La Rochelle Université

D'ici, on voit + loin !

Web Sémantique

Fouille de texte

2023-2024

Ahmed Hamdi
Gaël Lejeune

SOMMAIRE

Introduction

INTRODUCTION

Fouille de texte

- La **fouille de texte** (text mining) est une technologie d'intelligence artificielle (IA) qui utilise le traitement du langage naturel (NLP) pour transformer le **texte libre** (non structuré) des documents et des bases de données en **données normalisées** et structurées adaptées à l'analyse ou au pilotage de la machine.

- La **fouille de texte** est le processus d'organiser et d'extraire automatiquement des informations pertinentes à partir de texte non structuré (documents, commentaires clients, réseaux sociaux, emails…)

INTRODUCTION

Jargon

- Traitement automatique des langues (TAL)
- Traitement automatique du langage naturel (TALN)
- Natural Language Processing (NLP)
- Technologies de langage
- Linguistique computationnelle

INTRODUCTION

Quelques chiffres

- Entre 6 000 et 7 000 langues parlées dans le monde
- Environ 1 500 possèdent un système écrit
- Plus de 95% des locuteurs parlent moins de 5% des langues
- **Urgence** : doter les langues en voie de disparition

INTRODUCTION

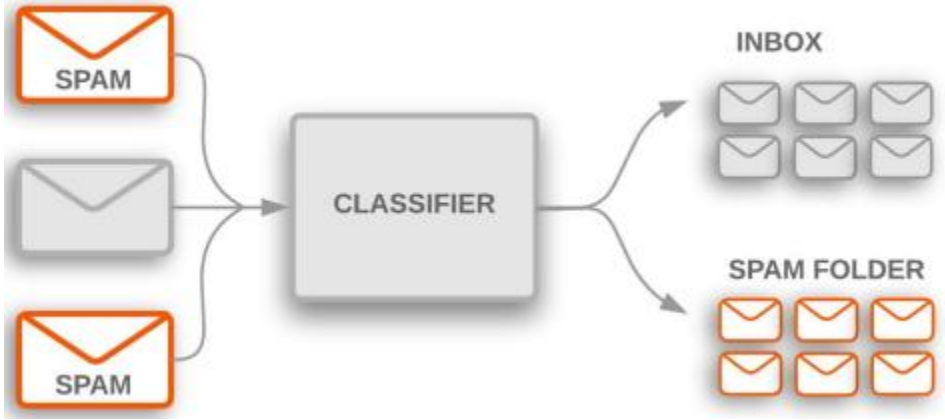
Applications

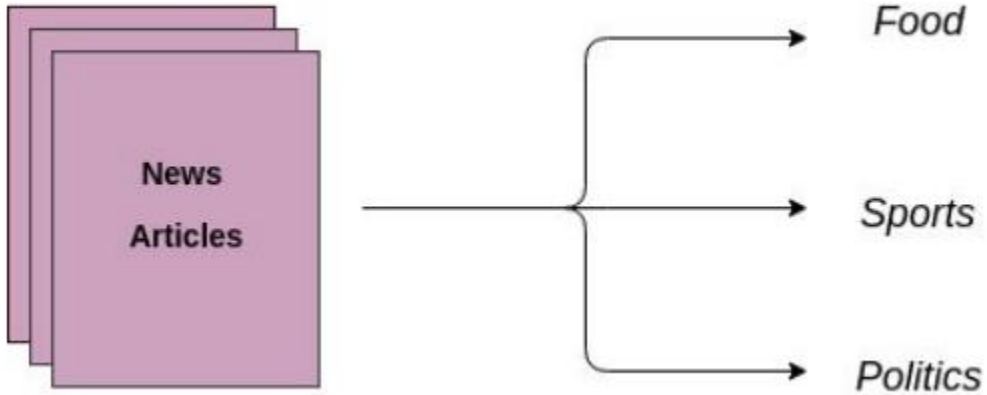
- Traduction automatique
- Recherche d'informations
- E-réputation
- Analyse de sentiments
- Indexation
- …

TAL

Applications

- **Classification de texte** : assigner une classe à un texte donné
  - Distinguer les spams des autres mails
  - Décider si un article appartient à une liste prédéfinie de classe (technologie, politique, sport)



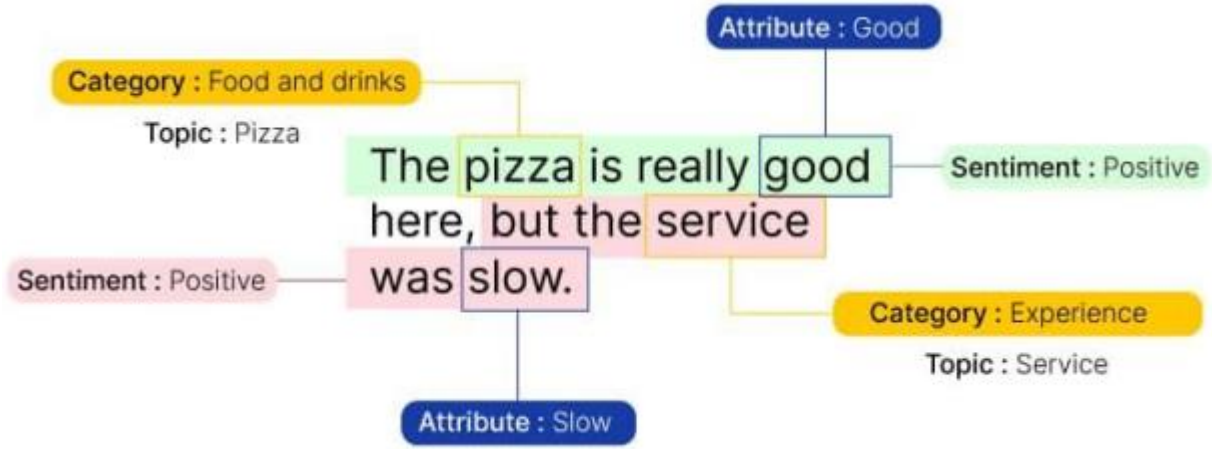


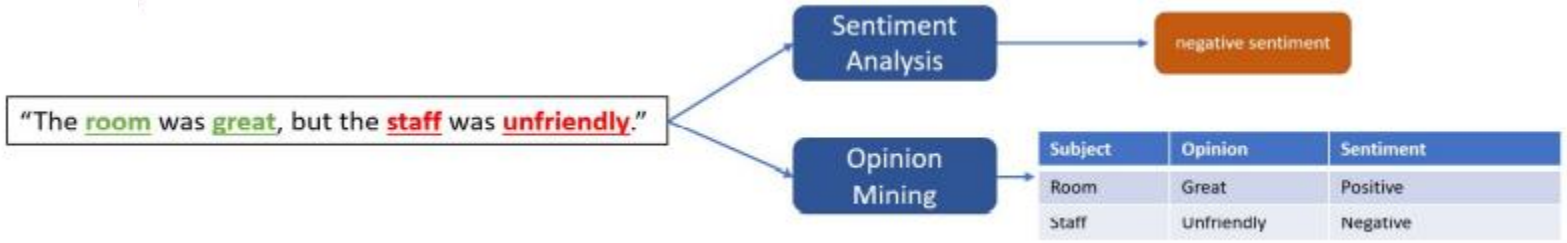
TAL

Applications

- **Analyse de sentiments (détection d'opinions)** : identifier et extraire des informations subjectives à partir d'une source de texte

  - Avis, évaluations et recommandations en ligne sur les sites des entreprises qui cherchent à commercialiser leurs produits, à identifier de nouvelles opportunités et à gérer leur réputation

