uc3m | Universidad Carlos III de Madrid

HULAT Human Language & Accessibility Technologies

# Cutting-edge Deep Learning for NLP learners.

Isabel Segura-Bedmar

1 y 2 Julio, Universidad Politécnica de Madrid (UPM)

# Presentación

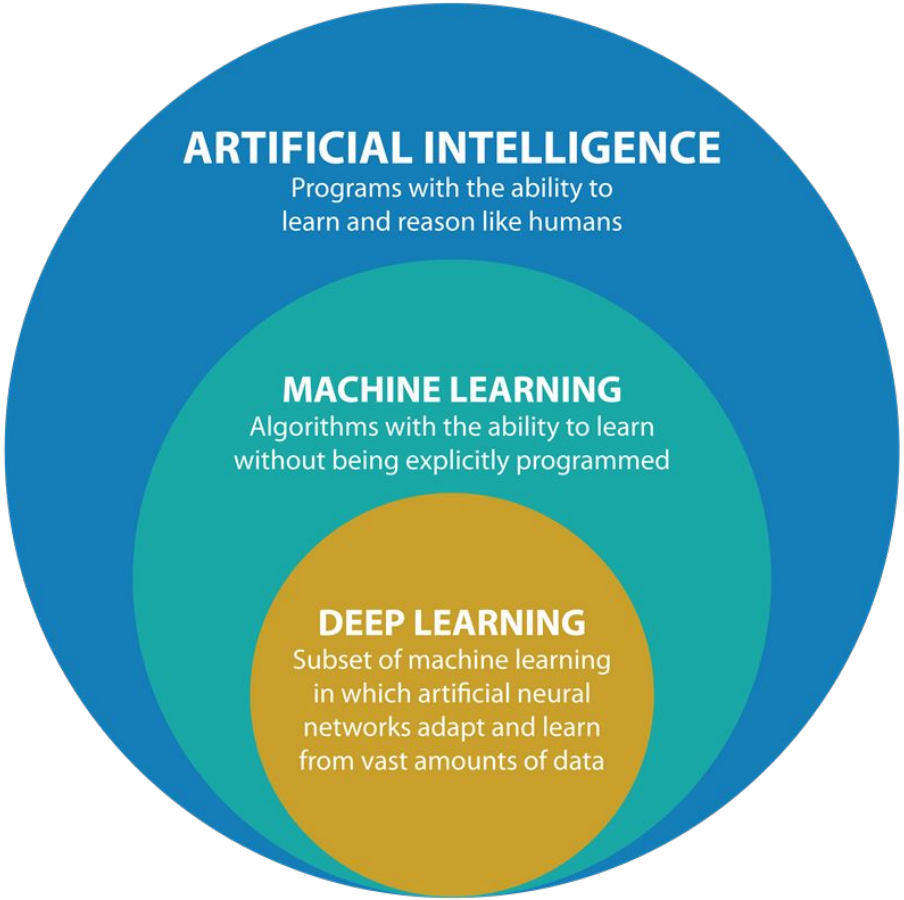
- 1998: Licenciada en Ciencias Matemáticas, Universidad Complutense de Madrid
- 1998-2004:Desarrollo Software en Telefónica I+D y Grupo Santander.
- 2004:... - Profesora Departamento Informática, Universidad Carlos III de Madrid
  - 2010: Doctora Europea en Ciencia y Tecnología Informática, Tesis: “Application of Information Extraction techniques to the pharmacological domain”, Premio Extraordinario de Doctorado y Premio Investigación SEPLN.
  - 2018: Profesora Titular.
  - Investigación centrada en Procesamiento de Lenguaje Natural
    - Más de 10 proyectos de investigación competitivos. coIP: DeepEMR, NLP4RARE-CM-UC3M
    - Publicaciones: https://scholar.google.es/citations?user=ZpbtnaUAAAAJ&hl=es,
    - Organización de competiciones como DDIEXTRACTION 2011 Y 2013.
    - Dirección de tres tesis (dos sobre técnicas de deep learning aplicadas a tareas de PLN).
- Enamorada del PLN y fascinada por las técnicas de machine learning (y en particular, deep learning).
- Mi principal motivación: ayudar a jóvenes investigadores.

# Outline (first season)

- **Introduction**
- Word Embeddings
- Deep learning architectures for NLP
  - Recurrent Neural Networks
  - Sequence to Sequence
  - Transformer
- Contextual languages models
  - BERT

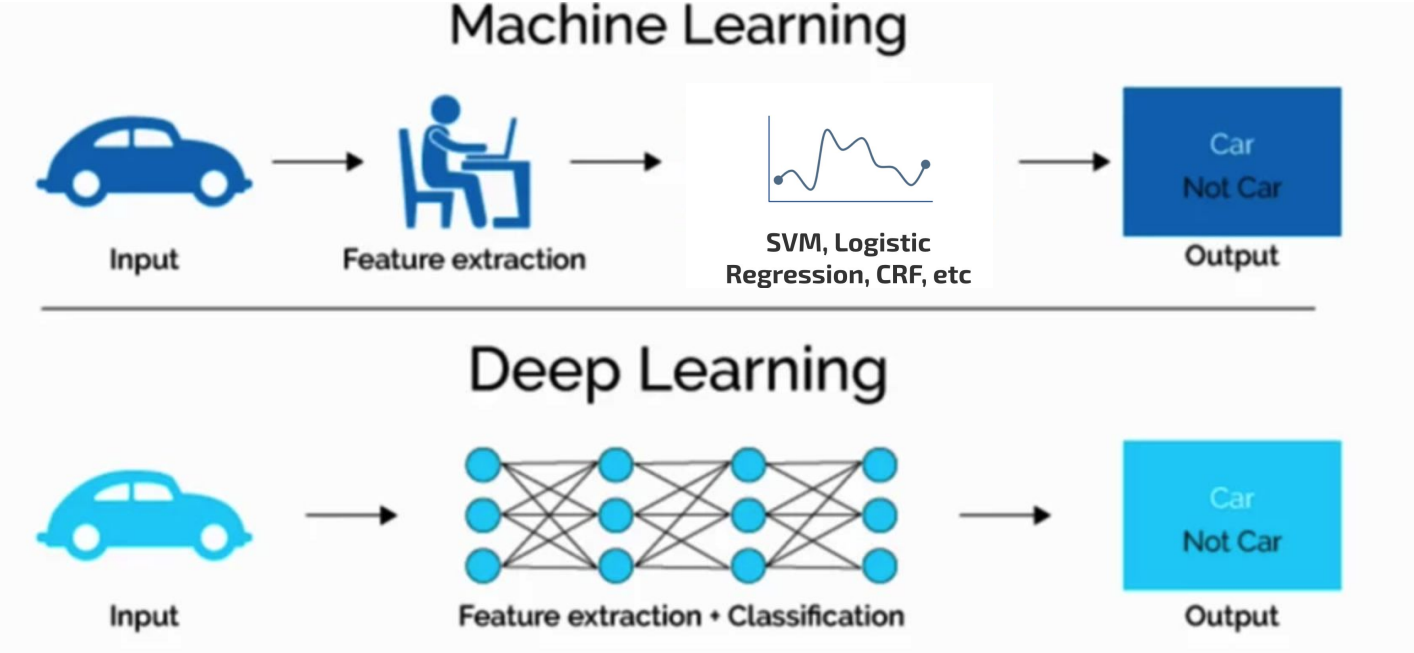
# Outline (Second season)

- TASK: Sentiment Analysis of tweets about Covid-19
  - Text representation
    - basic NLP tasks
    - Word Embeddings
  - Evaluation metrics for NLP tasks
  - Machine Learning algorithms
    - SVM, Logistic Regression
    - Deep Learning models:
      - BiLSTM with random initialization
      - BiLSTM with pre-trained word embeddings.
      - BERT



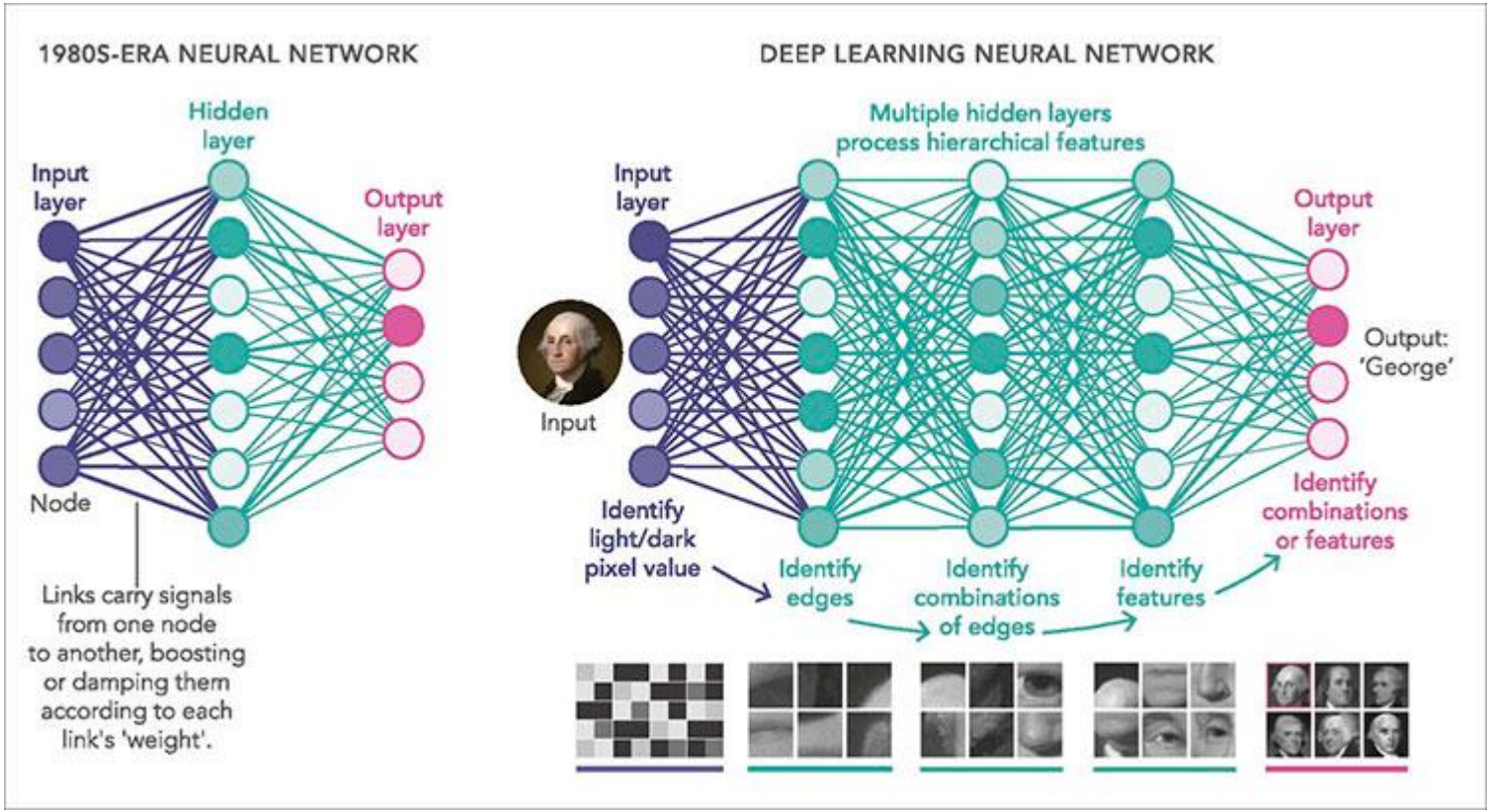

https://medium.com/@experienceIA18/diferencias-entre-la-inteligencia-artificial-y-el-machine-learning-f0448c503cd4

Traditional ML algorithms for NLP tasks heavily requiere on hand-crafted features. Feature engineering is a  time-consuming process. Moreover, it is not usually robust enough (low recall)




https://blog.bismart.com/es/diferencia-machine-learning-deep-learning

Employ multiple processing layers to learn hierarchical representations of data

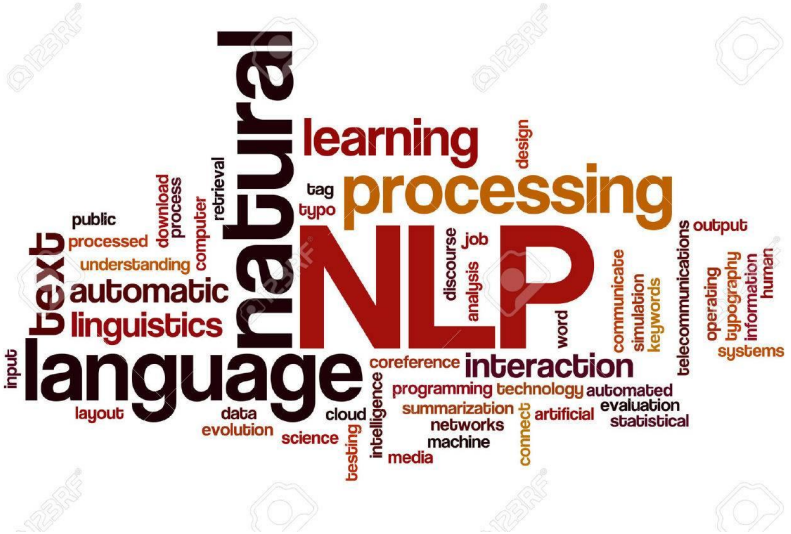



https://www.futurespace.es/redes-neuronales-y-deep-learning-capitulo-1-preludio/

Produce state-of-the-art results in many domains (such as computer vision, pattern recognition)

# Natural Language Processing

- Computational techniques for the automatic analysis and representation of human language
- Multidisciplinary: Linguistics, Mathematical and Computer Science, Psychology



# Why NLP?

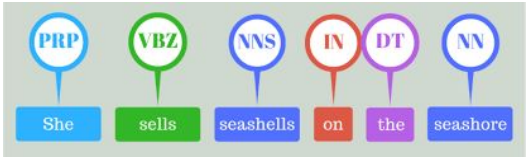

- Exponential Growth of data.
  - 2013, 3.5 ZB
  - 2022, 40 ZB
  - 2025, 180 ZB
- > 80% in unstructured form, primarily texts



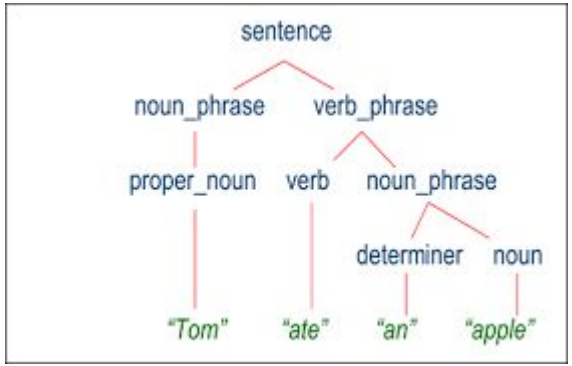
*1 ZB = 1 trillion GB

# NLP applications

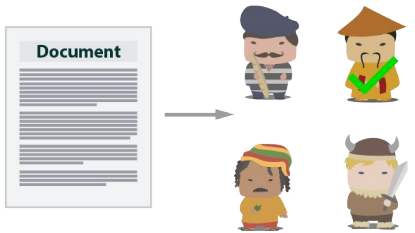
- NLP applications:
  - **part-of-speech (POS) tagging, parsing,**
  - **language identification**
  - Machine translation,
  - Information retrieval,
  - Information extraction,
  - Text classification,
  - Question answering,
  - Text summarization
  - Text simplification,
  - Conversational AI,
  - ...




https://github.com/dhirajhr/POS-Tagging

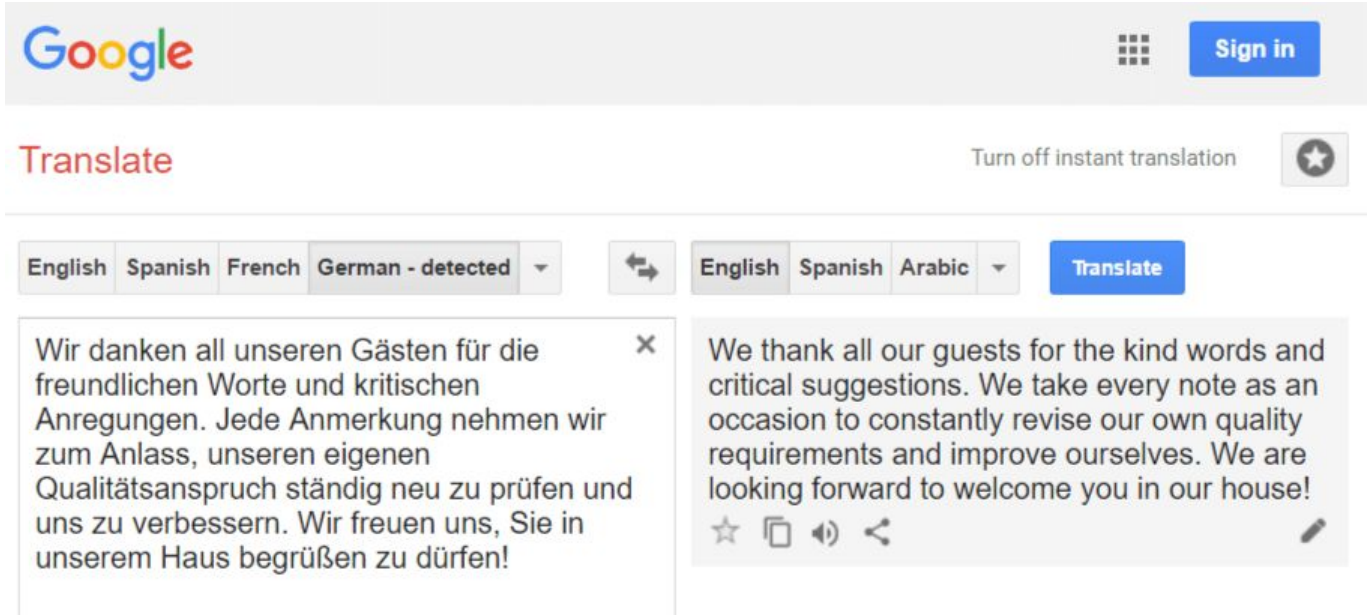



https://forum.huawei.com/

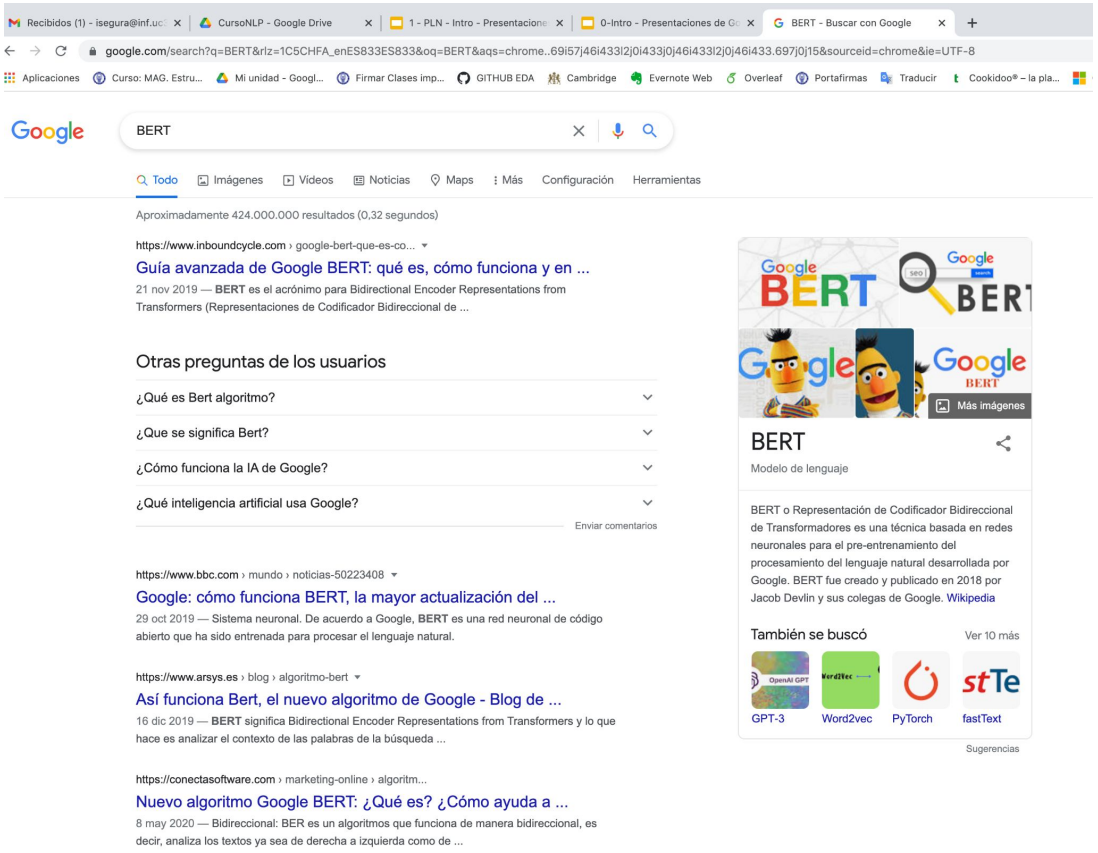



https://aylien.com/text-api/language-detection/

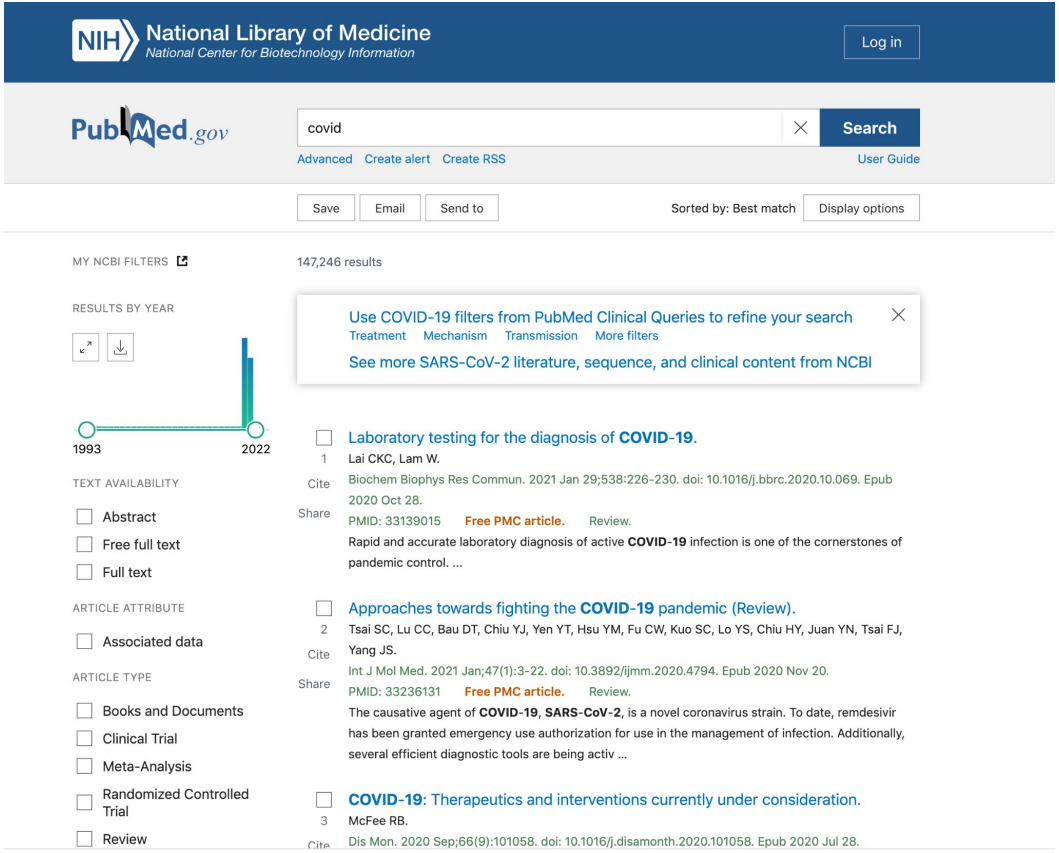
# Machine Translation (MT)



