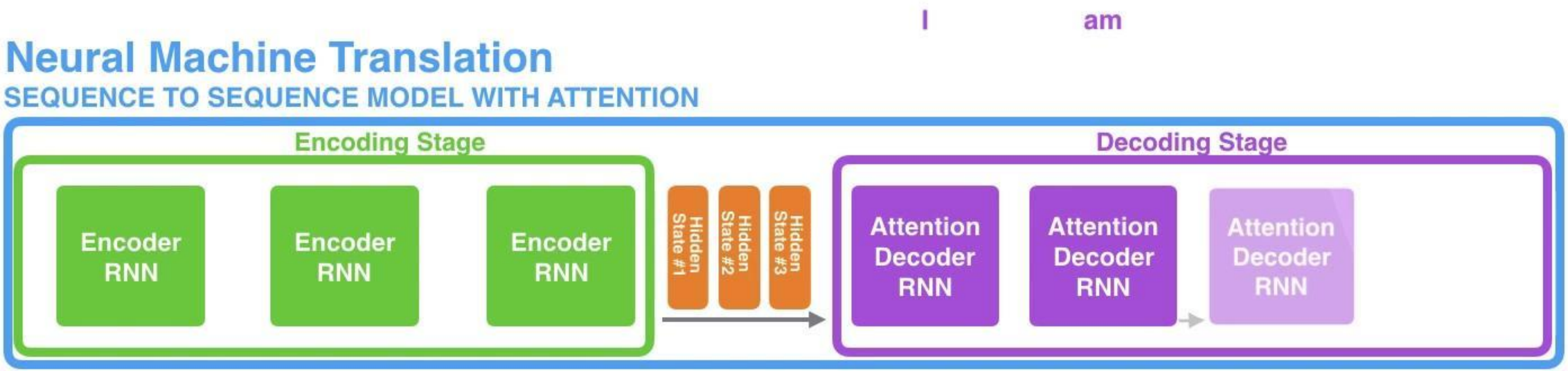


Credits: http://jalammar.github.io/illustrated-bert/

Time step: 7



Credits: http://jalammar.github.io/illustrated-bert/



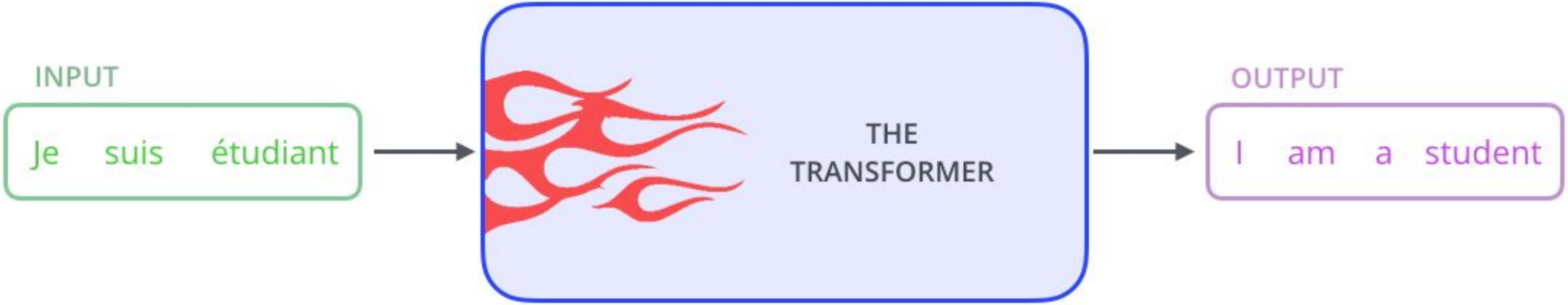
Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL WITH ATTENTION

Credits: http://jalammar.github.io/illustrated-bert/

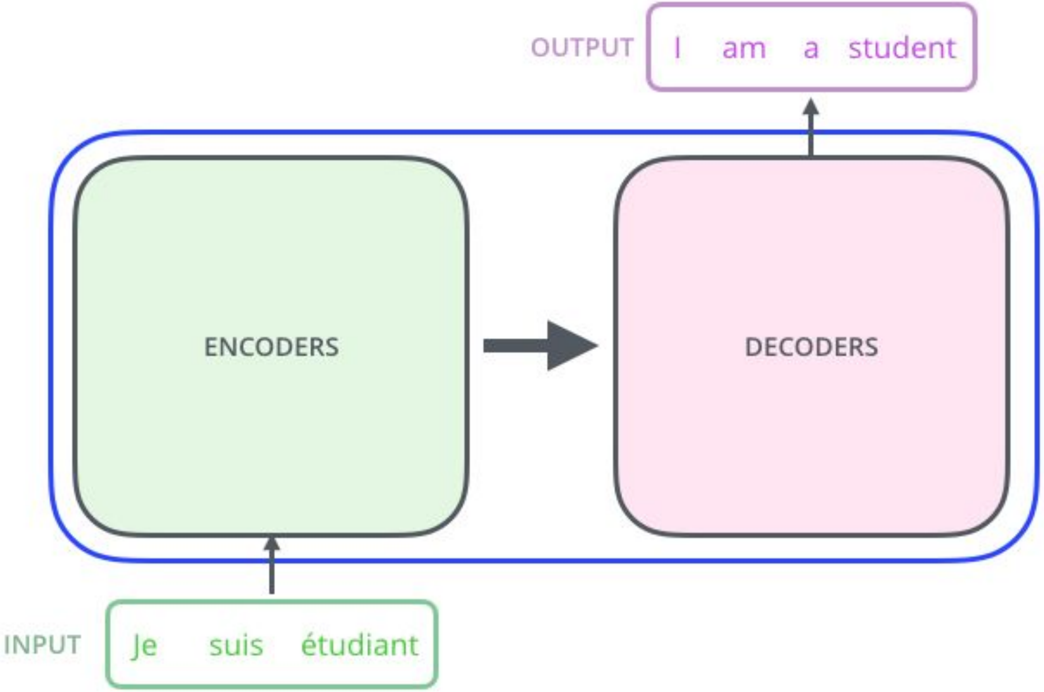
Credits: http://jalammar.github.io/illustrated-bert/

The Transformers!



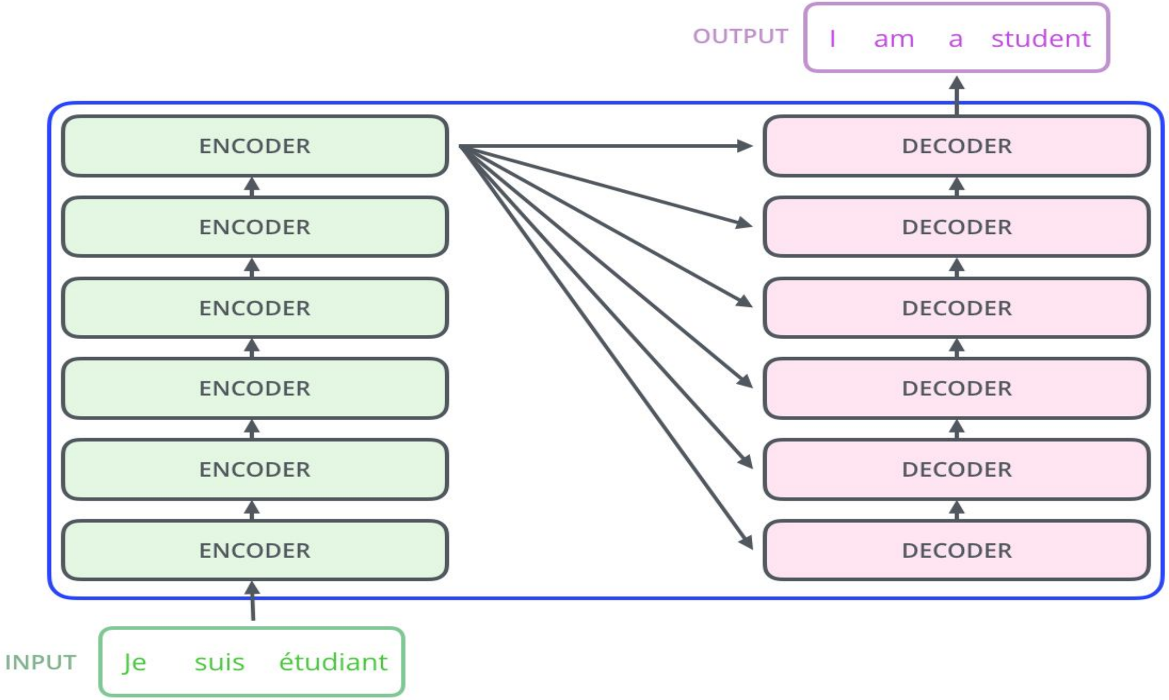
Credits: http://jalammar.github.io/illustrated-bert/

Inside the Transformer!



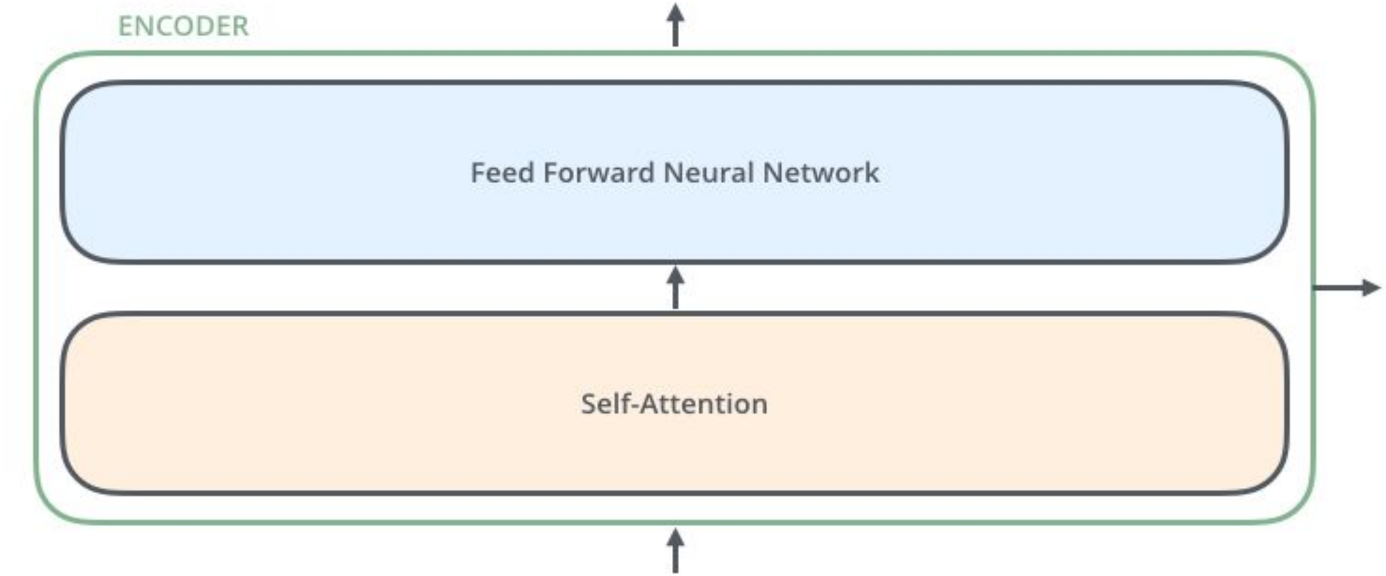
Credits: http://jalammar.github.io/illustrated-bert/

Inside the Transformer



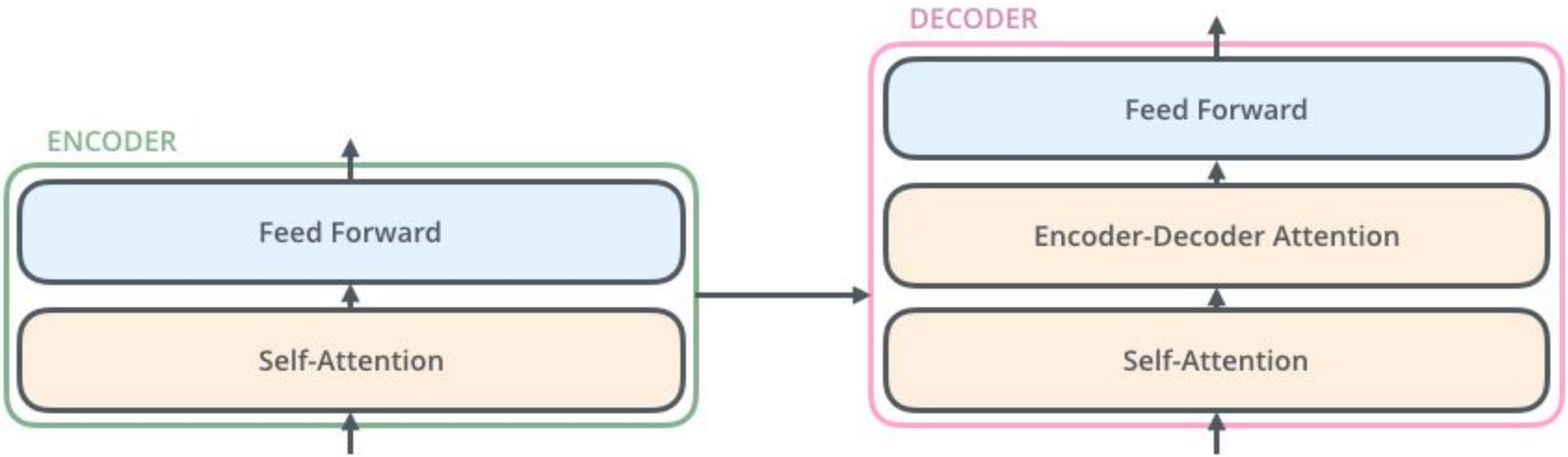
Credits: http://jalammar.github.io/illustrated-bert/

Inside the Encoder!



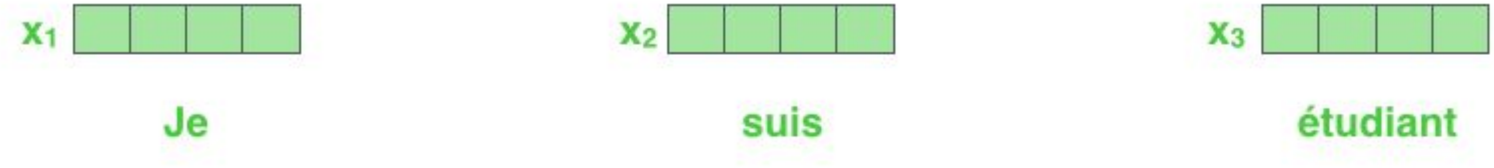
Credits: http://jalammar.github.io/illustrated-bert/

The Decoder!