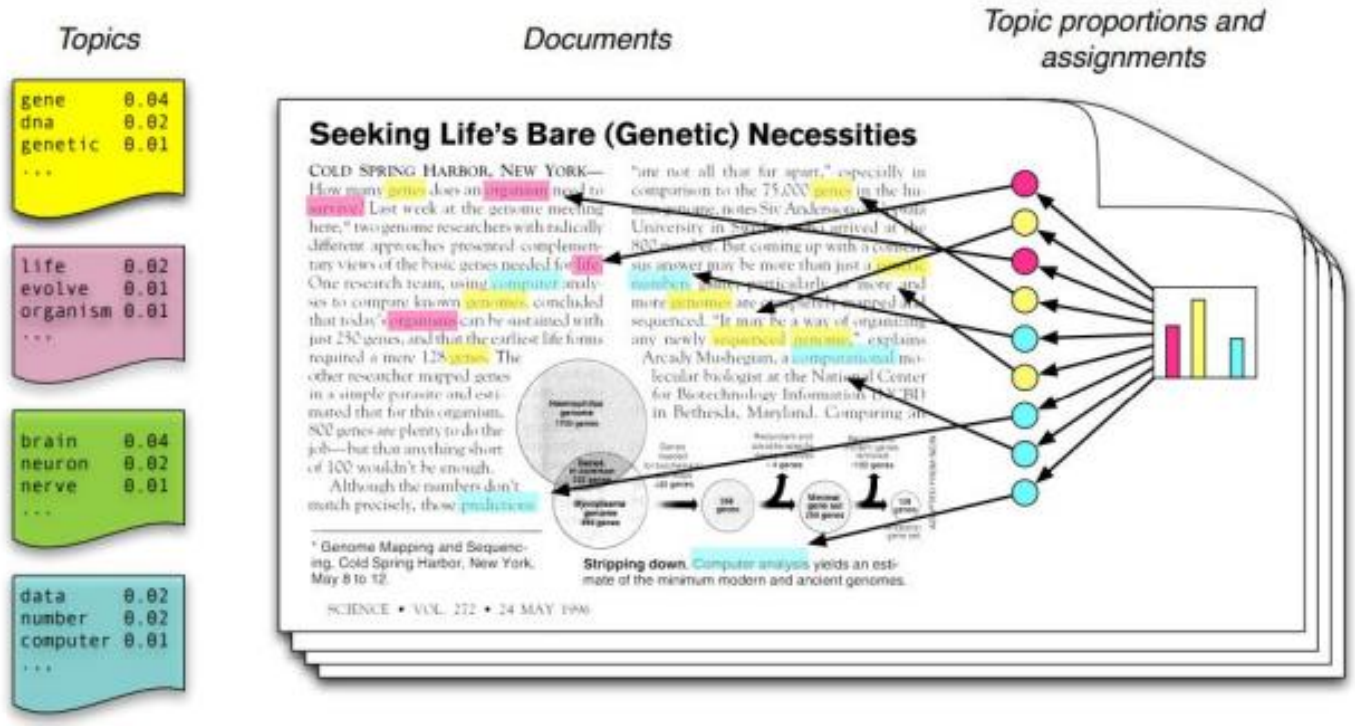




TAL

Applications

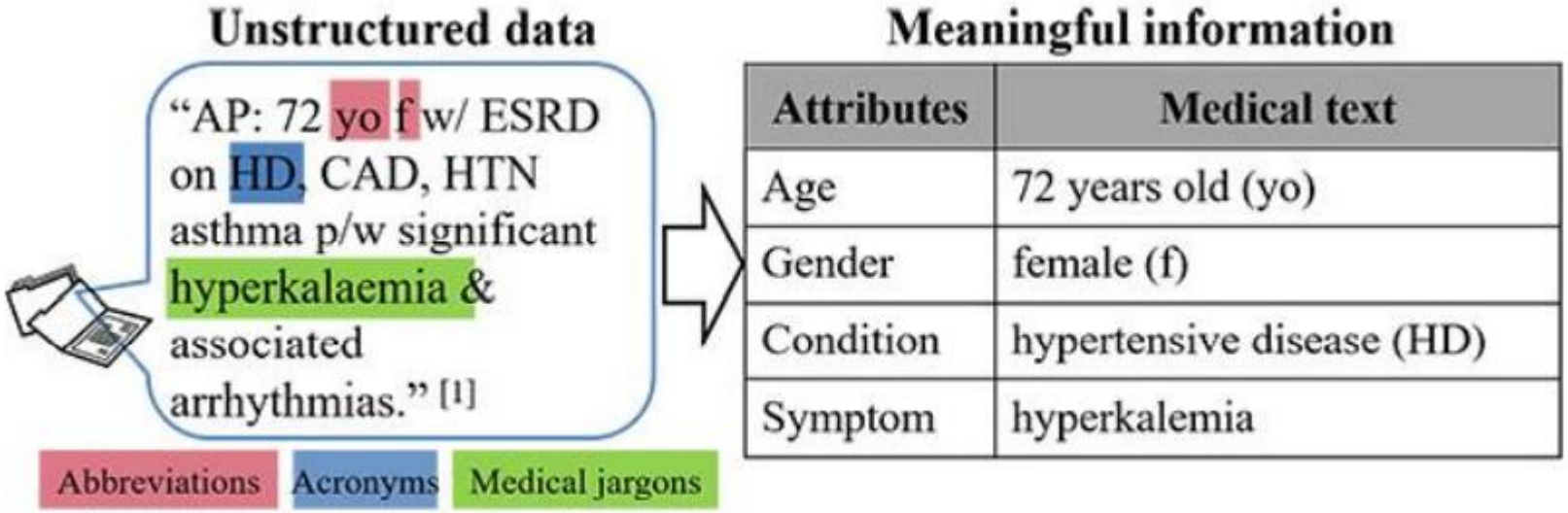
- **Topic modeling** : découvrir les principaux thèmes qui imprègnent une vaste collection de documents non structurée



TAL

Extraction d'information

- **Fouille de textes médicaux** : extraire des informations factuelles (qui fait quoi à qui pourquoi)
  - Identifier des informations pertinentes à partir des documents (finance, médecine, tourisme…)
  - tirer des informations de diverses sources et les regrouper sous une forme structurée

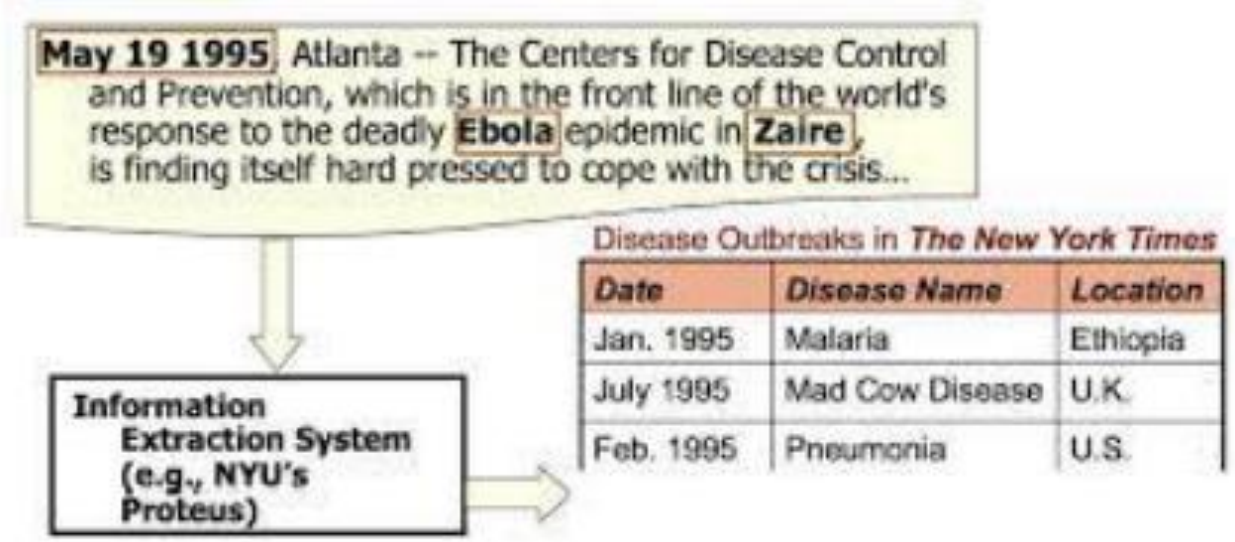


Attributes	Medical text
Age	72 years old (yo)
Gender	female (f)
Condition	hypertensive disease (HD)
Symptom	hyperkalemia

TAL

Applications

- **Fouille de textes médicaux** : fiches de patients
  - Relations de liaison aux protéines utiles pour la découverte de médicaments
  - Détection des relations gène-maladie à partir de la littérature biomédicale
  - Trouver des relations médicament-effets secondaires dans les dossiers de santé



Terminologie

TERMINOLOGIE

Corpus

- **Corpus** : ensemble de documents (ou « textes ») construit pour une étude

- Trois types :
  - Corpus d'apprentissage (ou d'entraînement) : apprendre un modèle pour une étude donnée
  - Corpus de développement (ou de validation) : ajuster les paramètres appris par le modèle
  - Corpus de test (ou de référence) : évaluer le modèle appris

TERMINOLOGIE

Forme, type, vocabulaire

- La notion de « mot » est très floue. On utilise plus couramment :
  - **Occurrence** (token)
  - **Forme** (type)
- L'ensemble des formes d'un corpus constitue son **vocabulaire**

TERMINOLOGIE

Forme, type, vocabulaire

- corpus
  - *Je mange une pomme chaque jour*
  - *Je mange aussi du pain chaque jour*

- Ensemble d'occurrences :

O = {*Je, mange, une, pomme, chaque, jour, Je, mange, aussi, du, pain, chaque, jour*}

- Vocabulaire (ensemble de formes) :

V = {*Je, mange, une, pomme, chaque, jour, aussi, du, pain*}

TERMINOLOGIE

lemme, partie du discours

- *Je mange une pomme chaque jour*
- *Je mange aussi du pain chaque jour*

+ Lemmes

Je, manger, une, pomme, chaque, jour

Je, manger, aussi, de+le, pain, chaque, jour

+ Parties du discours (POS-tag)

Pronom, verbe, déterminant, nom, adjectif, nom

Pronom, verbe, adverbe, déterminant, adjectif, nom

Fouille de texte : niveaux d'analyse

Ambiguïtés à tous les niveaux

Pas si simple

- Le bus a renversé un passant …
  - … je l'ai entendu freiner.
  - … je l'ai entendu crier.
- Le professeur a envoyé l'élève chez le proviseur
  - … il faisait trop de bruit.
  - … il était excédé.
  - … il l'avait convoqué.
- Non !
  - Si je viens demain en cours ? Non !
  - Vas-tu en cours demain ? Non !

TAL

Tokenisation (découpage en mots)

- *L'arbre*
- *Aujourd'hui*
- *Plateforme*
- *Dit-il*

Et les langues agglutinantes ?

+ أتتذكروننا

  « *est-ce que vous vous souvenez de nous ?*»

+ Vablaheibarvegavinnuverkfrageymsluskurautidyralyklakippuhringur

  « *porte-clés de la chaîne de clé pour la porte extérieure du hangar à outils des agents de la route sur le plateau Vablaheibi*»

TAL

Analyse morphologique

la porte

- Porte + Nom + Fem + Sg (j'ouvre la porte)
- Porter + Verbe + 1/3P + Sg (il la porte)

TAL

Analyse syntaxique

Jean regarde un homme sur la colline avec un télescope

- Qui est sur la colline ?
- Qui a un télescope ?

1. Jean regarde [un homme sur la colline avec un télescope]
2. Jean regarde [un homme sur la colline] avec un télescope
3. Jean regarde un homme [sur la colline avec un télescope]

TAL

Analyse sémantique

Tous les hommes aiment une femme

- **Chaque** homme aime une femme
- **Tous** les hommes aiment la même femme

J'ai demandé un gâteau au chocolat
J'ai demandé un gâteau au serveur

Solutions TAL

Solutions

Des techniques variées

+ Systèmes à base de **règles**
  - définies par l'humain (linguistes)
  - définies manuellement

+ Systèmes à base de **données**
  - *apprentissage supervisée ou non supervisé*
  - *à partir d'exemples (rédigés et/ou annotés par des humains)*
  - *Algorithmes (pensés par des humains)*

+ Systèmes hybrides

Solutions

Identification de la langue

✚ Loi de Zipf (observation empirique)

- Un corpus contenant 1M d'occurrences « *Brown Corpus* »
- « the » représente 7% du corpus
- 135 mots représentent la moitié des occurrences dans le corpus
- Inversement, la moitié du vocabulaire du corpus sont des **hapax**
- Les mots fréquents sont très rares… et inversement