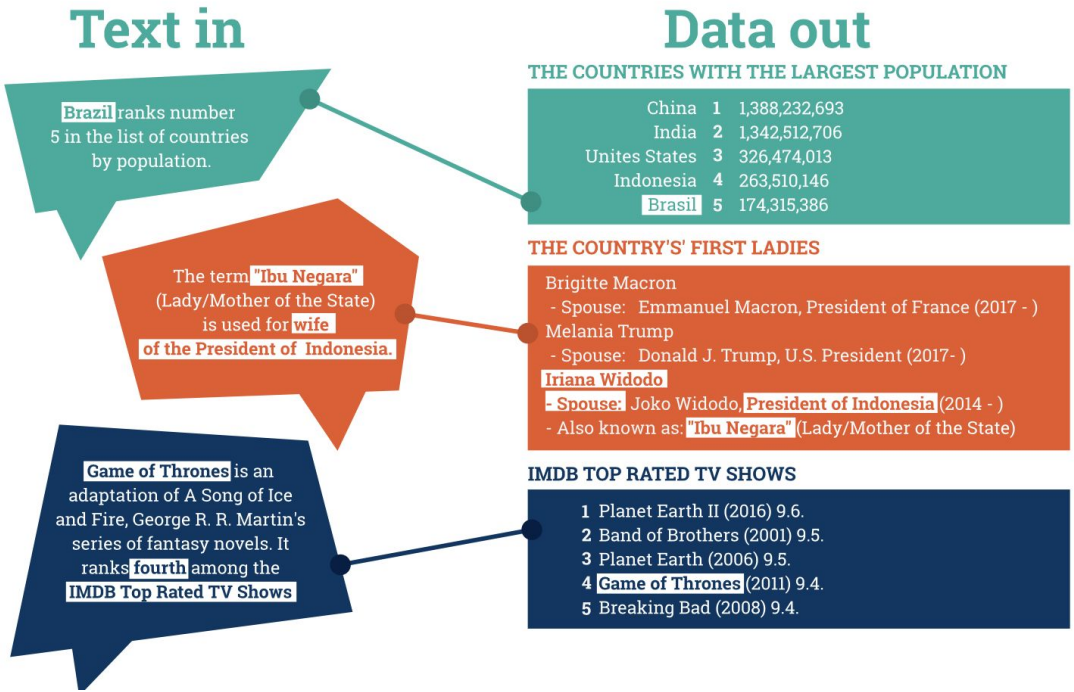
# Information Retrieval



# Information Retrieval



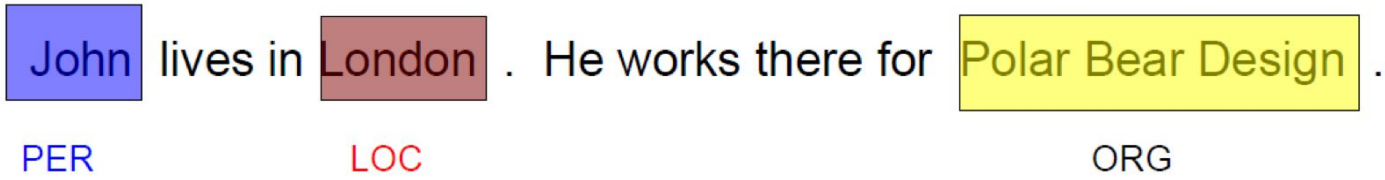
# Information Extraction



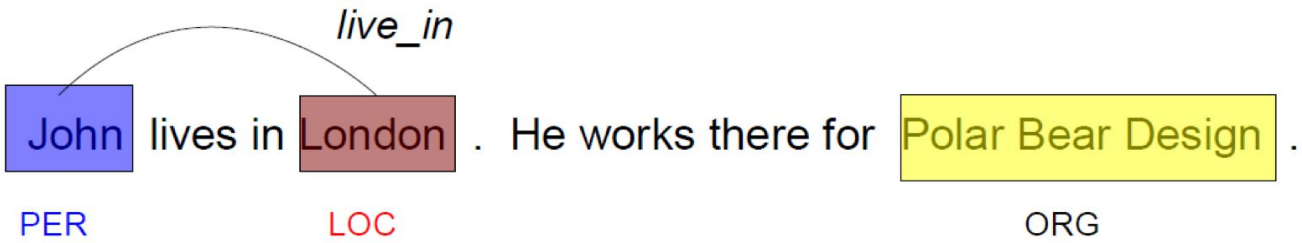

https://www.ontotext.com/knowledgehub/fundamentals/information-extraction/

# IE tasks

**1) Named Entity Recognition (NER)**



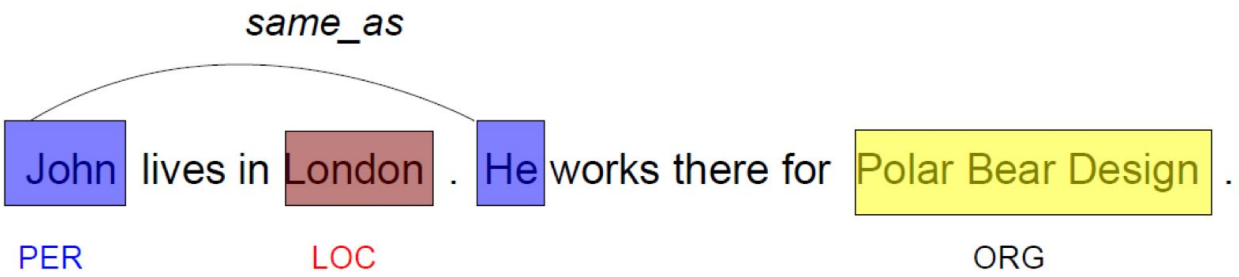

**2) Relation Extraction**

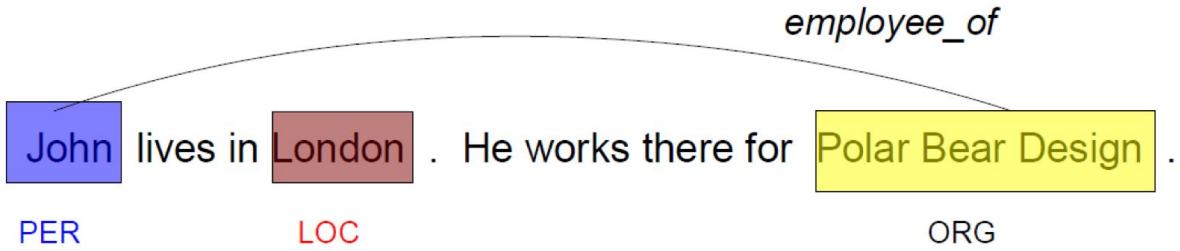



Source: https://gate.ac.uk/family/training.html

# IE tasks

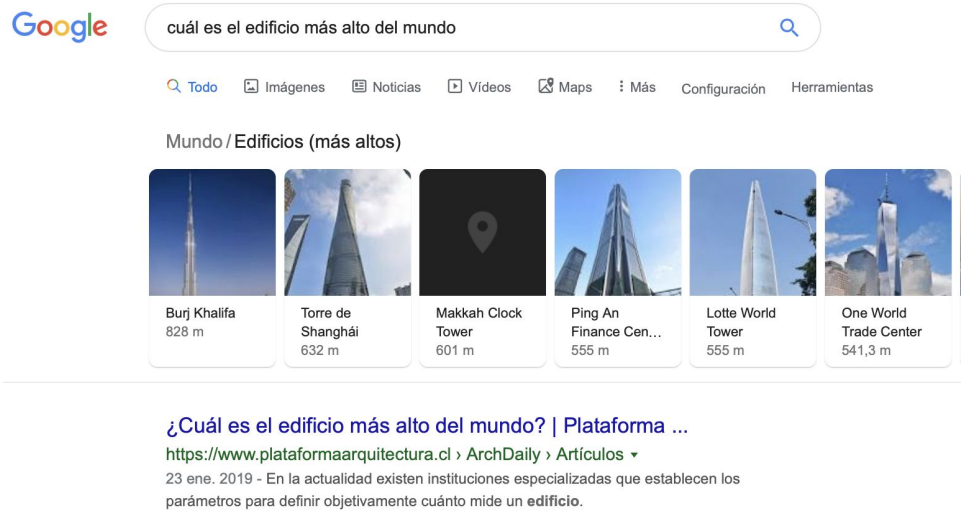
**3) Co-reference resolution**



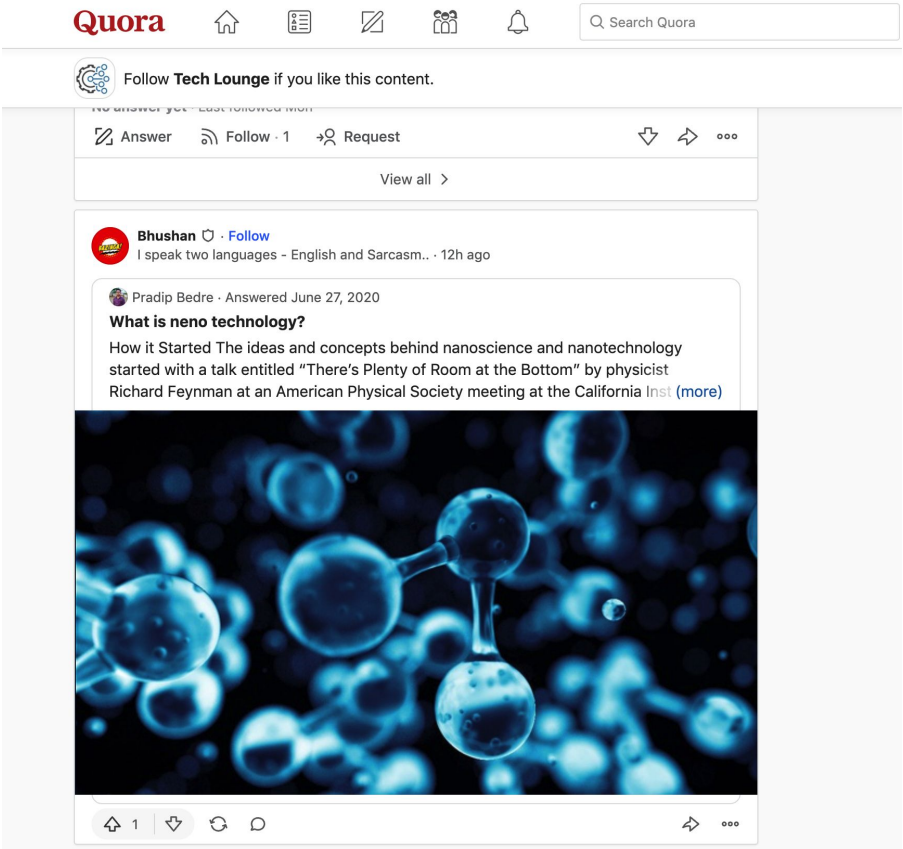




https://gate.ac.uk/family/training.html

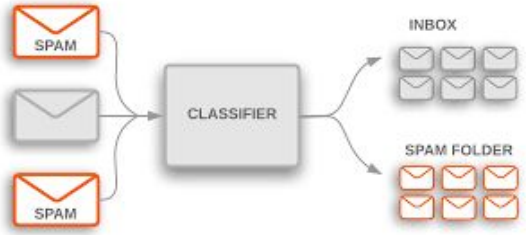
# Question Answering

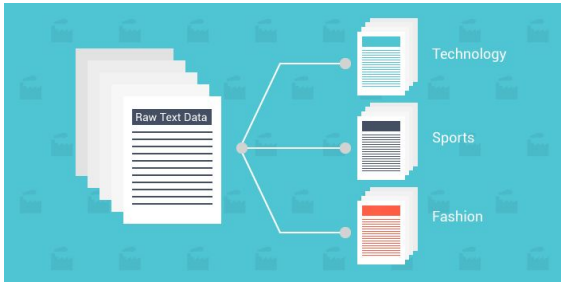


# Question Answering



# Text Classification






# Text summarization

**Source Text:** Peter and Elizabeth took a taxi to attend the night party in the city.

While in the party, Elizabeth collapsed and was rushed to the hospital.

**Summary:** Elizabeth was hospitalized after attending a party with Peter. 



https://blog.floydhub.com/gentle-introduction-to-text-summarization-in-machine-learning/

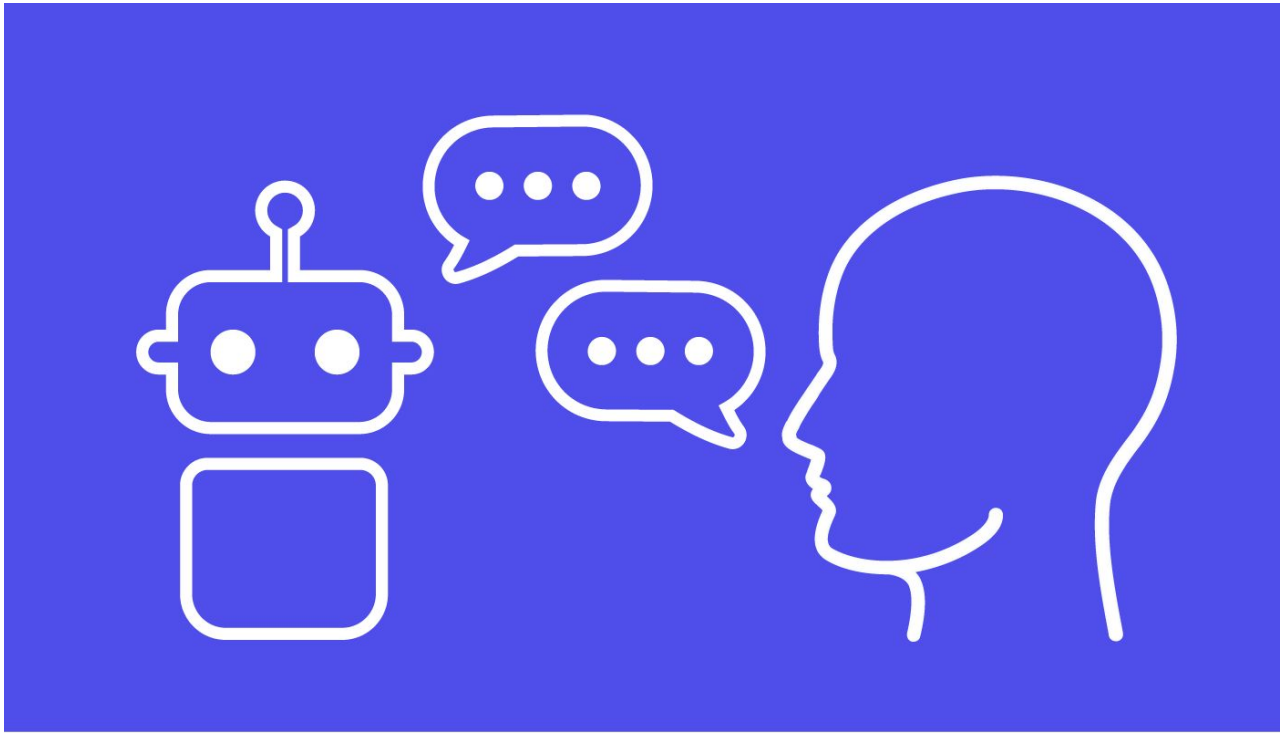
# Text simplification

| | |
|---|---|
| Normal: | Alfonso Perez *Munoz, usually referred to as Alfonso,* is a former Spanish *footballer, in the striker position.* |
| Simple: | Alfonso Perez is a former Spanish *football player.* |



Coster, W., & Kauchak, D. (2011, June). Simple English Wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2* (pp. 665-669). Association for Computational Linguistics.

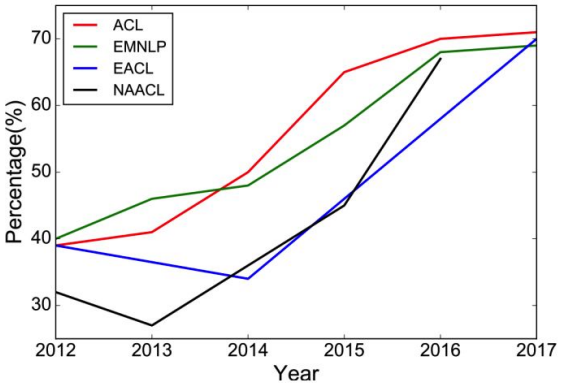
# Conversational AI (Chatbots)





https://medium.com/pat-inc/conversational-a-i-needs-meaning-not-keywords-part-1-e3b0d8825564

# DEEP LEARNING for NLP




## Deep Learning in NLP

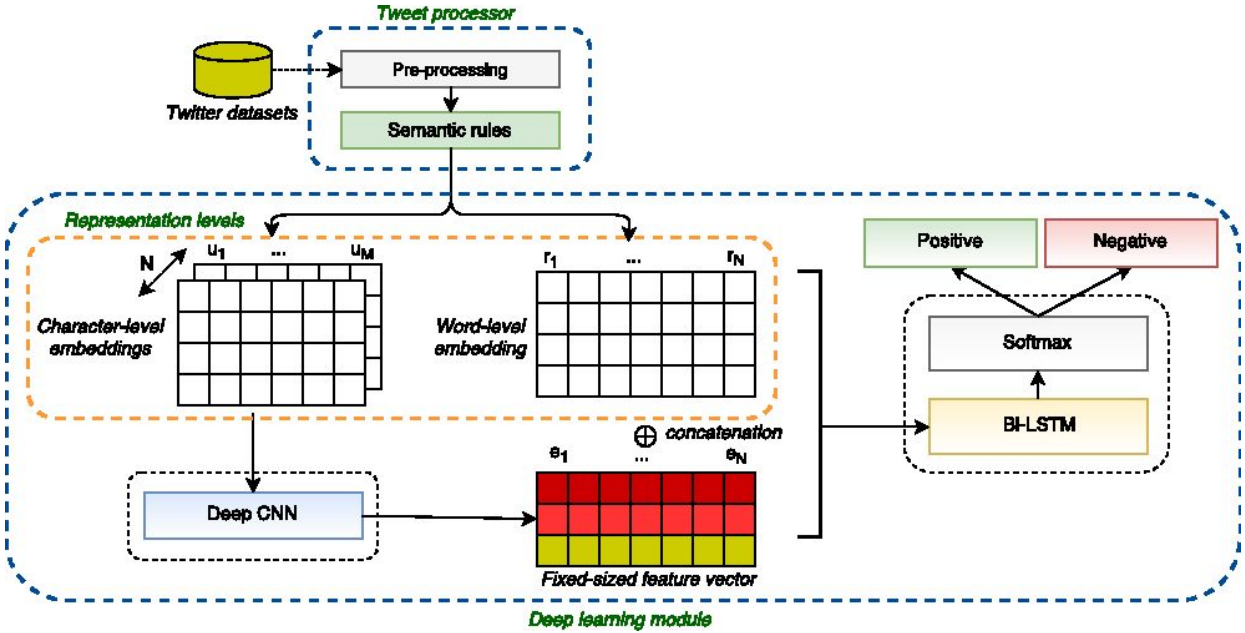
In the last years, NLP research has also exploited deep learning techniques

Produce state-of-the-art results in many NLP applications (machine translation, IE, text summarization, etc)

## Results

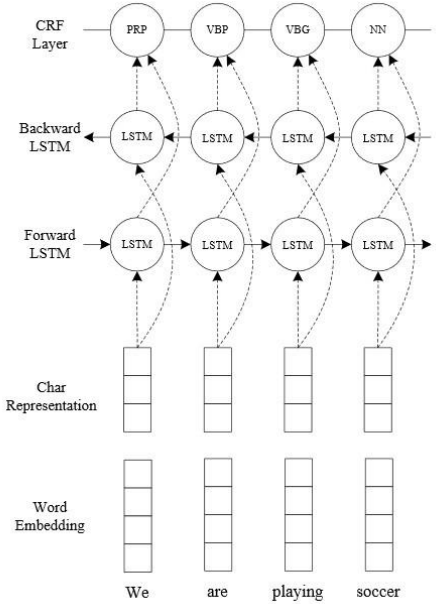
Figure from "Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. *ieee Computational intelligen Ce magazine*, *13*(3), 55-75."

