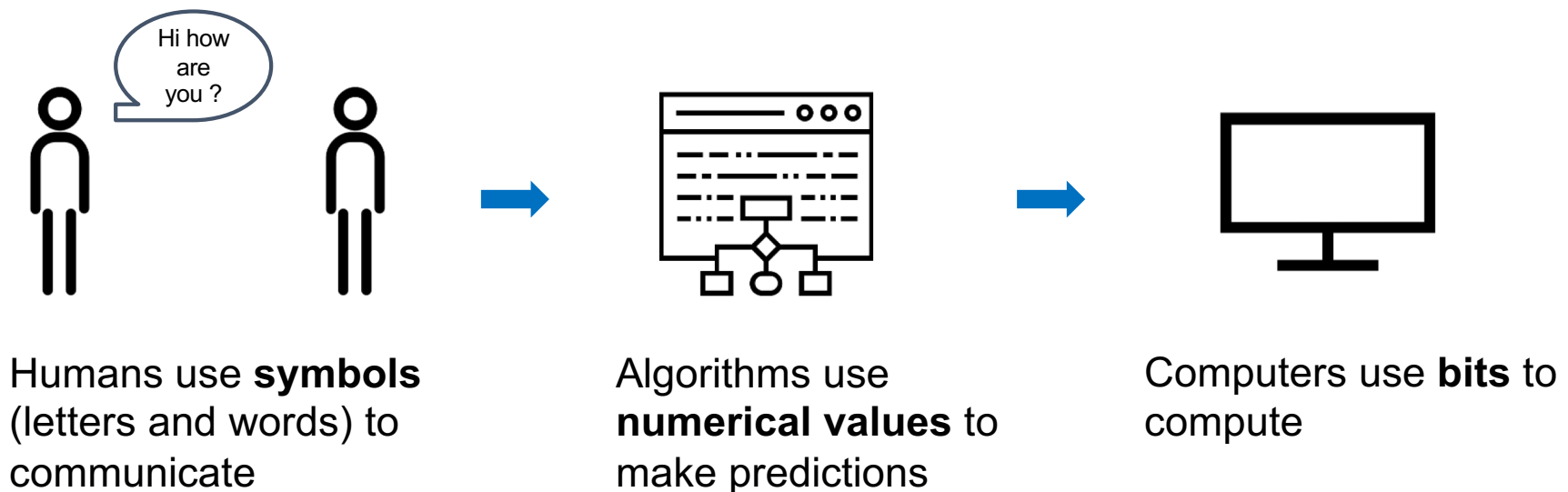


WORD REPRESENTATION FOR NATURAL LANGUAGE PROCESSING

Tristan Karch
BNP Paribas AI Ted Talk
January 28, 2019

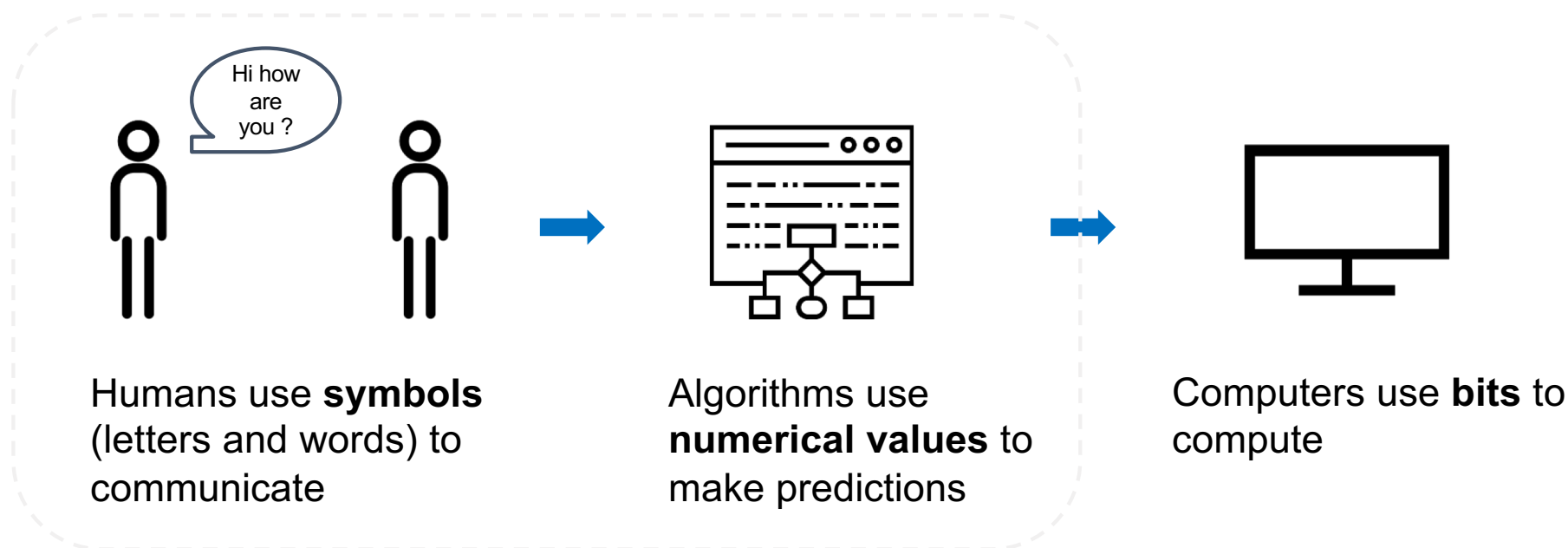
Natural Language Processing Challenge

Natural Language Processing (NLP) is a field of computer science that **programs computer to process textual information**.



