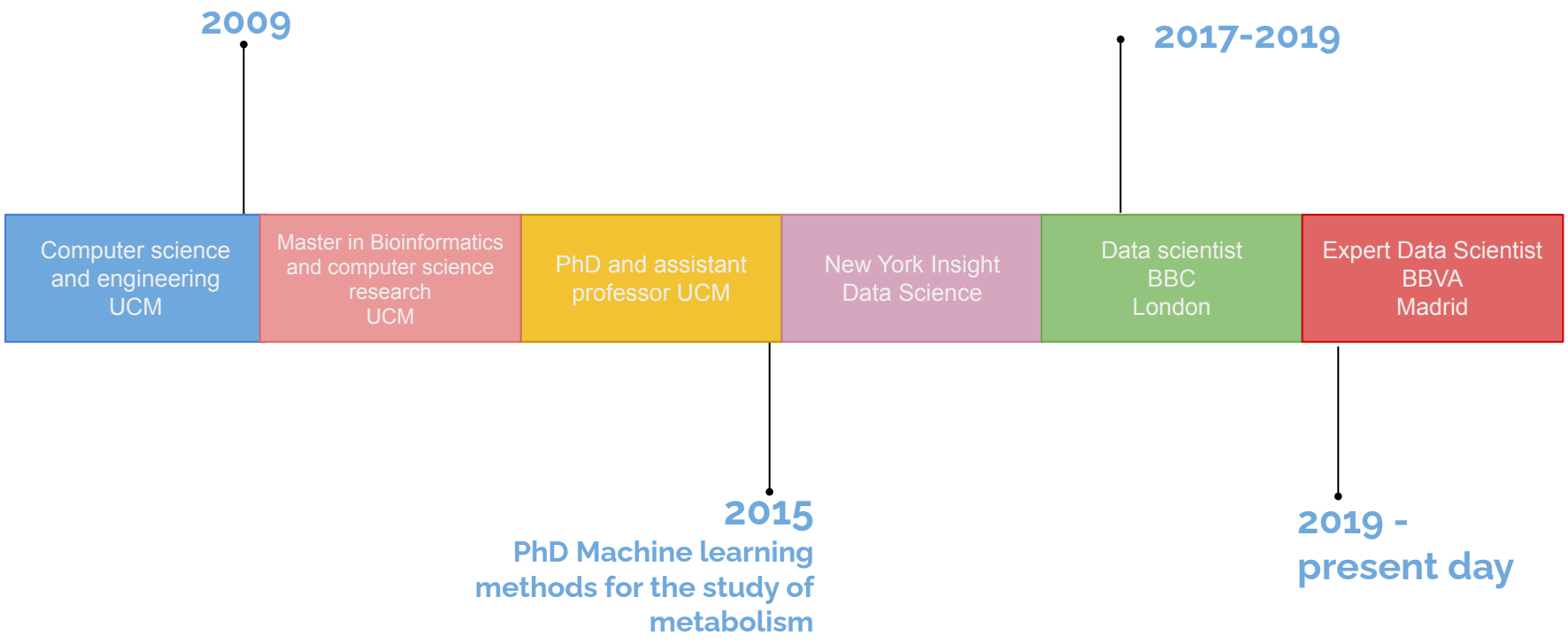
Introduction to Natural language processing

Session 1

Clara Higuera Cabañes, PhD
Barcelona Technology School, May 2021

About me


2009
2017-2019
Computer science and engineering UCM
Master in Bioinformatics and computer science research UCM
PhD and assistant professor UCM
New York Insight Data Science
Data scientist BBC London
Expert Data Scientist BBVA Madrid
2015
PhD Machine learning methods for the study of metabolism
2019 - present day

Class philosophy

Index

1. What is NLP?
2. NLP pipeline
    a. Data Acquisition
    b. Cleaning and preprocessing
    c. Text representation / Feature engineering

Break (9:50, 10min)

3. Hands on / live showcase nlp in action (20-30min)
4. NLP pipeline (30 min)
    a. Modeling
    b. Evaluation of supervised learning problems

Break 10 mins

Quiz (15 min, 10-15 mins review collaboratively)

5. Hands on / live showcase nlp in action (30 min)
6. NLP real use case
    a. Manual annotation

What is Natural Language Processing (NLP)?

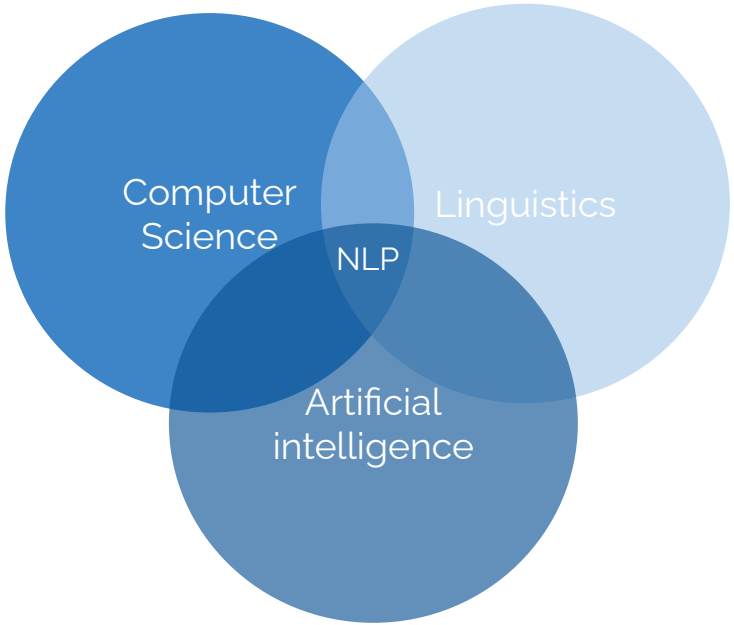
John: “How is the weather today?”