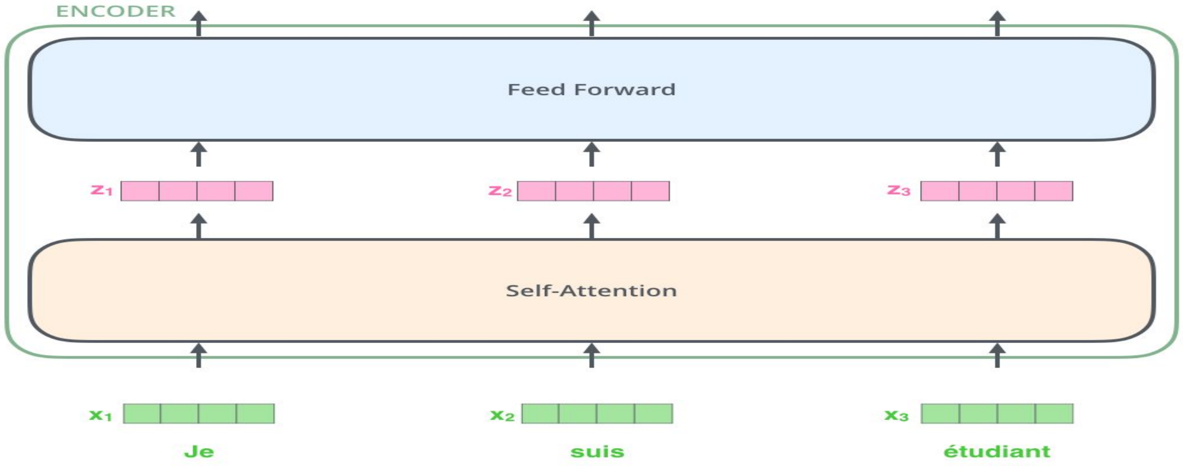
Credits: http://jalammar.github.io/illustrated-bert/

The Vectors



Credits: http://jalammar.github.io/illustrated-bert/

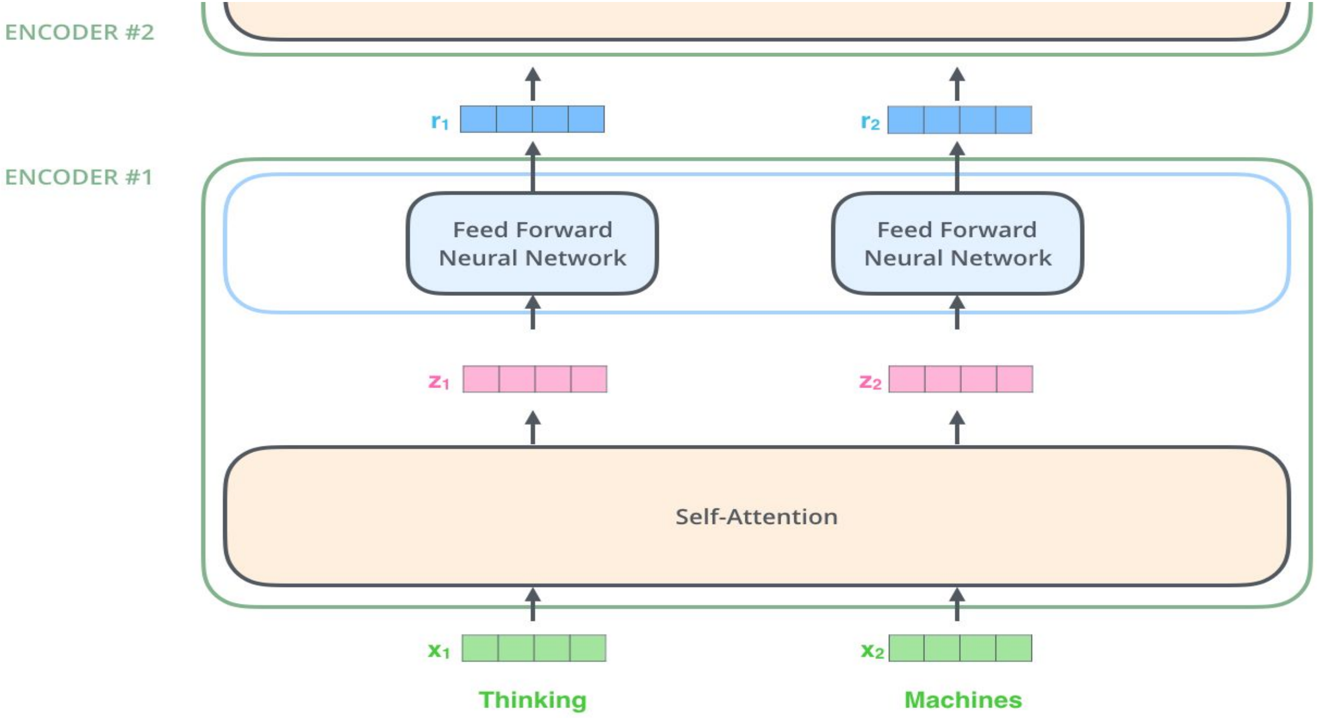
Encoder Flow



- Only the first encoder block will get the input vectors (word embeddings).
- The word in each position flows through its own path in the encoder.
- There are dependencies between each path in the self attention layer.
- The feed forward neural network does not have dependencies and thus it can be executed in parallel.

Credits: http://jalammar.github.io/illustrated-bert/

Encoder Flow



Credits: http://jalammar.github.io/illustrated-bert/

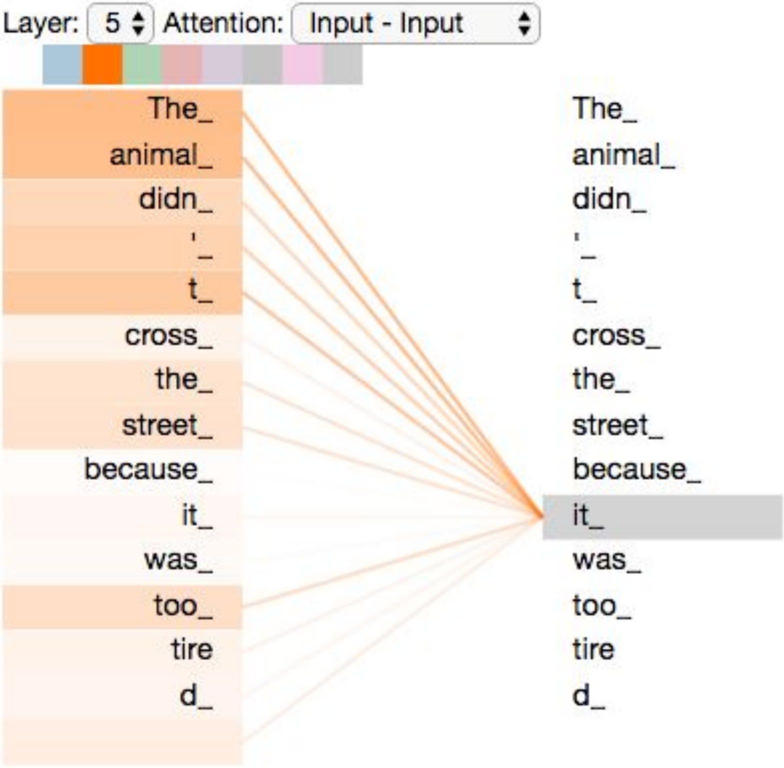
What exactly is “Self-Attention”?

“The animal didn't cross the street because it was too tired”

- What does “it” refer to? Street or animal?
- When model is processing “it”, the self attention allows it to associate “it” with animal.
- While processing a word, the self attention layer allows it to look at other positions in the input sequence for clues that can help encode that word better.

How to pay “attention”?

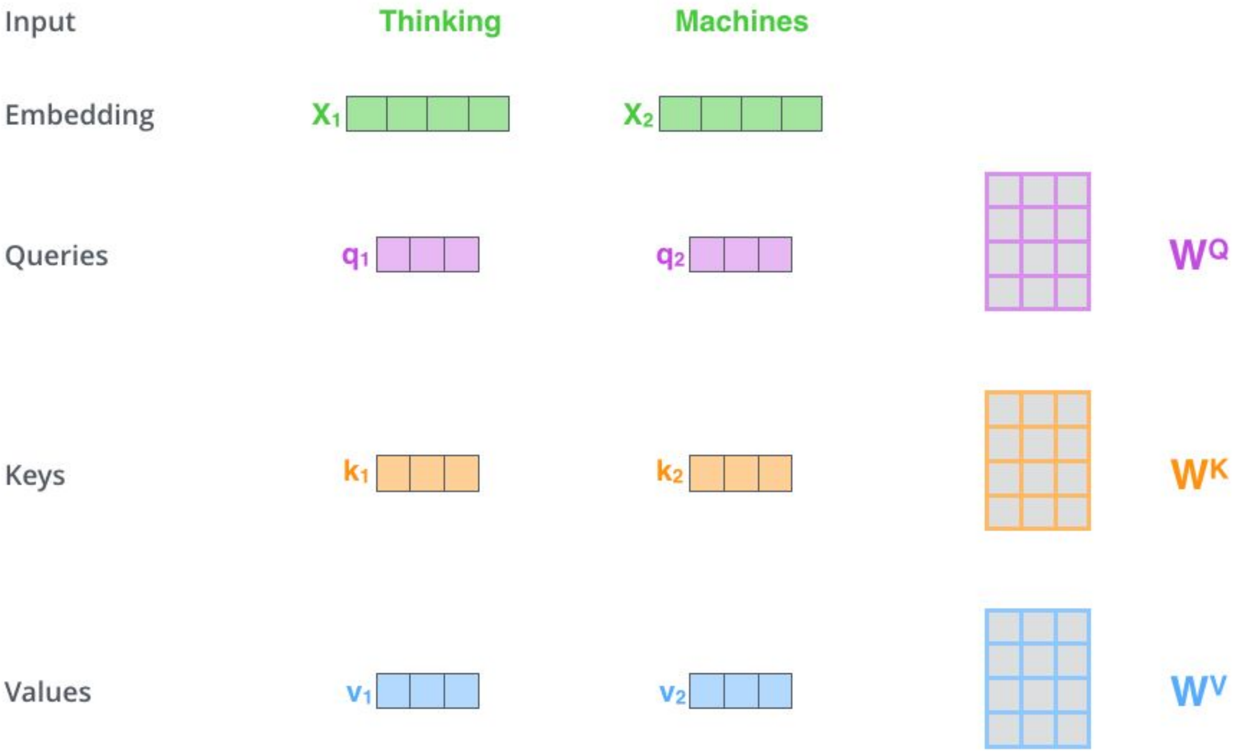
- While processing a word, the other words are scored to decide how much attention has to be paid to each of them.
- There is a method for scoring - which is done in the self attention layer.
- Each word is thus scored and a combination of vectors of all these words are then used as an extra information to process a word.



Layer: 5 Attention: Input - Input

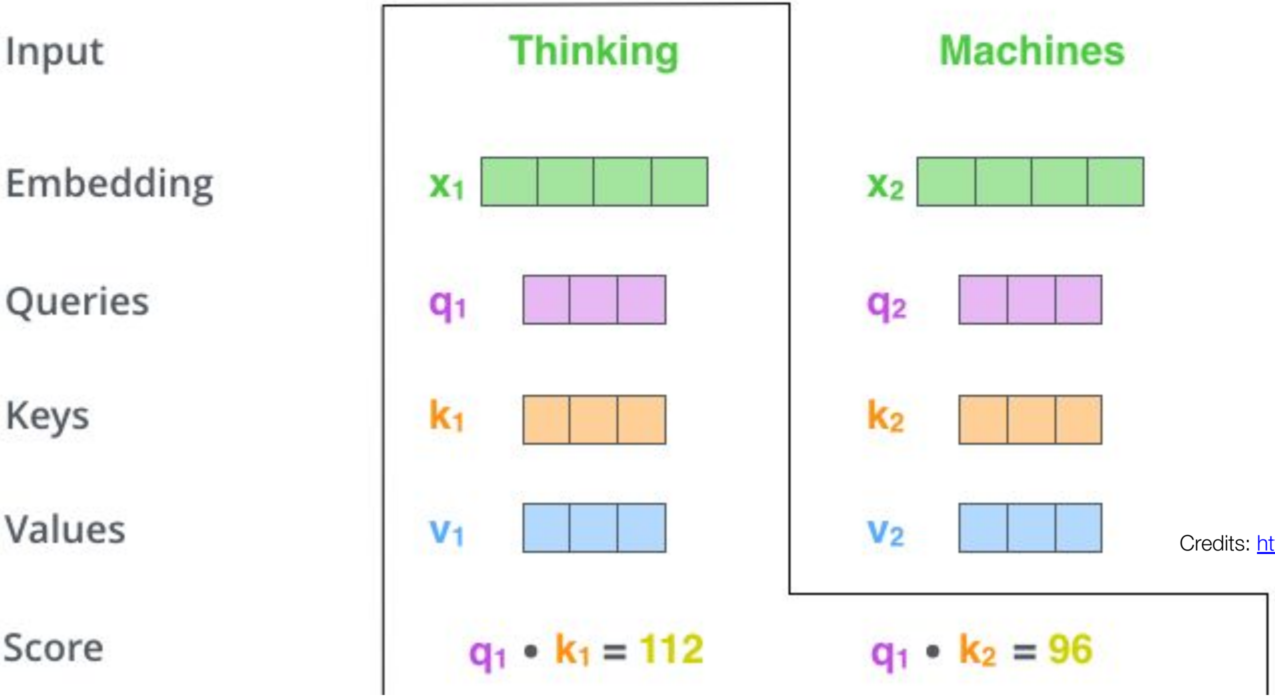
Credits: http://jalammar.github.io/illustrated-bert/

How to calculate attention-score?