Solutions

Identification de la langue

+ Loi de Zipf sur le Brown corpus

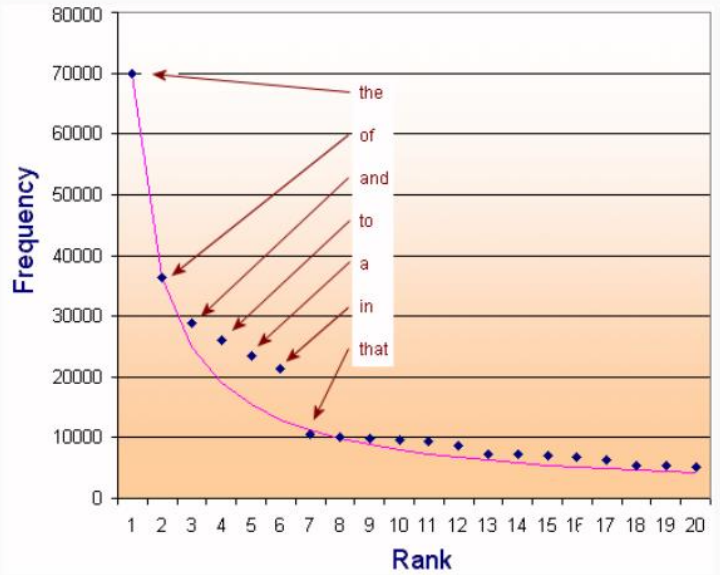


Figure 3 – Données très proches de l'attendu, surtout sur la longue traîne

Solutions

Identification de la langue

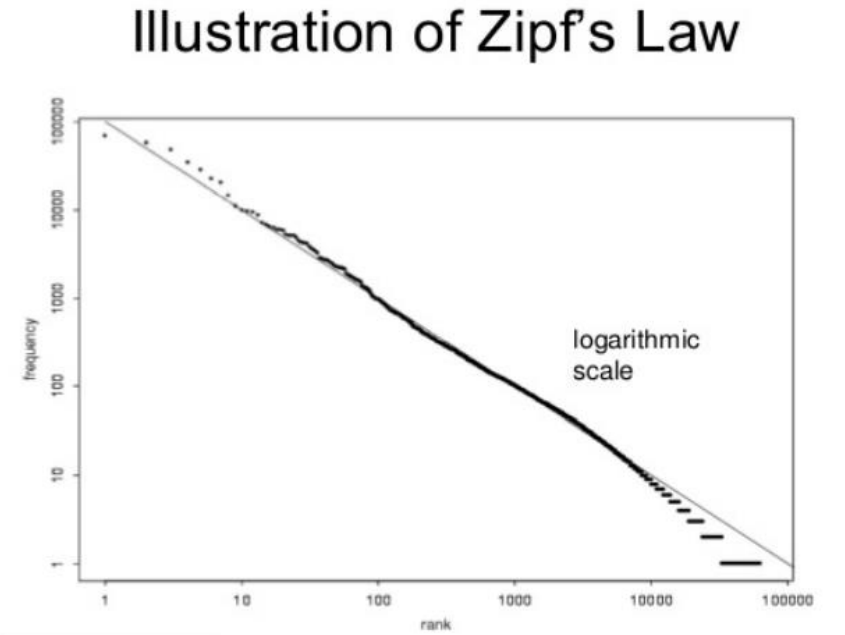


Figure 4 – Validité plus marquante encore en échelle logarithmique

Solutions

Identification de la langue

+ Méthode des short words / fequent words

- Short words : stop-words, mots-vides, mots-outils …
- Liste des mots grammaticaux pour chaque langue
- Compter les occurrences de ces mots outils dans le corpus
- Comparer avec des listes de référence (web scrapping)

Solutions

Identification de la langue

+ Implantation rapide

+ Données : corpus parallèle de l'Union Européenne (22 langues)

  - Découpage en deux parties (entraînement et test)
  - Entraînement : extraction d'un modèle de langue (les n mots plus fréquents) à partir de tous les textes de chaque langue
  - Test, pour chaque texte :
    - Calcul de l'intersection en mots
    - On prend la plus grande ➔ prédiction

Solutions

Identification de la langue

lg	#1	#2	#3	#4
bg	на (12593)	за (5657)	и (5529)	в (3919)
cs	a (5510)	v (3378)	na (2424)	se (1955)
da	og (5435)	i (4542)	at (4147)	af (3682)
de	der (5867)	die (5604)	und (5155)	in (2747)
en	**the (9547)**	and (5692)	of (5430)	to (4787)
es	de (16556)	la (8571)	en (5096)	y (5048)
et	ja (4295)	on (2746)	Euroopa (1658)	et (1240)
fi	ja (4952)	on (2623)	Euroopan (985)	EU :n (898)
fr	**de (11801)**	la (6466)	et (5177)	les (4999)
hu	a (9824)	az (4956)	és (4327)	A (2509)
it	di (7617)	e (4838)	in (2987)	la (2958)
lt	ir (4984)	Europos (1645)	kad (1311)	– (1293)
lv	un (5028)	ir (2448)	par (1658)	Eiropas (1473)
mt	u (5234)	li (4557)	ta' (2960)	ta' (1554)
nl	de (11253)	van (7093)	en (5167)	het (3986)
pl	w (5750)	i (3799)	na (2844)	z (1986)
pt	de (10488)	a (6684)	e (5153)	da (3785)
ro	de (10094)	în (5478)	și (5020)	a (4710)

Solutions

Identification de la langue

- 97% de bonne prédiction
- Autres méthodes :
  - Trigrammes de caractères : 96% de bonne prédiction
  - Méta-données, encodage…

lg	#1	#2	#3
bg	_на (12863)	на_ (11886)	ите (9741)
da	er_ (14032)	en_ (9306)	for (8681)
en	_th (13006)	the (11879)	he_ (11177)
es	_de (20787)	de_ (16648)	os_ (13741)
et	mis (6513)	se_ (5245)	ise (4791)
fi	en_ (11551)	ist (6937)	an_ (6291)
fr	es_ (21305)	_de (17707)	de_ (12042)

Solutions

Limitations

- Longueur du texte
- Textes multilingues
- Pièges de contexte
  - Barack Obama ➔ italien
  - Nicolas Sarkozy ➔ polonais
  - Barack Obama and Nicolas Sarkozy ➔ anglais

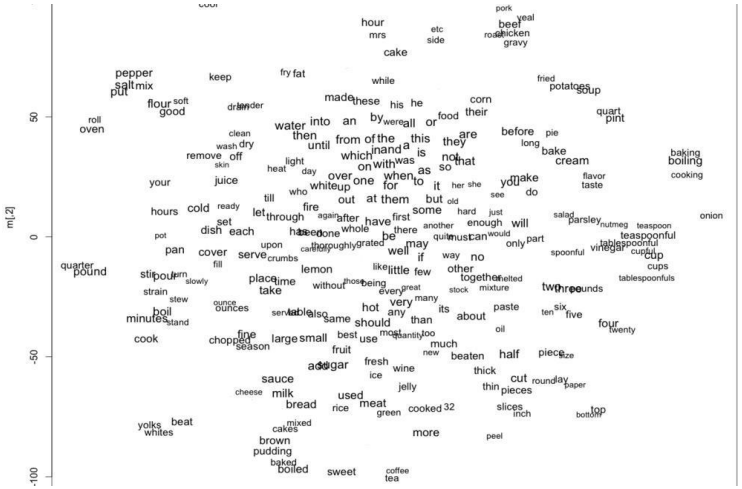
Word Embedding

Word embedding

What is it?

+ Converting each word into vector

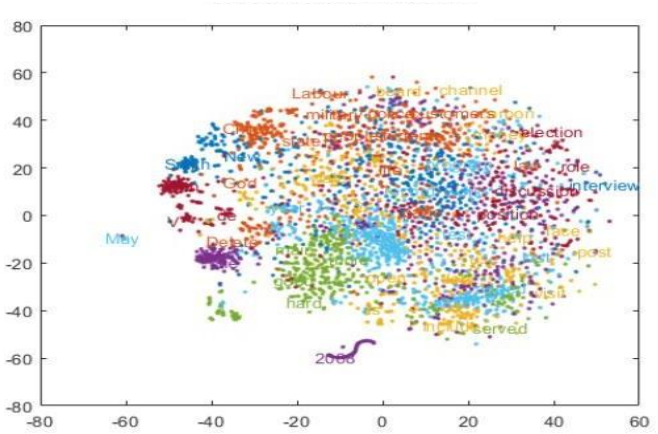
house →	0,01	0,55	0,08	0,19	0,98	0,67
course →	-0,1	0,33	0,09	0,27	0,78	0,19
horse →	0,08	-0,5	0,11	0,26	0,55	-0,7
mare →	0,24	0,87	-0,8	0,99	0,91	0,04
camel →	0,22	0,24	0,66	0,74	0,45	0,36



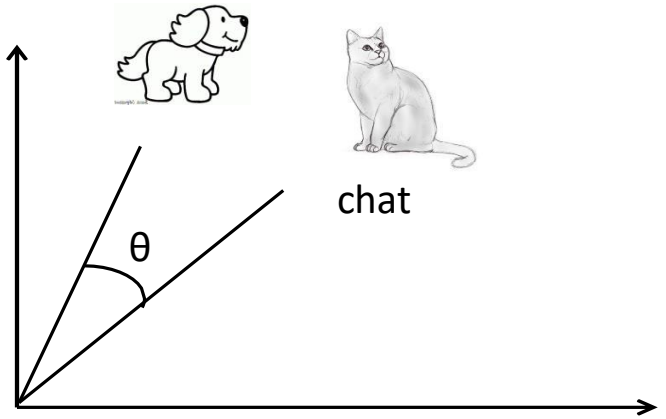
Word embedding

BUT why?

clustering



similarity



SIM (chien, chat) = cos(θ)

Mathematical operations

word + embedding = ?

Word embedding

Models

- Word frequency-based models
  - One-hot encoding
  - Bag-of-Words
  - TF-IDF
- Context-based models
  - Static : Word2vec, Glove, FastText
  - Dynamic : Bert, ELMo, Flair

Human-Readable

Pet
Cat
Dog
Fish
Bird
Cat

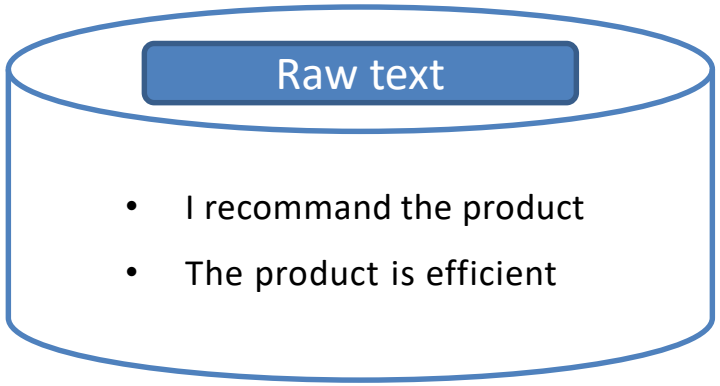
Machine-Readable

0,08	**-0,5**	**0,11**	**0,26**	**0,55**	**-0,7**
-0,1	-0,2	0,23	0,44	0,56	-0,3
0,97	-0,7	-0,1	0,21	0,23	0,87
0,12	0,53	0,64	0,23	0,67	-0,4
0,08	**-0,5**	**0,11**	**0,26**	**0,55**	**-0,7**

First approaches

One-hot Encoding

+ The simplest representation



Index	word
1	efficient
2	I
3	the
4	product
5	is
6	recommand

1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

➔ 6 word types (unique words) : **vocabulary**

But, if we have vocabulary of 50k words ? ~1M words in the English vocabulary

➔ Each word is converted into a vector with 49,999 zero and a unique 1

First approaches

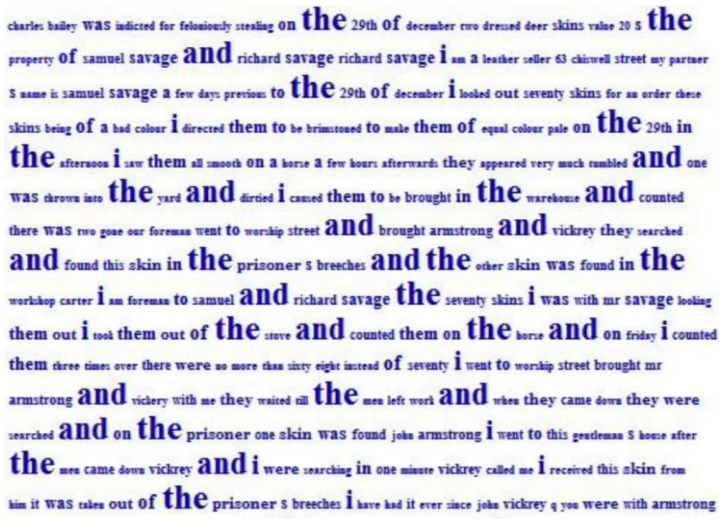
One-hot Encoding

- Vocabulary size

But, if we have vocabulary of 50k words ?