Natural Language Processing Challenge

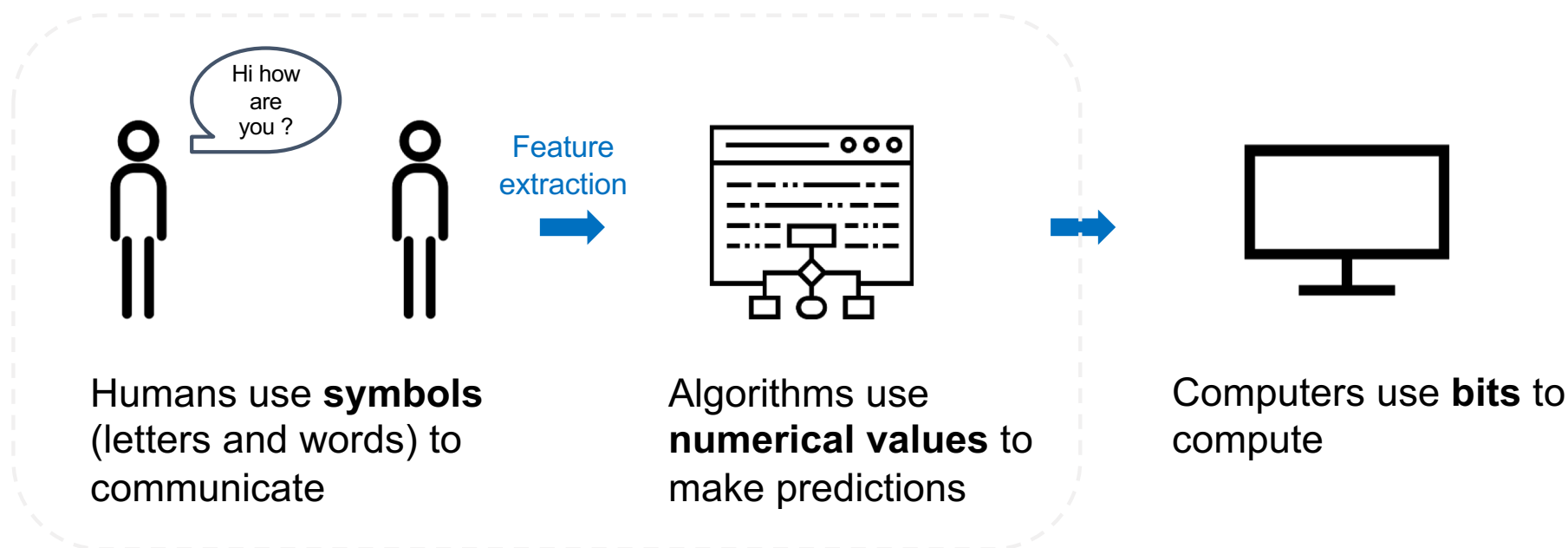
Natural Language Processing (NLP) is a field of computer science that **programs computer to process textual information**.



Goal: How can we convert words to numerical values without losing the internal language information of symbols? How can we encode the meaning/semantic of words?

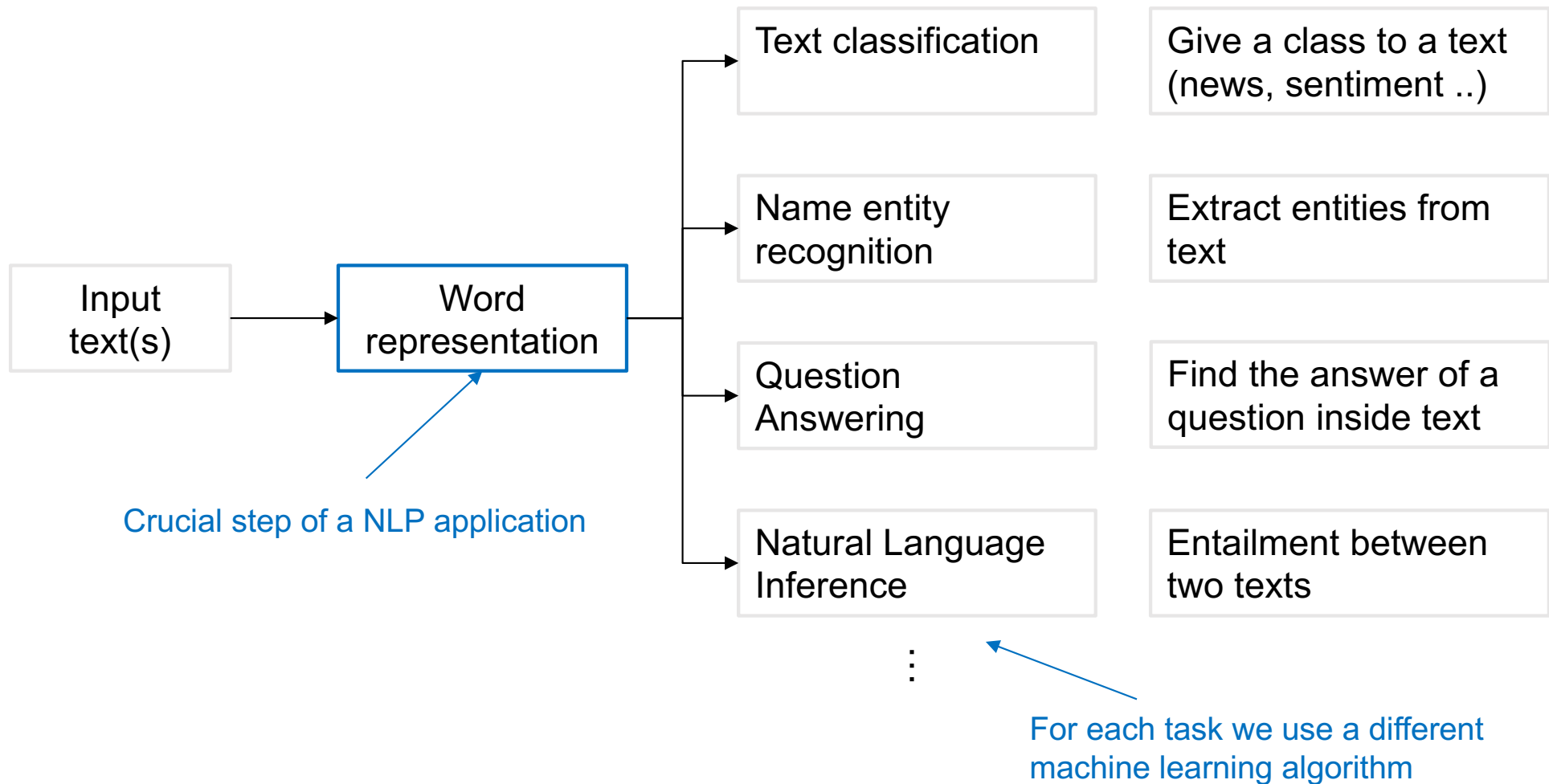
Natural Language Processing Challenge

Natural Language Processing (NLP) is a field of computer science that **programs computer to process textual information**.