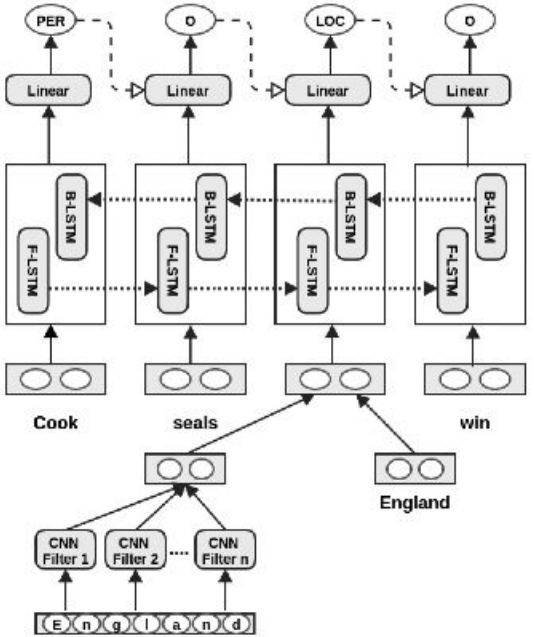
# BiLSTM for Text Classification (Sentiment Analysis of Tweets)




Dang, N. C., Moreno-García, M. N., & De la Prieta, F. (2020). Sentiment analysis based on deep learning: A comparative study. *Electronics*, *9*(3), 483.

# BiLSTM + CRF for Sequence Labeling tasks (PoS tagging and NER)

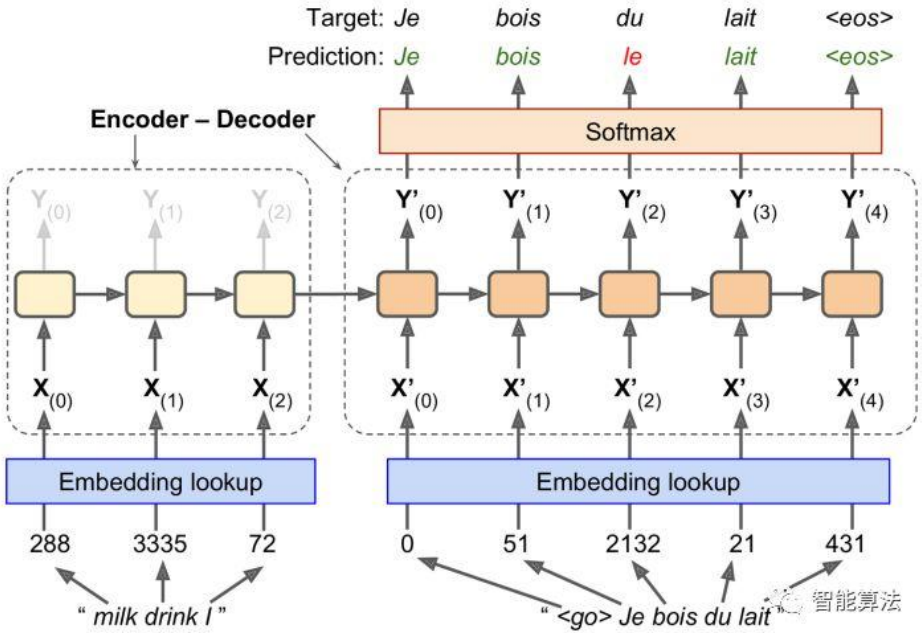



https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

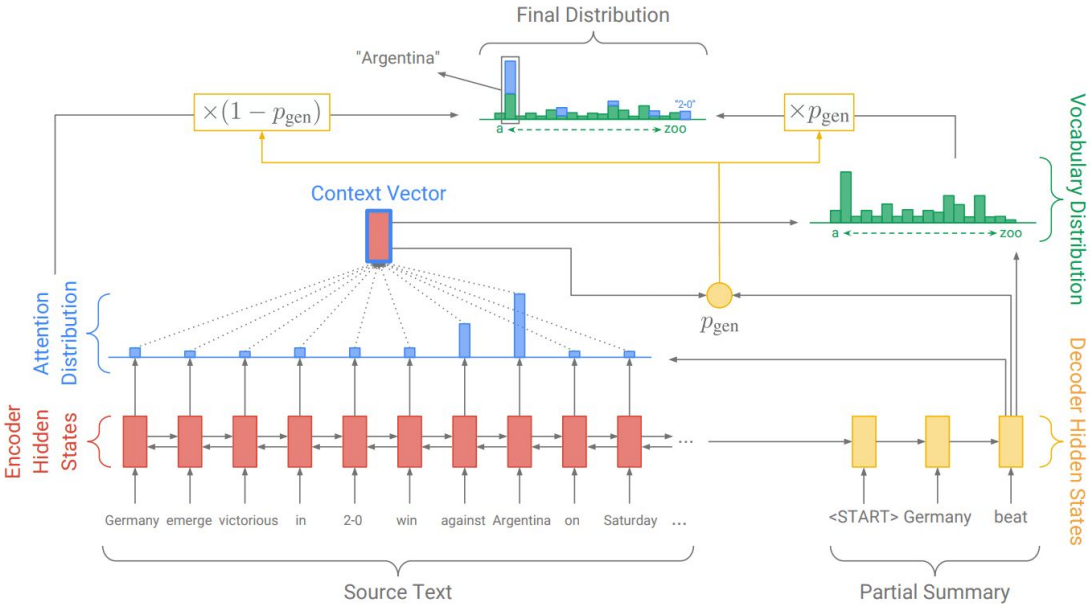
Murthy, R., Khapra, M. M., & Bhattacharyya, P. (2018). Improving NER tagging performance in low-resource languages via multilingual learning. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), 18*(2), 1-20.

# Seq2Seq for Machine translation




https://www.programmersought.com/article/83174154931/

# Transformers for Text summarization




https://machinelearningmastery.com/encoder-decoder-deep-learning-models-text-summarization/

# Outline (first season)

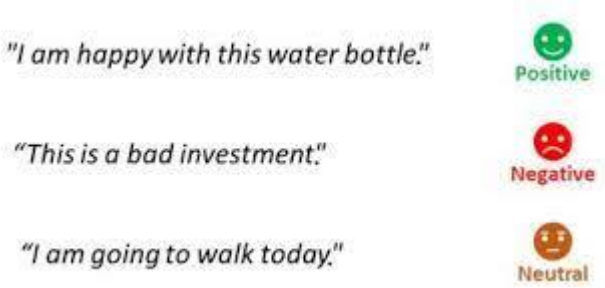
- Introduction
- **Word Embeddings**
- Deep learning architectures for NLP
  - Recurrent Neural Networks
  - Sequence to Sequence
  - Transformer
- Contextual languages models
  - BERT

# Machine Learning Pipeline

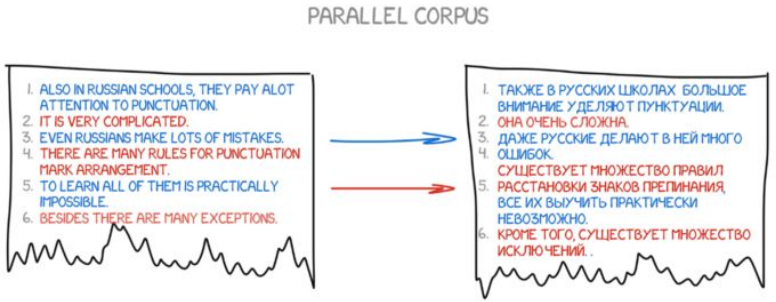
Texts
(training dataset)

Information
Extraction

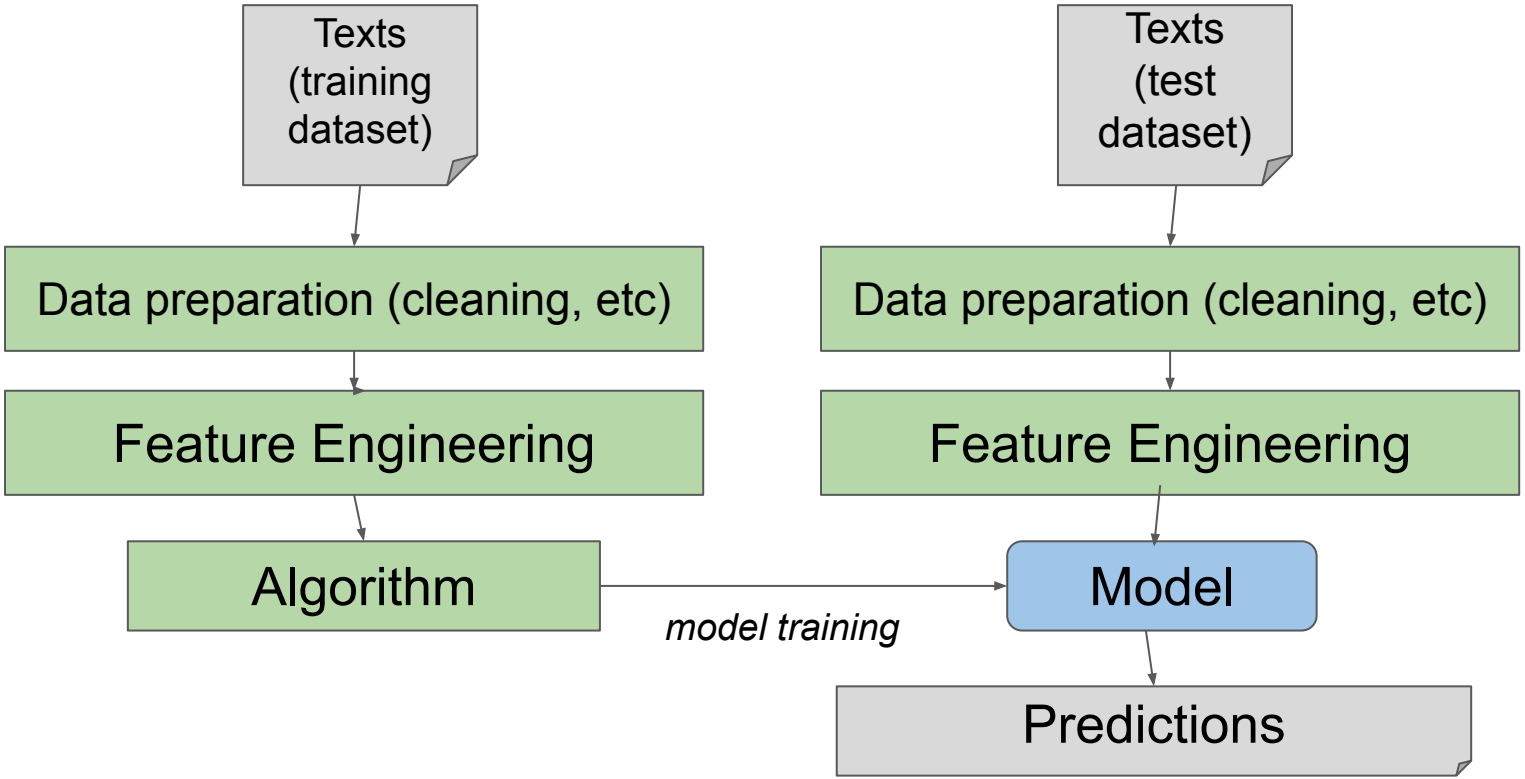

Text classification



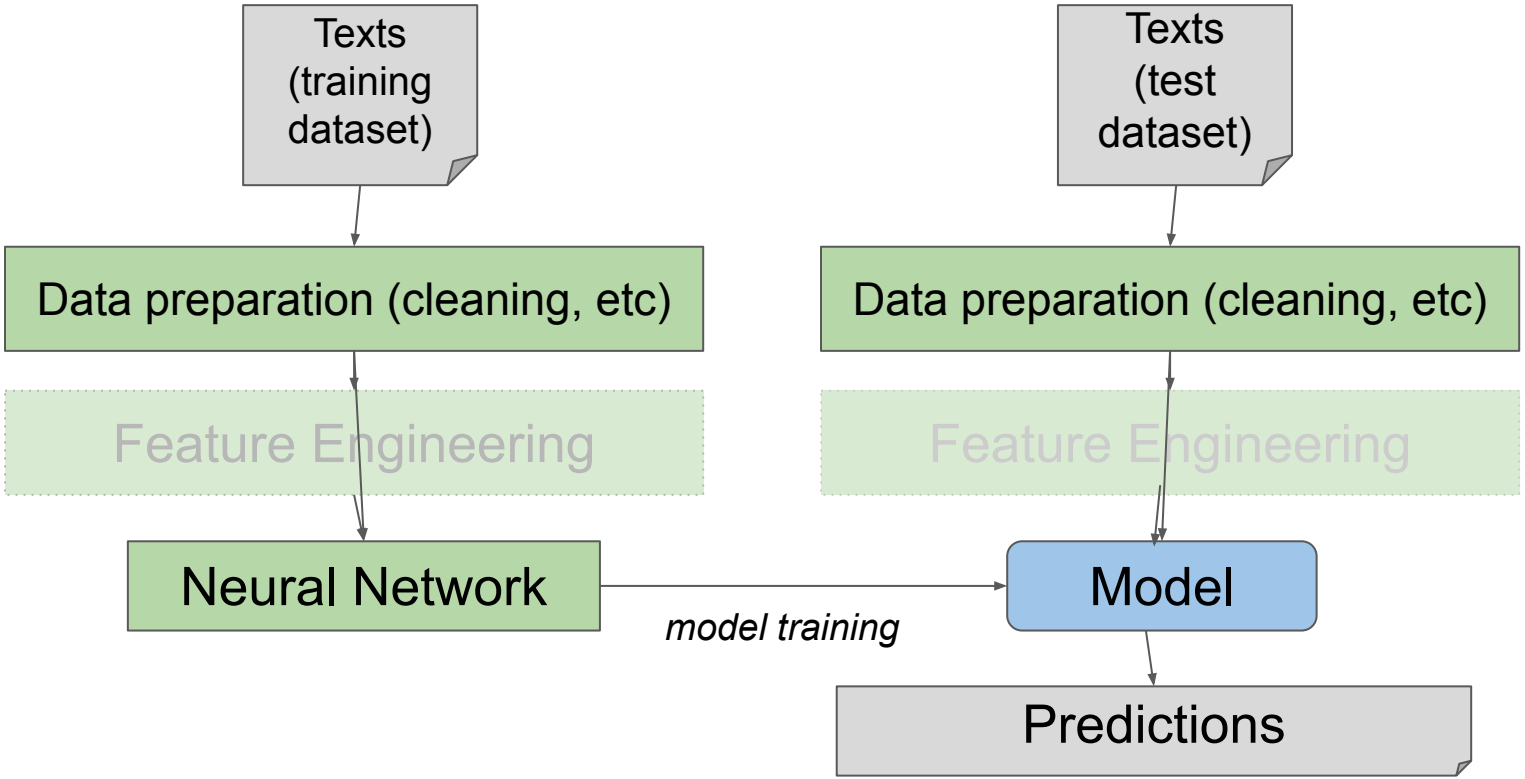

Machine translation or
Text summarization



# Machine Learning Pipeline



# Deep Learning Pipeline



# How to initialize a neural network?

- Two approaches:
  - random initialization
  - pre-trained word embeddings (vectors)

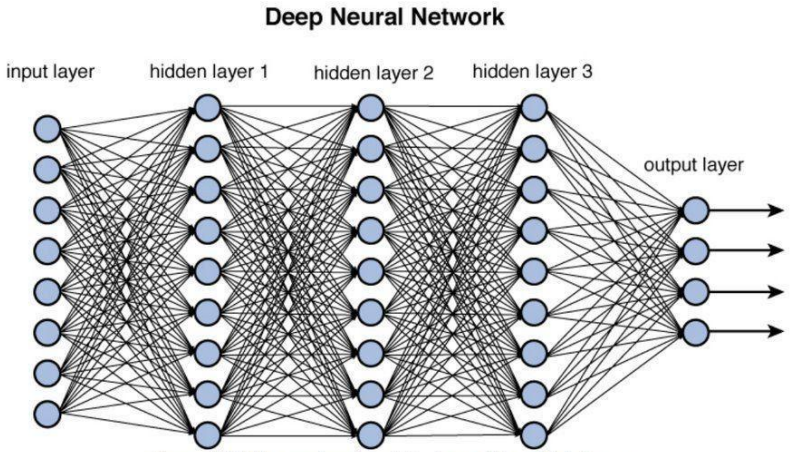
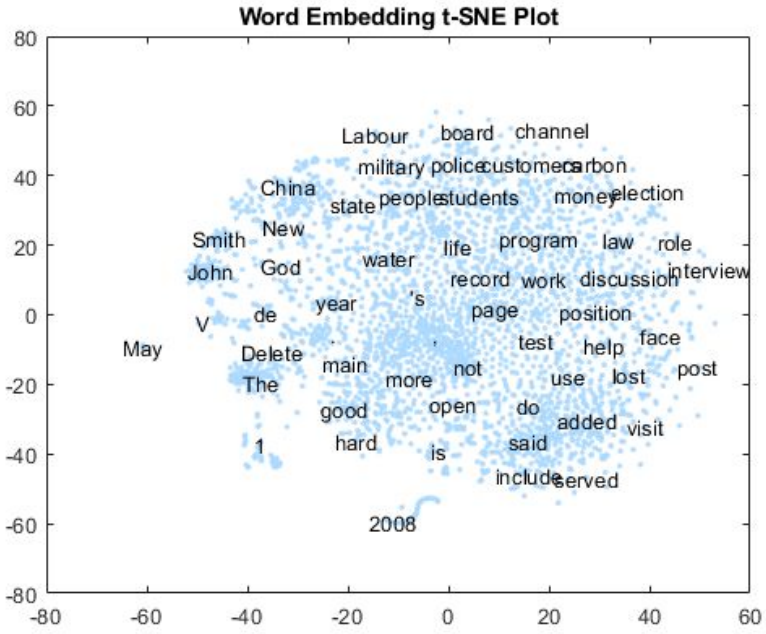


Figure 12.2 Deep network architecture with multiple layers.

https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964

# Word embeddings

- NLP techniques where words from the vocabulary are mapped to vectors of real number
- Traditional approaches: BoW and TF-IDF.
- Neural Networks: word2vec, glove, Fast2text




https://es.mathworks.com/help/textanalytics/ug/visualize-word-embedding-using-text-scatter-plot.html

# Bag of Words

- based on counting words in the document
- Steps:
  - Cleaning:
    - Remove stopwords, punctuation and special symbols.
    - Normalize texts (lemmatization or stemming).

# Lemmatization

- Obtain the lemma of a word.
  - house -> house
  - *smallest -> small*
  - *lawyer, lawsuit -> law*
  - *are -> be*
  - *women -> woman*
- *Decrease the vocabulary size and improve information retrieval*

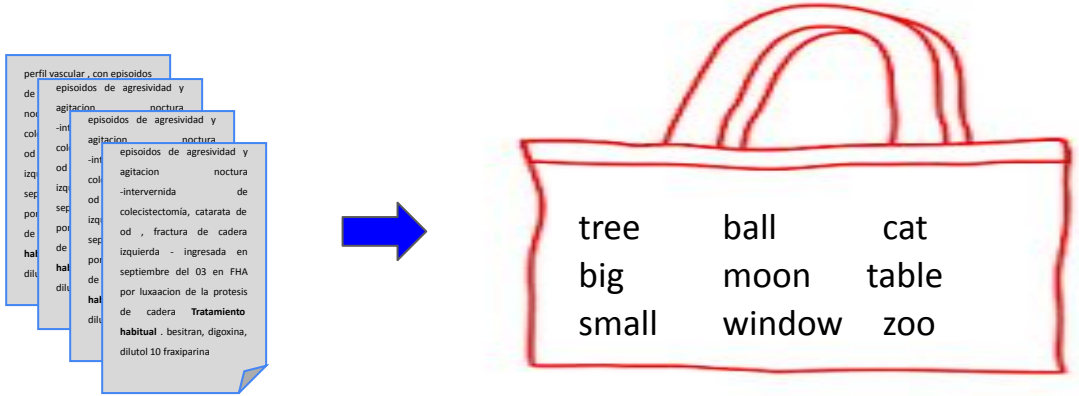
  *smaller, smallest -> small*
- online lemmatizer:
  - https://cst.dk/tools/index.php#output
  - http://www.gedlc.ulpgc.es/investigacion/scogeme02/lematiza.htm

# Stemming

- Obtain the stem of each word
  - *singing -> sing*
  - *singer -> sing*
  - *house -> hous*
  - *lawyer -> lawyer*
  - *lawsuit -> lawsuit*
  - *are -> are*
  - *women -> women*
- *Decrease the vocabulary size and improve information retrieval*
  - *poliposis nasal = pólipos nasales*
- Stemmer online:
  - *https://snowballstem.org/demo.html*

# Bag of Words

- Cleaning
- Obtain vocabulary (unique words) from all texts.




| ball | big | cat | moon | small | table | tree | window | zoo |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

# Bag of Words

- Each text is represented as a vector with the frequencies of their words

*D: The big cat in on the table and the small cat in the window.*
after cleaning:
*D: ~~The~~ big cat ~~is on the~~ table ~~and the~~ small cat ~~en la~~ window~~.~~*

Vector (features):

| ball | big | cat | moon | small | table | tree | window | zoo |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 0 | 1 | 1 | 0 | 1 | 0 |

# Bag of Words

*D1: ~~The~~ big cat ~~is on the~~ table ~~and the~~ small cat ~~in the~~ window*
*D2: ~~The~~ table ~~and the~~ window ~~are~~ small*
*D2: ~~The~~ moon ~~and the~~ small tree ~~are~~ big*

Vector (features):

| | ball | big | cat | moon | small | table | tree | window | zoo |
|---|---|---|---|---|---|---|---|---|---|
| D1 | 0 | 1 | 2 | 0 | 1 | 1 | 0 | 1 | 0 |
| D2 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| D3 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |

# TF-IDF

- Extended version of BoW.
- Every text is represented using tf-idf of its words
- TF-IDF decrease the weight of the very common words in the collection of texts
- Esta métrica, TF-IDF, consigue disminuir el peso de las palabras que son muy comunes en toda la colección de documentos.

# TF-IDF

- Term frequency - inverse document frequency.