➔ we need 50,000^2 = 2,5 billion units of memory space -- **Not efficient in terms of calculation**

Stop words ➔ most frequent



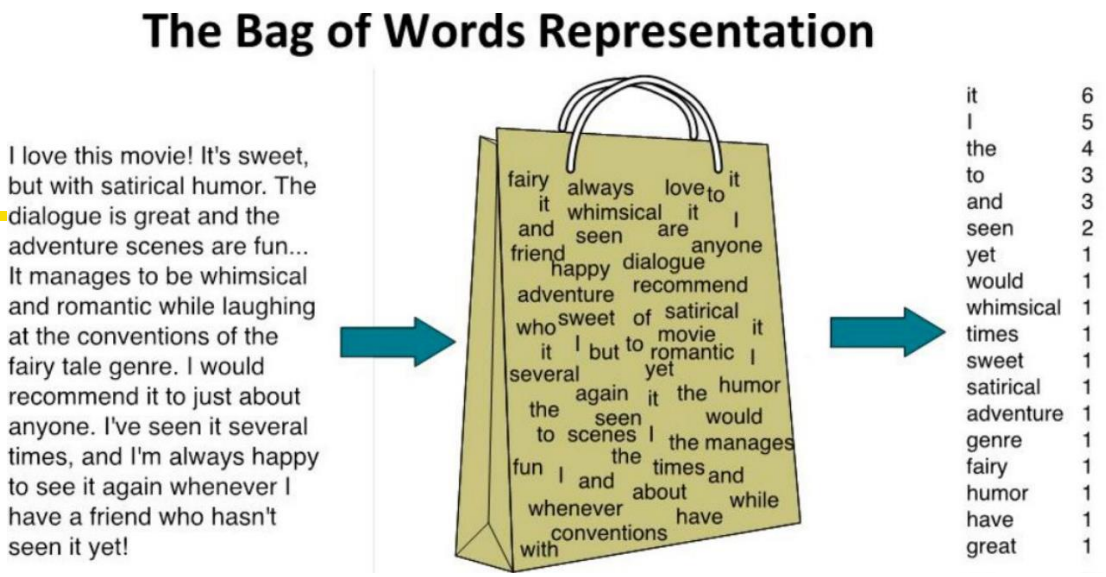
lemmas vs stems

First approaches

Bag-of-Words

- Unlike **one-hot encoding**, the **bag-of-words** is a representation of text that describes **the occurrence of words** within a document

- **Bag** ➔ because any information about the order or structure of words in the document is discarded



Two documents

1. I recommande the product
2. The product is efficient

➔ 6 word types (unique words) : **vocabulary**

Vocabulary

Indice	Terme	Fréquence
1	efficient	1
2	I	1
3	the	2
4	product	2
5	is	1
6	recommand	1

	Doc 1	Doc 2
efficient	0	1
I	1	0
the	1	1
Product	1	1
is	0	1
recommand	1	0

Premières approaches

TF-IDF

- The TF-IDF (Term Frequency-Inverse Document Frequency) statistical measure makes it possible to evaluate the importance of a term contained in a document, relative to a collection or a corpus.
- The weight increases in proportion to the number of occurrences of the word in the document. It also varies according to the frequency of the word in the corpus.
- For each word i in a document j

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

- Doc 1 : I recommand the product

I	➔ TF = 1/6 = 0,17	➔ IDF = log(2/1) = 0.3	➔ TF*IDF = 0.17*0.3
recommand	➔ TF = 1/6 = 0,17	➔ IDF = log(2/1) = 0.3	➔ TF*IDF = 0.17*0.3
the	➔ TF = 2/6 = 0.33	➔ IDF = log(2/2) = 0	➔ TF*IDF = 0
product	➔ TF = 2/6 = 0.33	➔ IDF = log(2/2) = 0	➔ TF*IDF = 0

- Doc 2 : the product is efficient

the	➔ TF = 2/6 = 0.33	➔ IDF = log(2/2) = 0	➔ TF*IDF = 0
product	➔ TF = 2/6 = 0.33	➔ IDF = log(2/2) = 0	➔ TF*IDF = 0
is	➔ TF = 1/6 = 0.17	➔ IDF = log(2/1) = 0.3	➔ TF*IDF = 0.17*0.3
efficient	➔ TF = 1/6 = 0.17	➔ IDF = log(2/1) = 0.3	➔ TF*IDF = 0.17*0.3

First approaches

Limitations

- Large vocabulary ➔ Large vectors
- **SPARSE** vectors

w_1	-0.55	0	0	0.26	0	…	0
w_2	0	0.89	0	0	0	…	0
w_3	0	0	0.67	0	-0.45	…	0
…	…	…	…	…	…	…	…
w_n	0	-0.11	0	0	0	…	0.98

➔ **Encoding that completely ignores the word meaning**

First approaches

Limitations

- Encoding that completely ignores the word meaning

  - The **altitude** of the Eiffel Tower is 330 meters
  - The **height** of the Eiffel Tower is 330 meters

- In a classic vector representation:
  - the word **altidtude** is as close to the word **height** as it is to other words
  - distance (altitude, height) = distance (altitude, xxx)

  No meaning ➔ No semantic proximity

- But, what is meaning? How to encode it in a vector?

Context-based embedding

What is meaning?

John Rupert Firth (1890-1960) was an English linguist and a leading researcher in British linguistics in the 1950s.

"You shall know a word by the company it keeps!" [Firth, 1957]

➔ Learning the meaning of a word through its contexts of use

context = neighbor words

Context-based embedding

The word2vec model

- Word2vec (Mikolov et al.2013)

Efficient estimation of word representations in vector space

T Mikolov, K Chen, G Corrado, J Dean

arXiv preprint arXiv:1301.3781, 2013 · arxiv.org

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors

AFFICHER PLUS

☆ Enregistrer Citer Cité 37741 fois Autres articles Les 45 versions

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

1 Introduction

Many current NLP systems and techniques treat words as atomic units - there is no notion of similarity between words, as these are represented as indices in a vocabulary. This choice has several good reasons - simplicity, robustness and the observation that simple models trained on huge amounts of data outperform complex systems trained on less data. An example is the popular N-gram model used for statistical language modeling - today, it is possible to train N-grams on virtually all available data (trillions of words [3]).

However, the simple techniques are at their limits in many tasks. For example, the amount of relevant in-domain data for automatic speech recognition is limited - the performance is usually dominated by the size of high quality transcribed speech data (often just millions of words). In machine translation, the existing corpora for many languages contain only a few billions of words or less. Thus, there are situations where simple scaling up of the basic techniques will not result in any significant progress, and we have to focus on more advanced techniques.

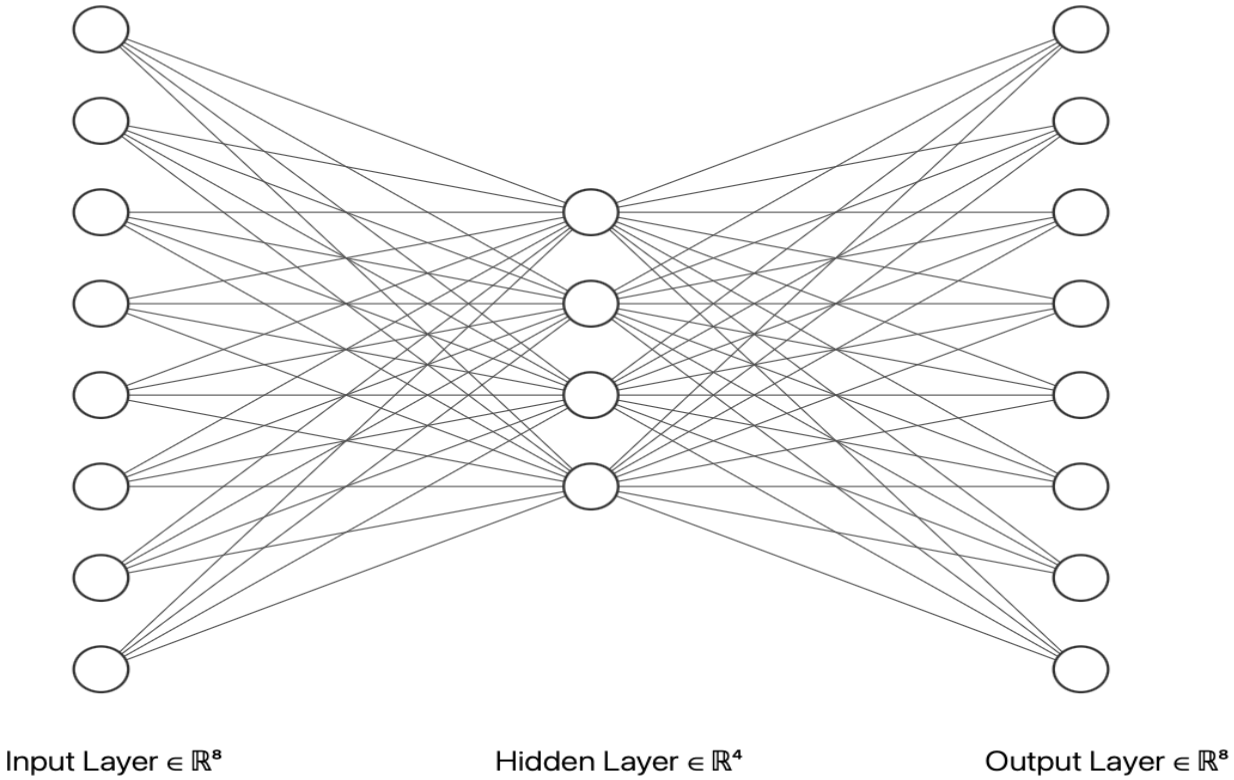
With progress of machine learning techniques in recent years, it has become possible to train more complex models on much larger data set, and they typically outperform the simple models. Probably the most successful concept is to use distributed representations of words [10]. For example, neural network based language models significantly outperform N-gram models [1, 27, 17].