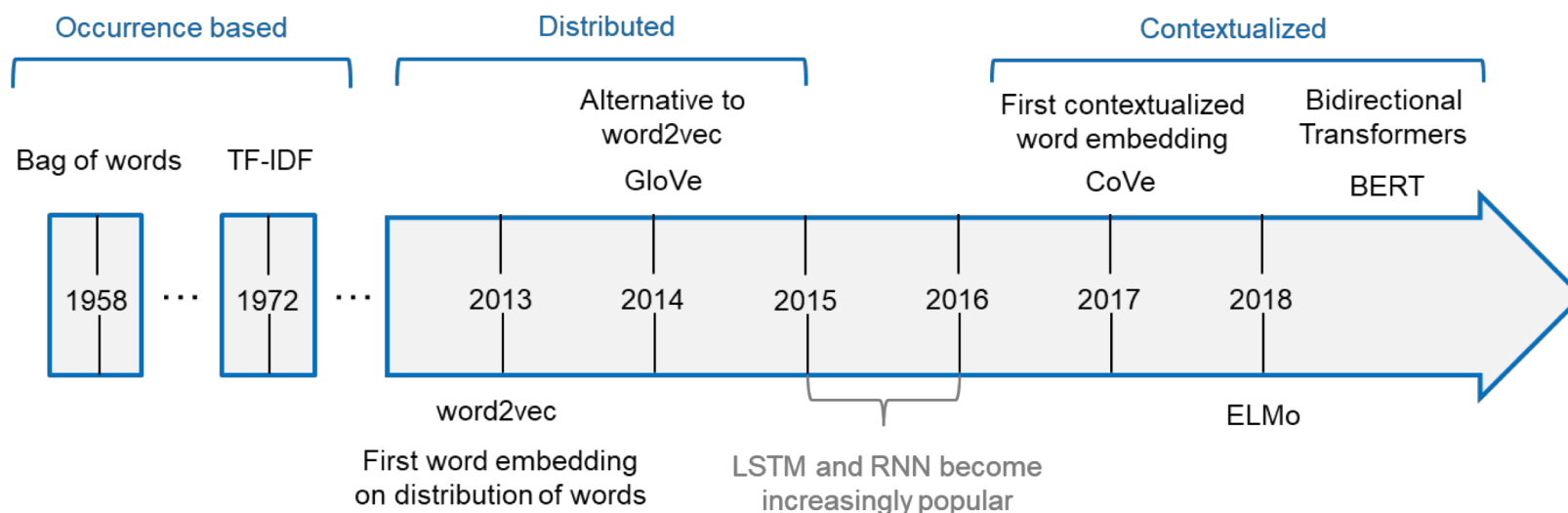
Feature extraction: Building derived values intended to be **informative** and **non-redundant**.

Example of Natural Language Processing Tasks



Different Types of Word Representations

1. **Occurrence based word representation:** words are encoded based on their occurrence count inside a document
2. **Distributed word representation:** word **meanings** are encoded into a vector space thanks to a neural network.
3. **Contextualized word representation:** the meaning of words **inside their context** is encoded into a vector space thanks to a neural network



Some NLP Jargon

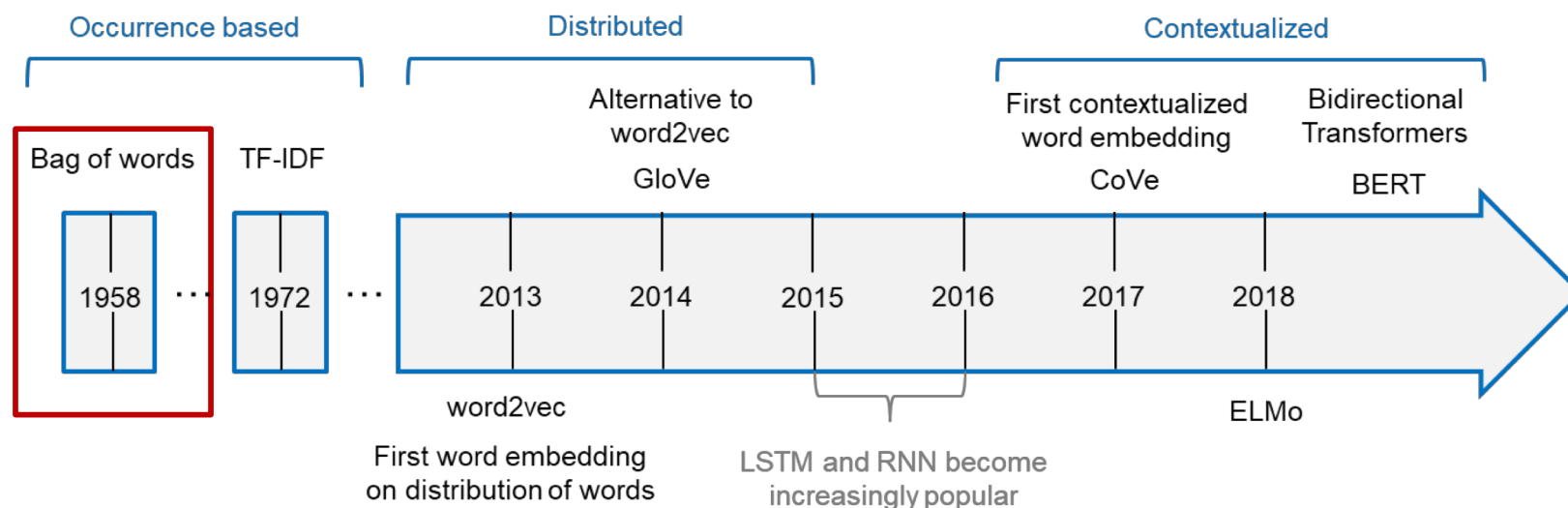
Definitions:

- *Corpus*: a collection of documents (written text) on which the analysis is performed
- *Document*: a part of the corpus
- *Vocabulary*: a set or subset of the words of the **corpus**

Example: A **corpus** could be a **full book**. In this case the **documents** would be its **paragraphs**.