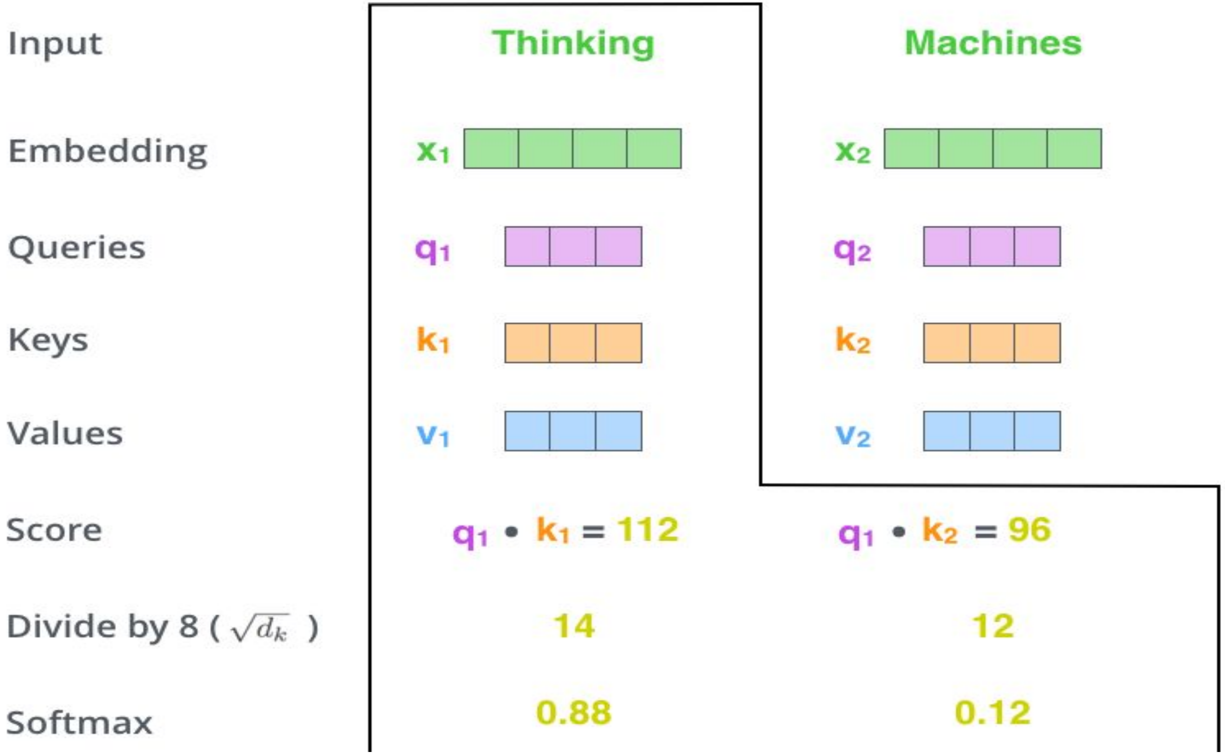
Credits: http://jalammar.github.io/illustrated-bert/

How to calculate attention-score?



Credits: http://jalammar.github.io/illustrated-bert/

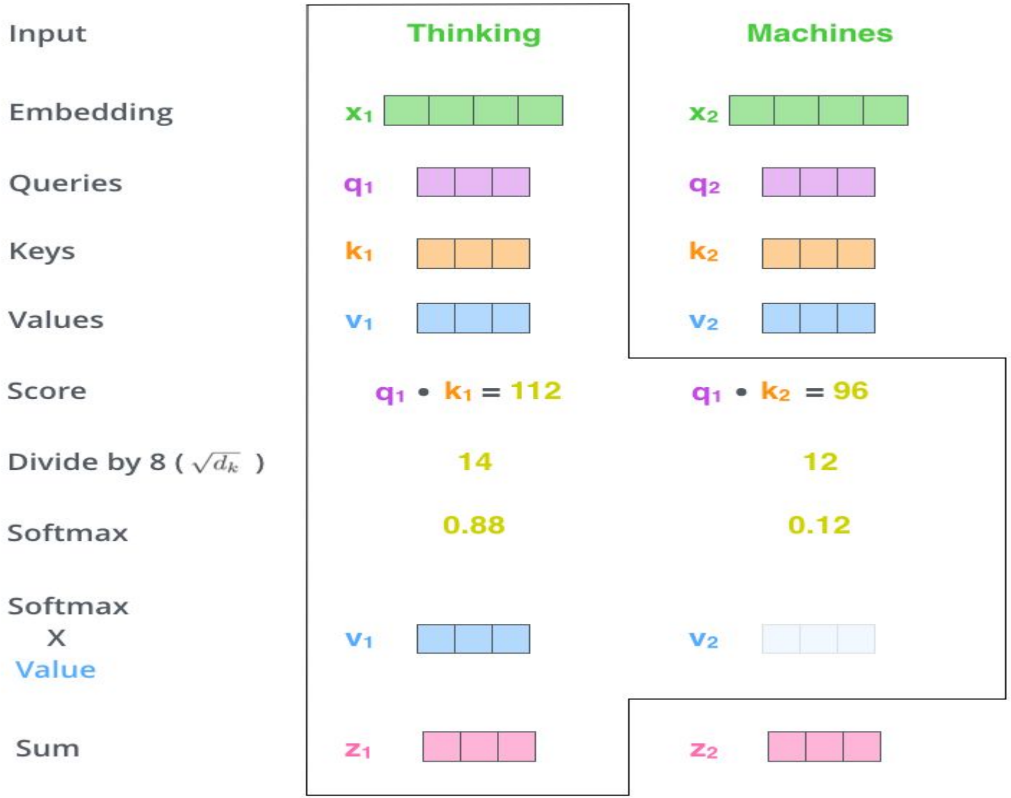
How to calculate attention-score?



Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \bullet k_1 = 112$	$q_1 \bullet k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12

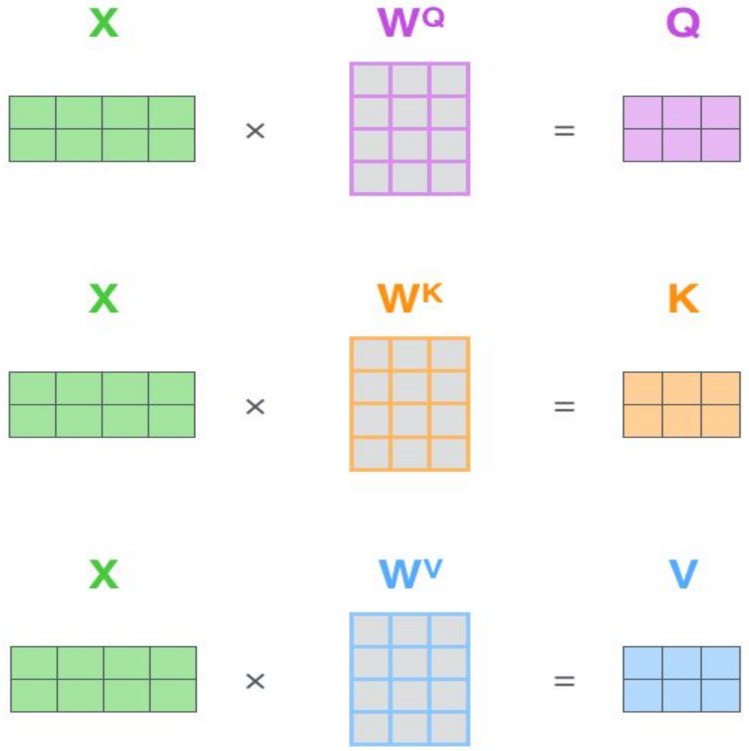
Credits: http://jalammar.github.io/illustrated-bert/

How to calculate attention-score?



Credits: http://jalammar.github.io/illustrated-bert/

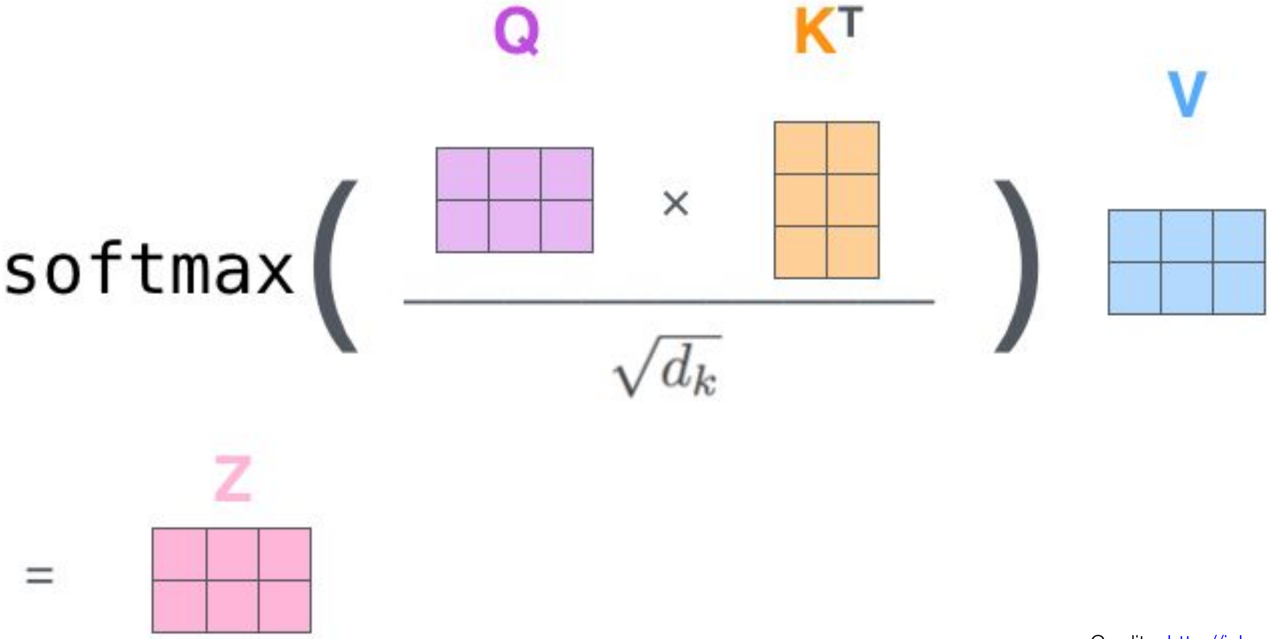
Matrix calculation of Self-Attention



Credits: http://jalammar.github.io/illustrated-bert/

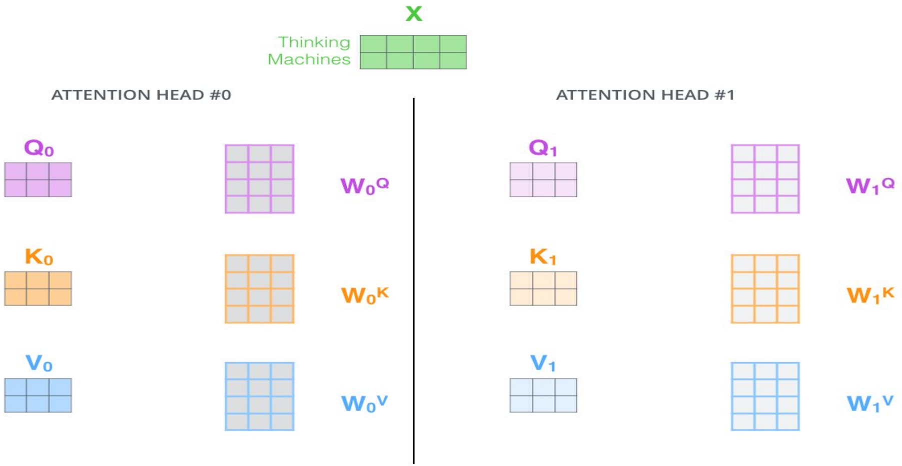
©InFoCusp Innovations Pvt. Ltd. 2019

Matrix calculation of Self-Attention



Credits: http://jalammar.github.io/illustrated-bert/

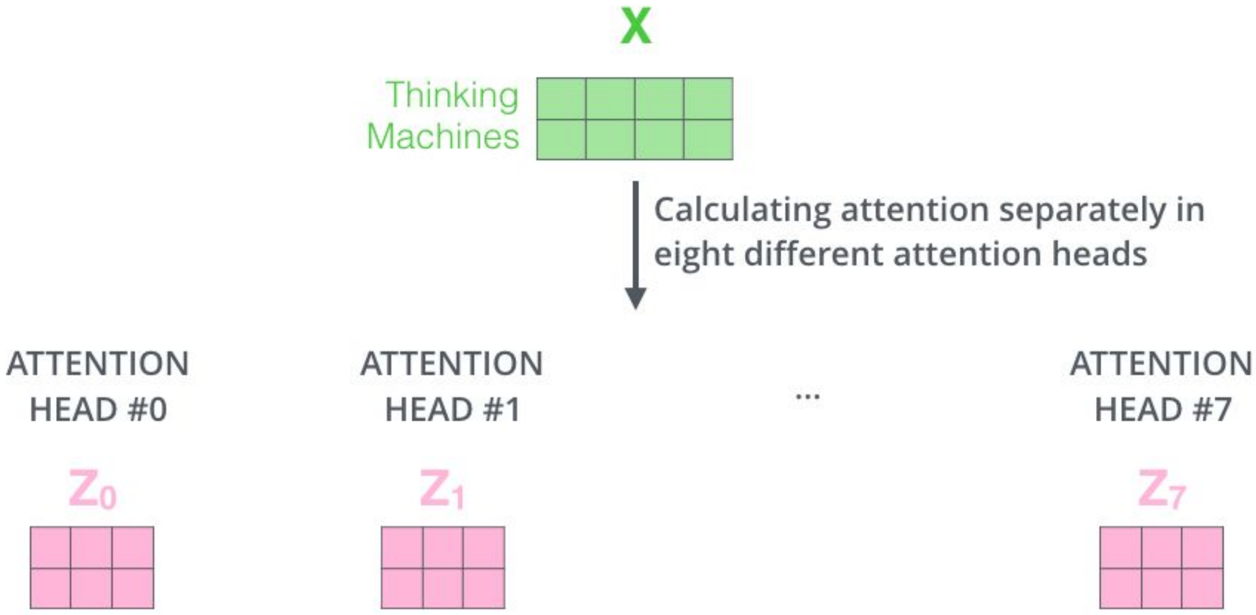
Multiple attention heads



- Multiple attention heads expands the models ability to focus on different positions.
- It gives the model multiple representation subspaces because corresponding to each space, we will have different query/key/value vectors.
- In the paper, they have used 8 attention heads