1.1 Goals of the Paper

The main goal of this paper is to introduce techniques that can be used for learning high-quality word vectors from huge data sets with billions of words, and with millions of words in the vocabulary. As far as we know, none of the previously proposed architectures has been successfully trained on more

Word2vec

Shallow Learning

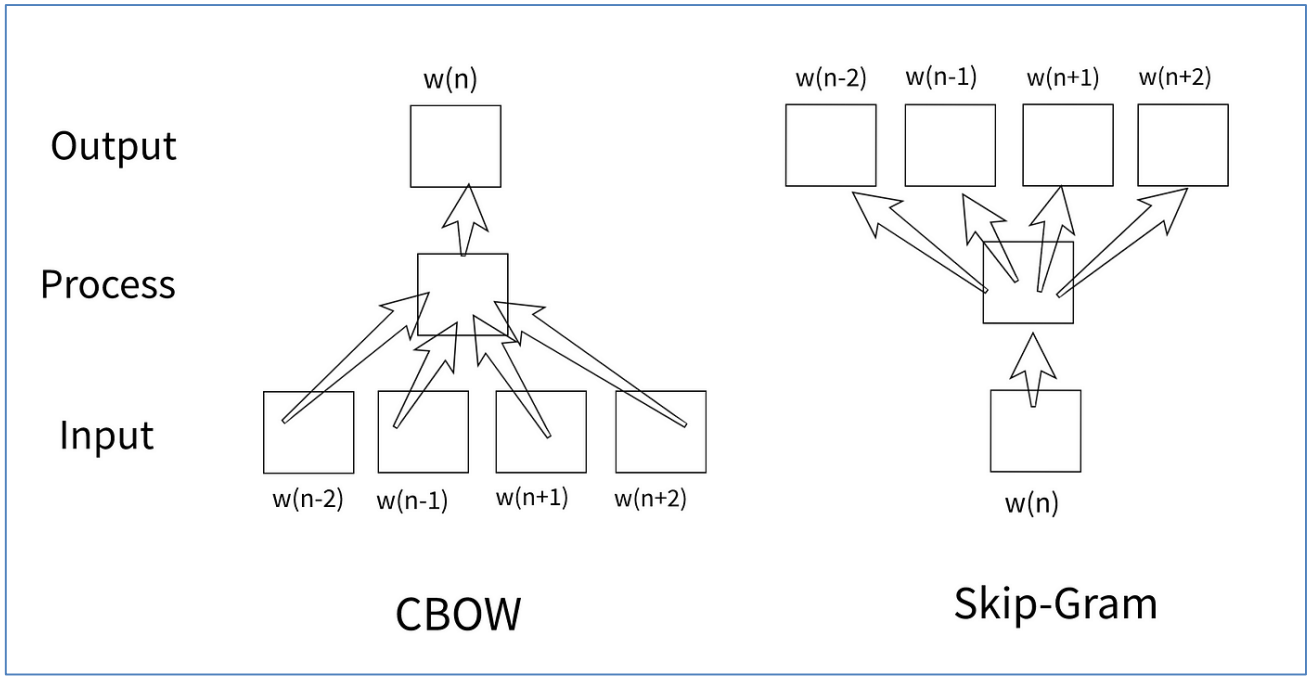


Word2vec

Approaches

+ **Approach 1** : CBOW
Continuous Bag-of-Words

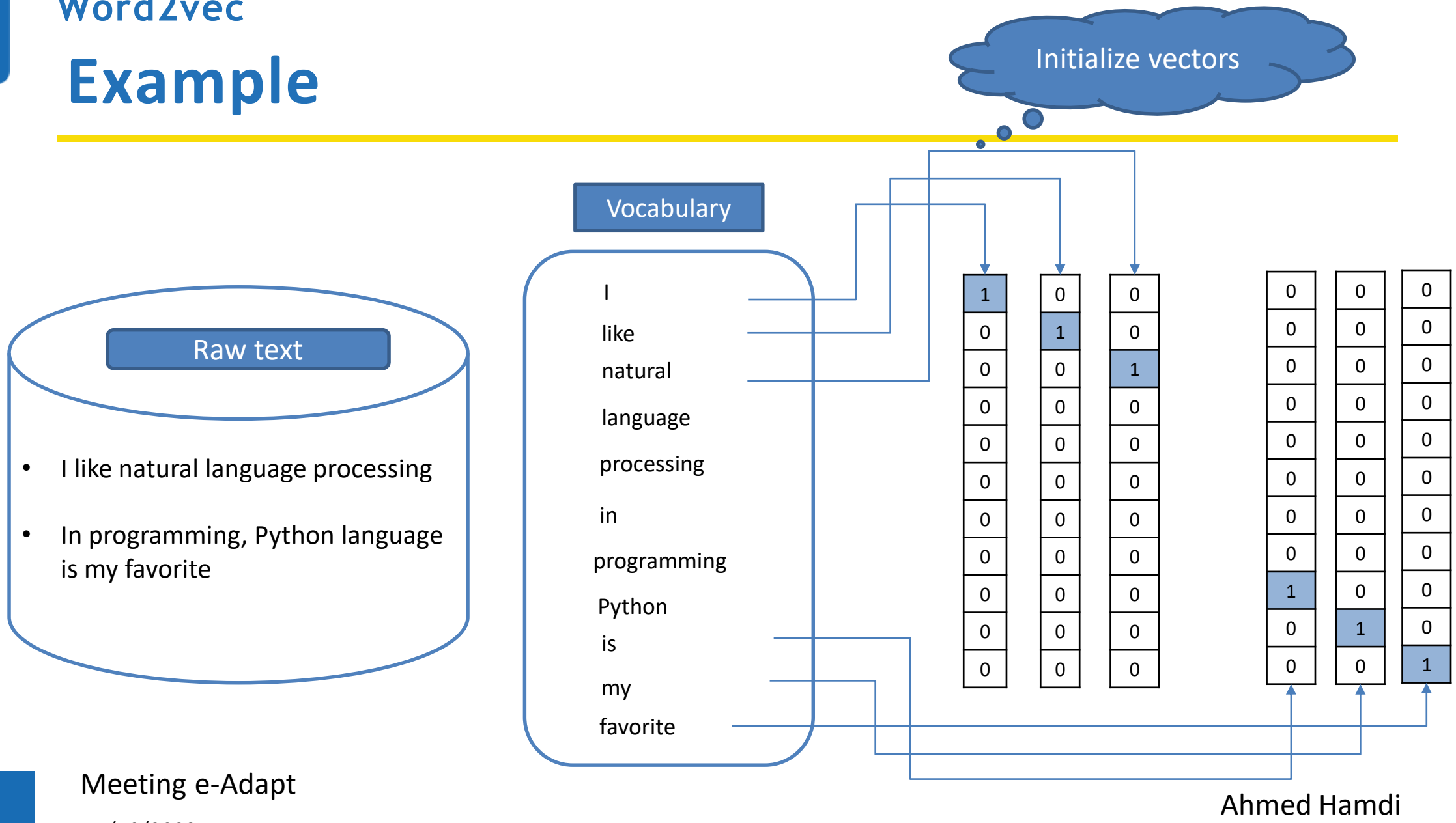
+ **Approach 2** : Skip-gram



Prediction of the **current word** according to the **right and left context**

Prediction of the **right and left context** according to the **current word**

Example


Initialize vectors
Vocabulary
Raw text
I like natural language processing
In programming, Python language is my favorite
I
like
natural
language
processing
in
programming
Python
is
my
favorite

Example

CBOW: predict the current word according the context

window = 2 words

context = 4 words

- I like natural language processing
- I like natural language processing
- I like natural language processing
- I like natural language processing
- I like natural language processing
- In programming, Python language is my favorite