**TF-IDF (W) = TF(W,d) * IDF(W)**

  - TF(W,d) = term frequency of the word *W* in the document d.
  - IDF(W) = inverse document frequency. The logarithm of the quotient of the total number of documents and the number of documents that contains the word *W*.

$$\mathrm{idf(W)} = \log \frac{\#(\text{documents})}{\#(\text{documents containing word W})}.$$

https://en.wikipedia.org/wiki/Tf%E2%80%93idf

# TF-IDF

*D1: ~~The~~ big cat ~~is on the~~ table ~~and the~~ small cat ~~in the~~ window*
*D2: ~~The~~ table ~~and the~~ window ~~are~~ small*
*D2: ~~The~~ moon ~~and the~~ small tree ~~are~~ big*

**Bag of Words**

| | ball | big | cat | moon | small | table | tree | window | zoo |
|---|---|---|---|---|---|---|---|---|---|
| D1 | 0 | 1 | 2 | 0 | **1** | 1 | 0 | 1 | 0 |
| D2 | 0 | 0 | 0 | 0 | **1** | 1 | 0 | 1 | 0 |
| D3 | 0 | 1 | 0 | 1 | **1** | 0 | 0 | 0 | 0 |

**TF-IDF (W) = TF(W,d) * IDF(W)**

$$\text{idf(W)} = \log \frac{\#(\text{documents})}{\#(\text{documents containing word W})}.$$

| | ball | big | cat | moon | small | table | tree | window | zoo |
|---|---|---|---|---|---|---|---|---|---|
| D1 | 0 | 0.17 | 0.95 | 0 | **0** | 0.17 | 0 | 0.17 | 0 |
| D2 | 0 | 0 | 0 | 0 | **0** | 0.17 | 0 | 0.17 | 0 |
| D3 | 0 | 0.17 | 0 | 0.47 | **0** | 0 | 0.47 | 0 | 0 |

# Drawbacks of traditional approachess

- Have high dimensionality and are very sparse.
- Don’t capture semantics
  - *Edema de glotis != hinchazón de la laringe*
- Don’t position of occurrence of words
  - *The hotel was very good and not expensive != The hotel was very expensive and not good*

Figure taken from https://laptrinhx.com/word-embeddings-part-2-3902951508/

# Neural Networks for Word embeddings

- Shallow neural networks trained on a large unlabeled corpus to predict a word based on its context (or given a context, to predict the most appropriate word for it).
- Word2Vec, Glove, FastText.
- Efficient in capturing context similarity.
- Cosine distance (fast and efficient).

**Distributional Hypothesis**
(Harris, 1954)
Words with similar meanings tend to occur in similar context

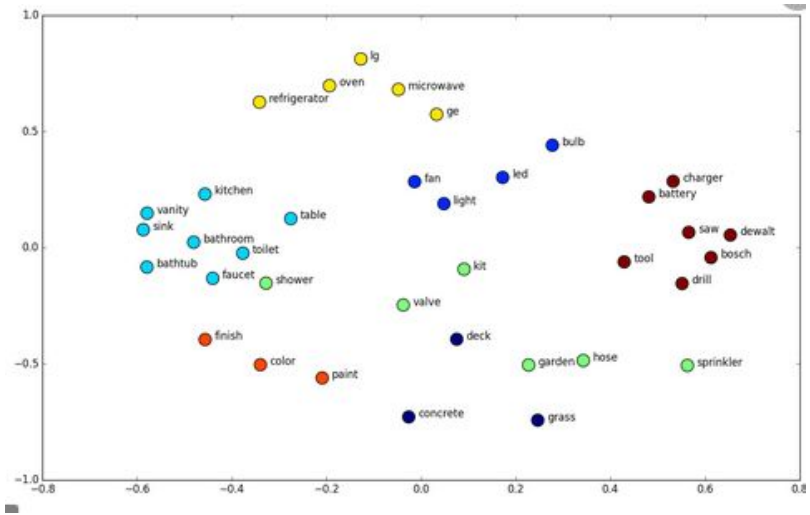



Figure taken from https://laptrinhx.com/word-embeddings-part-2-3902951508/

# Neural Networks for Word embeddings

- Capture syntactic and semantic information.

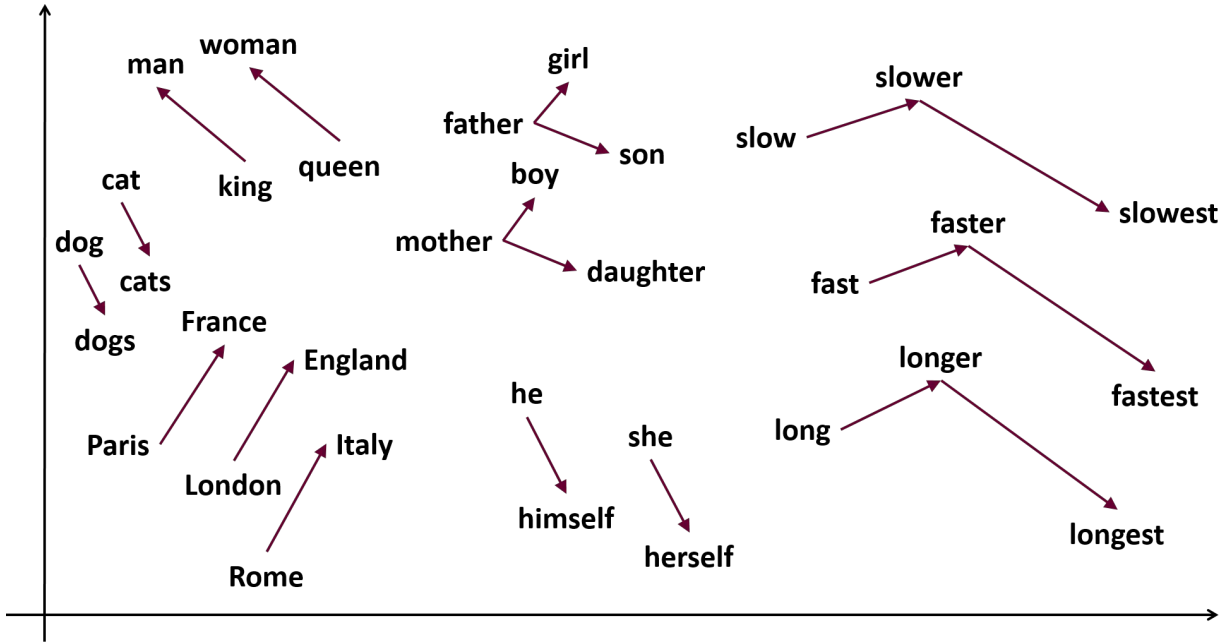
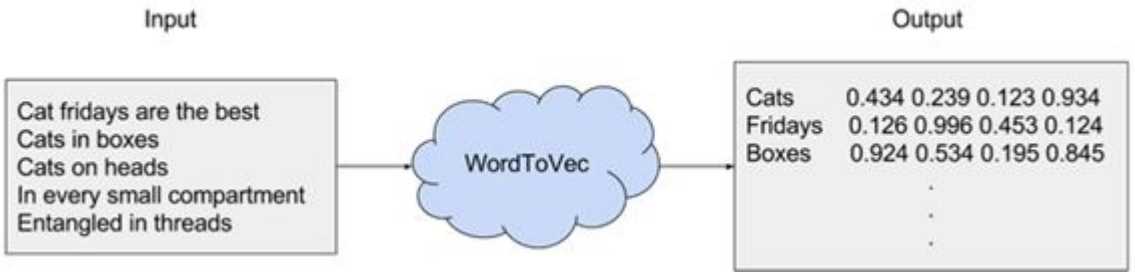



Figure taken from https://medium.com/analytics-vidhya/implementing-word2vec-in-tensorflow-44f93cf2665f

# Word2Vec




https://medium.com/@zafaralibagh6/simple-tutorial-on-word-embedding-and-word2vec-43d477624b6d

# Word2Vec

- Proposed two architectures to efficiently create word embeddings: continuous bag-of-words (CBOW) and skip-grams models.
- CBOW computes the conditional probability of a target word given the context words surrounding it across a window of size k.
- Skip-gram model predicts the surrounding context words given the central target word

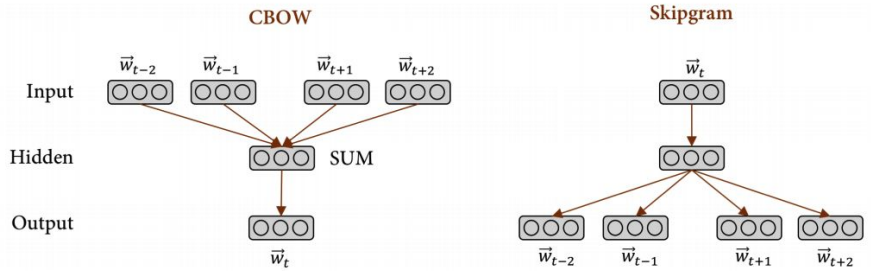



Figure taken from Thomás Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean: Efficient Estimation of Word Representations in Vector Space. ICLR 2013.

# Word2Vec: CBOW

- A simple fully connected neural network with one hidden layer.
- The input layer takes a one-hot vector for each context word. The output layer is softmax probability over all words in the vocabulary
- Finally, each word from the vocabulary is represented as two learned vectors, corresponding to the context and target word representations.

$$\mathbf{v_c} = \mathbf{W}_{(\mathbf{k},.)} \quad and \quad \mathbf{v_w} = \mathbf{W}'_{(.,\mathbf{k})}$$

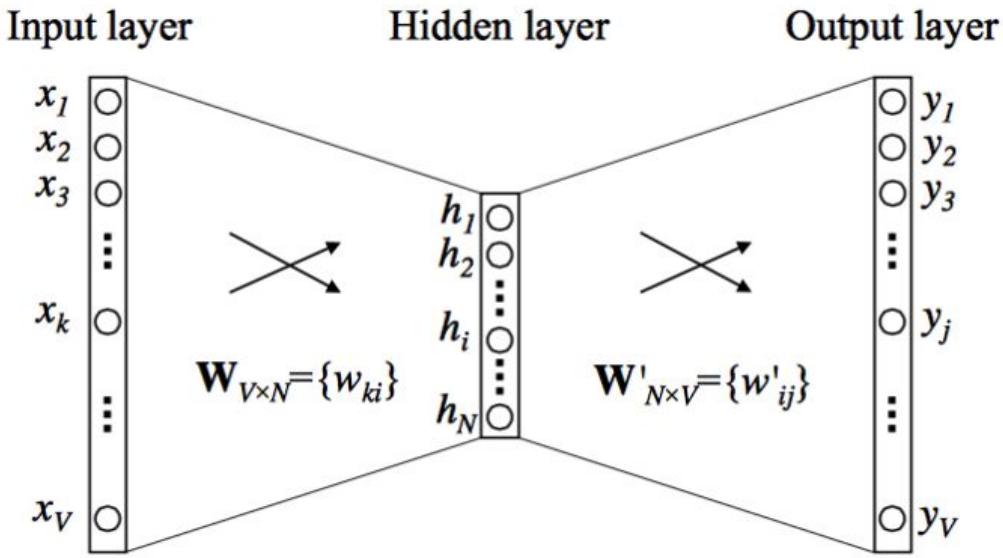
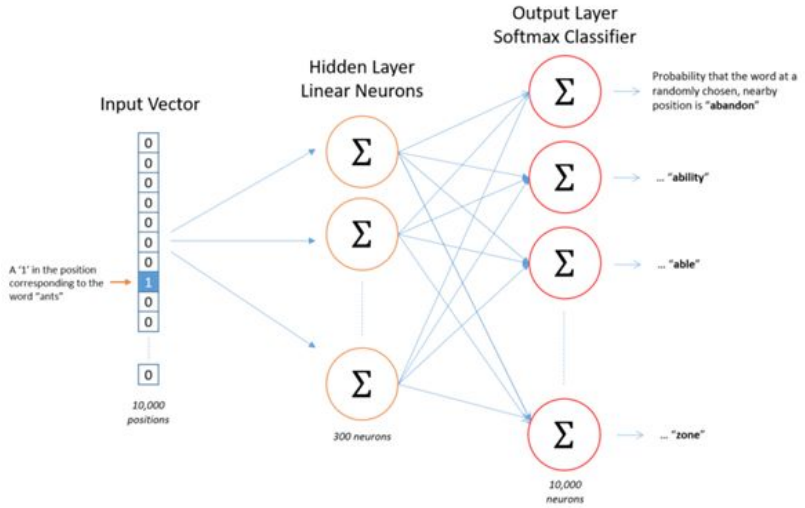



Figure taken from Thomás Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean: Efficient Estimation of Word Representations in Vector Space. ICLR 2013.

# Word2Vec: Skip-gram

- input word is represented with a hot-vector, with dimension the the size of the vocabulary (total unique words)
- the output is a vector containing for each word in the vocabulary, its probability as neighbour of the input word.
- Produces more accurate results on large datasets.




https://towardsdatascience.com/skip-gram-nlp-context-words-prediction-algorithm-5bbf34f84e0c

# Drawbacks of neural word embeddings

- Training word embeddings models require large amount of texts and time.
- Inability to represent phrases (“Joe Biden”, “American Airlines”, “Ford Motor Company”).
- Do not handle polysemy and homonyms correctly.
- Some words such as ***good*** and ***bad*** have very similar word embeddings (very small context window).

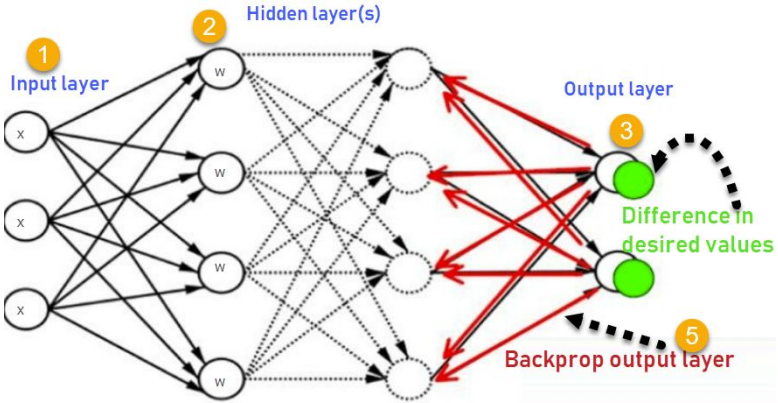
# Pre-trained word embedding models

- Repository: http://vectors.nlpl.eu/repository/

# Outline (first season)

- Introduction
- Word Embeddings
- **Deep learning architectures for NLP**
  - **Recurrent Neural Networks**
  - Sequence to Sequence
  - Transformer
- Contextual languages models
  - BERT

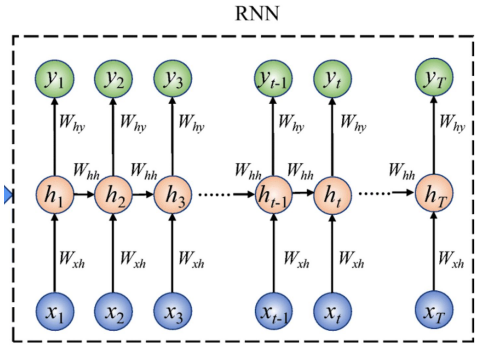
# Artificial Neural Network




https://www.guru99.com/backpropogation-neural-network.html

# Recurrent Neural Network

- Processes **sequences** (words, symbols, images, sensor measurements, etc) of different length.
- Remembers the past and uses it to take the next decision.
- Influenced not just by weights applied on inputs like a regular NN, but also by a “hidden” state vector representing the context based on prior input(s)/output(s)




Yang, B., Yin, K., Lacasse, S., & Liu, Z. (2019). Time series analysis and long short-term memory neural network to predict landslide displacement. *Landslides*, *16*(4), 677-694.

# Applications of RNNs

- Computer vision: image and video caption generation, image detection, etc
- Prediction problems (power demand, airline traffic volume, etc)
- Speech recognition
- NLP: ***PoS tagging, NER, language modelling, text summarization, machine translation, text classification, etc***

# RNN for NLP

*text summarization*
*machine translation*

*multilabel text classification*
*Sequence labeling task: NER, PoS tags*

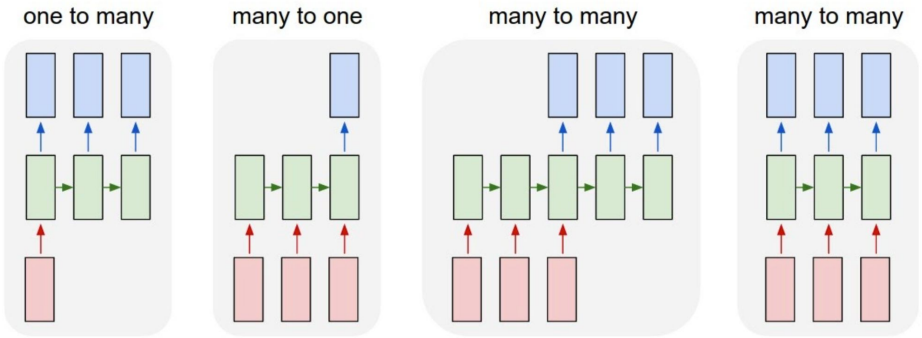
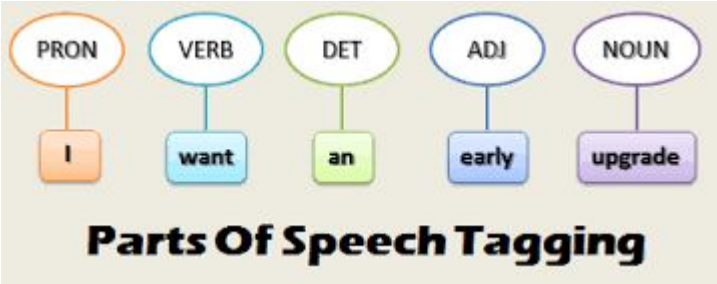


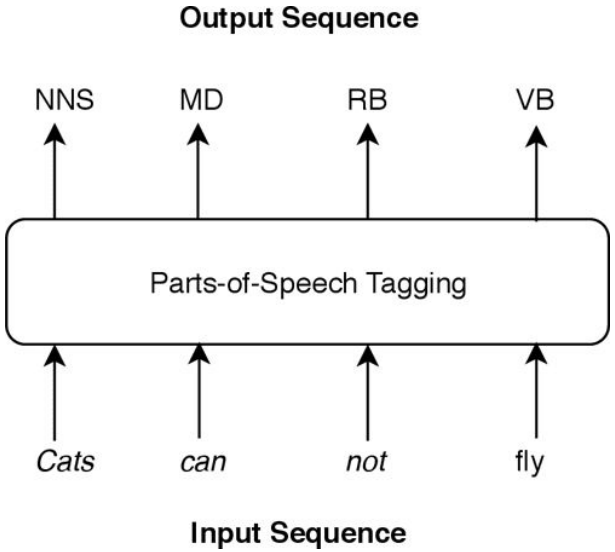

*image captioning*

*text classification*
*next word prediction (language modelling)*

# Why RNN for NLP?

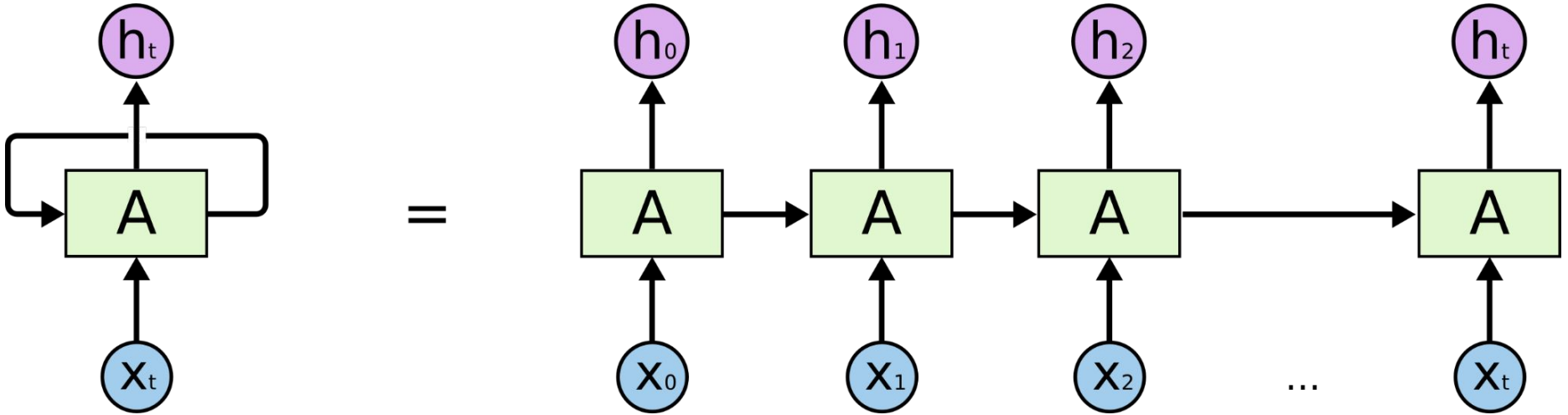



Sources: https://thinkinfi.com/extract-custom-keywords-using-nltk-pos-tagger-in-python/



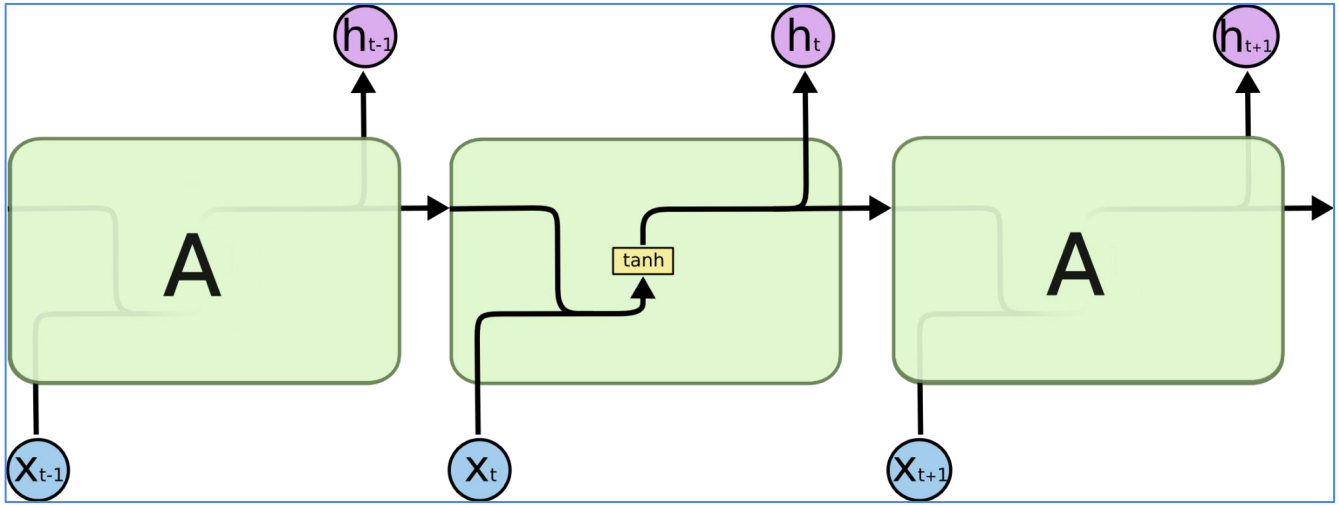

Source: Ahmed, M., Samee, M. R., & Mercer, R. (2018, December). Improving Neural Sequence Labelling Using Additional Linguistic Information. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 650-657). IEEE.