Credits: http://jalammar.github.io/illustrated-bert/

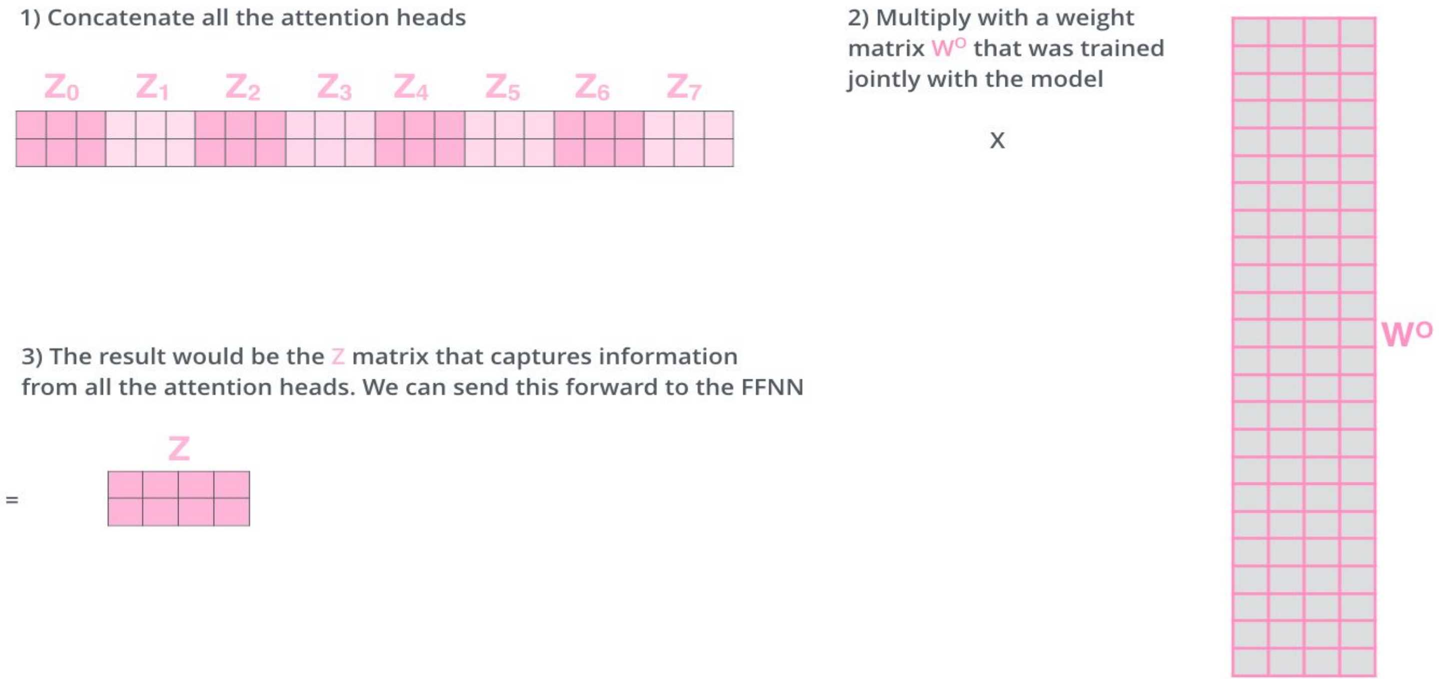
Multiple attention heads



Credits: http://jalammar.github.io/illustrated-bert/

©InFoCusp Innovations Pvt. Ltd. 2019

Multiple attention heads



1) Concatenate all the attention heads

2) Multiply with a weight matrix W^O that was trained jointly with the model

3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

Credits: http://jalammar.github.io/illustrated-bert/

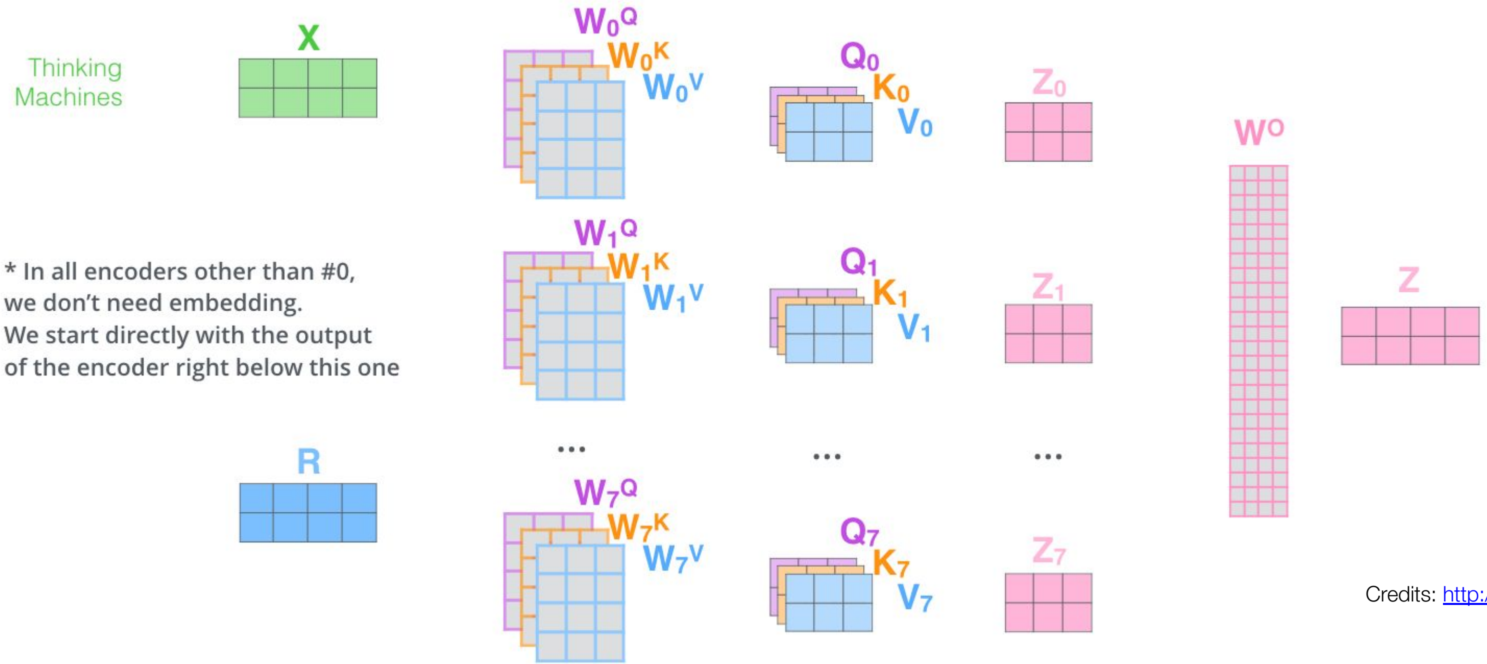
1) This is our input sentence*

2) We embed each word*

3) Split into 8 heads. We multiply X or R with weight matrices

4) Calculate attention using the resulting Q/K/V matrices

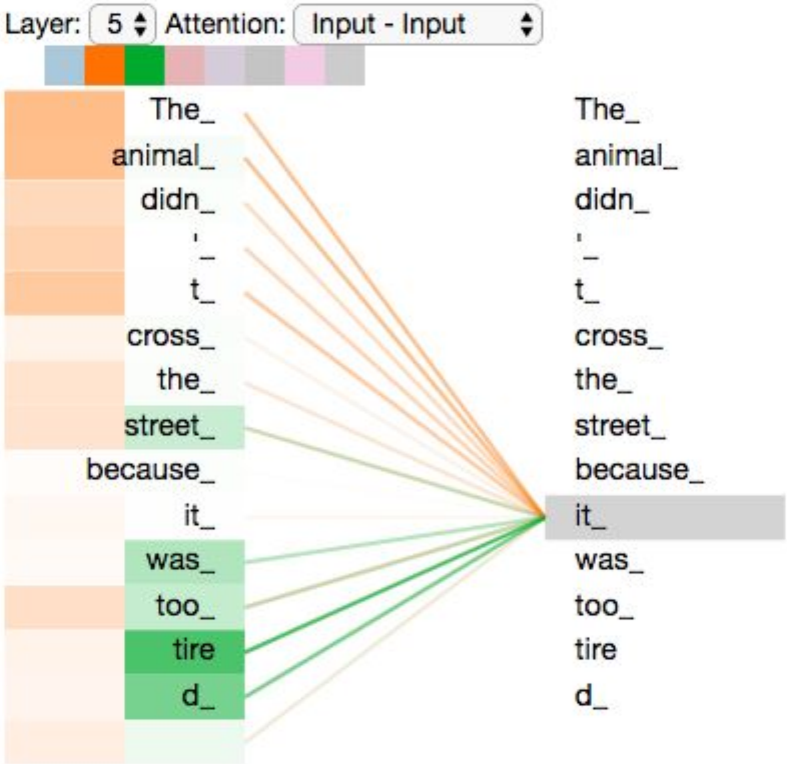
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

Credits: http://jalammar.github.io/illustrated-bert/

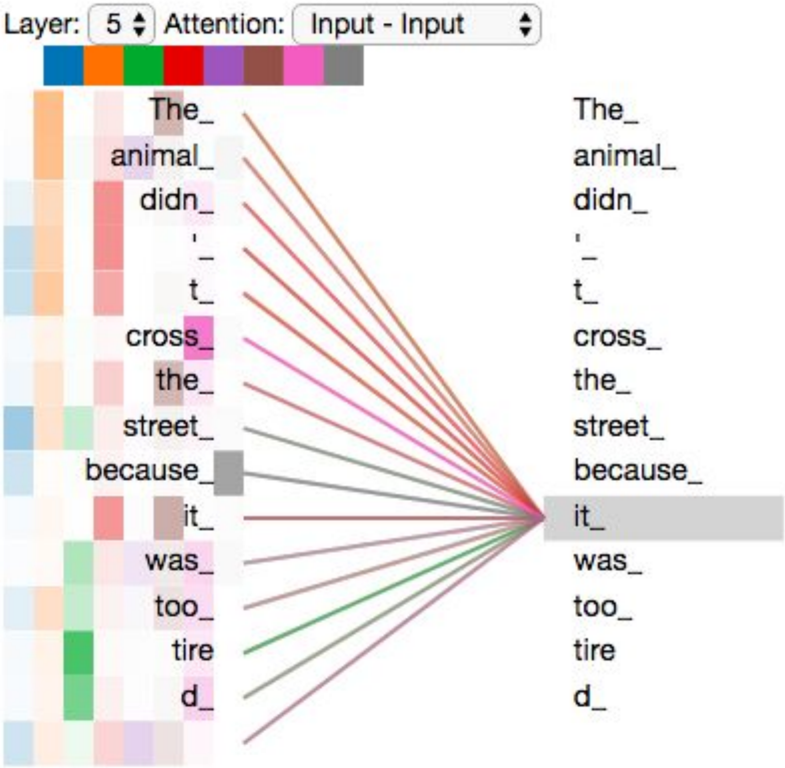
Multiple attention heads



Layer: 5 Attention: Input - Input

Credits: http://jalammar.github.io/illustrated-bert/

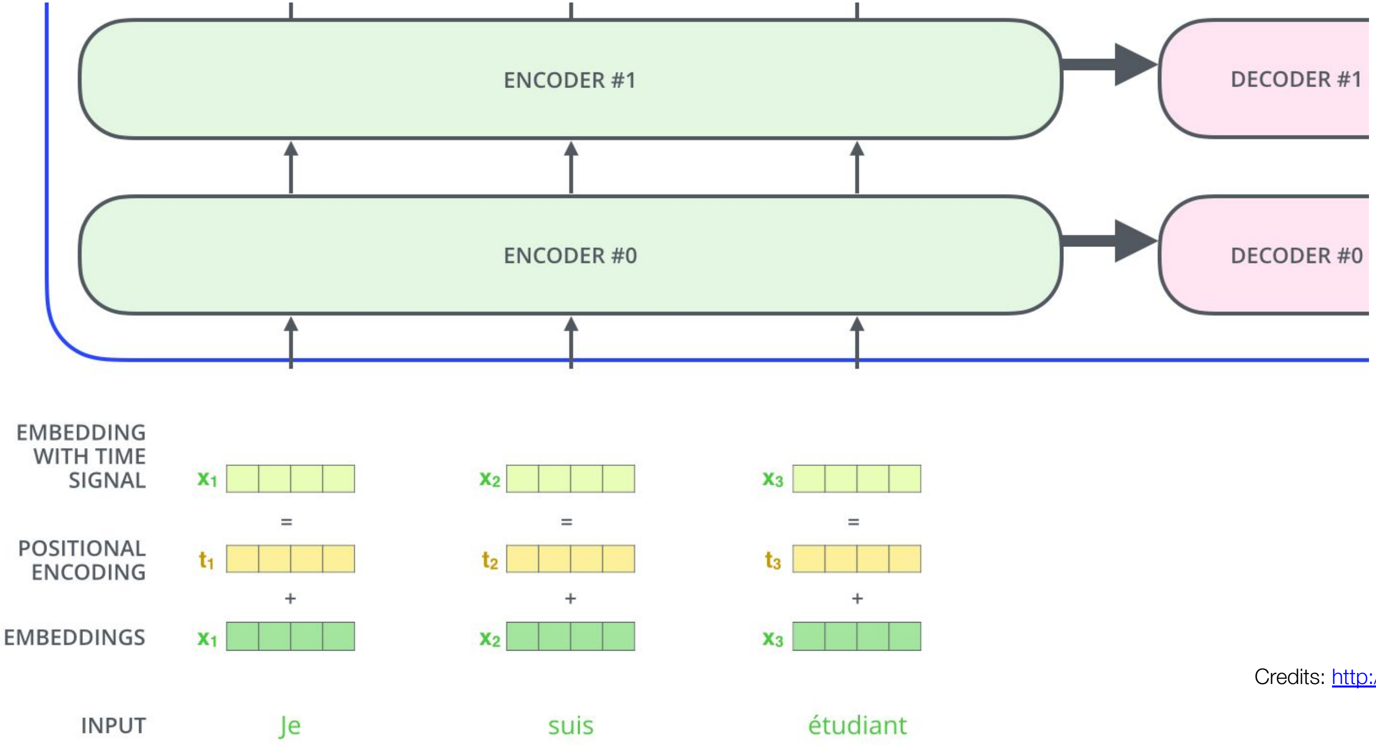
Multiple attention heads



Layer: 5 Attention: Input - Input

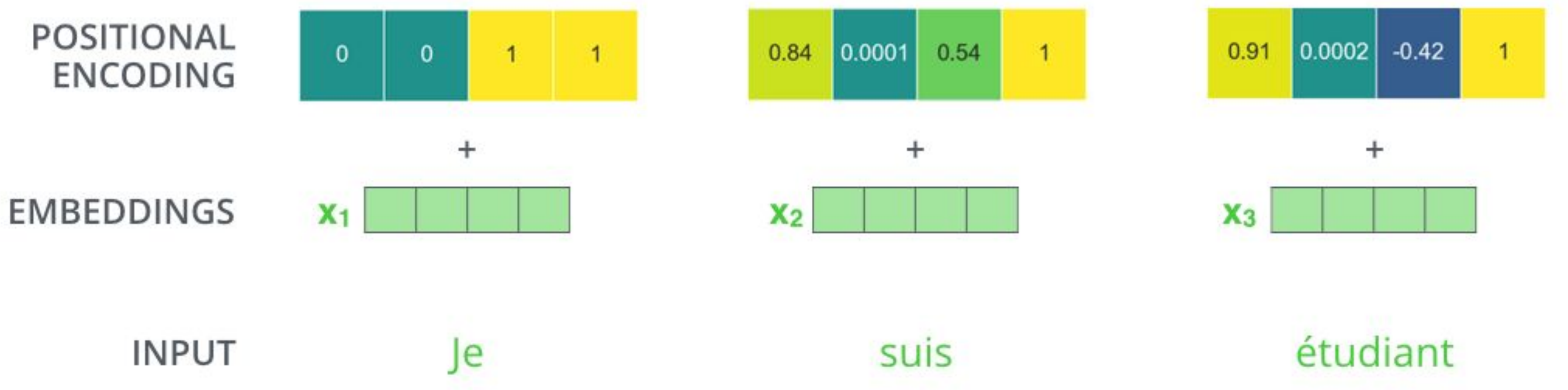
Credits: http://jalammar.github.io/illustrated-bert/