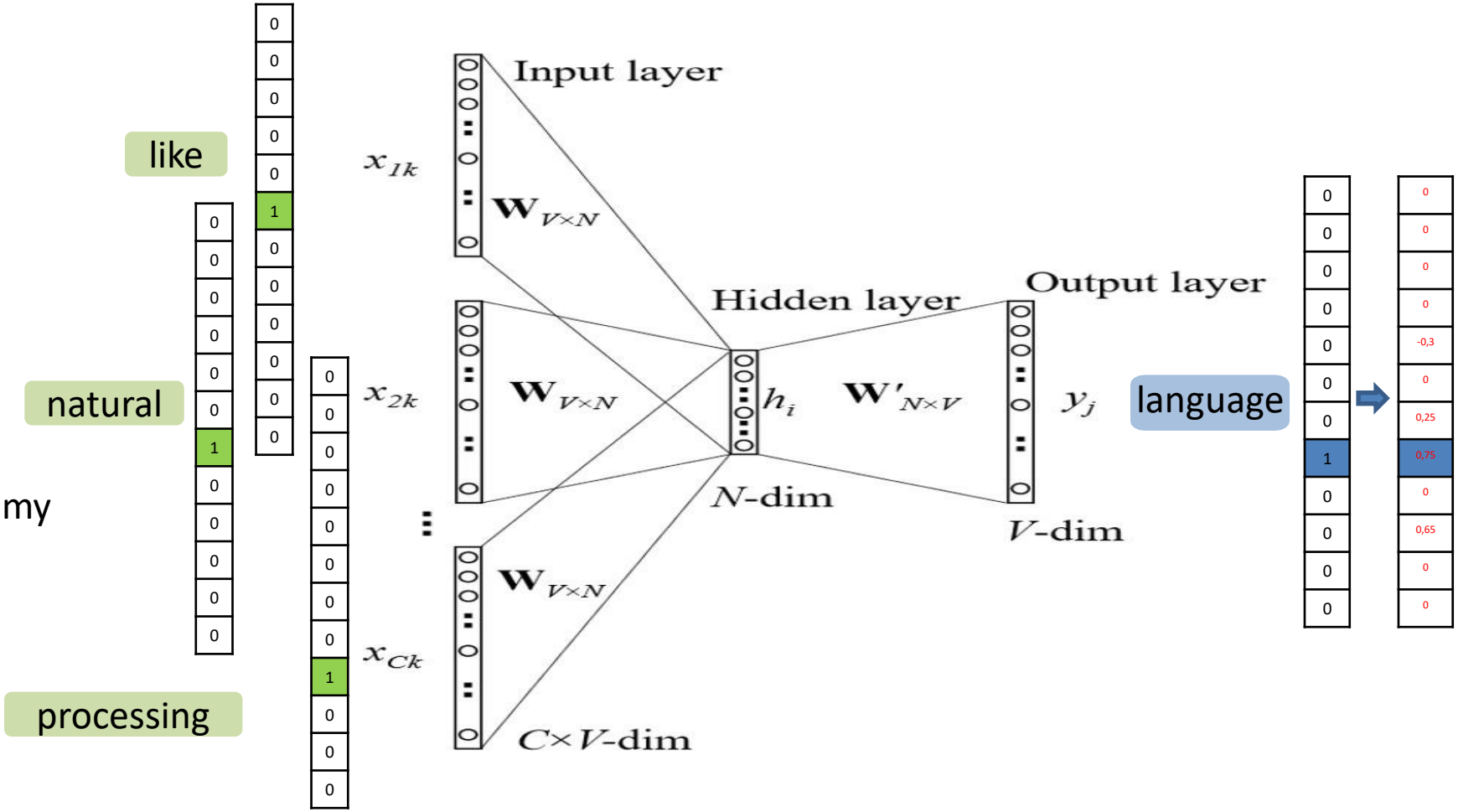
Word2vec

Example

CBOW ➔ learning W

I like natural language processing

In programming, Python language is my favorite



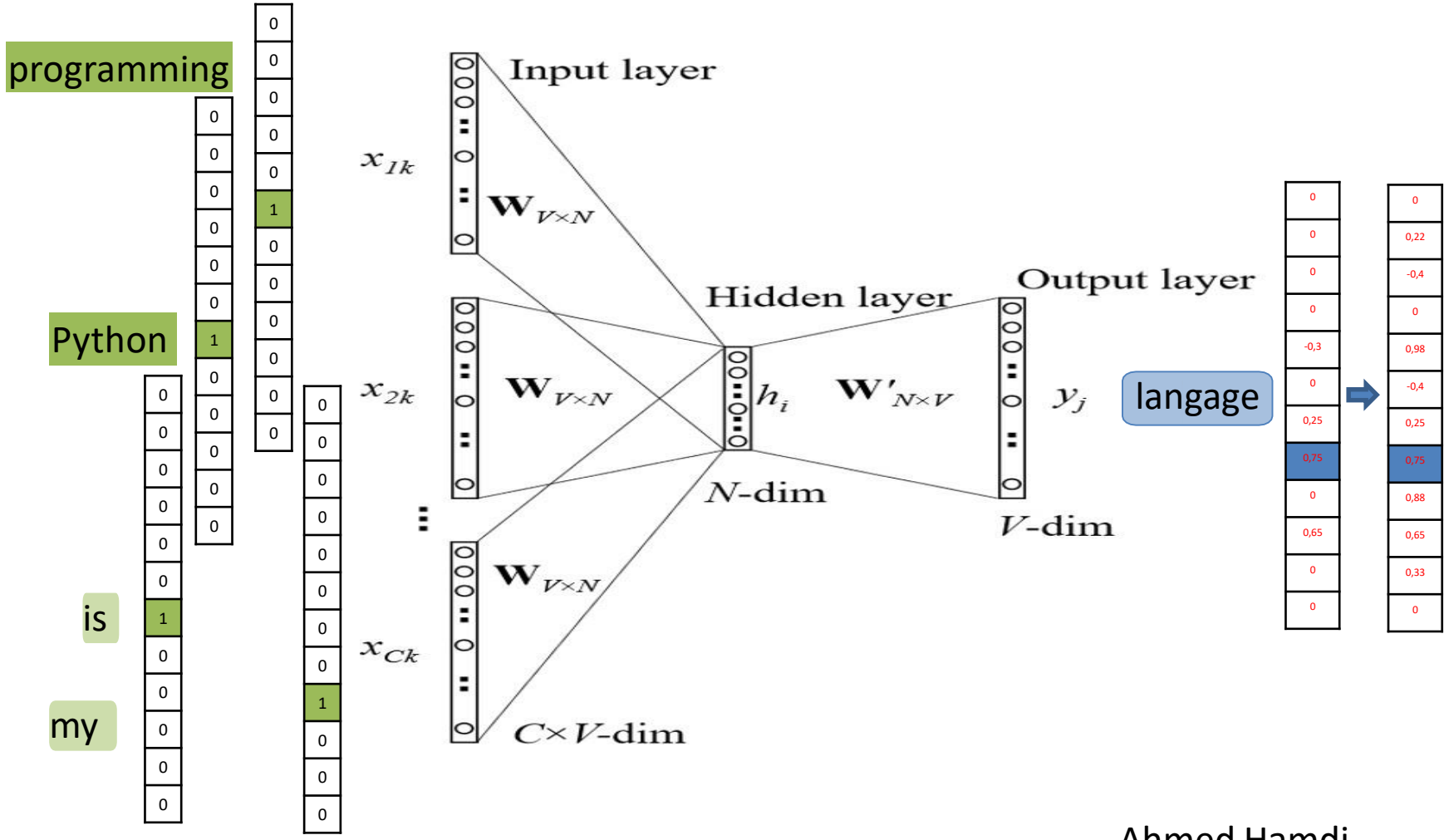
Word2vec

Example

CBOW ➔ learning W

I like natural language processing

In programming, Python language is my favorite

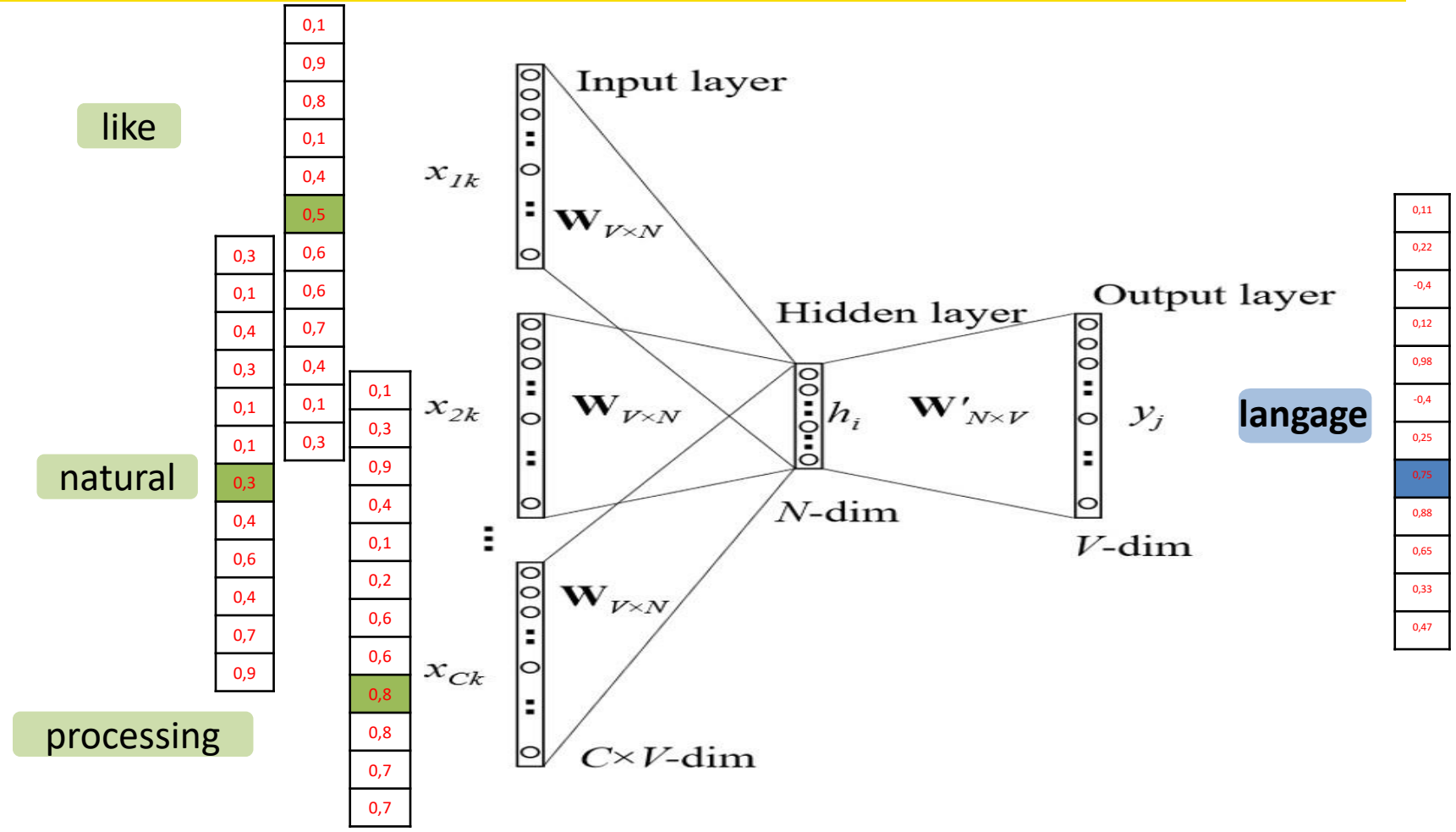


Word2vec

Example

Skip-gram ➔ learning W’

I like natural language processing



Word2vec

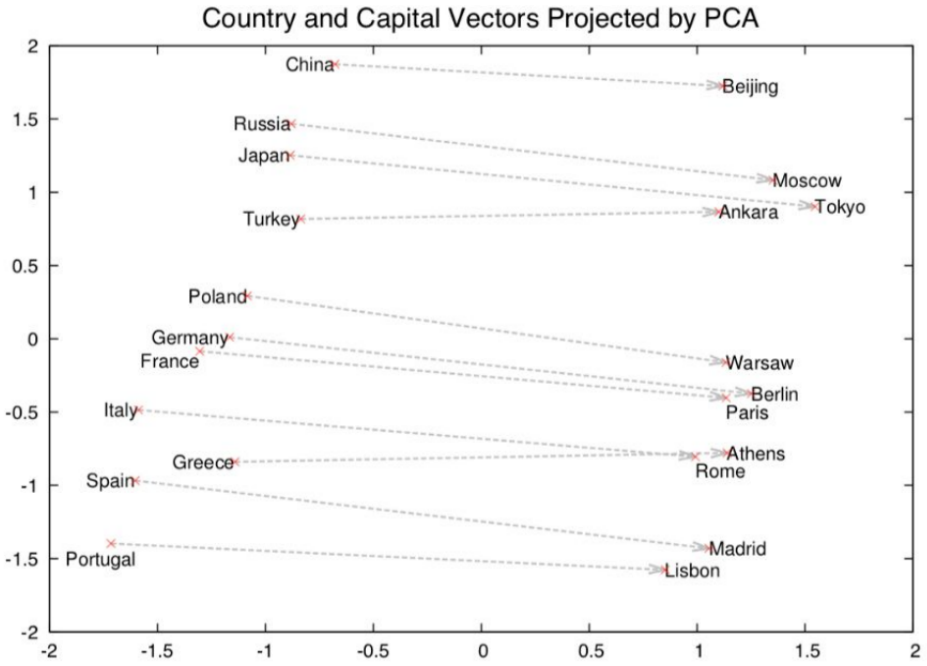
Model training

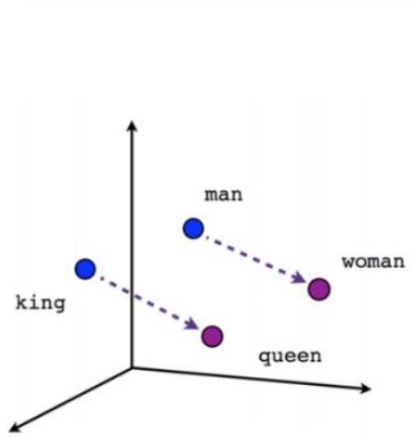
- Persue learning of W_{VxN} and $W'_{V\times N}$
  - > Couvrir tous les mots du vocabulaire
  - > Contexte plus large
- *Use* W_{VxN} or $W'_{V\times N}$ as word representation
  - > Or use the average of both

vocabulaire	$W_{V \times N}$				
nous	0.1	0.5	..	..	0.6
convertissons	2.4	2.6	..	..	1.8
chaque	1.6	1.4	…	…	2.7
mot	0.5	1.5	…	…	2.4
en	0.9	3.6	..	..	2.0
vecteur					