Different Types of Word Representations

- **Occurrence based word representation:** words are encoded based on their count occurrence inside a document
- **Distributed word representation:** word **meanings** are encoded into a vector space thanks to a neural network.
- **Contextualized word representation:** the meaning of words **inside their context** is encoded into a vector space thanks to a neural network

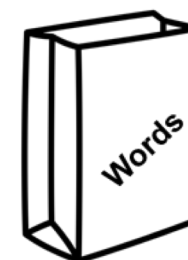


Occurrence Based Word Representation

Bag of Words (BoW):

Each document is represented as a vector that counts the occurrence of words in it
How to construct the bag?

1. Define a vocabulary from **the corpus**
2. Create the vector that represents the document by counting the words of the vocabulary that belongs to the document.



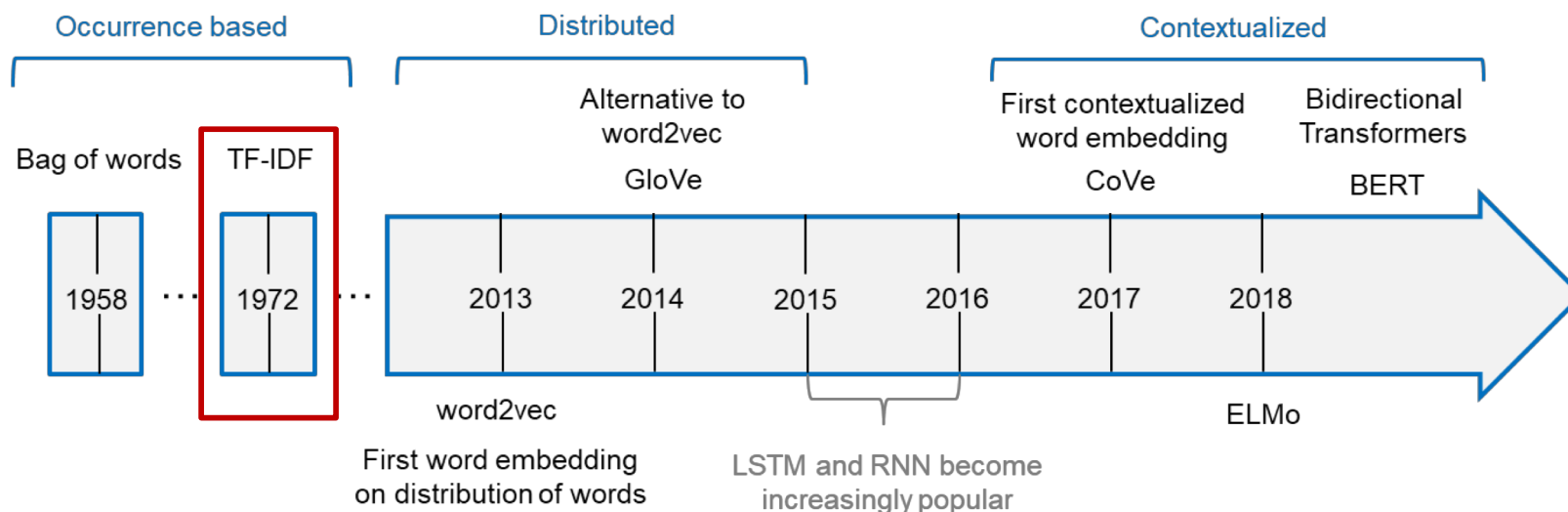
Example: Given the following corpus of two documents:

1. John likes to watch movies. Mary likes movies too.
2. John likes to watch football.

Vocabulary:	<div>“John” “likes” “to” “watch” “movies” “Mary” “too” “football”</div>	$BoW_1 =$	$\begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 2 \\ 1 \\ 0 \\ 0 \end{bmatrix}$	$BoW_2 =$	$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$
-------------	---	-----------	--	-----------	--

Different Types of Word Representations

- **Occurrence based word representation:** words are encoded based on their count occurrence inside a document
- **Distributed word representation:** word **meanings** are encoded into a vector space thanks to a neural network.
- **Contextualized word representation:** the meaning of words **inside their context** is encoded into a vector space thanks to a neural network



Occurrence Based Word Representation

Term Frequency – Inverse Document Frequency (TF - IDF):

BoW computes the **frequency** of words to define their importance

Issue: It is **not necessarily the most frequent words that are the most important**.
We have to take into account **rare** words across the corpus

→ Instead of using the frequency of BoW, TF-IDF uses **another score** which solves this problem:

$$tfidf(w, d) = tf(w, d) \times idf(w, C) \quad w \text{ is a word, } d \text{ a document and } C \text{ the corpus}$$

$$\left\{ \begin{array}{ll} tf(w, d) & \text{Term frequency of word } w \text{ inside document } d \text{ (similar to Bow)} \\ idf(w, C) & \text{Measure of **how rare** word } w \text{ is **across all documents**} \end{array} \right.$$

Occurrence Based Word Representation

Term Frequency – Inverse Document Frequency (TF - IDF):

BoW computes the **frequency** of words to define their importance

Issue: It is **not necessarily the most frequent words that are the most important**. We have to take into account **rare** words across the corpus

→ Instead of using the frequency of BoW, TF-IDF uses **another score** which solves this problem:

$$tfidf(w, d) = tf(w, d) \times idf(w, C) \quad w \text{ is a word, } d \text{ a document and } C \text{ the corpus}$$

$$\begin{cases} tf(w, d) = \frac{n_w}{|d|} \\ idf(w, C) = \log\left(\frac{|D|}{1 + |d \in D : w \in d|}\right) + 1 \end{cases}$$