# RNN cells




http://colah.github.io/posts/2015-08-Understanding-LSTMs/

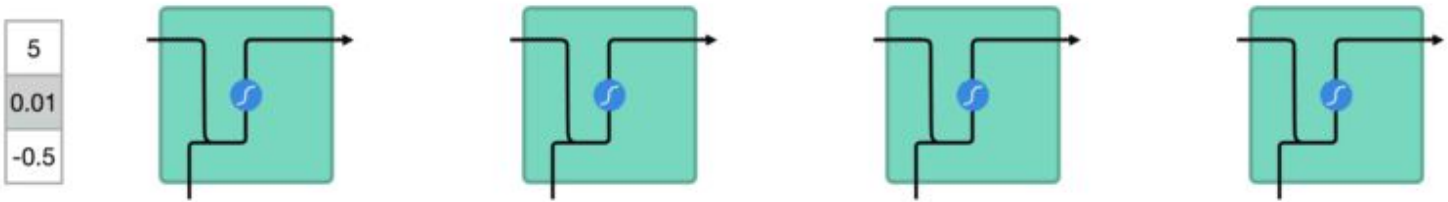
# RNN cell




http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# RNN cell

vector transformations without tanh

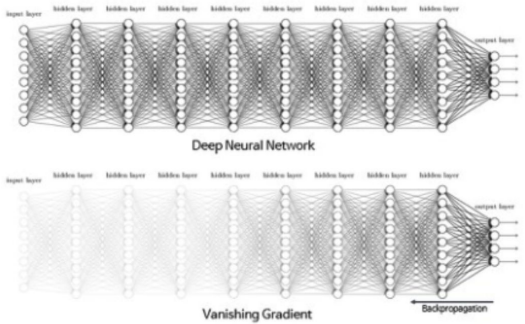



vector transformations with tanh

https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

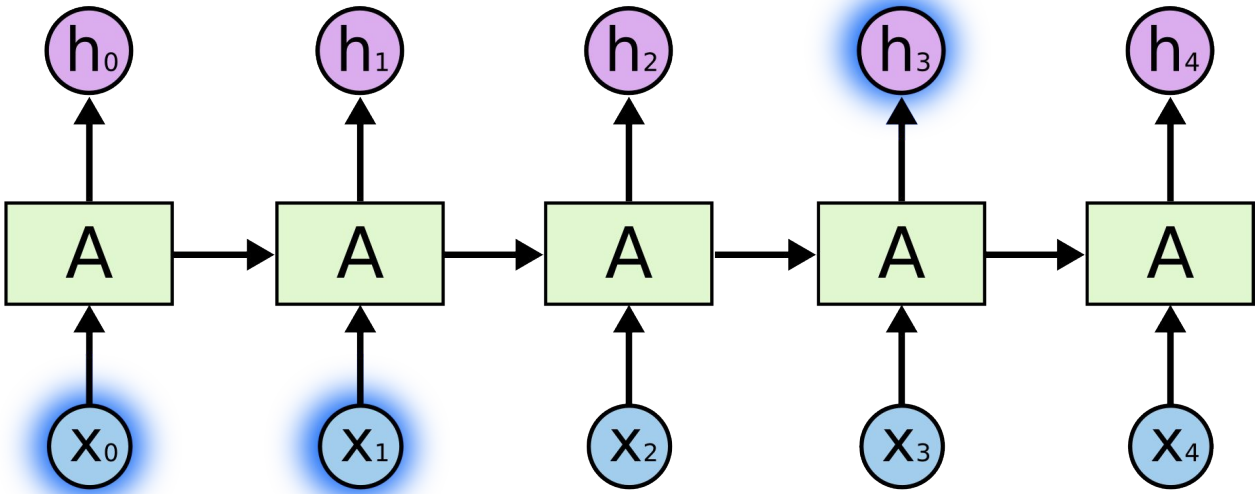
# Vanishing gradient problem

- RNNs suffer from the vanishing gradient.
- **Vanishing gradient** leads to slow convergence



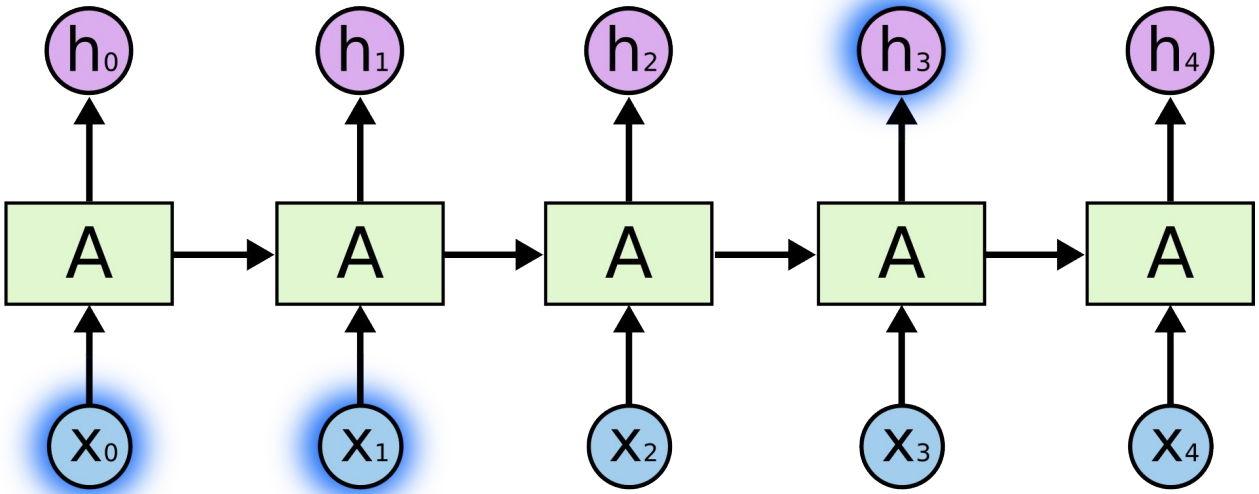

https://hackernoon.com/tackle-bias-and-other-problems-solutions-in-machine-learning-models-f4274c5fe538

# The problem of Long-term dependencies




http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# The problem of Long-term dependencies

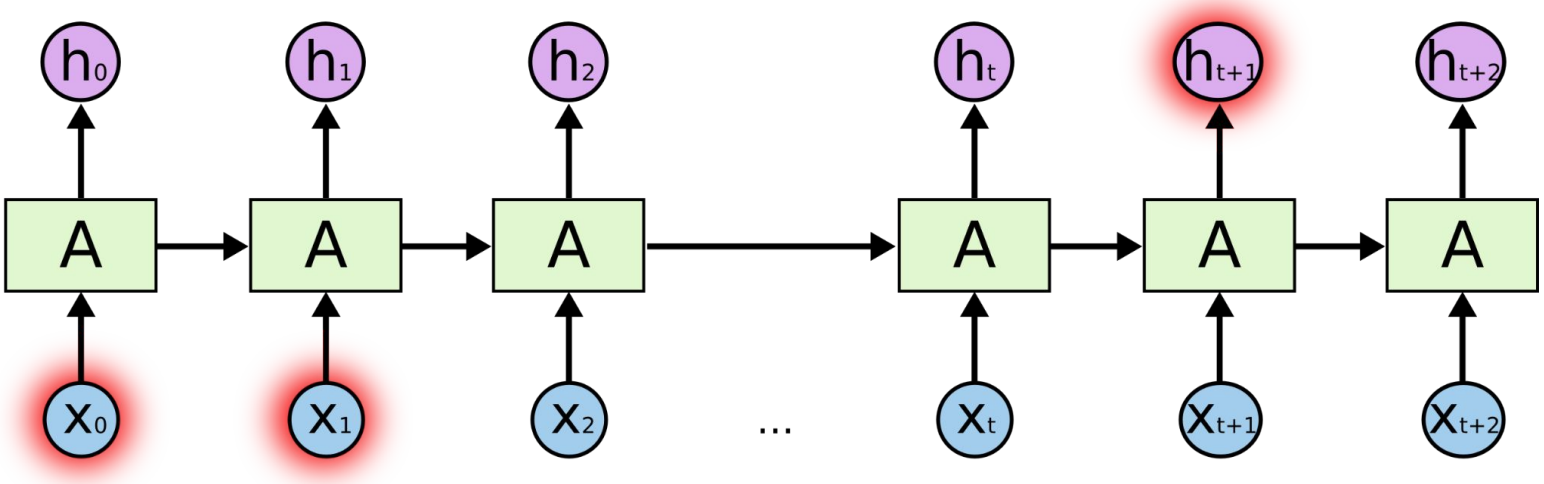



The clouds are in sky

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# The problem of Long-term dependencies

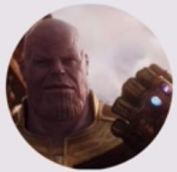
*I **grew up in <u>France</u>**, in Lyon, surrounded by vineyards, markets and my parents' bistro. I **speak fluent** French.*




http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# The problem of Long-term dependencies

**Customers Review** 2,491

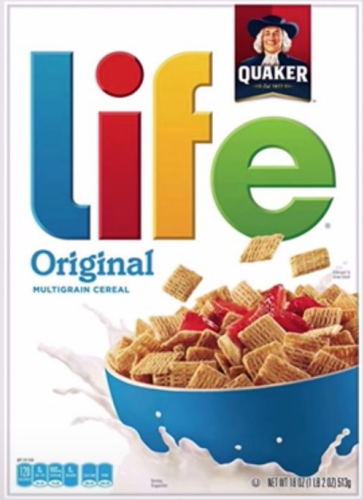


**Thanos**

September 2018

Verified Purchase

**Amazing!** This box of cereal gave me a **perfectly balanced breakfast**, as all things should be. I only ate half of it but **will definitely be buying again!**



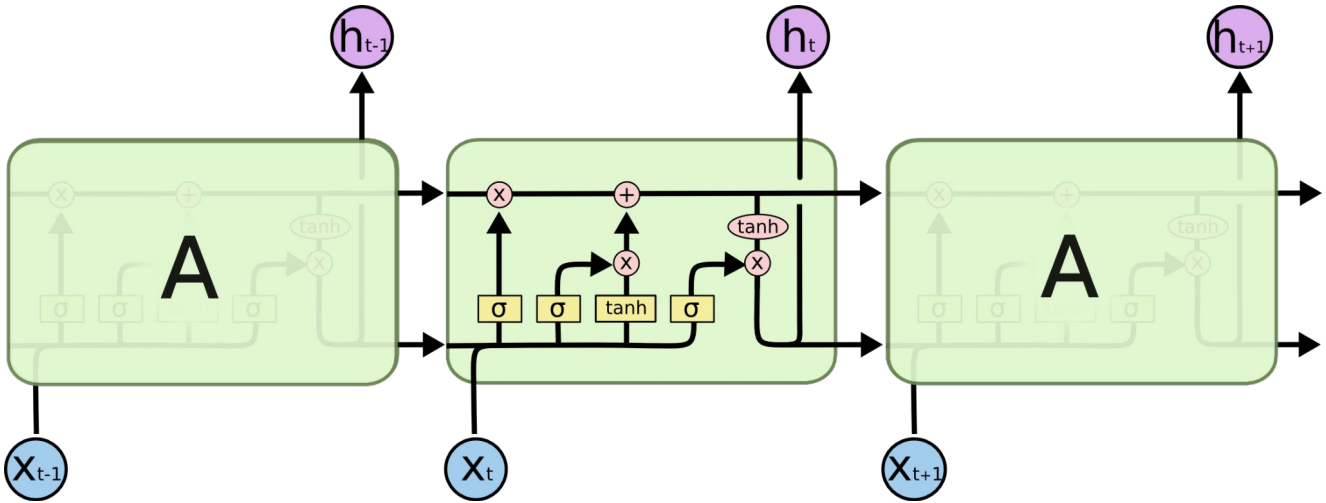

**A Box of Cereal**

**$3.99**

https://youtu.be/8HyCNIVRbSU

# Long-short Term Memory Networks (LSTM)

Gates allow LSTM cells to control what information to remove, store, and output to the next cell

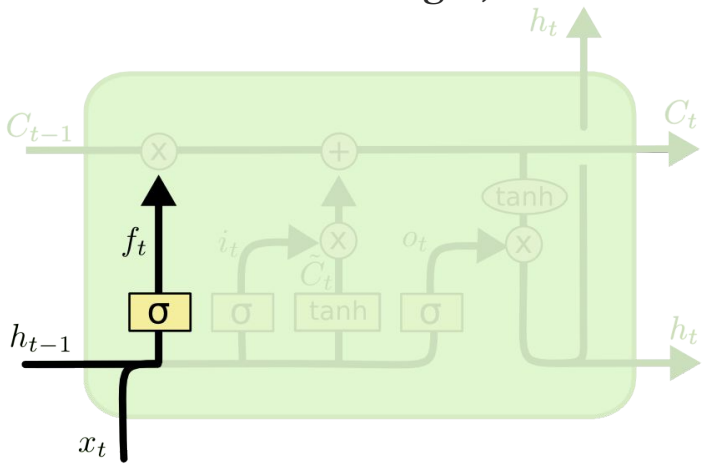



Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

# Forget gate layer

which information to forget

To do this, it uses a sigmoid layer that outputs a number between 0 and 1: the closer to 0 means to forget, and the closer to 1 means to keep.

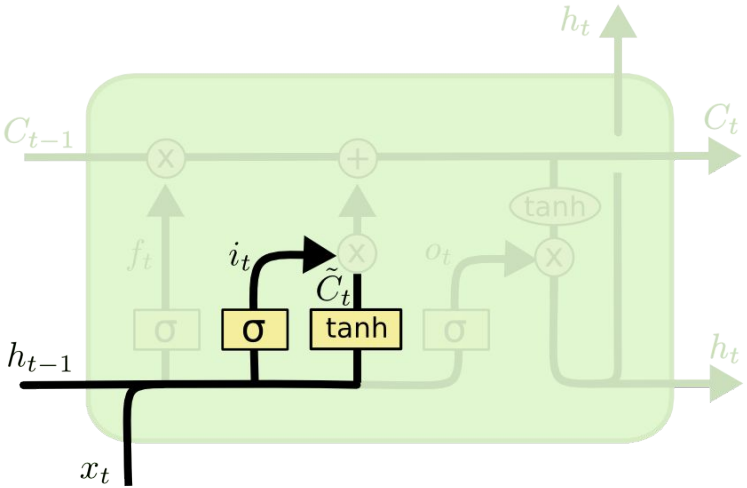



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# Input gate layer

what information is relevant to update in the current cell state of the **LSTM** unit.



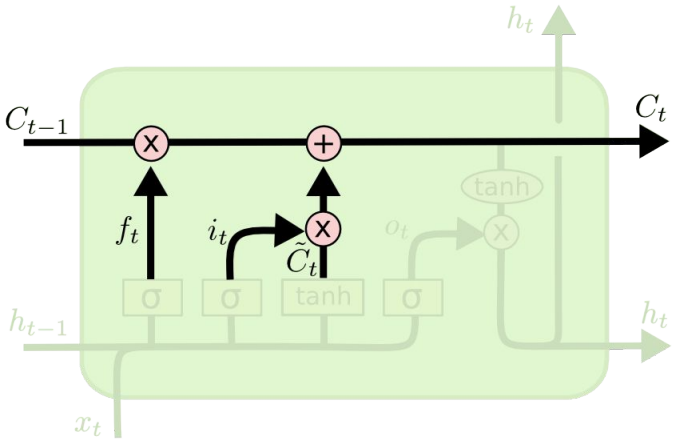

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# The new cell state

We update the old cell state, $C_{t-1}$ into the new cell state $C_t$ (multiplying the old state by $f_t$, forgetting the things we decided to forget earlier). Then we add $i_t * \tilde{C}_t$, which is the new candidate values.

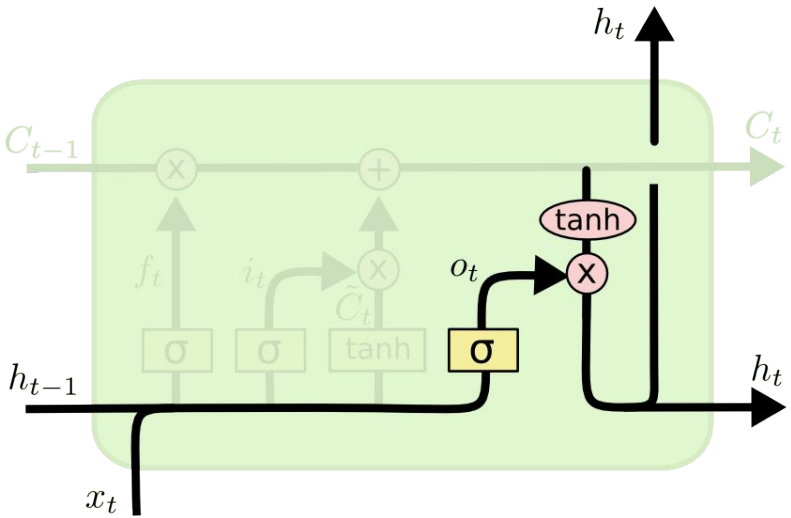



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# Output gate layer

what information will be passed to the next



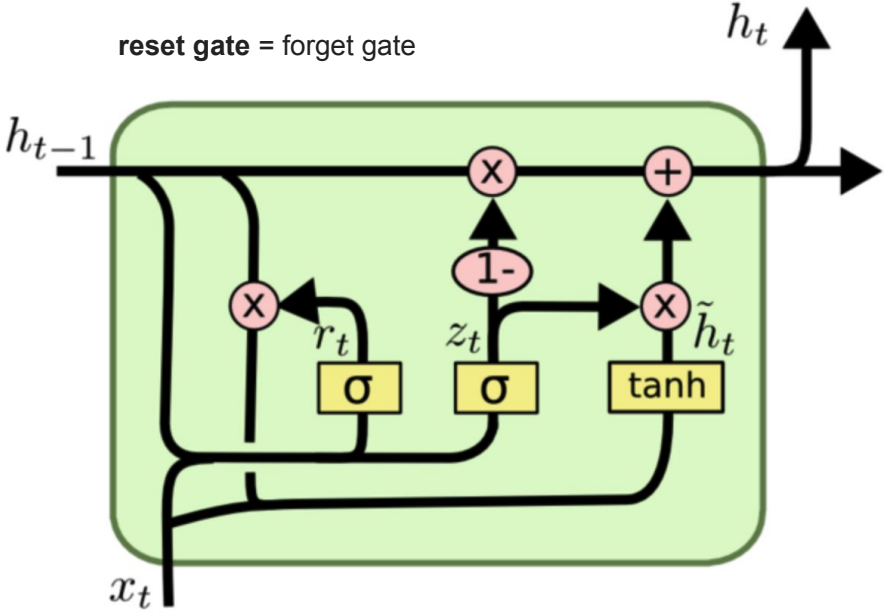

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# GRU unit

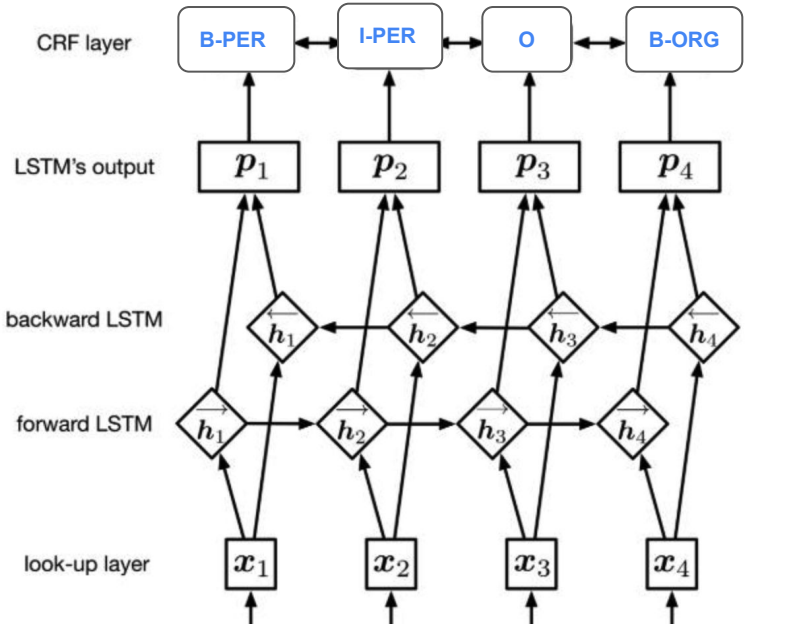
GRUs are slightly faster and easier to run than LSTM




https://primo.ai/index.php?title=Gated_Recurrent_Unit_(GRU)

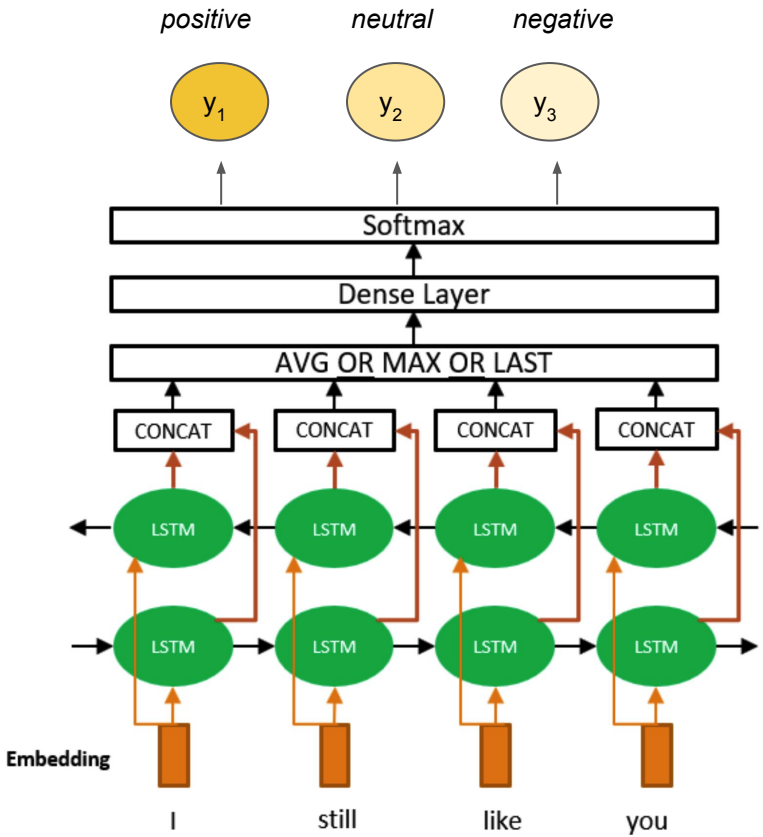
**update gate** determines both how much information to keep from the last state and how much information to let in from the previous layer.