Positional Embedding



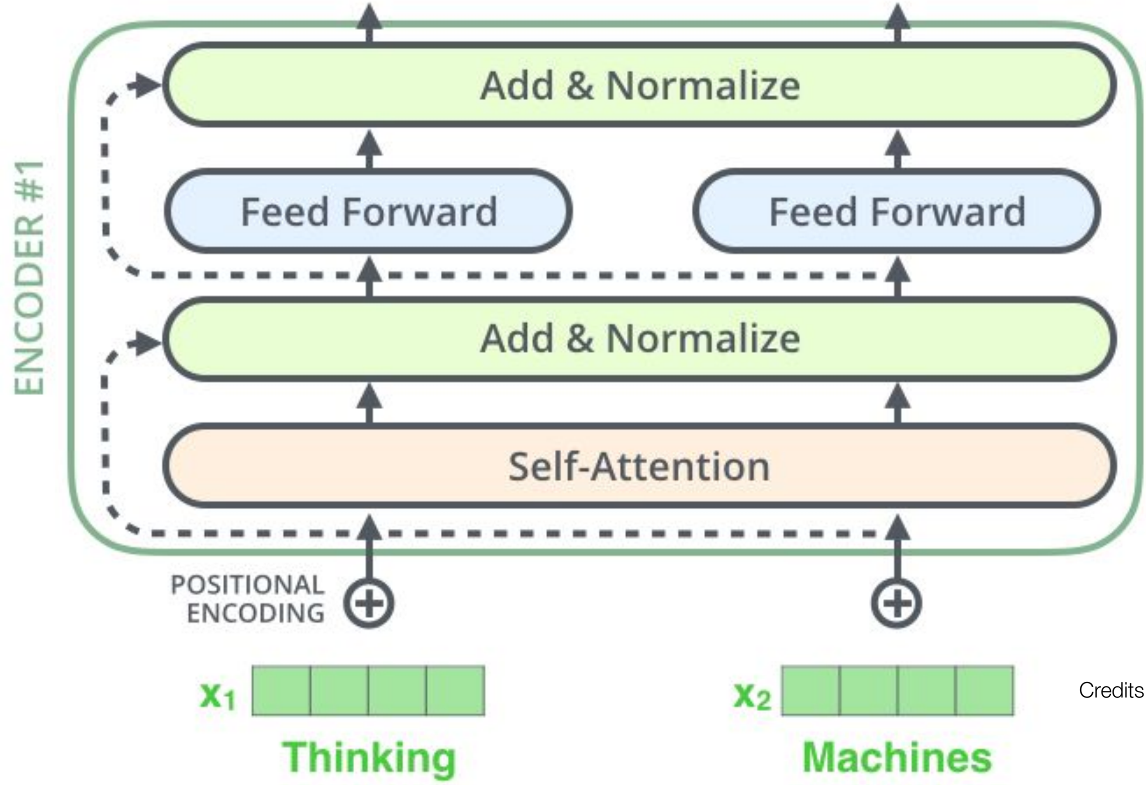
Credits: http://jalammar.github.io/illustrated-bert/

Positional Embedding



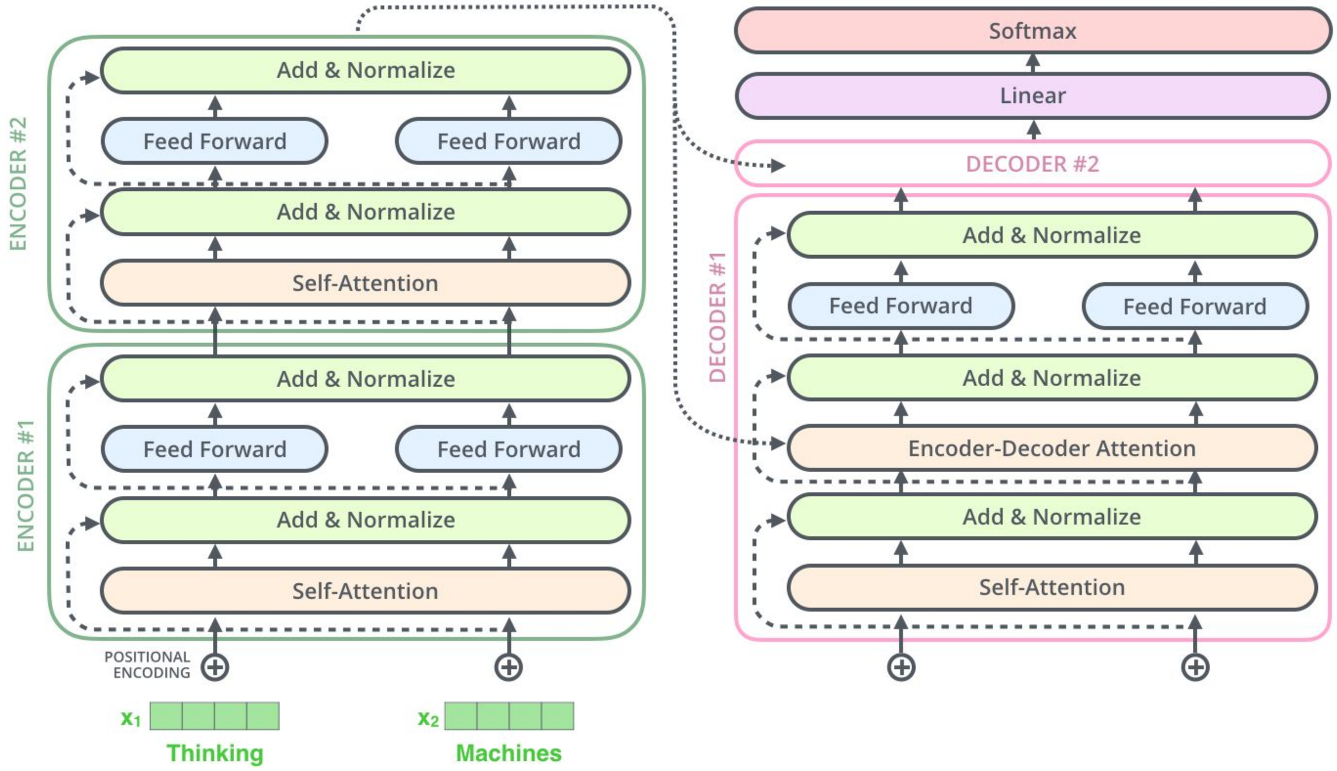
Credits: http://jalammar.github.io/illustrated-bert/

Residual Connections



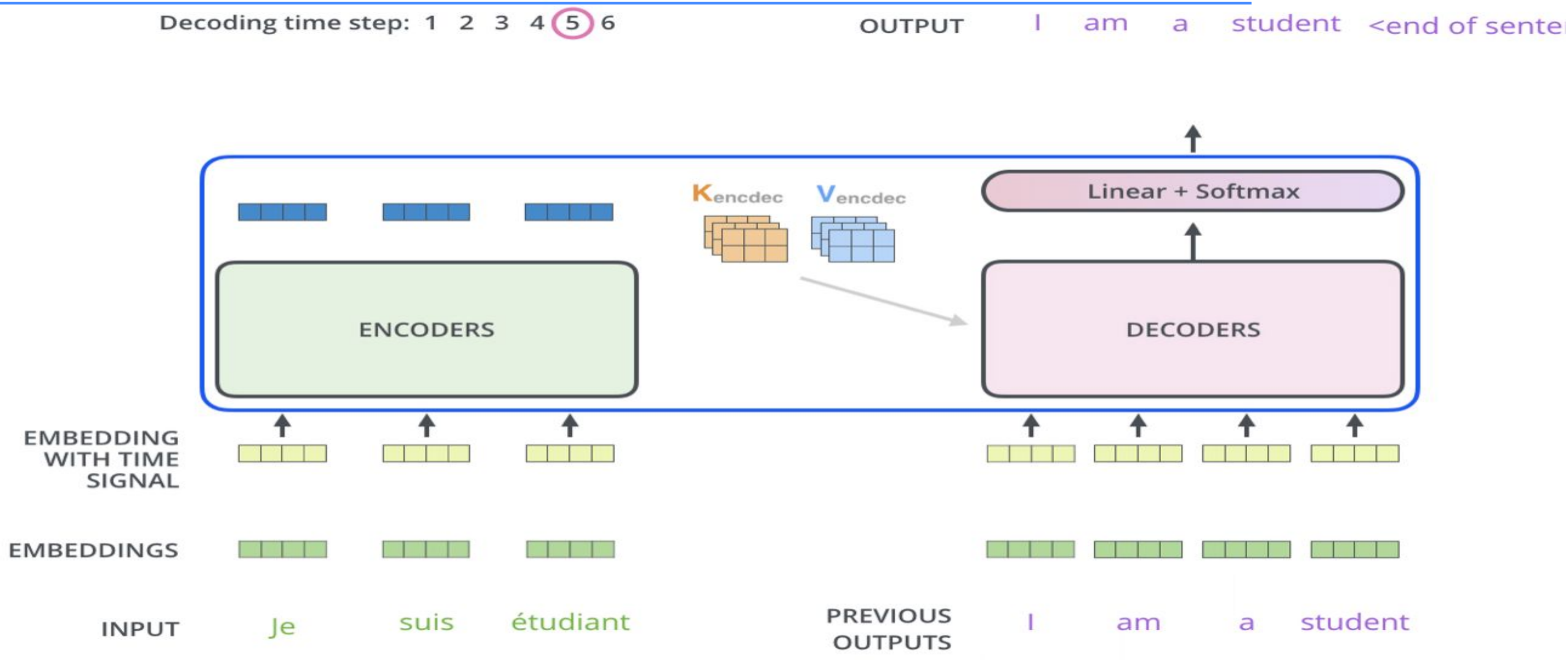
Credits: http://jalammar.github.io/illustrated-bert/

Residual Connections



Credits: http://jalammar.github.io/illustrated-bert/

The Decoder



Credits: http://jalammar.github.io/illustrated-bert/

The BERT!

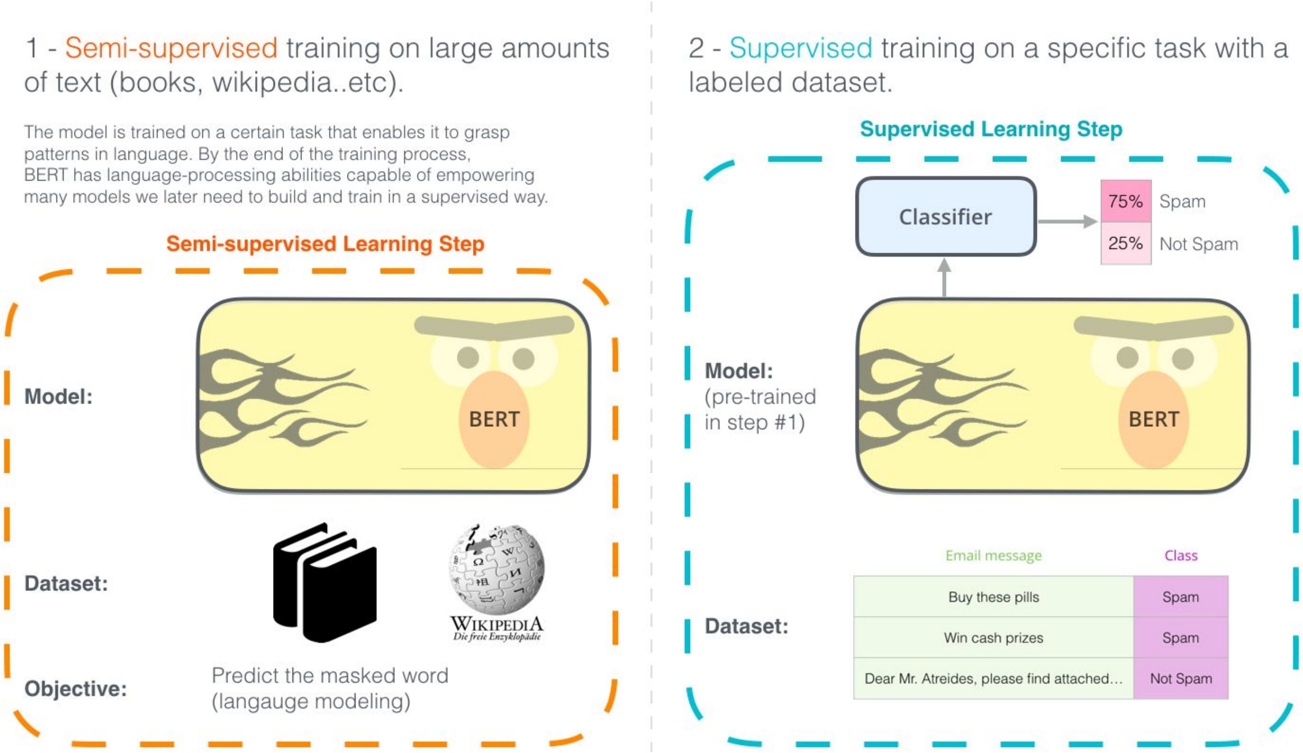
Bidirectional Encoder Representations from Transformers

- Bidirectional - because it looks at both, the left and the right side of the word for context.
- It does this using the self-attention layer.

Transfer Learning

- Transfer Learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

How BERT uses transfer learning?



1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

Semi-supervised Learning Step

Model:

Dataset:

Objective: Predict the masked word (langauge modeling)

2 - Supervised training on a specific task with a labeled dataset.