The score is divided by the number of documents which contain word w

The log function lowers the scaling for really frequent words

Occurrence Based Word Representation

Term Frequency – Inverse Document Frequency (TF - IDF):

BoW computes the **frequency** of words to define their importance


Issue: It is **not necessarily the most frequent words that are the most important**.
We have to take into account **rare** words across the corpus

Example:

1. John likes to watch movies. Mary likes movies too.
2. John likes to watch football.

Vocabulary:

"John"	1
"likes"	1
"to"	1
"watch"	1
"movies"	0
"Mary"	0
"too"	0
"football"	1

$$BoW_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$
$$tfidf_2 = \begin{bmatrix} 1/5 \times (\log(2/(1+2))+1) \\ 1/5 \times (\log(2/(1+2))+1) \\ 1/5 \times (\log(2/(1+2))+1) \\ 1/5 \times (\log(2/(1+2))+1) \\ 0 \\ 0 \\ 0 \\ 1/5 \times (\log(2/(2))+1) \end{bmatrix} = \begin{bmatrix} 0.12 \\ 0.12 \\ 0.12 \\ 0.12 \\ 0 \\ 0 \\ 0 \\ 0.2 \end{bmatrix}$$


The TF – IDF scores focus on the word "football" because it is rare in the corpus

Occurrence Based Word Representation

Tricks to improve the models:

- **N-grams:** Build a vocabulary that is a combination of successive words

Example: It might be more useful for the model to consider “New York” as one term instead of two

- **Stemming/Lemmatization:** Keep only the root form of the inflected words

Example: Inside documents it might be useful to aggregate the score of “playing”, “play”, “played” to a single term “play”

- **Stop words removal:** Remove commonly used words from vocabulary when building the BoW or TFIDF vector

Example: Words such as “to”, “and”, “a” or “what” etc. do not give much information about documents

Occurrence Based Word Representation

Pros and cons:

Pros	Cons
<ul style="list-style-type: none">• Really easy to compute• Fast to compute as calculation are straightforward	<p>Does not capture:</p> <ul style="list-style-type: none">• Position in text• Semantic <p>The representation is sparse For big corpus the size of the vocabulary can be an issue</p>

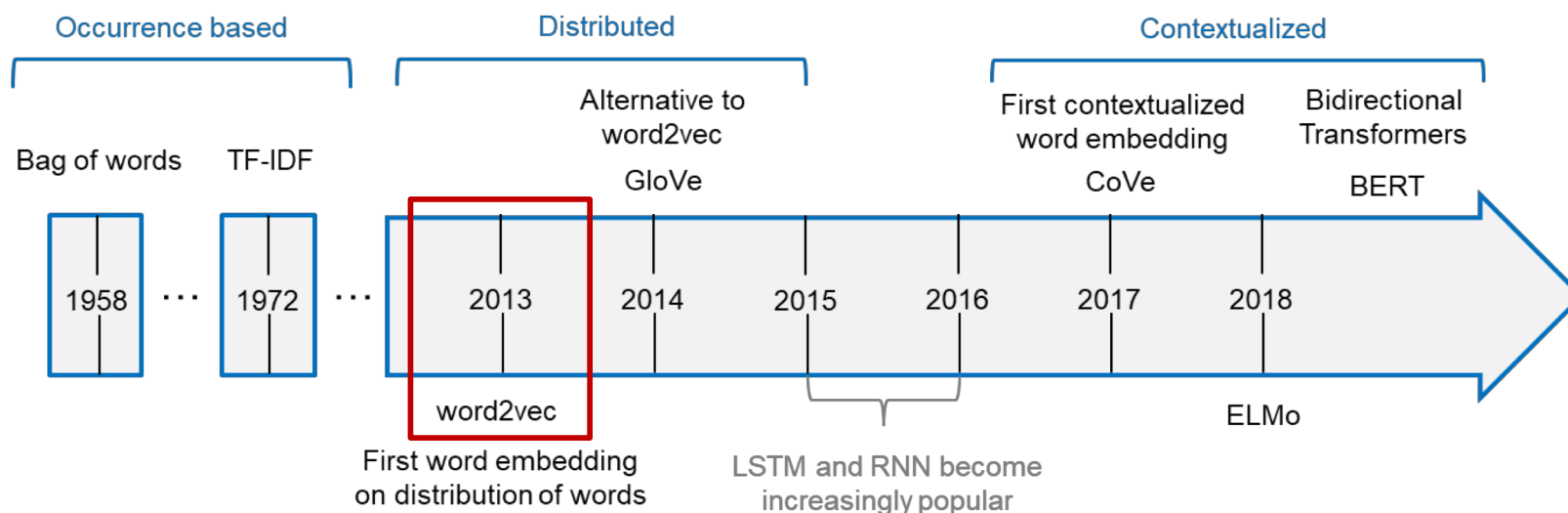
Occurrence based word representations are

- Really good for text classification
- particularly bad at text comparison because they work as keyword matching and don't consider the semantic of words

To solve more complex NLP tasks we need models that work at a semantic level in order to catch synonyms.

Different Types of Word Representations

- **Occurrence based word representation:** words are encoded based on their count occurrence inside a document
- **Distributed word representation:** word **meanings** are encoded into a vector space thanks to a neural network.
- **Contextualized word representation:** the meaning of words **inside their context** is encoded into a vector space thanks to a neural network



Distributed Word Representation

The intuition behind word embedding:

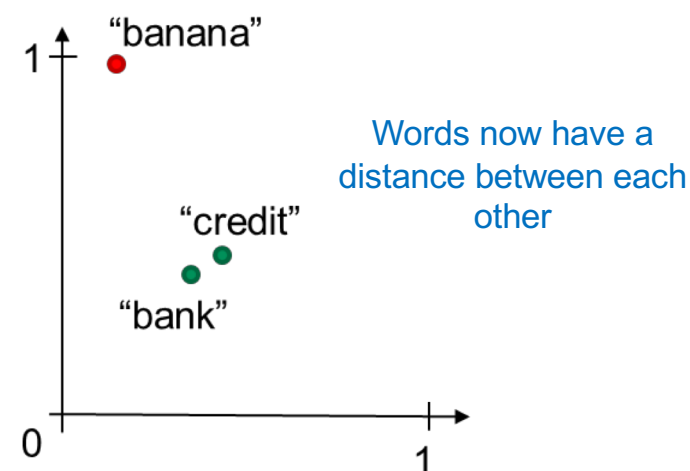
A word embedding encodes **the meaning** of words into a vector of numerical values

The model learns from analyzing the **context** in which words appear.