# Recurrent Neural Network (bidirectional)




https://programmerclick.com/article/6956466983/

# Recurrent Neural Network (bidirectional)




Saroufim, C., Almatarky, A., & Hady, M. A. (2018, October). Language independent sentiment analysis with sentiment-specific word embeddings. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis* (pp. 14-23).

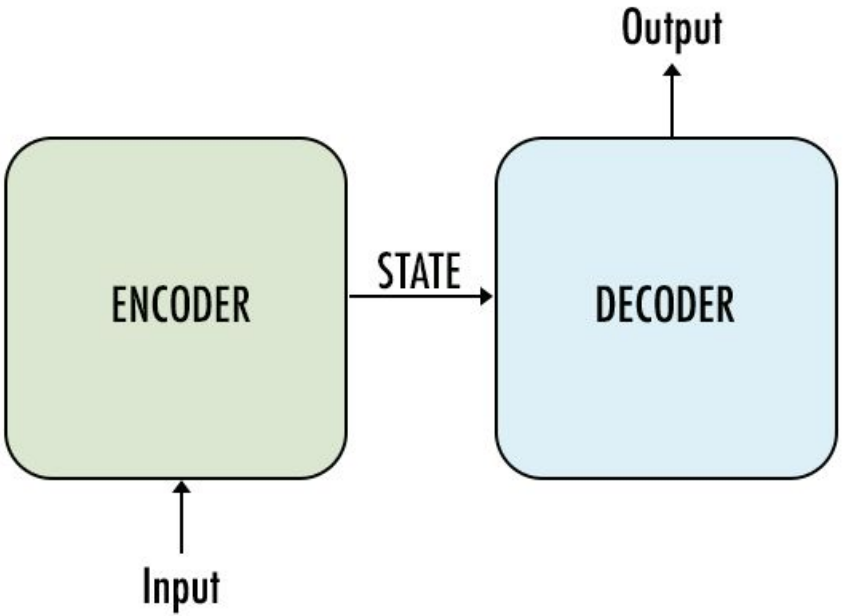
# Keys ideas in RNN

- RNN can process sequences.
- RNN takes its decisions based on the learnt from the past.
- Vanilla RNN (tanh) suffers from vanishing gradient problem.
- LSTM capable to learn long term dependencies
- GRU is simpler and more efficient  than LSTM
- The main drawback of RNNs is that they **cannot be parallelized**. Then, **computational cost** becomes **critical** dealing with long sequences.

# Outline (first season)

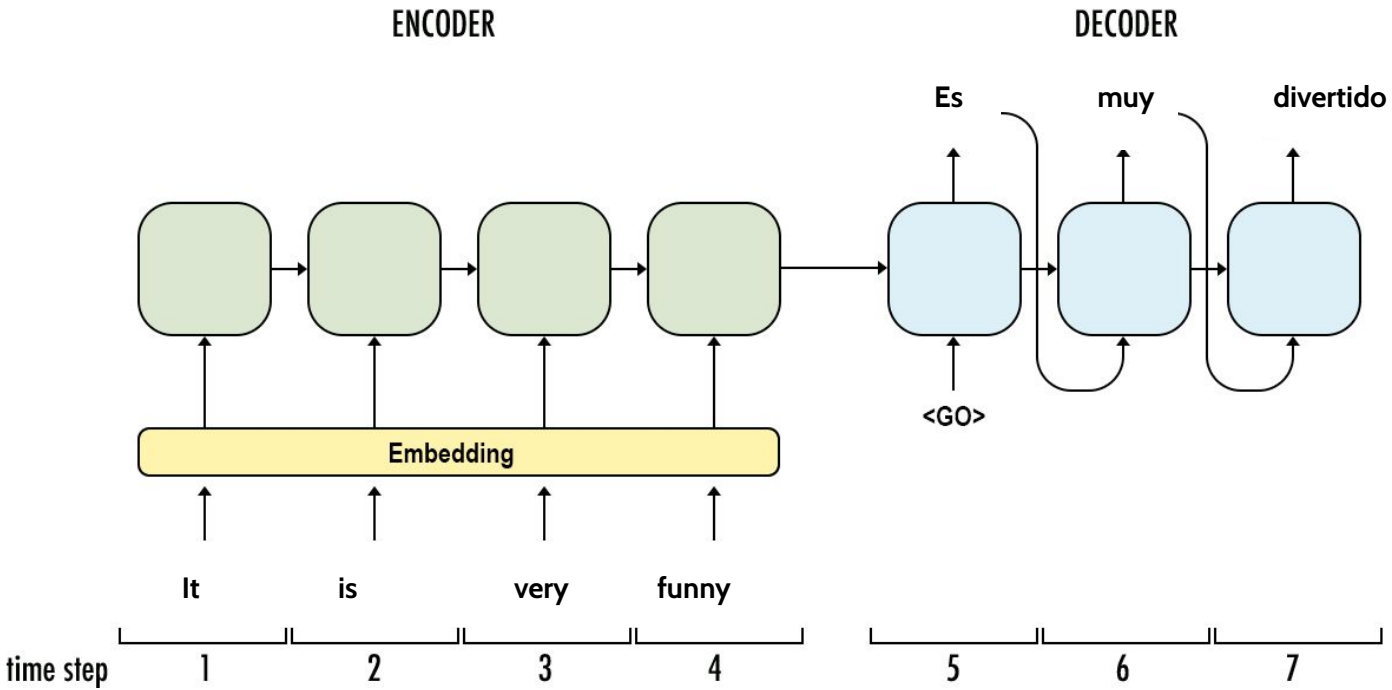
- Introduction
- Word Embeddings
- Deep learning architectures for NLP
  - Recurrent Neural Networks
  - **Sequence to Sequence**
    - Attention mechanism
  - Transformer
- Contextual languages models
  - BERT

# Sequence to Sequence Model (Seq2Seq)



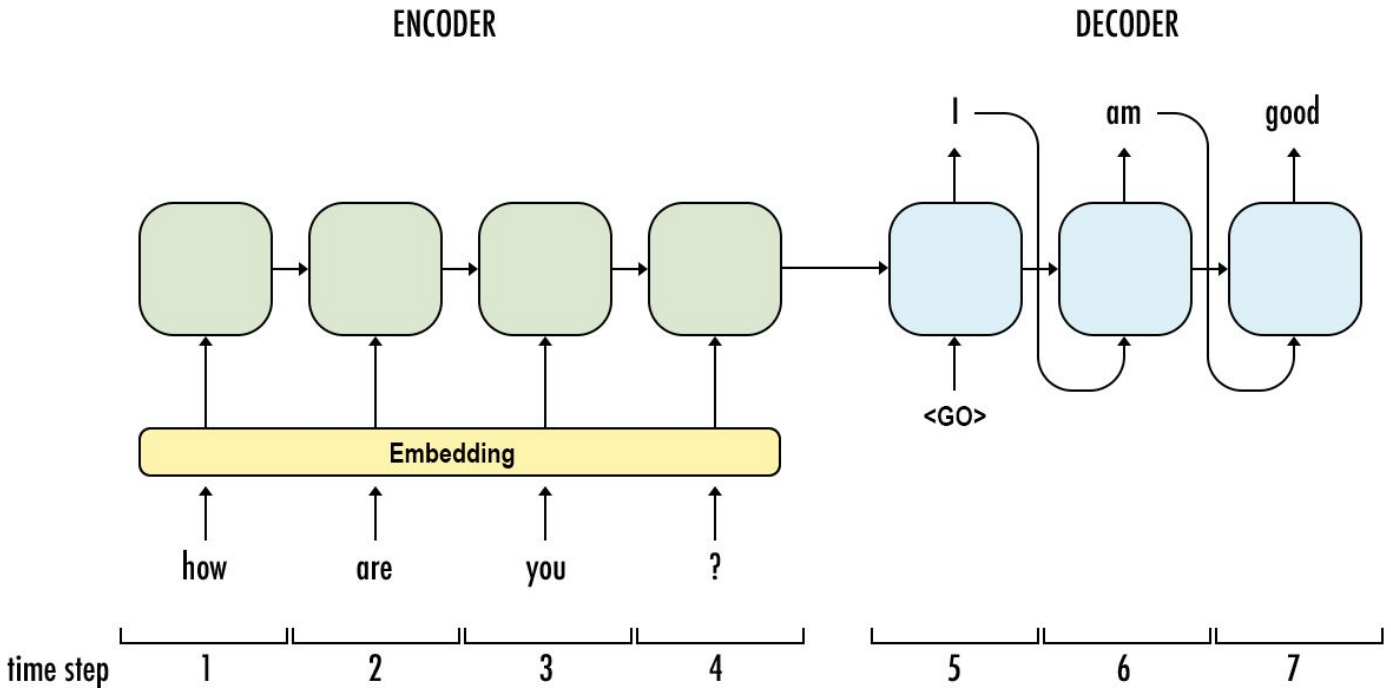

source: https://towardsdatascience.com/sequence-to-sequence-model-introduction-and-concepts-44d9b41cd42d

# Seq2Seq for machine translation



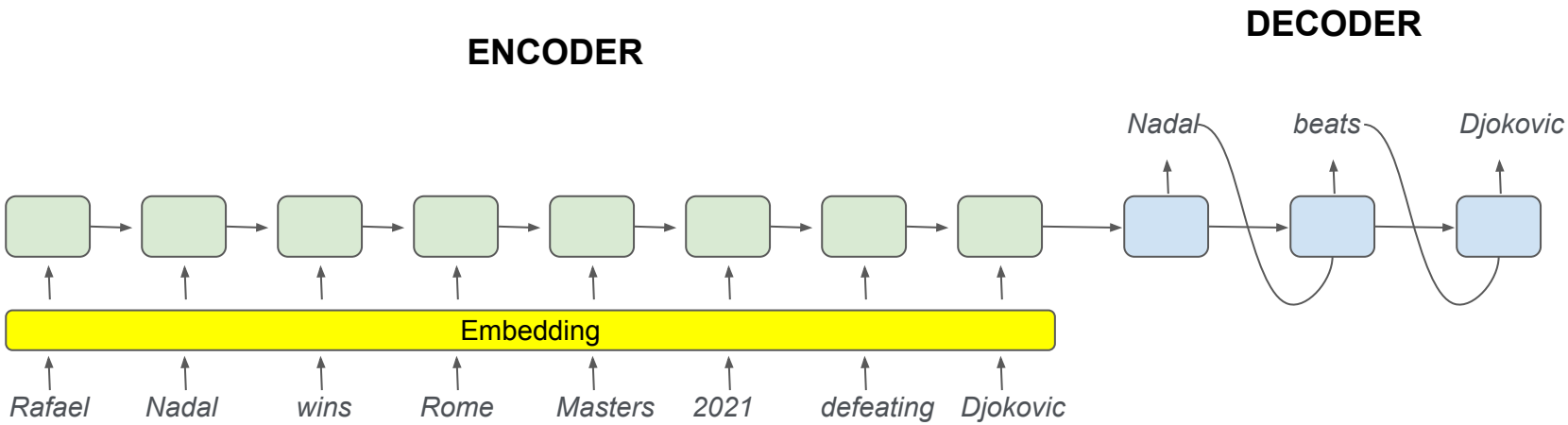

source: https://towardsdatascience.com/sequence-to-sequence-model-introduction-and-concepts-44d9b41cd42d

# Seq2Seq for chatbot

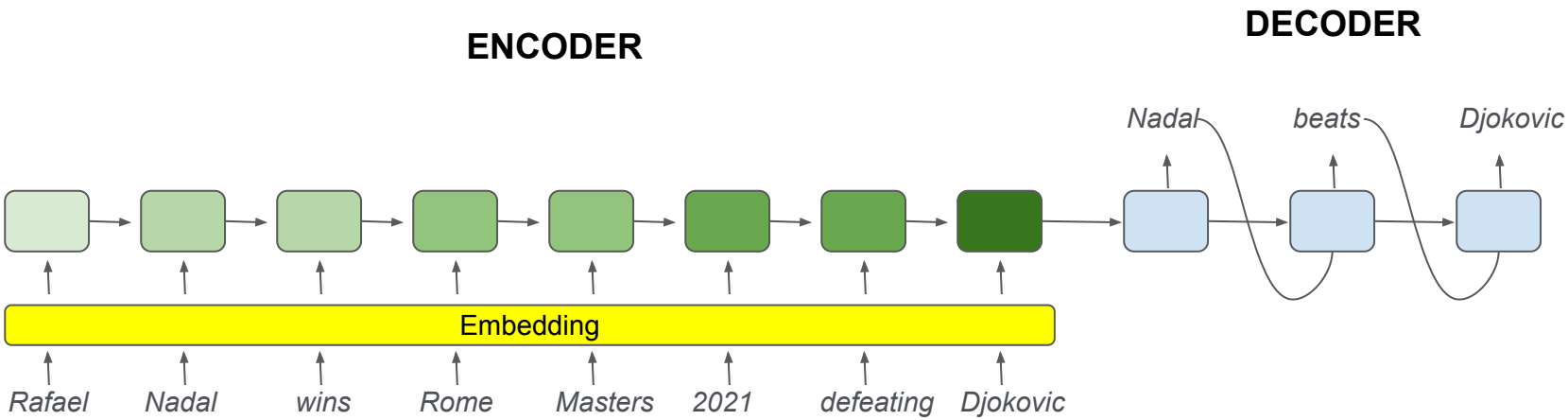



source: https://towardsdatascience.com/sequence-to-sequence-model-introduction-and-concepts-44d9b41cd42d

# Seq2Seq for text summarization



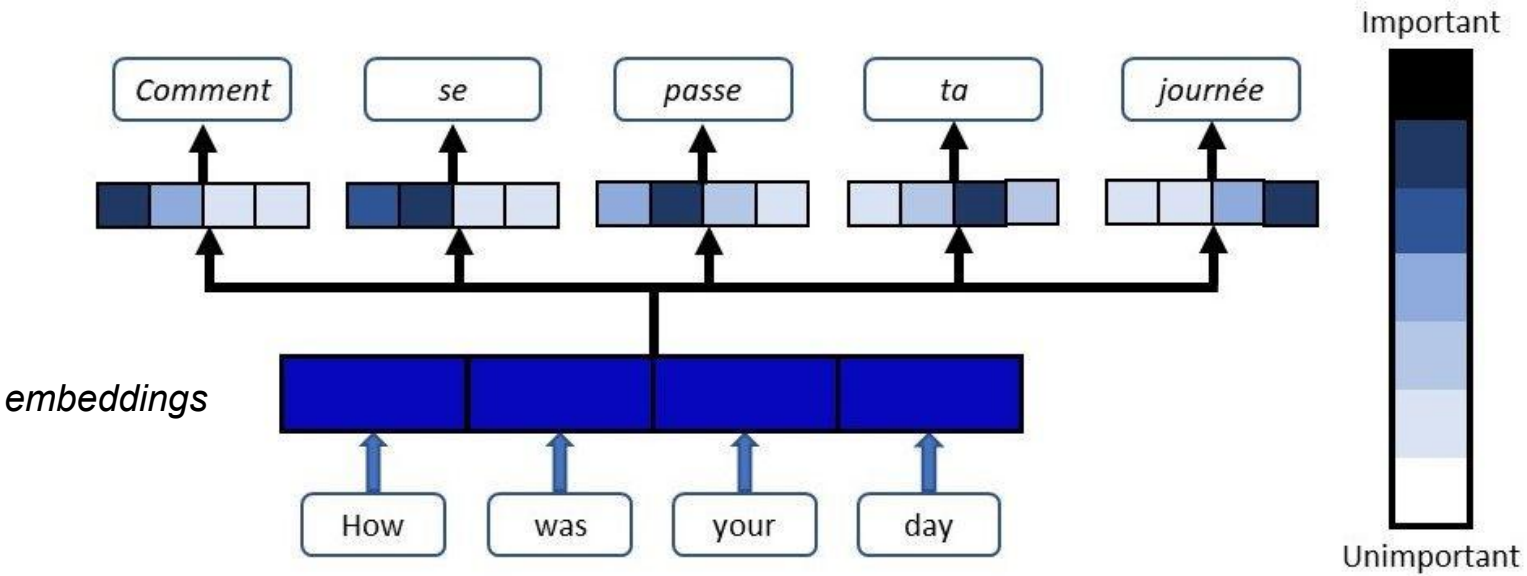
# Seq2Seq




**Drawbacks**:

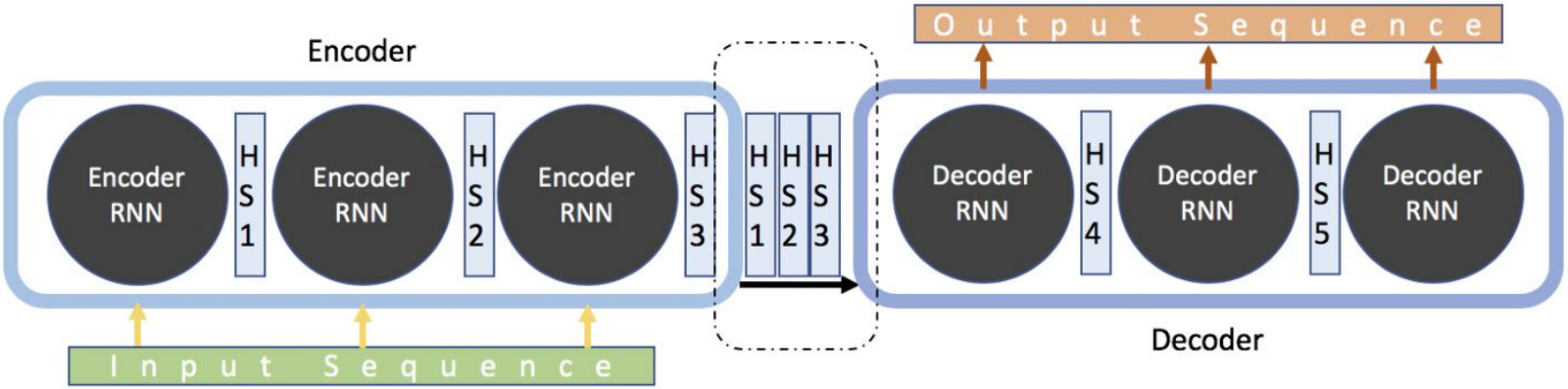
1) The output sequence depends heavily on the context defined by the hidden state in the final output of the encoder, making it challenging for the model to **deal with long sequences**.
2) **RNN steps cannot** be parallelized. Then, **computational cost** becomes **critical** for long sequences.

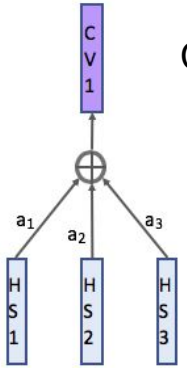
# Basic idea of attention mechanism




https://blog.floydhub.com/attention-mechanism/

# Attention mechanism



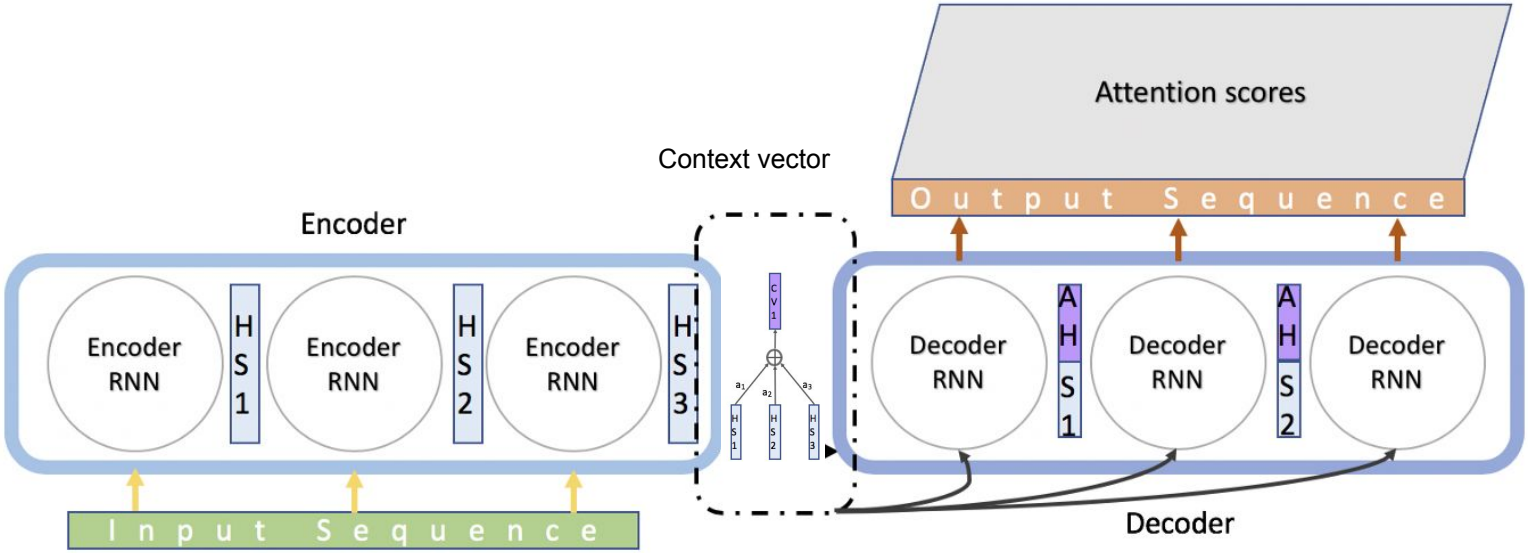




Context vector

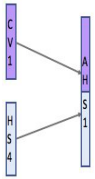
Attention scores: a1, a2, a3

Source: https://towardsdatascience.com/day-1-2-attention-seq2seq-models-65df3f49e263