Supervised Learning Step

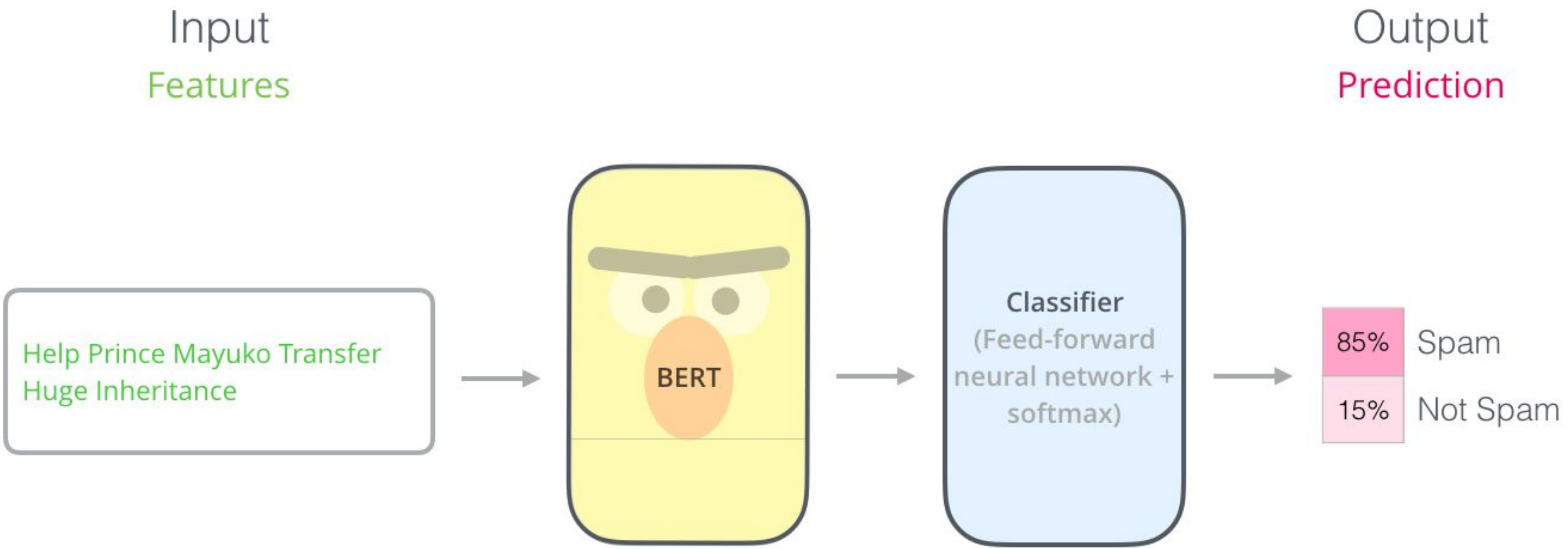
Model: (pre-trained in step #1)

Dataset:

Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached…	Not Spam

Credits: http://jalammar.github.io/illustrated-bert/

Classification Example



Input

Features

Help Prince Mayuko Transfer Huge Inheritance

BERT

Classifier

(Feed-forward neural network + softmax)

Output

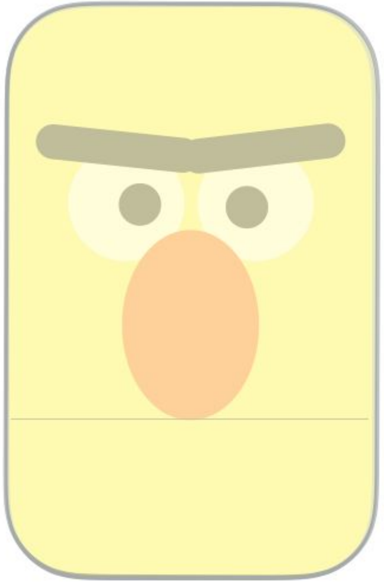
Prediction

85%	Spam
15%	Not Spam

Credits: http://jalammar.github.io/illustrated-bert/

BERT Variants

BERT$_{\text{BASE}}$

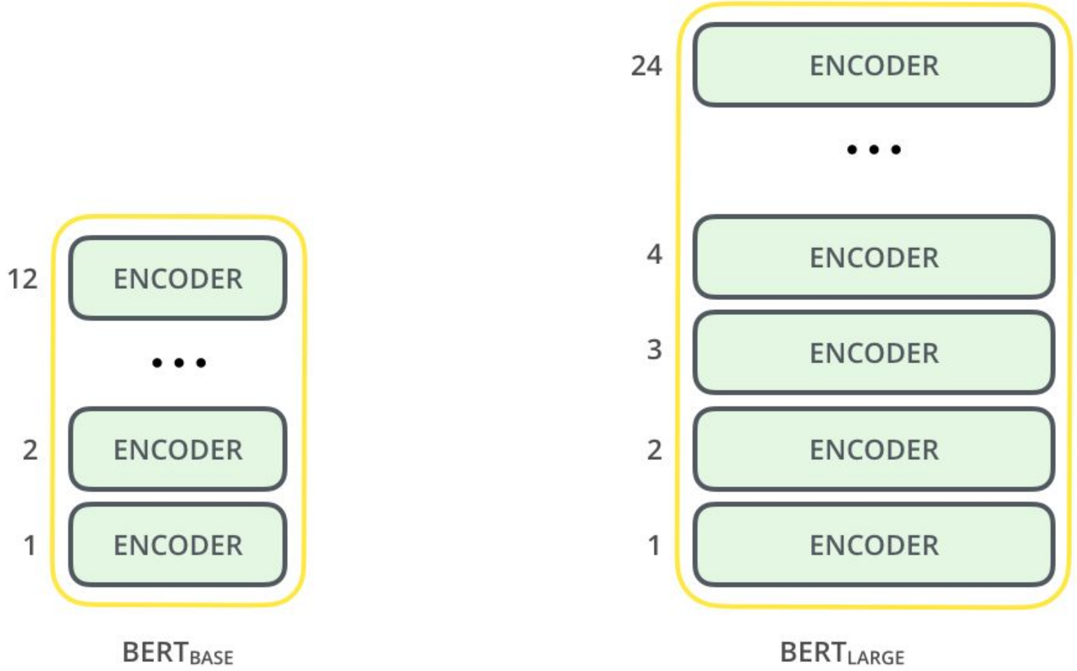


BERT$_{\text{LARGE}}$

Credits: http://jalammar.github.io/illustrated-bert/

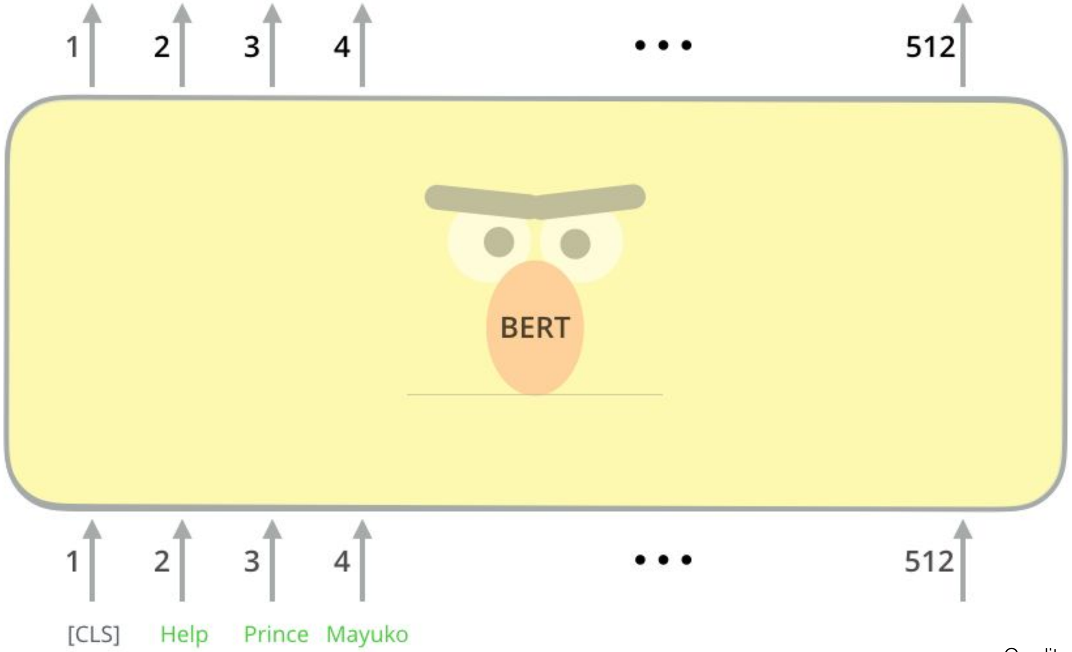
©InFoCusp Innovations Pvt. Ltd. 2019

BERT Variants



Credits: http://jalammar.github.io/illustrated-bert/

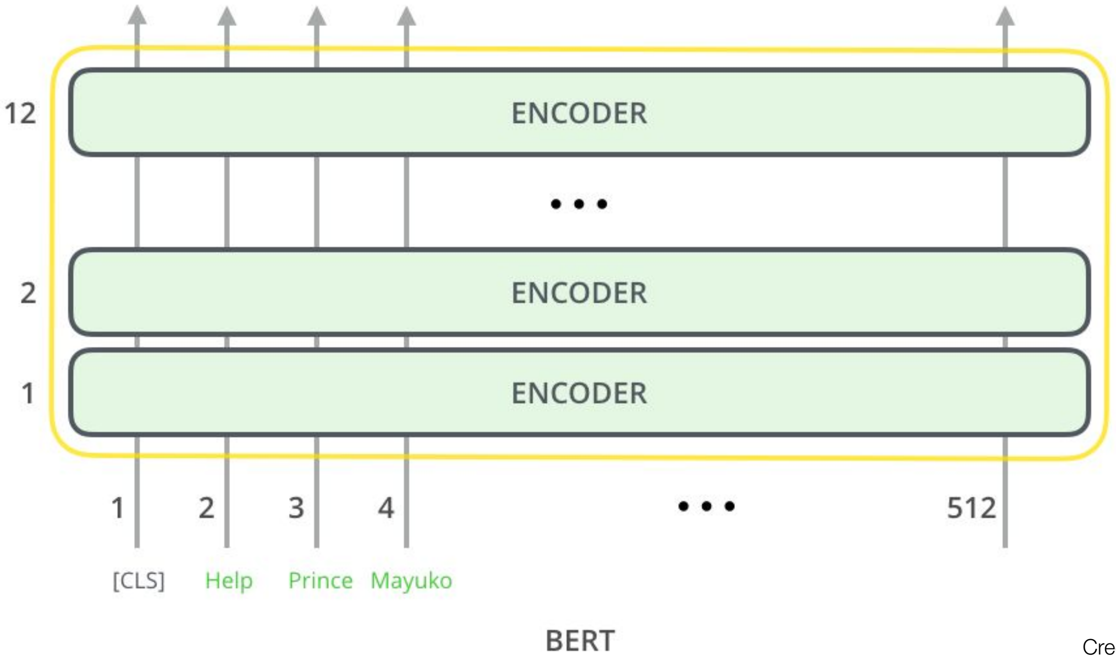
Special Token - CLS



Credits: http://jalammar.github.io/illustrated-bert/

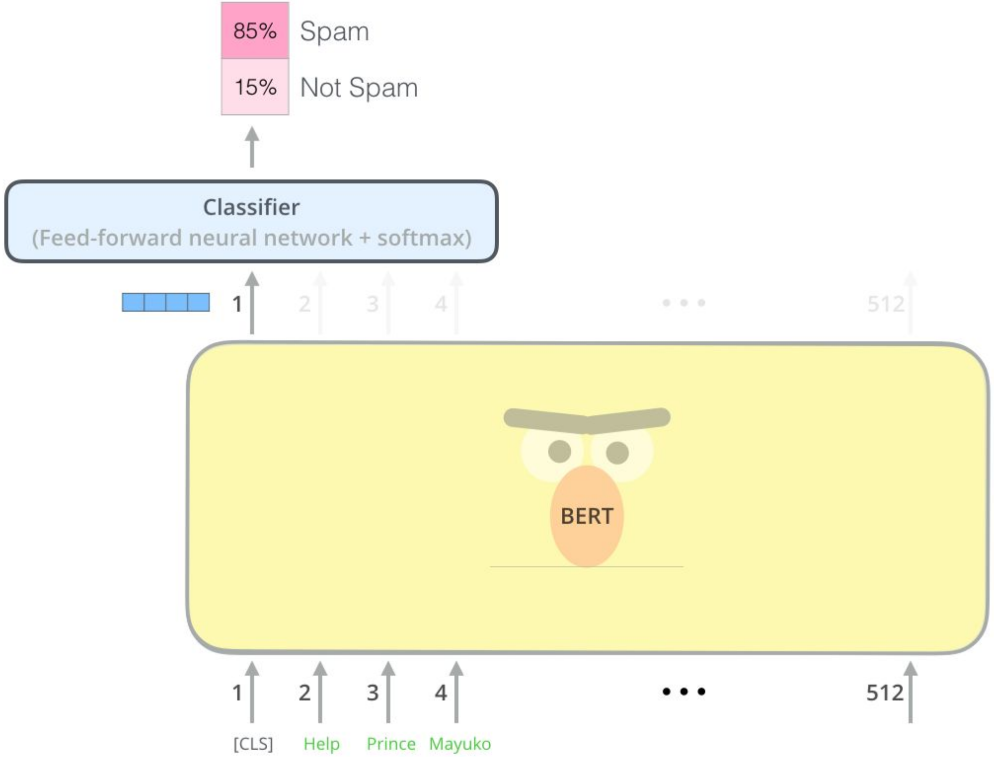
©InFoCusp Innovations Pvt. Ltd. 2019

Special Token - CLS



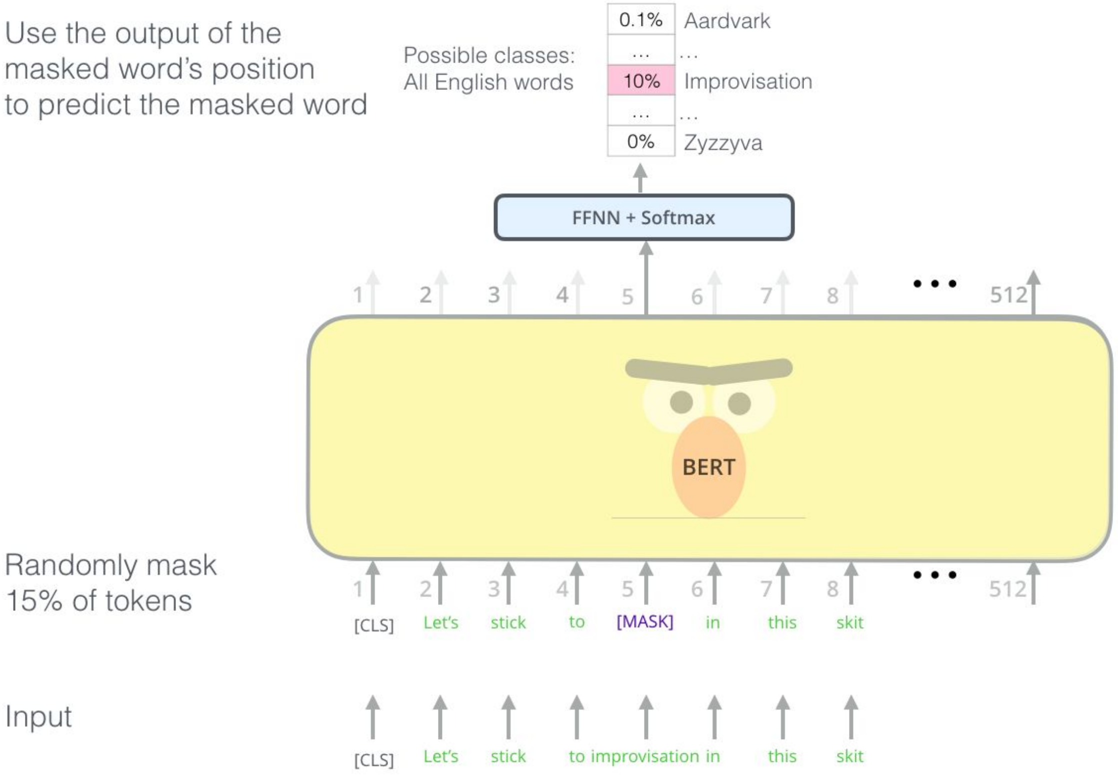
Credits: http://jalammar.github.io/illustrated-bert/