“two words that appear in the same context have similar meanings / are close to each other”

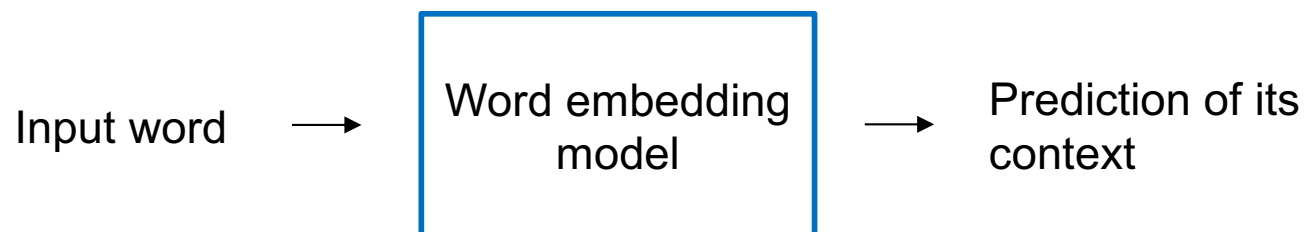
Mathematically: We want map words to vectors such that words that share **common contexts in the corpus** are **close neighbors in the vector space**

“bank”		(0.4, 0.4)
“credit”	Word embedding in 2D	(0.46, 0.48)
“banana”		(0.15, 0.98)



Distributed word representation

Training Data: What does the model takes as input to learn the vector representation?



→ We need **training samples (input word, context)** in order to teach the model how to predict the context in which the input word appears

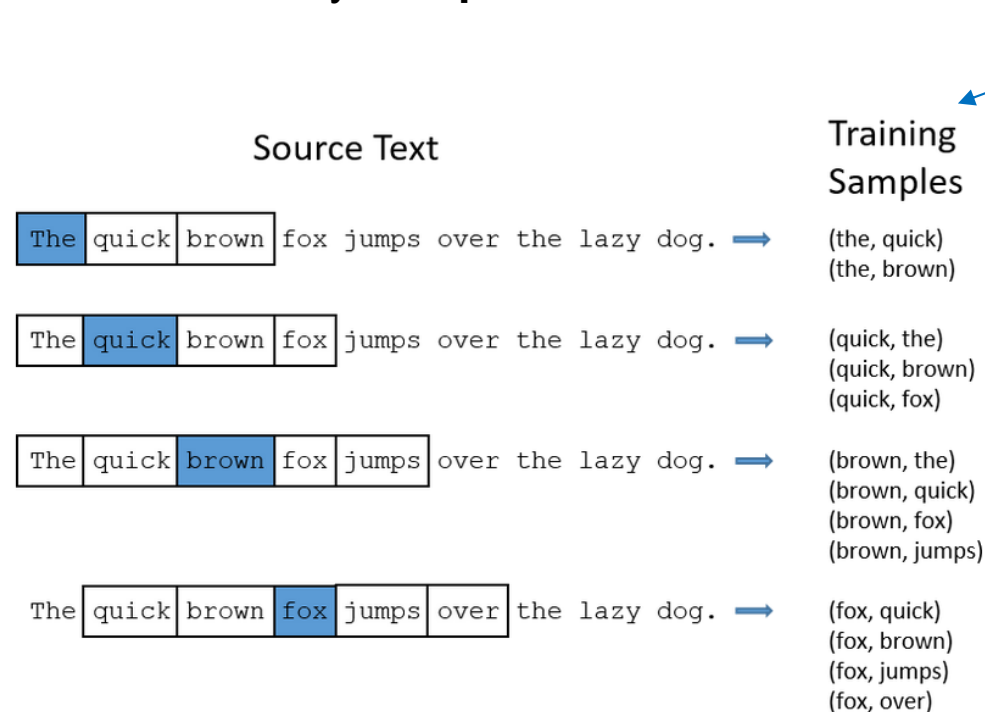
Where can we find numerous different contexts for words?



Distributed word representation

Training Data: What does the model takes as input to learn the vector representation?