# Seq2Seq with attention (attention scoring)

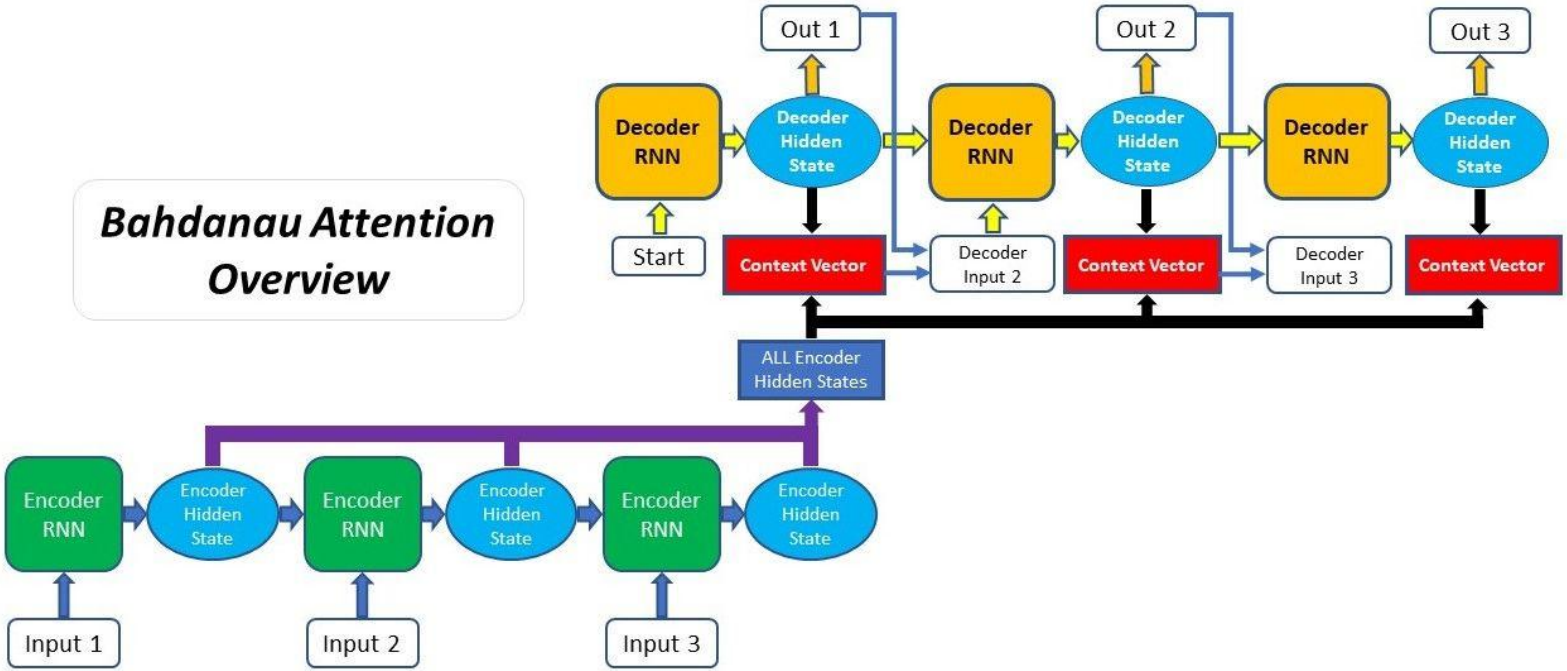



**Computational cost** are still **critical** for long sequences.




attention hidden vector

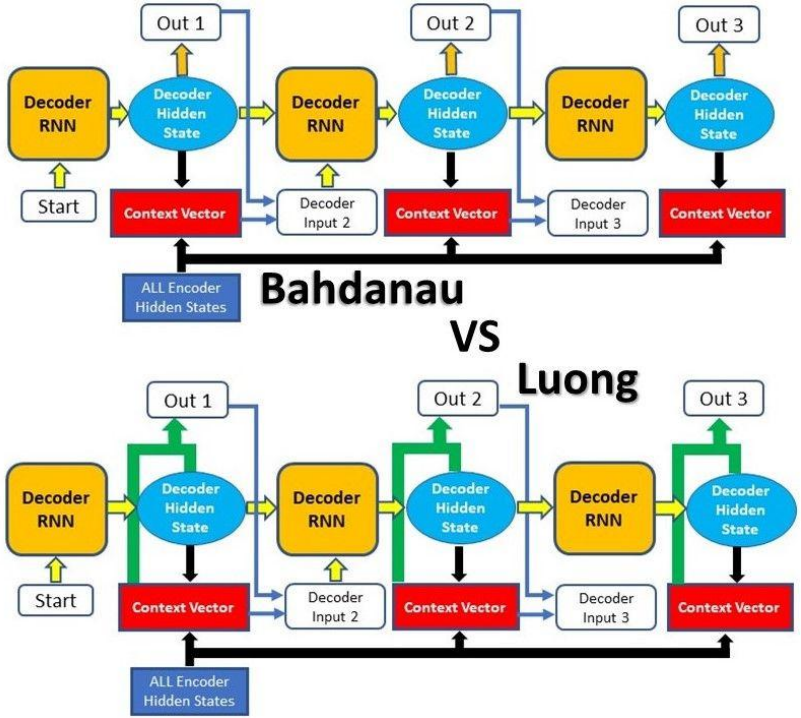
# Bahdanau's attention mechanism




https://arabicprogrammer.com/article/3033619795/

Bahdanau, D., Cho, K., & Bengio, Y. Neural machine translation by jointly learning to align and translate. ICLR 2015

# Bahdanau's attention versus Luong's attention

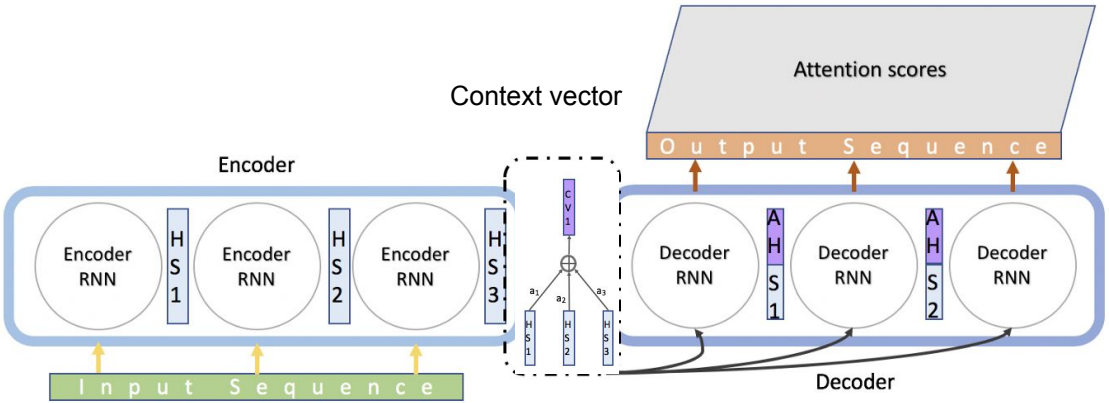



https://arabicprogrammer.com/article/3033619795/

Effective Approaches to Attention-based Neural Machine Translation. Thang Luong, Hieu Pham, Christopher D. Manning. EMNLP 2015.

# Seq2Seq with attention (attention scoring)

**Computational cost** are still **critical** for long sequences!!!.



# Outline (first season)

- Introduction
- Word Embeddings
- Deep learning architectures for NLP
  - Recurrent Neural Networks
  - Sequence to Sequence
  - **Transformer**
    - self-attention and multi-head attention mechanisms
- Contextual languages models
  - BERT

# Transformer

- It is a type of network that applies **attention mechanisms** to gather information about the **relevant context of a given word**, and then **encode that context** in a rich vector that smartly represents the word.

- It is also a Seq2Seq model (encoder-decoder) that only uses attention mechanisms instead of RNNs.
  - **self-attention**
  - **multi-head attention**

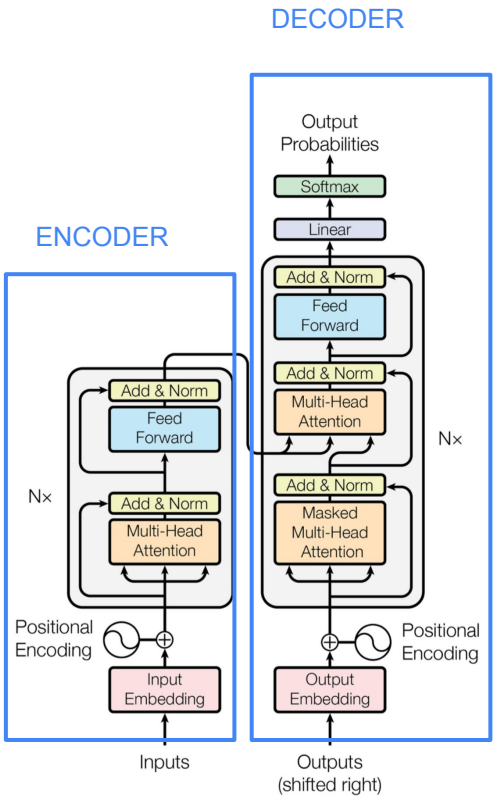
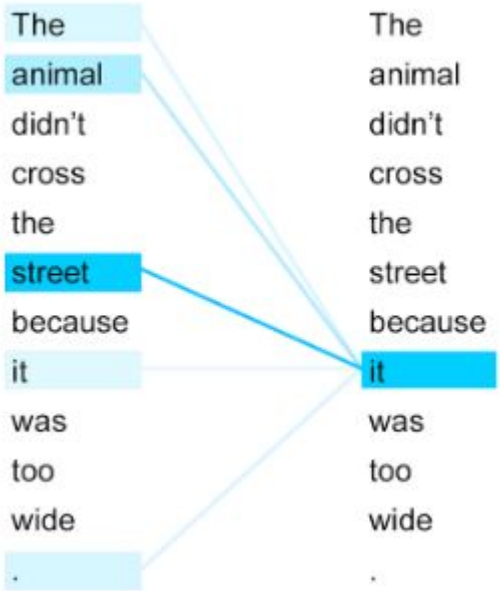



Figure 1: The Transformer - model architecture.



Waswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *NIPS*

# Basic idea of self-attention




**Advantages:**

- Can compute the weights in **parallel**, for all tokens at once.
- As it also sees all the other inputs, it can easily **preserve long-term dependencies** in long text sequences.
- Therefore, **self-attention** could completely replace RNN

source: https://www.kaggle.com/residentmario/transformer-architecture-self-attention

# Encoder Self-attention and Decoder self-attention

**Encoder Self-Attention** computes the interaction between each input word with other input words.

**Decoder Self-Attention** operates on each word of the target sequence. That is, it computes the interaction between each target word with other target words

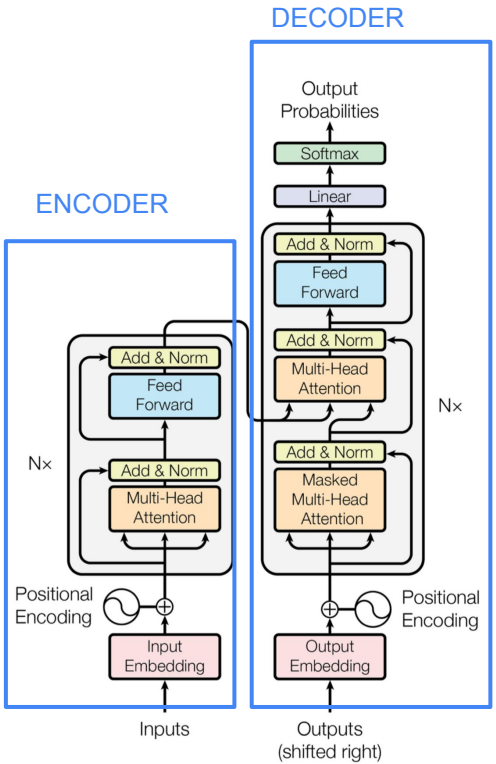



Figure 1: The Transformer - model architecture.



Waswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *NIPS*

# Encoder-Decoder Attention

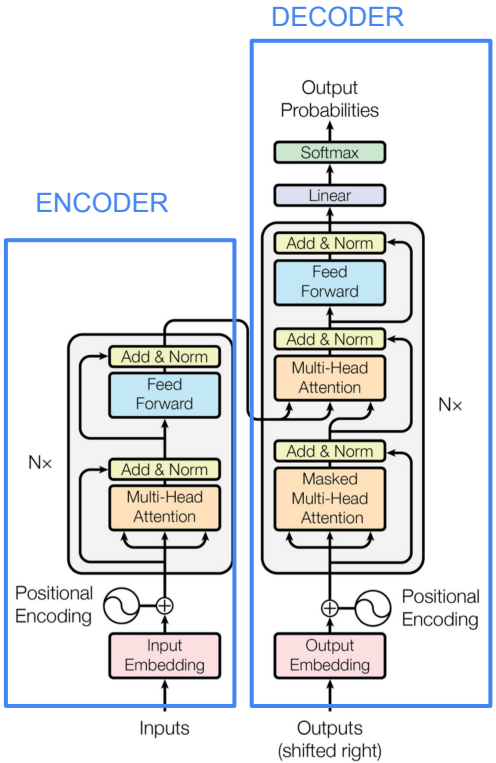
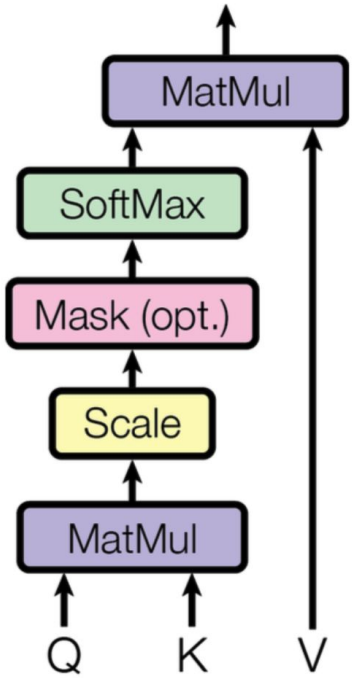



Figure 1: The Transformer - model architecture.

The **Encoder-Decoder Attention** computes the interaction between each target word with each input word. Therefore, it masks out the later words in the target output.

https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853

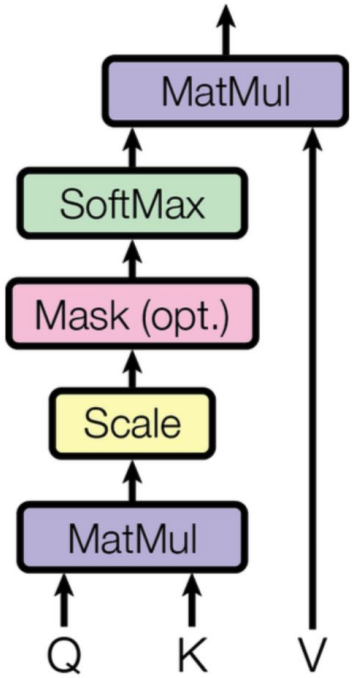
# How to implement self-attention




$$Attention(Q, K, V) = \overbrace{softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)}^{\text{attention weights}}V$$

https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04
https://medium.com/analytics-vidhya/masking-in-transformers-self-attention-mechanism-bad3c9ec235c

# Why masking?




$$Attention(Q,K,V) = \overbrace{softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)}^{\text{attention weights}}V$$

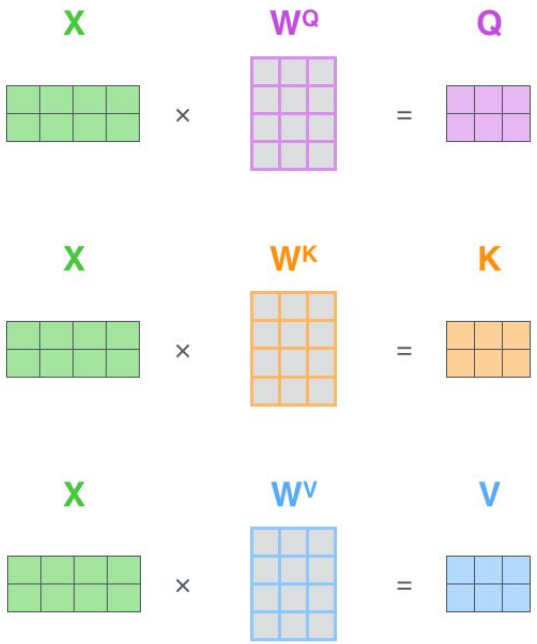
- all vectors must have the same dimensión (sequences could have different number of tokens)
- the **Encoder-Decoder Attention** should only use the previous tokens to the current token to predict its output. Thus, we have to hide the next tokens.

https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04
https://medium.com/analytics-vidhya/masking-in-transformers-self-attention-mechanism-bad3c9ec235c

# How to implement self-attention

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

X: embedding matrix (each row is the word embedding of a word in the input sequence)
dimension= 512 (only 4 in figure)




WQ, WK, WV: weight matrices we've trained

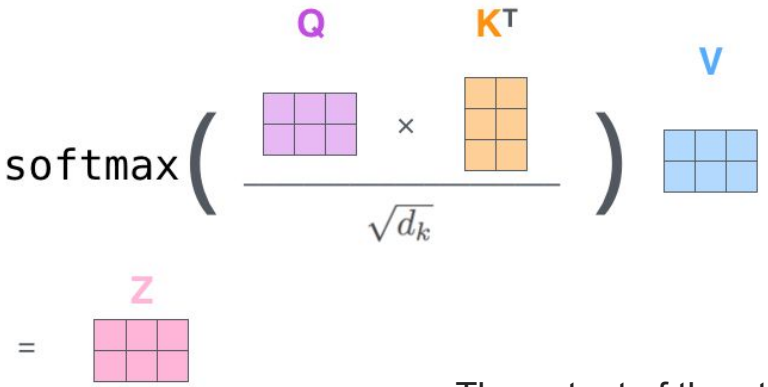
Three vectors for each of the input vector: the query vector, the key vector, and the value vector.

- (Q) represents the current word to be represented
- the key (K) are the input vectors fo all words in the input sequence. It will represent the relevance to the query,
- the value (V) represent the actual contents of the input.

https://jalammar.github.io/illustrated-transformer/

# How to implement self-attention

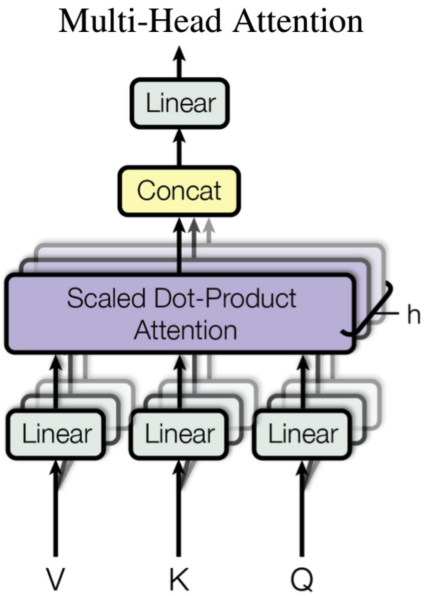
$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$




The output of the attention head is a weighted sum of the values vector

https://jalammar.github.io/illustrated-transformer/

# Multi-head attention




Attention module repeats its computations multiple times in parallel. Each of these is called an Attention Head.

https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04

# Keys ideas of transformers

- Transformers is a Seq2Seq model (encoder-decoder) that only uses attention mechanisms instead of RNNs.
- Self-attention allows to preserve long-term dependencies.
- Multi-head attention allows to jointly attend to information from different representation **subspaces** at different positions and more parallelization.

# Outline (first season)

- Introduction
- Word Embeddings
- Deep learning architectures for NLP
  - Recurrent Neural Networks
  - Sequence to Sequence
  - Transformer
- **Contextual languages models: BERT**

# Why are important model languages?

- Proper language representation is key for developing general-purpose language understanding methods.

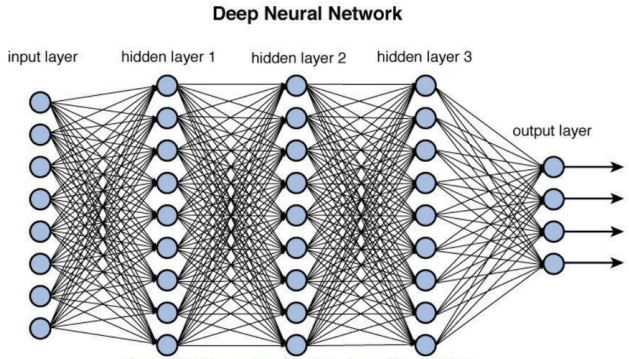


Figure 12.2 Deep network architecture with multiple layers.

http://esl.fis.edu/teachers/support/vocabPoly.htm

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# Types of language models

- Approaches:
  - Context-free language models
  - Contextual language models

http://esl.fis.edu/teachers/support/vocabPoly.htm

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# Contextual-free models (such as Word2Vec or Glove)

- Generate a single word embedding representation for each word.
- Cannot appropriately represent **polysemy words**.
- **Ambiguity** is one of the **biggest** challenges in **NLP**

- Work out the *solution* in your head.
- Heat the *solution* to 75° Celsius.

- The *key* broke in the lock.
- The *key* problem was not one of quality but of quantity.

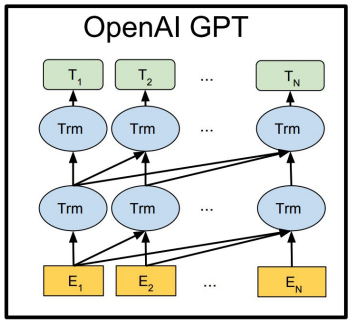
- There are many non-native *pupils* in the class.
- *Pupils*' size changes according to the brightness of light.

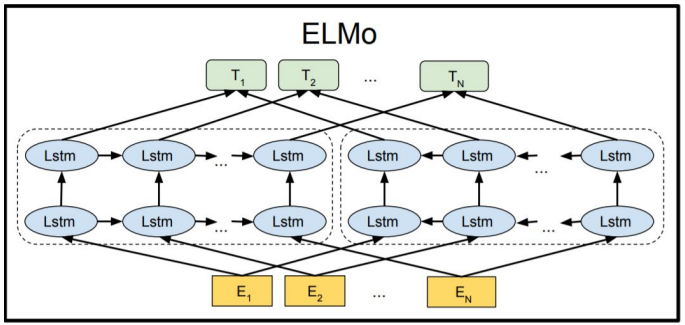
http://esl.fis.edu/teachers/support/vocabPoly.htm

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# Contextual language models

- The **word vector** is based on the **context** of the word
- Previous contextual language models were **unidirectional** (ULMFit, GPT, OpenAI GPT) or shallow bidirectional model (ELMo).