Special Token - CLS for classification



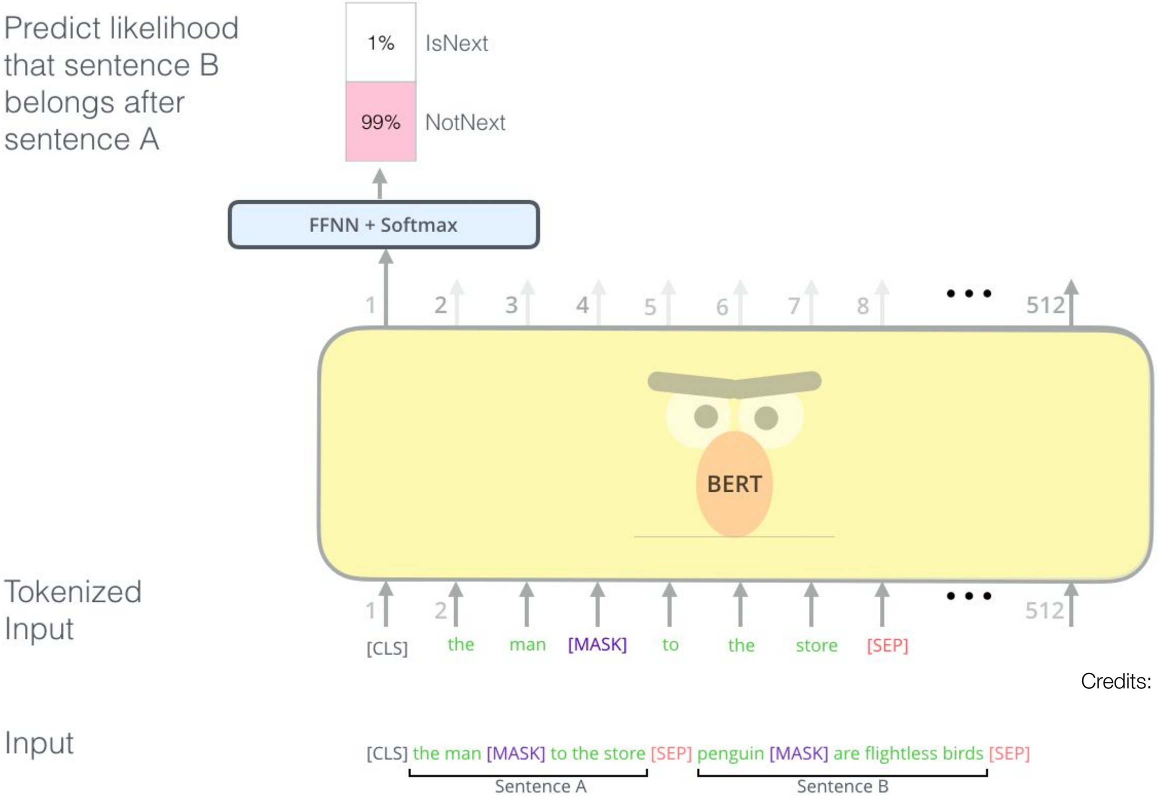
Credits: http://jalammar.github.io/illustrated-bert/

BERT pre-training: Masked Language Model


Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0.1% Aardvark
… …
10% Improvisation
… …
0% Zyzzyva
FFNN + Softmax
1 2 3 4 5 6 7 8 • • • 512
BERT
Randomly mask 15% of tokens
1 2 3 4 5 6 7 8 • • • 512
[CLS] Let's stick to [MASK] in this skit
Input
[CLS] Let's stick to improvisation in this skit

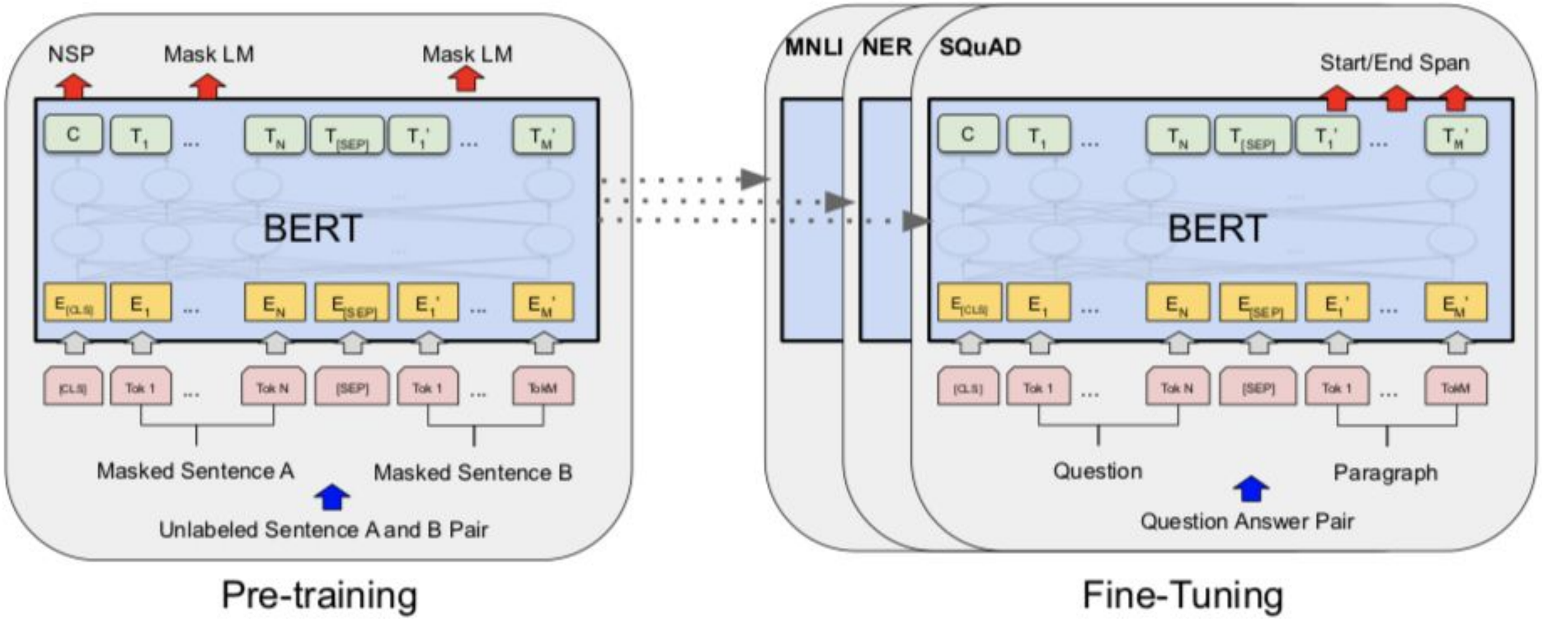
Credits: http://jalammar.github.io/illustrated-bert/

BERT pre-training: Next sentence prediction with special token SEP

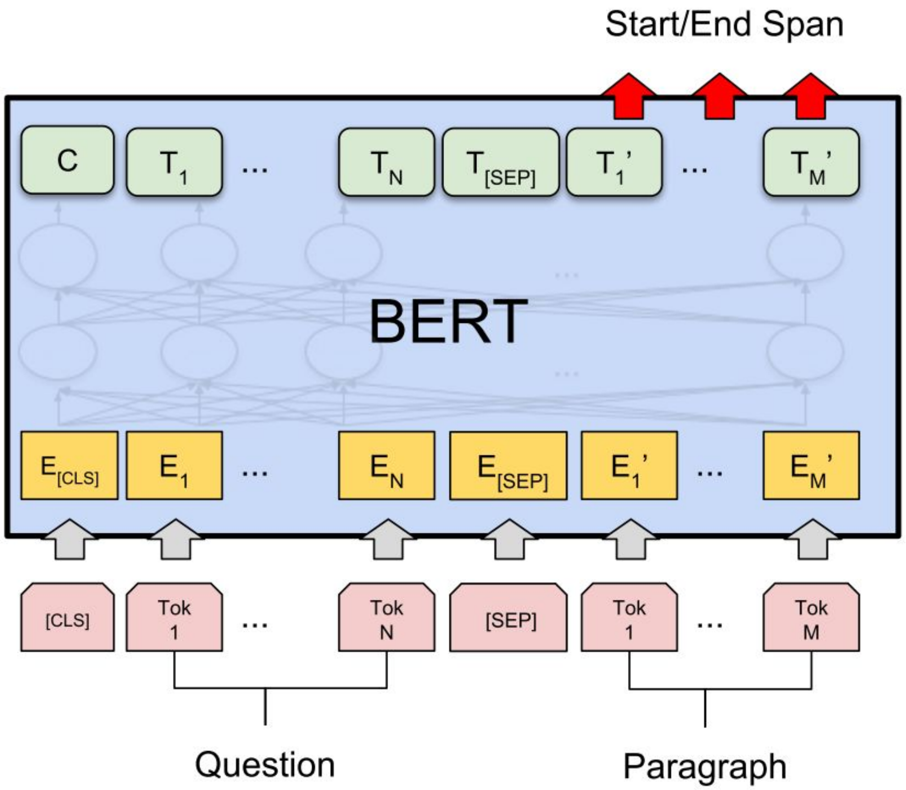

Predict likelihood that sentence B belongs after sentence A
1% IsNext
99% NotNext
FFNN + Softmax
1
2
3
4
5
6
7
8
• • •
512
BERT
Tokenized Input
1
2
512
[CLS]
the
man
[MASK]
to
the
store
[SEP]
Input
[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flightless birds [SEP]
Sentence A
Sentence B

Credits: http://jalammar.github.io/illustrated-bert/

Task specific BERT



BERT for Question-Answering



Getting the Start and End positions

$$Pi = e^{S \cdot Ti} / \sum e^{S \cdot Tj}$$

$$S \cdot Ti + E \cdot Tj$$

©InFoCusp Innovations Pvt. Ltd. 2019

Topic Modeling

- Aim : We have N “documents”, we want to categorize them into K topics, based on their similarity.
- Assumptions : We do not know have any prior information about the topics. We only have the N documents, and we have a fixed “K”.
- Applications
  - News classification
  - Document type classification (Legalsifter)
  - Extremely important step in search (Google, Bing ….)
  - Email filters , spam classification ….
- Pro tip : This is *very* similar to clustering. All we have to figure out is how to cluster on text!

Topic Modelling methods

- Clustering after some pre-processing (Simplest and most popular)
- Explicit Semantic Analysis (ESA) we will cover this today. Rest tomorrow.

- Latent Semantic Indexing (LSI)
- pLSI - probabilistic LSI
- LDA (Latent Dirichlet Allocation)

Key : Don’t go by the big and complicated names, most concepts are simple :)

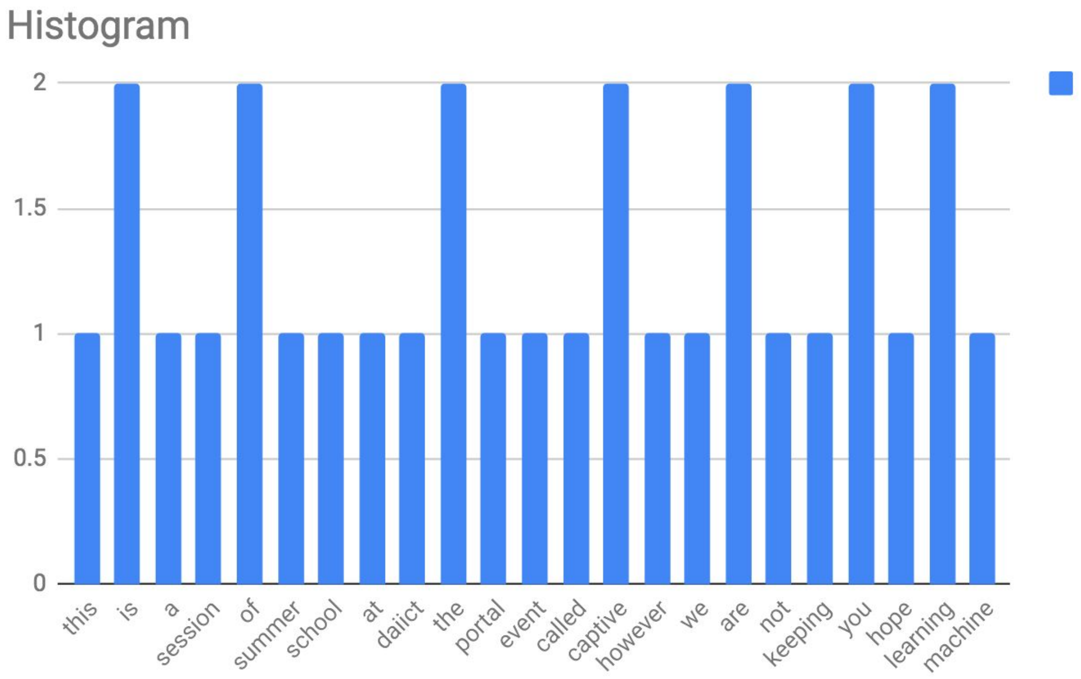
Term frequency Histogram

Let’s say there is a “document” (Document in NLP could be as small as a sentence or two!)

Our document is:

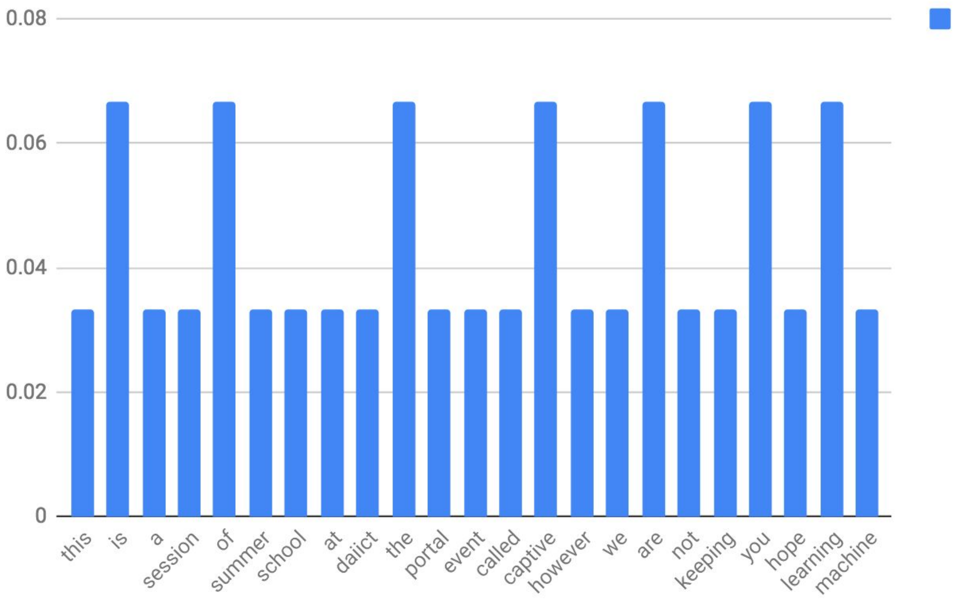
“This is a session of summer school at DAIICT. The portal of the event is called "Captive". However, we are not keeping you captive! Hope you are learning machine learning!”