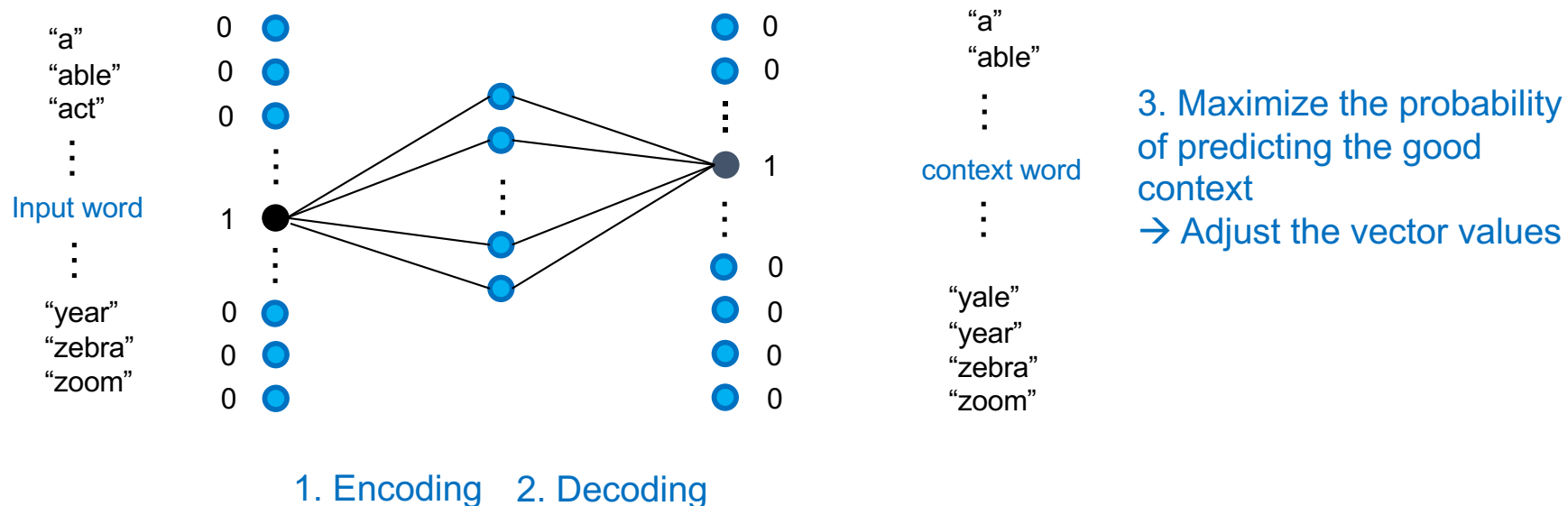
We use a **sliding window** over many **Wikipedia articles** to construct pairs (word, context).



Distributed word representation

Model Architecture: How do we transform the meaning of words into vectors of real values?

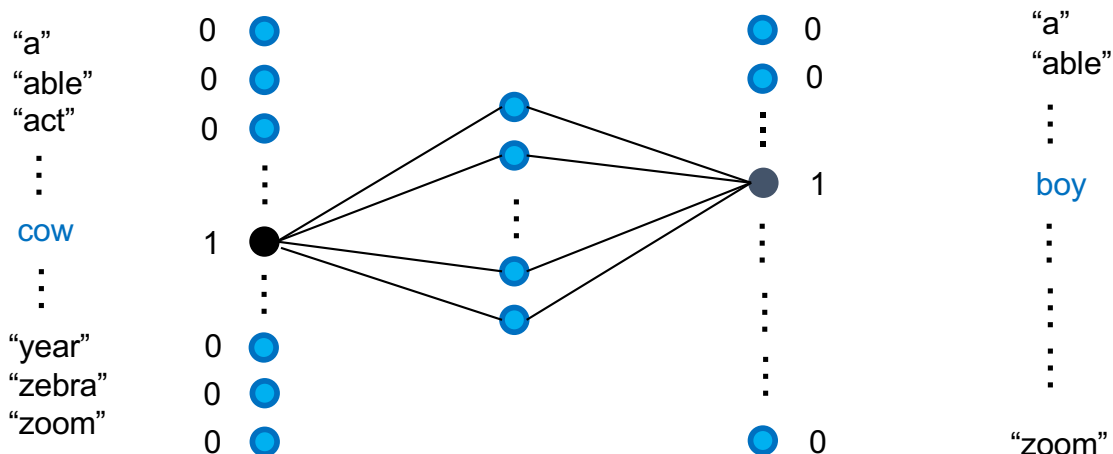
→ We use a Neural Network as an auto-encoder



Distributed word representation

Model Architecture: How do we transform the meaning of words into vectors of real values?

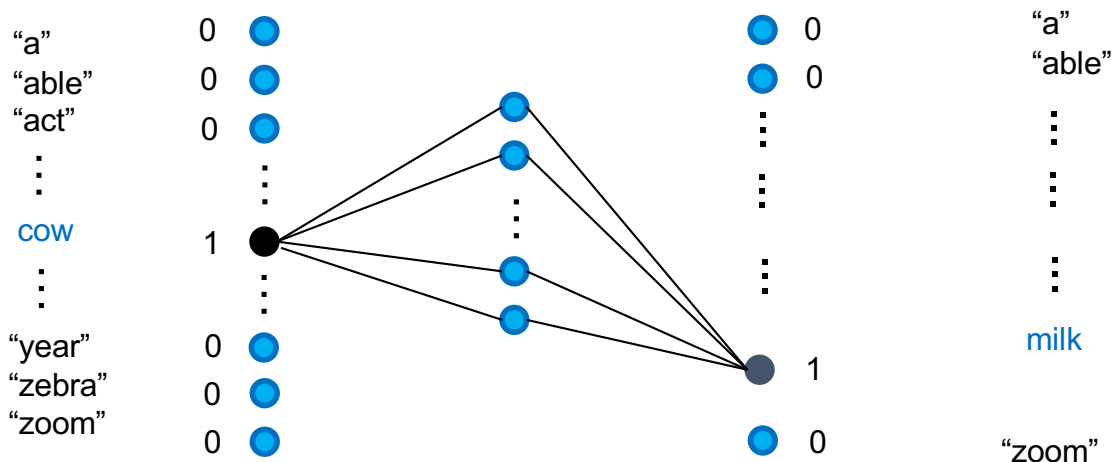
→ We use a Neural Network as an auto-encoder



Distributed word representation

Model Architecture: How do we transform the meaning of words into vectors of real values?