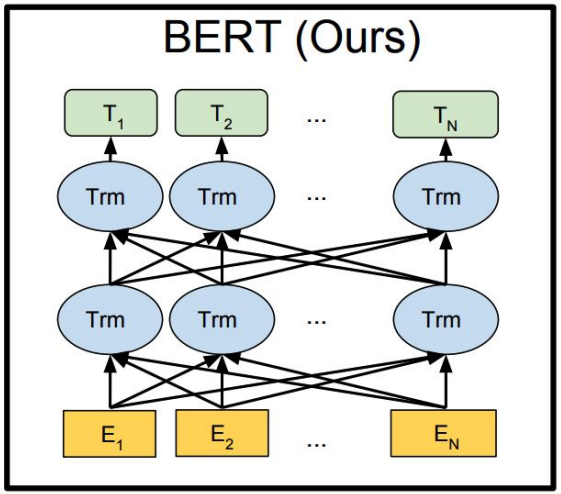



Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# Contextual language models: BERT

- BERT is fully bidirectional
- Based on **attention mechanisms** to gather information about the **relevant context of a given word**
- BERT has become a state-of-the-art in many NLP tasks

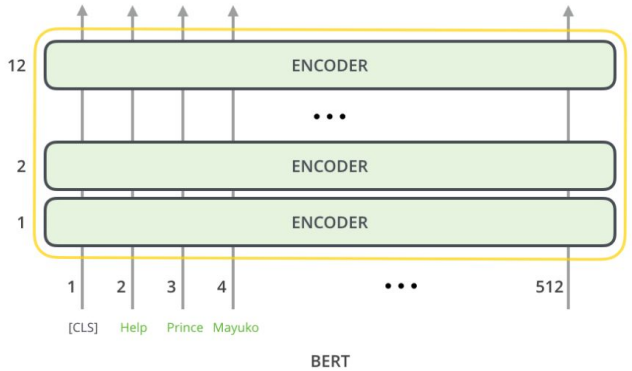



http://esl.fis.edu/teachers/support/vocabPoly.htm

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# BERT architecture

- A stack of encoders
- A vector is generated for each input token.

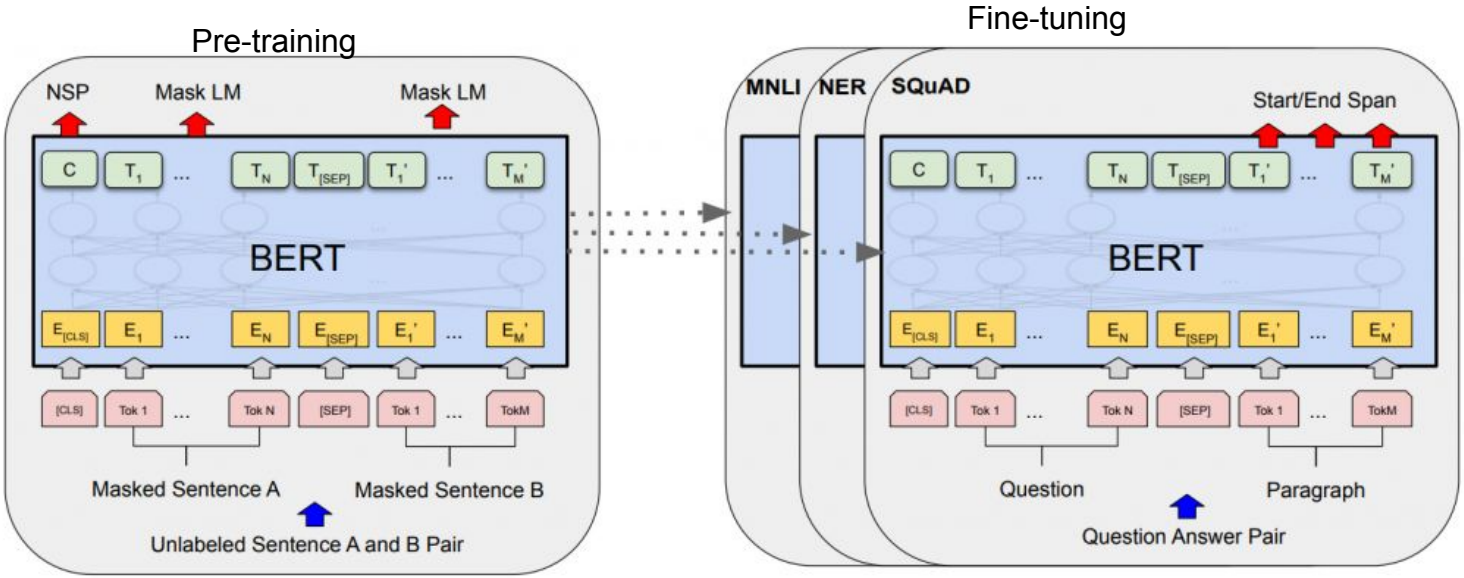



source: http://jalammar.github.io/illustrated-bert/

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# BERT

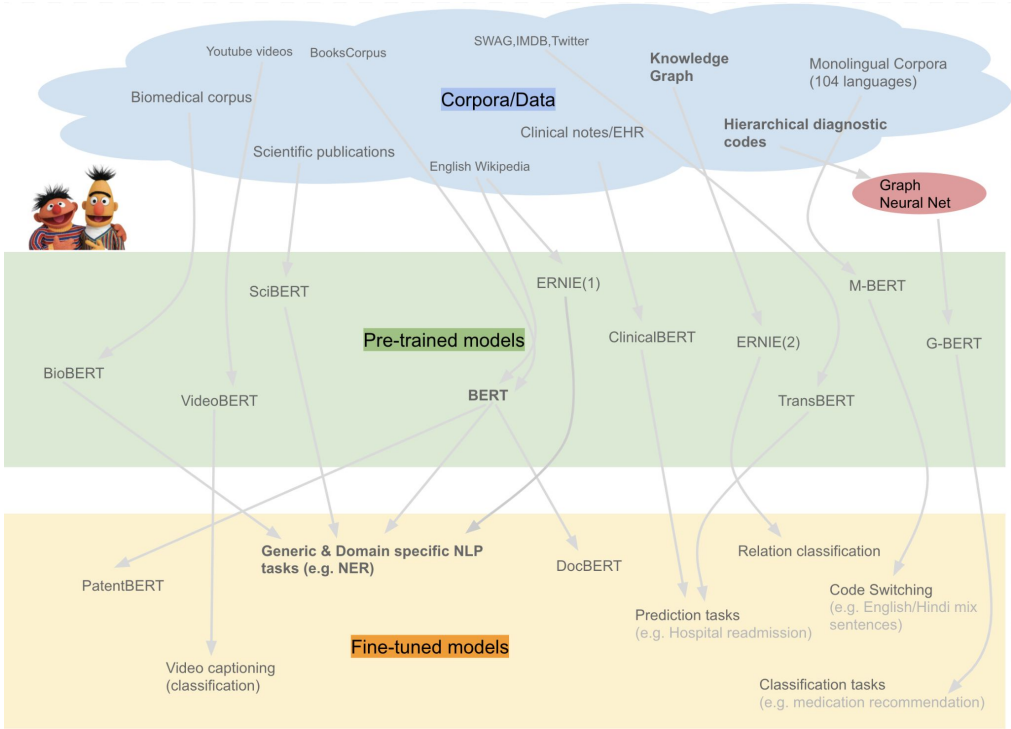
- These vectors can be used in many different NLP tasks: text classification, text summarization, NER, etc.




Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# Training BERT

BERT was trained on Wikipedia (2,500 million words) and BookCorpus (800 million words)




https://towardsdatascience.com/bert-for-dummies-step-by-step-tutorial-fb90890ffe03

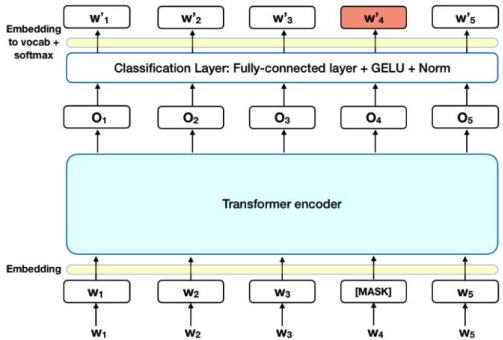
# Pre-Training BERT

- To train BERT, two approaches are used simultaneously:
  - Masked Language Model (MLM)
  - Next Sentence Prediction (NSP)

https://towardsdatascience.com/understanding-bert-bidirectional-encoder-representations-from-transformers-45ee6cd51eef

# Masked Language Model (MLM)

- The model is fed with a sentence such that 15% of the words in the sentence are masked.
- Then, BERT has to predict the masked words correctly given the context of unmasked words.




https://towardsdatascience.com/understanding-bert-bidirectional-encoder-representations-from-transformers-45ee6cd51eef

**Input:** The $[MASK]_1$ is not working. It's unable to $[MASK]_2$

**Labels:** $[MASK]_1$ = computer; $[MASK]_2$ = start.

https://towardsdatascience.com/understanding-bert-is-it-a-game-changer-in-nlp-7cca943cf3ad

# Next Sentence Prediction (NSP)

- The model is fed with 2 sentences.
- The model has to predict the order of the 2 sentences.

**Sentence A** = The computer is not working.

**Sentence B** = It's unable to start.

**Label** = IsNextSentence

**Sentence A** = The computer is not working.

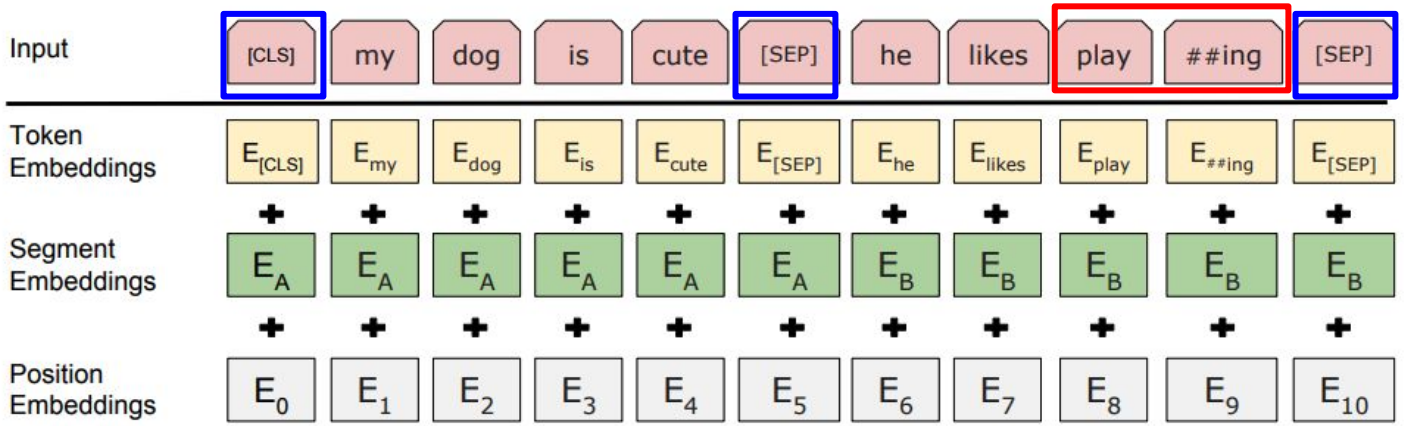
**Sentence B** = Coffee is very tasty.

**Label** = NotNextSentence

https://towardsdatascience.com/understanding-bert-is-it-a-game-changer-in-nlp-7cca943cf3ad

# BERT input

[CLS] marks the beginning of an input sentence, [SEP] marks the separation/end of sentences.

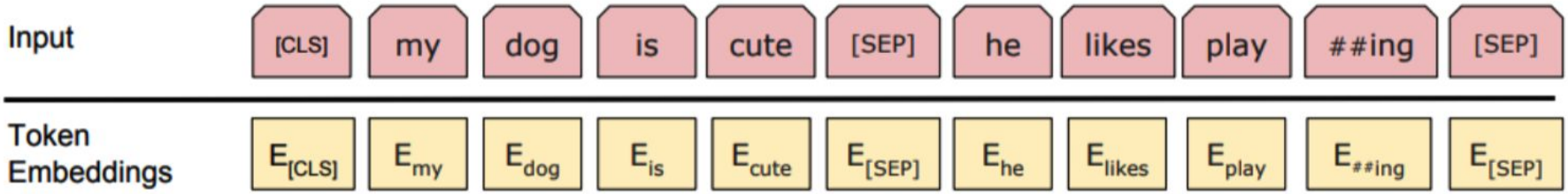



Input: *"My dog is cute. He likes playing"* => **['[CLS]', 'My', 'dog', 'is', 'cute' [SEP]','He', 'likes', 'play', '##ing', '[SEP]']**

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# BERT input - token embeddings

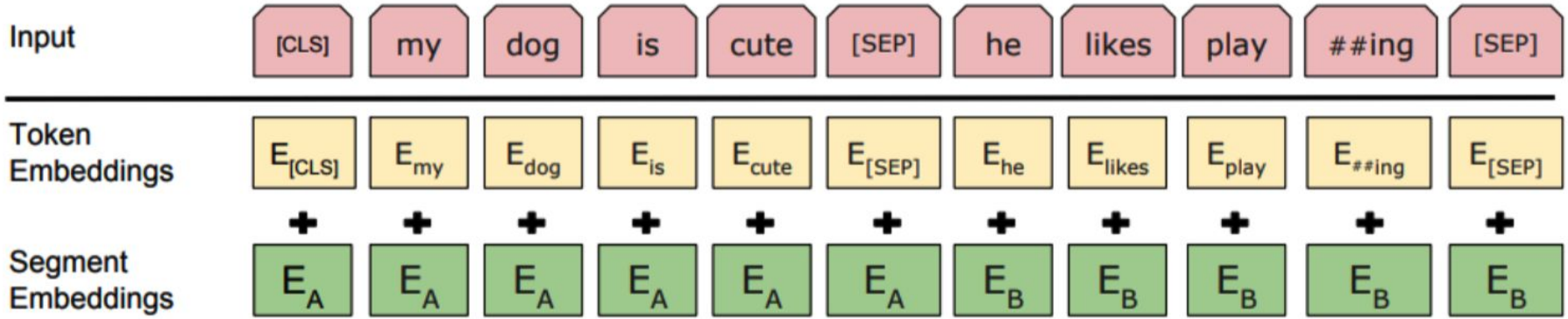
- generates the token embedding (a numerical vector for each word in the input sentence).




Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# BERT input - segment embeddings

- help to distinguish between the different sentences in a single input.
- For the input:
  **[‘[CLS]’, ‘my’, dog, ‘is’, ‘cute’, ‘[SEP]’, ‘he’, ‘likes’, ‘play, ‘##ing’, ‘[SEP]’],**
  the segment embeddings will be:
  **[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1]**




Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

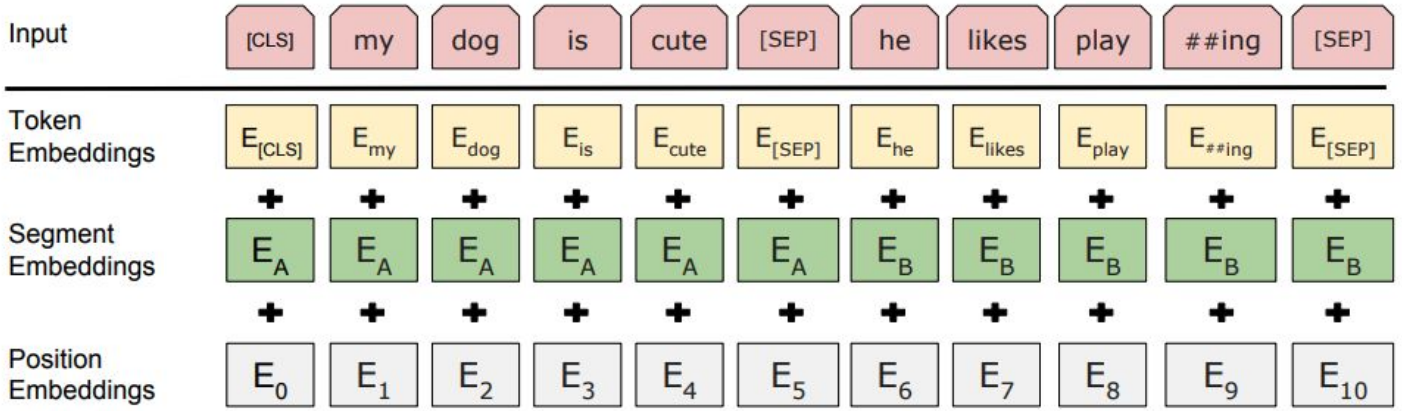
# BERT input - mask embeddings

- The input size for BERT is 512.
- We must add padding of size 512 - len(our input) at the end.
- We also generate a mask token of size 512 in which the index corresponding to the relevant words will have **1**s and the index corresponding to padding will have **0**sil

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# BERT, position embeddings

- Position Embeddings generated internally in BERT and that provide the input data a sense of order.



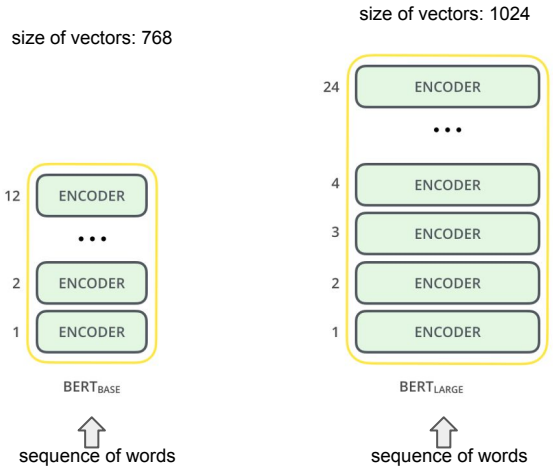

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# Different versions of BERT

- BERT base: 12 encoders, 12 head attention heads. It has an output size of 768 dimensions.
- BERT large: 24 encoders, 16 head attentions. It has an output size of 1024 dimensions.




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

Download from:
https://huggingface.co/
https://github.com/google-research/bert

Spanish version of BERT:
https://github.com/dccuchile/beto

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# Fine-tuning for a specific task

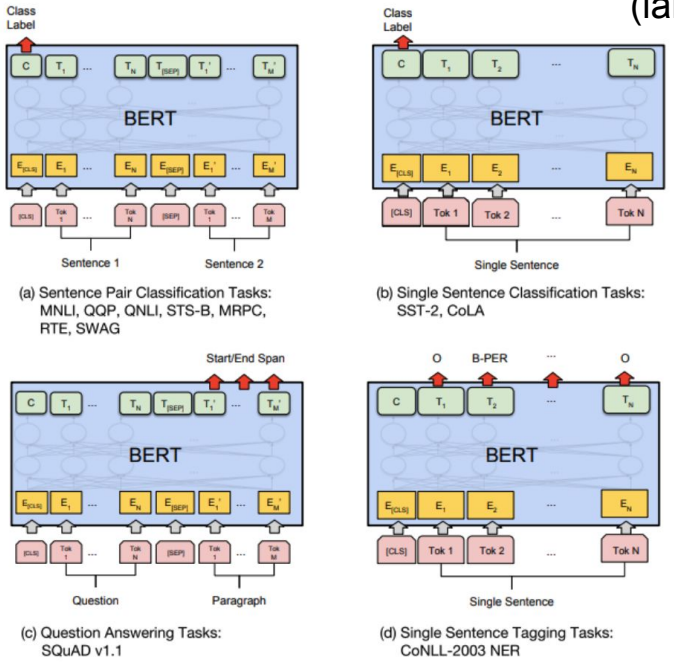
- As we feed input data, the entire pre-trained BERT model and **the additional untrained classification layer** is trained on our specific task.
- For example, if the task if text classification, we can add a linear softmax layer on top of the BERT for predicting the class label of the input text.
- If our task is NER, we can add a CRF layer on the top of BERT.
- Training the model is relatively inexpensive: the bottom layers (BERT) have already great words representation, and we only really need to train the last layer .

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL 2019.

# Fine-tuning for a specific task

output dimension=768 (base), 1024 (large

Text similarity




Text classification: spam detection, fake news detection, hate speech detection, sentiment analysis,...

QA
Text summarization
Machine Translation

Sequence labelling tasks: NER, PoS tagging

https://towardsdatascience.com/understanding-bert-bidirectional-encoder-representations-from-transformers-45ee6cd51eef

# Later models based on BERT

- ROBERTa
- Dilistbert
- XML/mBERT
- ALBERT

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

# ROBERTa

- Developed by Facebook,
- Based on BERT.
- Do not use Next Sentence Prediction (NSP)
- Use dynamic amskin during the training epochs.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

# DistilBERT

- Developed by HuggingFace,
- Learns a distilled (approximate) version of BERT, retaining 95% performance but using only half the number of parameters
- The concept is that once a large neural network has been trained, its full output distributions can be approximated using a smaller network.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.

# XLM/mBERT

- Developed by Facebook,
- XLM a dual-language training mechanism with BERT in order to learn relations between words in different languages.
- The model outperforms other models in a multi-lingual classification task
- Significantly improves machine translation if a pre-trained model is used for the initialization of the translation model.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

# ALBERT

- Developed by Google Research and Toyota Technological Institute,,
- Based on BERT.
- It is much smaller and lighter and smarter than BERT.
- Improves parameter efficiency by sharing all parameters (feed forward and attention parameters), across all layers.
- Replace NSP with **Sentence-Order Prediction (SOP)**
- ALBERT represents a new state of the art for NLP on several benchmarks and a new state of the art for parameter efficiency.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

# Conclusions

- Deep learning can learn unsupervised features effectively and provides state-of-the-art techniques for many NLP.
- Proper language models are key for developing NLP systems.
- Recurrent Neural Network requires high computation cost to process long sequences.
- Transforms based on attention mechanisms can replace RNN
- BERT is a contextual language model capable to correctly represent polysemy words.
- BERT is fully bidirectional and obtains state-of-the-art results in many NLP tasks.
- Fine-tuning a BERT model for a specific task is relatively inexpensive

# Thank you
# Question time!!!

isegura@inf.uc3m.es
https://hulat.inf.uc3m.es/nosotros/miembros/isegura
https://github.com/isegura