Term frequency Histogram



©InFoCusp Innovations Pvt. Ltd. 2019

Term frequency Histogram



Important to normalize the document! We shall see soon.

What is a vocabulary in topic modeling context

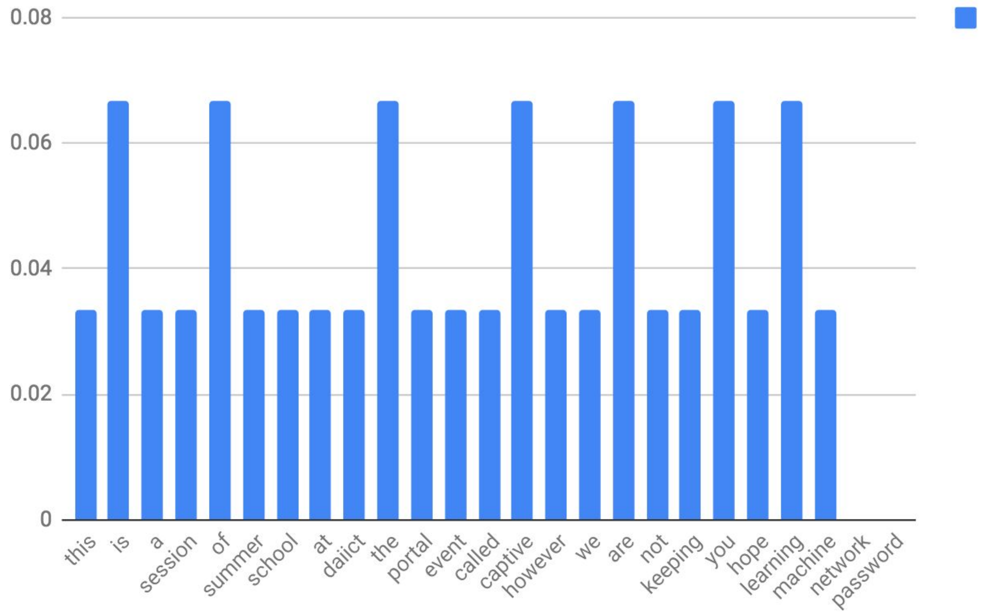
Let’s make things a “slightly more” interesting : Let’s add one more document

“DAIICT captive portal network password”

“network” and “password” are the new words.

Vocabulary is union of **all the words from all the documents**.

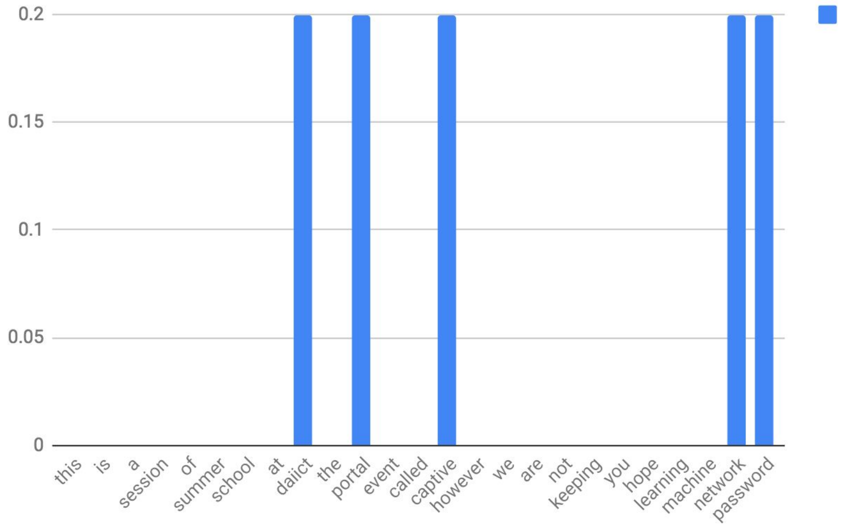
Explicit Semantic Analysis. Our term frequency histogram has to change!



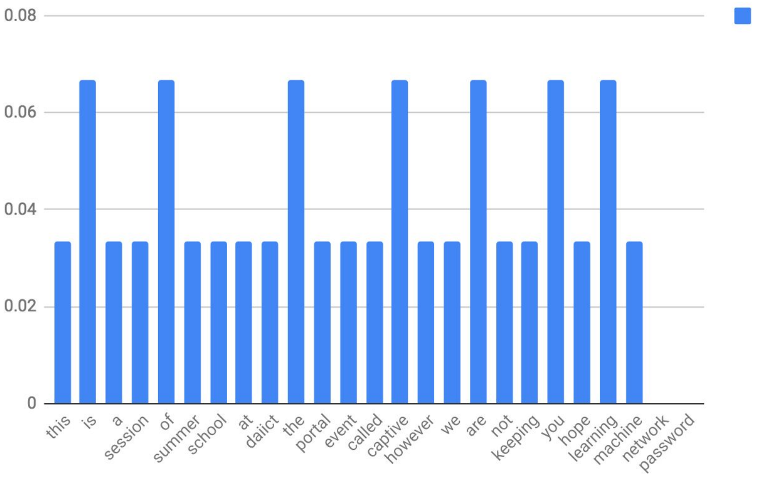
Note: network and password now come into our histogram. With 0 probability. This is called a full-vocabulary histogram.

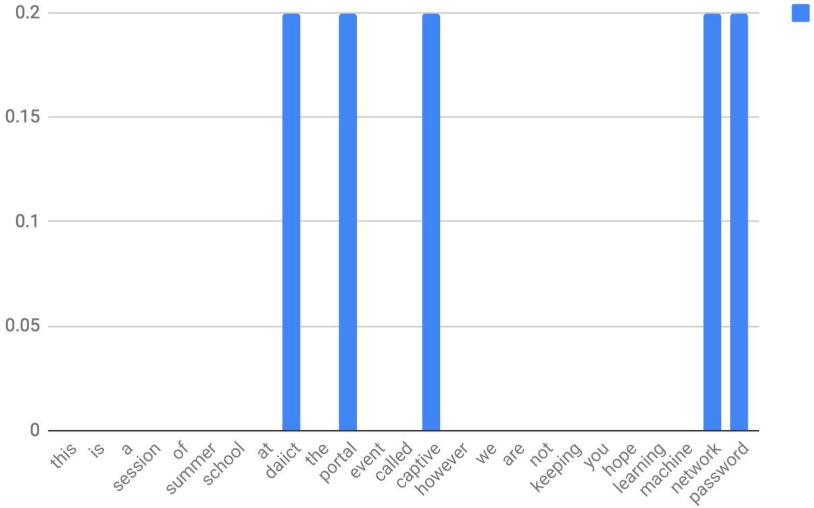
Full-vocabulary Histogram for sentence 2

“DAIICT captive portal network password”



This can be formed as a vector! Perfect for Linear Algebra





Clustering for topic modeling

- Like any topic modeling problem : we just start with N documents first.
- First step: Find out the total vocabulary V
- Second step : Make a full-vocabulary term frequency histogram out of it.
- Third step : We already have vectorized our text, Voila! Apply k-means (and many other utilities)

Explicit semantic analysis

- Like any topic modeling problem : we just start with N documents first.
- First step: Find out the total vocabulary V
- Second step : Make a full-vocabulary term frequency histogram out of it.
- Third step : Make an “inverted index” : This finds occurrences of words within the document. It’s a word’s representation in the document space
-

infocusp
Innovations Pvt. Ltd.

Inverted Index

Document1 : “This is a session of summer school at DAIICT. The portal of the event is called "Captive". However, we are not keeping you captive! Hope you are learning machine learning!””

Document2 : “DAIICT captive portal network password”

	Document1	Document2
Captive	1	1
Portal	1	1
Network	0	1
summer	1	0
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...

Normalize the inverted Index

Document1 : “This is a session of summer school at DAIICT. The portal of the event is called "Captive". However, we are not keeping you captive! Hope you are learning machine learning!””

Document2 : “DAIICT captive portal network password”

	Document1	Document2
Captive	0.003	0.2
Portal	0.003	0.2
Network	0	0.2
summer	0.003	0
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...

Inverted Index in the real world

In reality, there are MANY documents (N is large).

	Document1	Document2	Document3	Document4	Document5	...	...	...	DocumentN
Captive	0.03	0.2	0.004	0	0.001	0.001	0.1	0	

Portal	0.03	0.2	0.003	0	0.001	0	0.05	0	0

Side tidbit : Google

©InFoCusp Innovations Pvt. Ltd. 2019

Linear Algebra Alert!

	Document1	Document2	Document3	Document4	Document5	...	...	...	DocumentN
Captive	0.03	0.2	0.004	0	0.001	0.001	0.1	0	

Portal	0.03	0.2	0.003	0	0.001	0	0.05	0	0.2

Each word is of fixed N-dimensions. We can apply linear algebra to find "similarity" between words. Similarity here is words co-occurring in the set of document together.

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \, ||\mathbf{v}||} = \frac{\sum_{i=1}^{N} u_i v_i}{\sqrt{\sum_{i=1}^{N} u_i^2} \sqrt{\sum_{i=1}^{N} v_i^2}}$$

©InFoCusp Innovations Pvt. Ltd. 2019

Explicit Semantic Analysis

We are able to find association of the words as seen by our document. Keep the words that have high similarity (greater than the threshold) among each other in one “group”.

For example : groupA (“captive”, “cell”, “dungeon”, “bunker”, “prisoner”, “DA-IICT”,”portal”)

groupB (“flower”, “bloom”, “leaves”, “photosynthesis”, ….)

We can sum up the group vectors directly.

Vector Summing up

	Document1	Document2	Document3	Document4	Document5	...	...	...	DocumentN
Captive	0.03	0.2	0.004	0	0.001	0.001	0.1	0	

+

Portal	0.03	0.2	0.003	0	0.001	0	0.05	0	0.2

groupA	0.06	0.4	0.007	0	0.002	0.001	0.15	0	0.2

“Group” Similarity

groupA	0.06	0.4	0.007	0	0.002	0.001	0.15	0	0.2
groupB	0.1	0.4	0.027	0	0.3	0.001	0	0.2	0.3
...	...	...	...	...	...	...	...	...	...
groupG	0.1	0.4	0.027	0	0.3	0.001	0	0.2	0.3

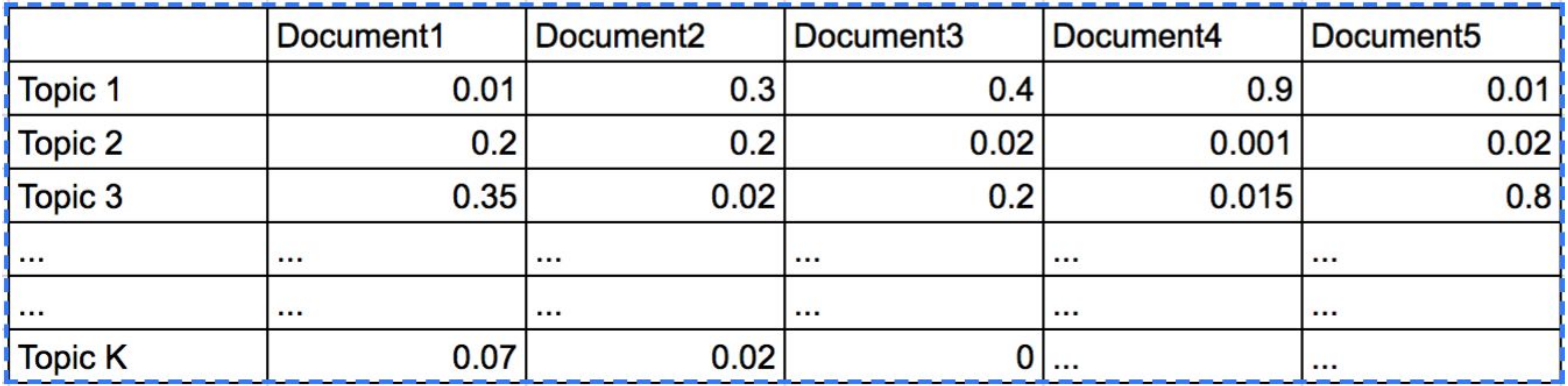
Now the cosine similarity can be applied across “groups”.

©InFoCusp Innovations Pvt. Ltd. 2019

Explicit Semantic Analysis

We came from words to “groups” similarity, then “groups” similarity can give us more “groups”.

This iterative process finally gives us “K” known concepts, with each document’s participation in it.



	Document1	Document2	Document3	Document4	Document5
Topic 1	0.01	0.3	0.4	0.9	0.01
Topic 2	0.2	0.2	0.02	0.001	0.02
Topic 3	0.35	0.02	0.2	0.015	0.8
...	...	...	...	...	...
...	...	...	...	...	...
Topic K	0.07	0.02	0	...	...

Egozi, Ofer; Markovitch, Shaul; Gabrilovich, Evgeniy (2011). "Concept-Based Information Retrieval using Explicit Semantic Analysis" (pdf). *ACM Transactions on Information Systems*. **29**

Latent Semantic Indexing

- LSI is very similar to ESA, it is a time-and-space optimized version of ESA

- LSI had to exist, because as we see in ESA
  - the first step was to make a vector **for each word, each having D dimensions.** Very quickly, that entire matrix becomes large enough not to fit in RAM.
  - Moreover, the *word x document* matrix is really a sparse matrix.

- A linear Algebra technique called SVD (Singular Value Decomposition) is used to reduce a large matrix into smaller, representative constituent matrices.
- This is similar to PCA you learned earlier.

Tutorials

- NLP-Basic Tutorial : Sentiment Analysis of IMDB reviews
    - Dataset can be downloaded from here: https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
- BERT-AQuA Tutorial

Directions for Self-Explorations!

- https://machinelearningmastery.com/what-are-word-embeddings/
- https://arxiv.org/pdf/1810.04805.pdf
- https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/
- https://jalammar.github.io/illustrated-transformer/
- http://jalammar.github.io/illustrated-bert/

infocusp
Innovations Pvt. Ltd.

©InFoCusp Innovations Pvt. Ltd. 2019

Any Questions?

infocusp
Innovations Pvt. Ltd.

Thank You!

infocusp
Innovations Pvt. Ltd.

©InFoCusp Innovations Pvt. Ltd. 2019