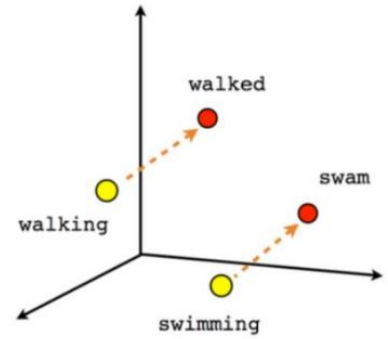
Word2vec

Semantic and Geometric Relationships





Male-Female



Verb tense

Word2vec

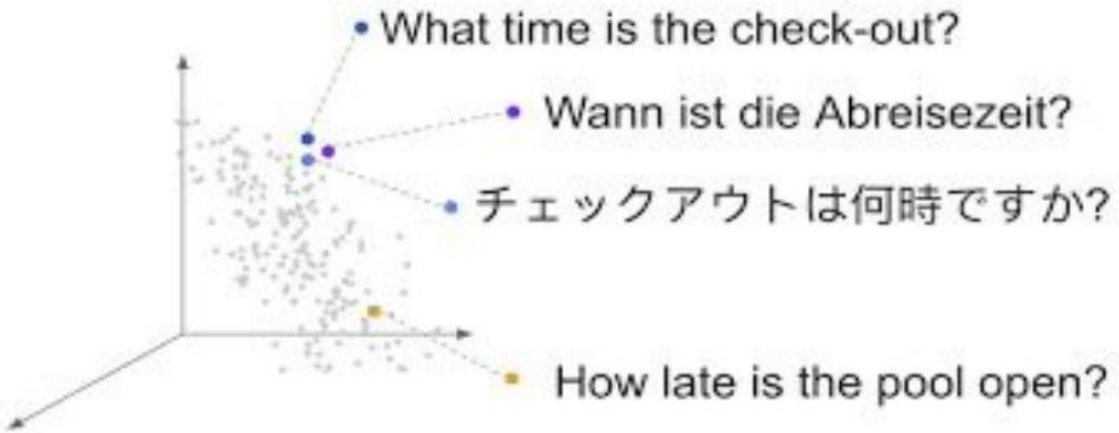
Vector Arithmetic and Semantics

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

- ✓ $\overrightarrow{\text{France}} - \overrightarrow{\text{Paris}} \approx \overrightarrow{\text{Germany}} - \overrightarrow{\text{Berlin}}$
- ✓ $\overrightarrow{\text{Euro}} - \overrightarrow{\text{France}} + \overrightarrow{\text{India}} \approx \overrightarrow{\text{Rupee}}$
- ✓ $\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}} \approx ?$

Word2vec

Multilingual concept similarity



Word2vec

Pretrained models

- **Word2Vec** (French, English, German)
  - Google News corpus → 3B mots
  - Wikipedia français → 500M mots
  - Wikipedia allemand → 651M mots
- **Glove** (English only)
  - Wikipedia 2014 + Gigaword → 6B mots
  - Twitter → 27B mots (2B tweets)
  - Common Crawl (2 models) → 840B mots & 42B mots
- **FastText** (157 languages)
  - Common Crawl
  - Wikipedia

```
1 import gensim.downloader as api
2
3 # télécharger un modèle
4 word2vec_model = api.load('word2vec-google-news-300')
```

```
1 # utiliser le modèle
2 # calculer la similarité
3 word2vec_model.similarity('Ahmed','Hamdi')
```

```
]: 0.5047506
```

```
1 # les mots les plus similaires
2 word2vec_model.most_similar('Ahmed', topn=3)
```

```
]: [('Mohammed', 0.8453757762908936),
 ('Ibrahim', 0.8363413214683533),
 ('Mohamed', 0.8346195220947266)]
```

```
1 # trouver l'intrus de point de vue sémantique
2 word2vec_model.doesnt_match("Ahmed Marwa Ronan Noura Mourad Yacine".split())
```

```
]: 'Ronan'
```

```
1 # analogie sémantique
2 word2vec_model.most_similar(positive=['euro','India'], negative=['France'], topn=1)
```

```
]: [('rupee', 0.680260181427002)]
```

```
1 # le vecteur d'un mot
2 word2vec_model['Ahmed']
```

```
]: array([-0.15917969,  0.4609375 ,  0.14746094, -0.05615234,  0.06933594,
         0.06494141, -0.09521484, -0.18652344, -0.22363281,  0.05371094,
        -0.17871094, -0.04150391, -0.20898438,  0.03149414, -0.24023438,
         0.11474609,  0.06103516, -0.22949219,  0.10449219,  0.27929688,
        -0.21875   , -0.10498047,  0.00402832, -0.05053711,  0.18457031,
```

Document embeddings

Doc2vec

- Doc2vec (Mikolov et al.2013)
- Vector representation model for:
  - ✓ Phrases
  - ✓ Paragraphs
  - ✓ Documents

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

Abstract

The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. In this paper we present several extensions that improve both the quality of the vectors and the training speed. By subsampling of the frequent words we obtain significant speedup and also learn more regular word representations. We also describe a simple alternative to the hierarchical softmax called negative sampling.

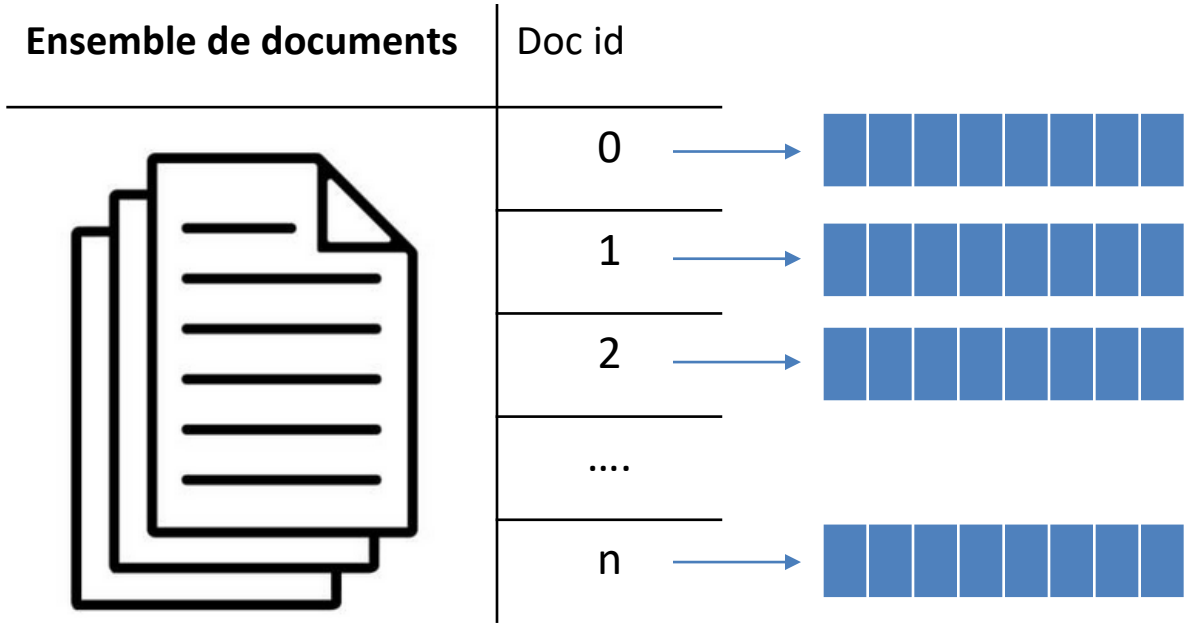
An inherent limitation of word representations is their indifference to word order and their inability to represent idiomatic phrases. For example, the meanings of "Canada" and "Air" cannot be easily combined to obtain "Air Canada". Motivated by this example, we present a simple method for finding phrases in text, and show that learning good vector representations for millions of phrases is possible.

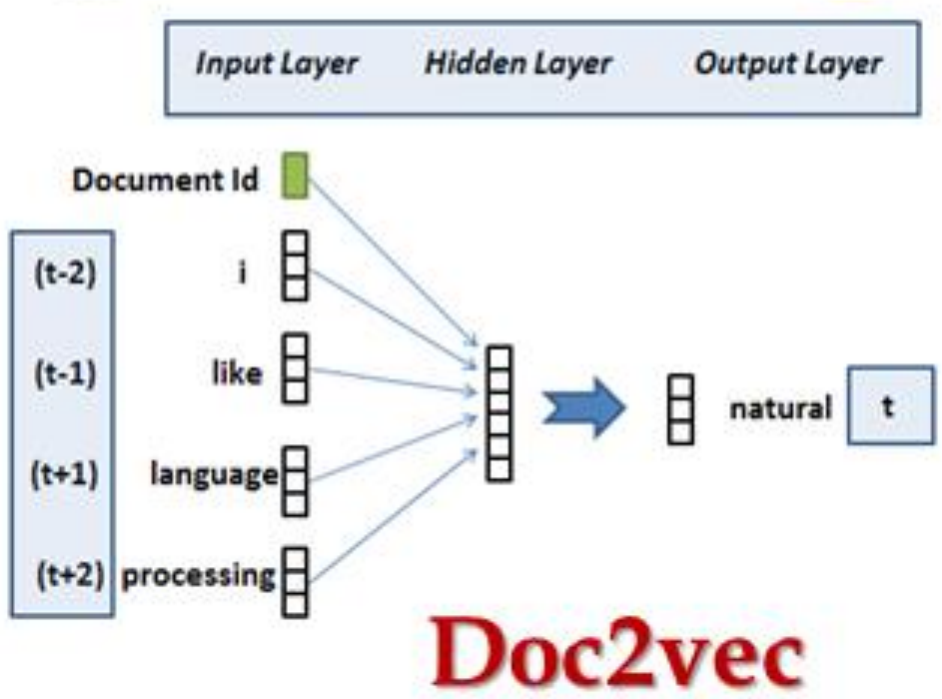
1 Introduction

Distributed representations of words in a vector space help learning algorithms to achieve better performance in natural language processing tasks by grouping similar words. One of the earliest use of word representations dates back to 1986 due to Rumelhart, Hinton, and Williams [13]. This idea has since been applied to statistical language modeling with considerable success [1]. The follow up work includes applications to automatic speech recognition and machine translation [14, 7], and

Document embeddings

Doc2vec

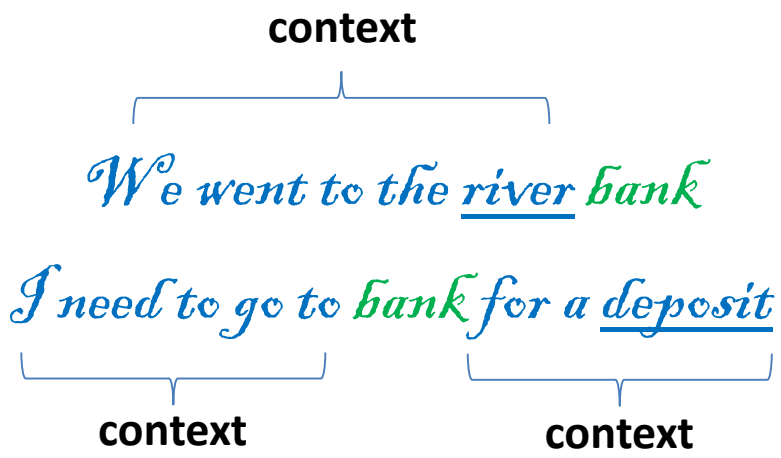




Limitations

Ambiguity

Word2vec assigns one vector for each word that encodes all contexts ➔ Static embeddings