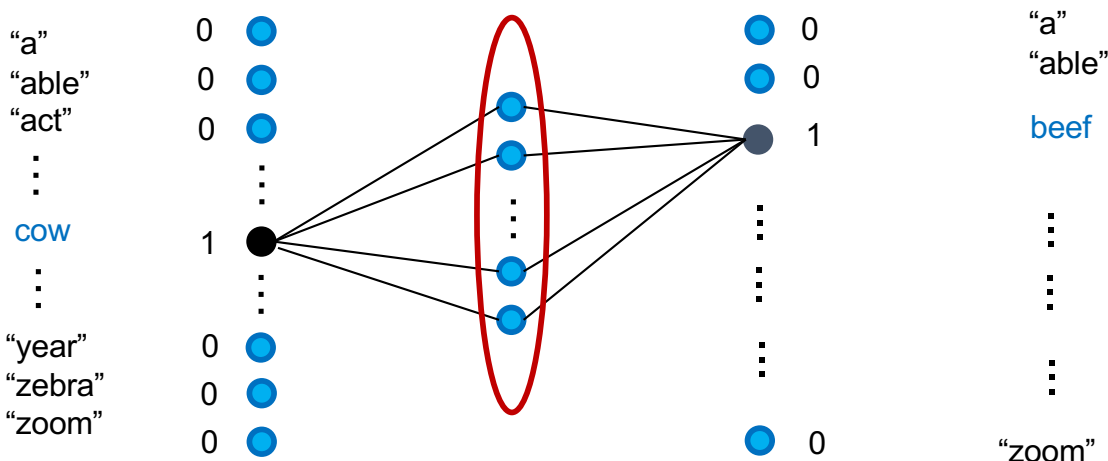
→ We use a Neural Network as an auto-encoder



Distributed word representation

Model Architecture: How do we transform the meaning of words into vectors of real values?

→ We use a Neural Network as an auto-encoder



By showing the network enough pairs (input, context) we can condensate the information of the various context in which the word appears in order to **encode its meaning**.

Thanks to the encoding decoding mechanism the model is able to predict different context words from the **same input words** → **encoded vectors contains the semantic of the word**.

Distributed word representation

Illustration of Word Embedding capabilities

Semantic similarity: Let's ask the model what are the closest words to “bnp”:

```
> embedding_model.(positive=[bnp'], topn=3)
```

Word	Similarity
paribas	0.78
khaleda	0.54
awami	0.53
societe	0.51
barclays	0.48



BNP PARIBAS



9th Prime Minister of Bangladesh

Leader of the Bangladesh Nationalist Party

Often written “bnp” on Wikipedia 😊

Distributed word representation

Illustration of Word Embedding capabilities

Linear operation: We can do basic operations on word vector to recover semantic information

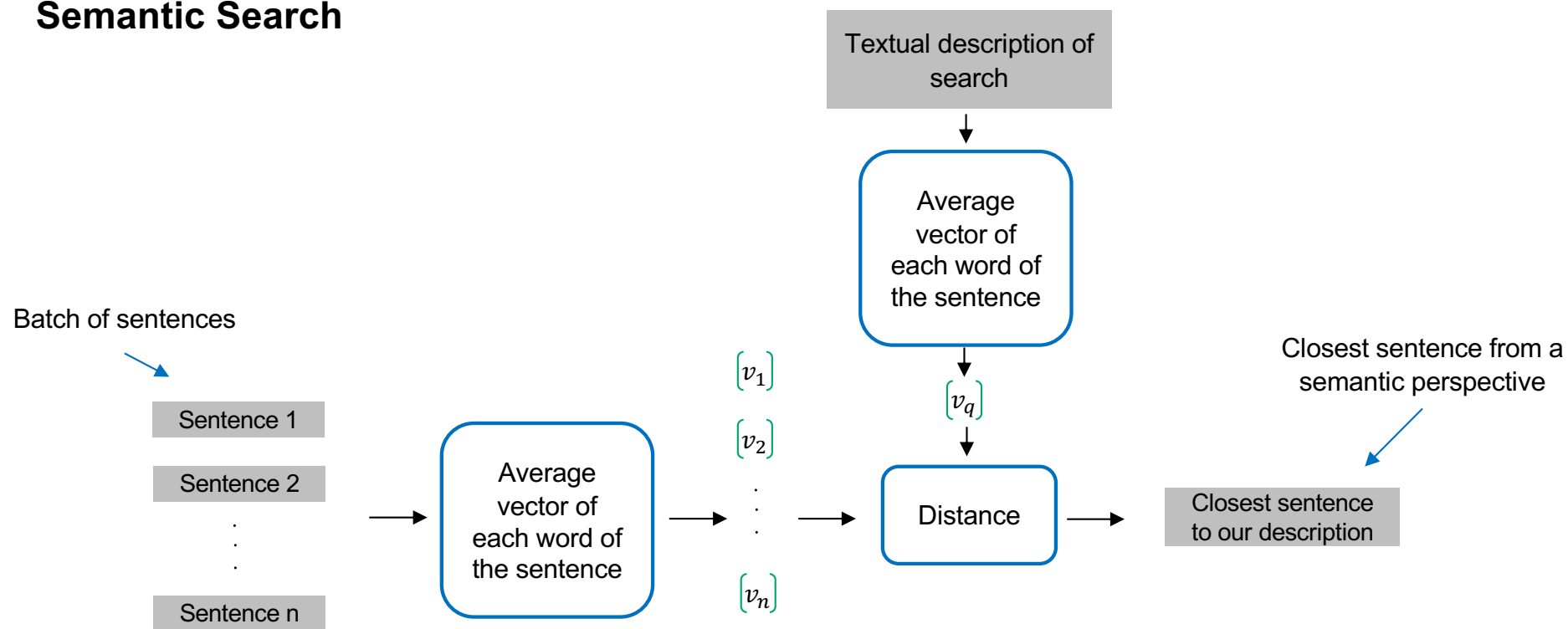
Example: “woman”+”king” – “man”=“queen”
 “paris”+”spain” - “france”=“madrid”

- This means that we can combine the word of a sentence to compute its representation by taking the average vector of all the words of the sentence
- We can do **semantic search** inside text

Distributed word representation

Illustration of Word Embedding capabilities

Semantic Search



Distributed word representation

Pros	Cons
<ul style="list-style-type: none">Distance in vector space allow to compare text at a semantic level	<ul style="list-style-type: none">Harder to implementEncode a single vector for each word and thus a single meaning

Word embedding are

- Less efficient for text classification when using simple models
- Very powerful for text comparison (Question answering, Natural Language Inference)

Conclusion

- Word representation are constantly getting better allowing to encode abstract information about language such as the meaning of word
- This does not mean that we should replace the old by the new → Occurrence based word representation works great for text classification
- It is important to adapt the word representation to the NLP application we want to consider