Dynamic embeddings will be able to distinguish and capture **the two different semantic meanings** by producing **two different vectors** for **the same word** bank

OCR errors

OCRed words are unknown words (OOVs)

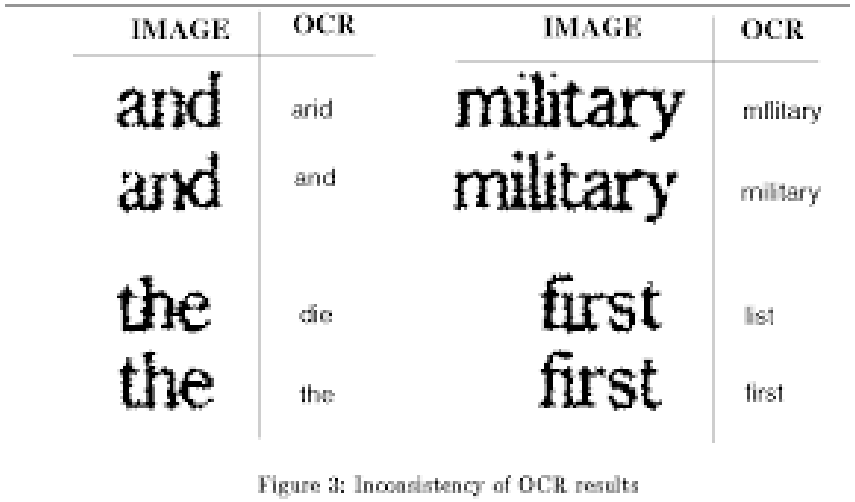


IMAGE	OCR	IMAGE	OCR
and	arid	military	millitary
and	and	military	military
the	die	first	list
the	the	first	first

Figure 3: Inconsistency of OCR results

Subword embeddings proposes vector representations for parts of words. The unknown words are tokenized into parts with known embeddings.

Dynamic embeddings

Language Models

Languages models compute word embeddings using deep neural networks.

	Word2vec	ELMo	FLAIR
Embedding	Static	Dynamic	
Sub-words	No	Character	Sub-words
Source	Wikipedia, news	Web, livres	Tout type
Corpus size	Billion	Tens of billions	
Architecture	Shallow	RNN	BiLSTM
Context	2 - 10	Variable	
Vector size	50 - 300	1024	4096
Use	Initial features	Fine-tuned	

Large Language Models

BERT

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

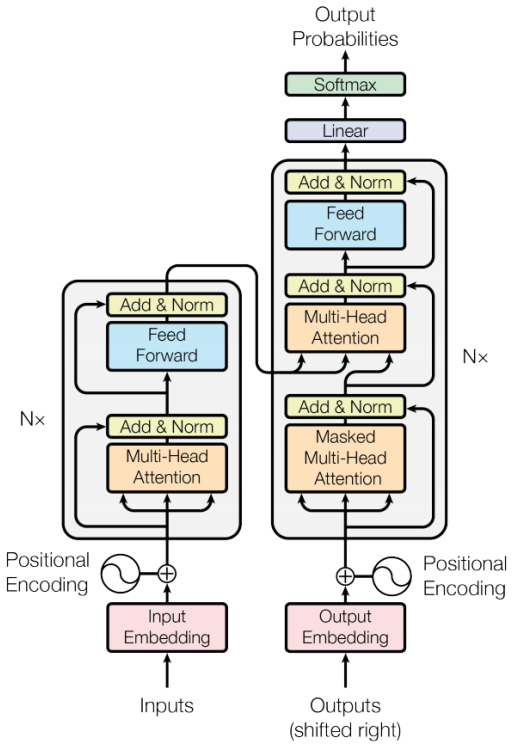
Abstract

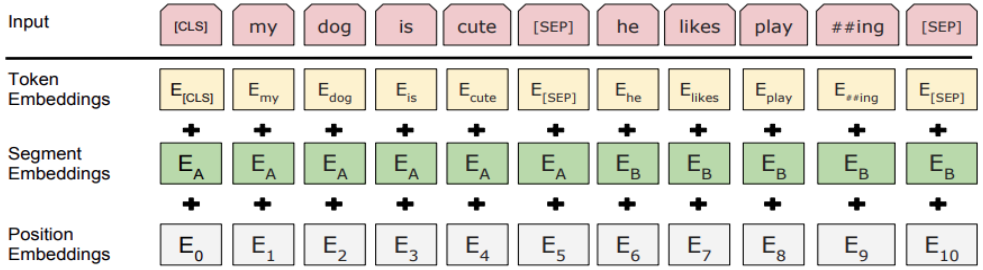
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

1 Introduction

Recurrent neural networks, long short-term memory [12] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [29, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder

Transformers



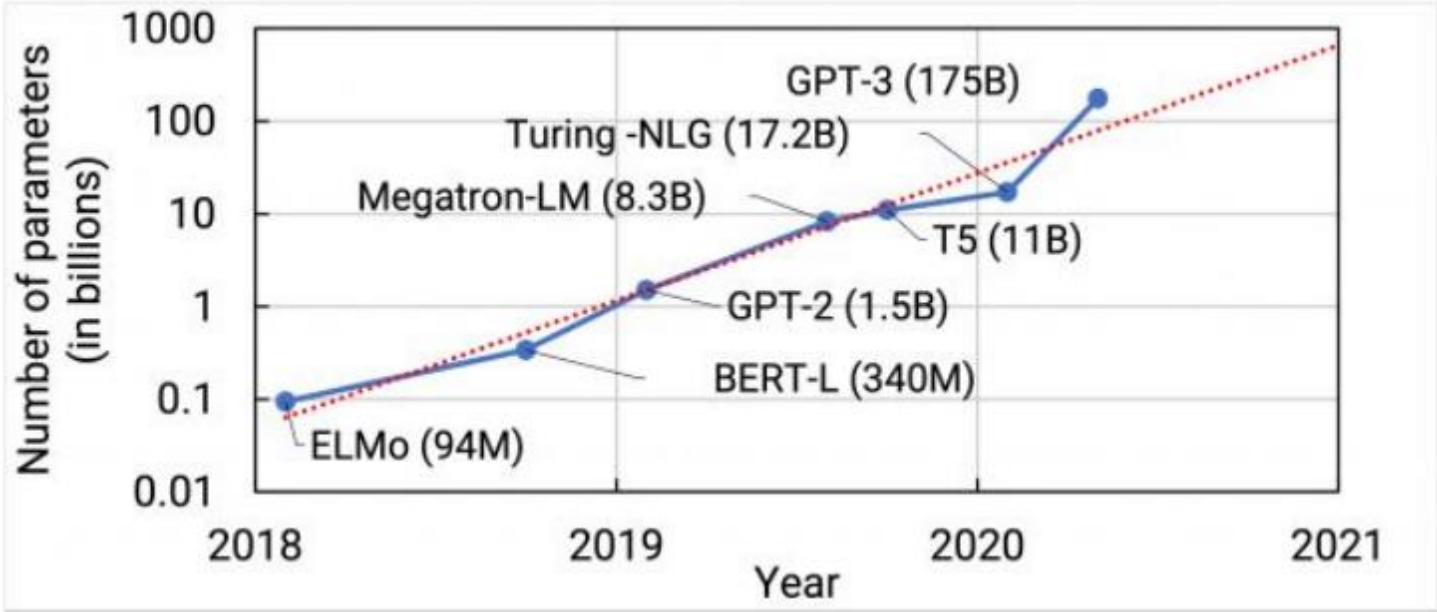


Pretrained models

Language	Pretrained model
English	BERT-base, BERT-large
	RoBERTa
	AlBERT
	GPT-2
French	CamemBERT
	FlauBERT
Arabic	AraBERT
	AraELECTRA
Multi-lingual	M-Bert

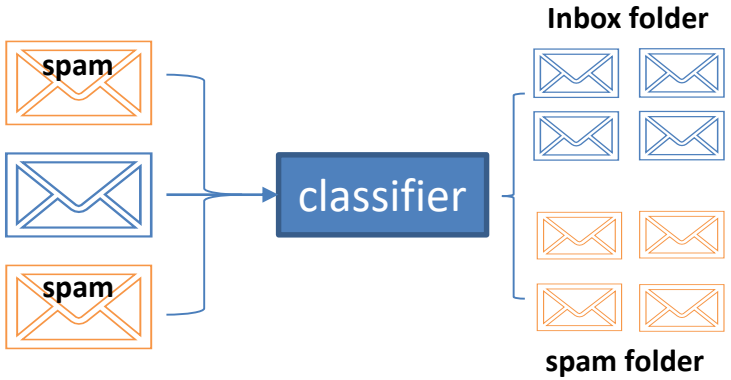
Large language models

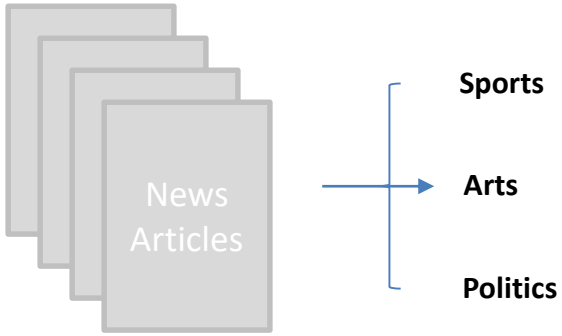
➔ Large corpora
➔ Large context
➔ Large number of parameters



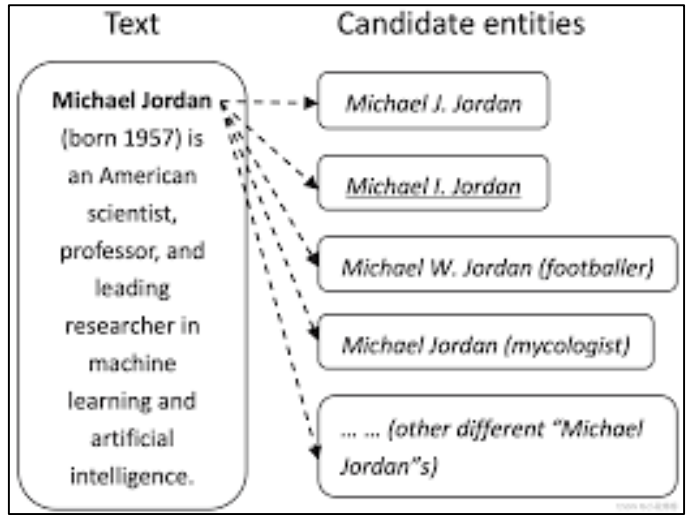
Word embedding
NLP Tasks

TEXT CLASSIFICATION





TEXT MINING



The room was great
but the staff was unfreindly

Sentiment analysis → **Negative sentiment**

Opinion mining →

Subject	Opinion	Sentiment
Room	Great	Positive
Staff	Unfreindly	Negative

Embedding for downstream tasks

Question answering

BUT NOT ONLY NLP, even for image processing

SO, HOW CAN WE TAKE ADVANTAGE FROM **EMBEDDING** IN **PROCESS MINING**?

D’ici, on voit + loin !

La Rochelle Université

univ-larochelle.fr