Digital assistant: “It is 37 degrees centigrade outside with no rain today.”

John: “What does my schedule look like?”

Digital assistant: “You have a strategy meeting at 4 p.m. and an all-hands at 5:30 p.m. Based on today’s traffic situation, it is recommended you leave for the office by 8:15 a.m.”

What is Natural Language Processing?



It concerns building systems that can process and understand human language.

Sectors: retail, healthcare, finance, media, law, marketing, human resources, and many more.

Applications of NLP

Customer service

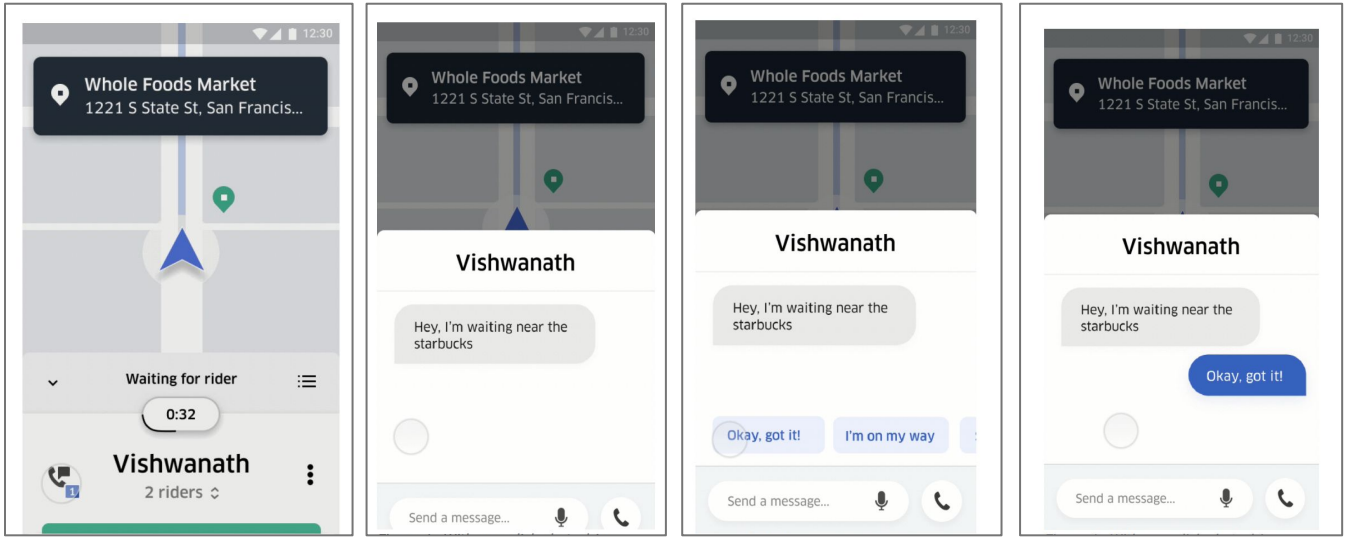
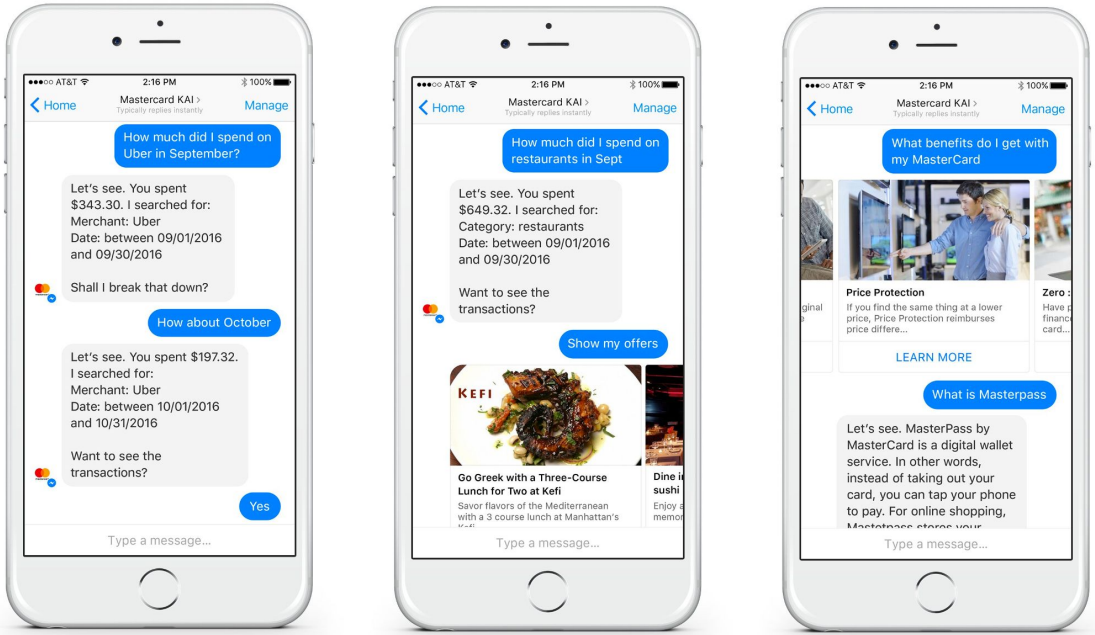


Figure 1: With one-click chat, driver-partners can more easily respond to rider messages.

Uber one-click-chat

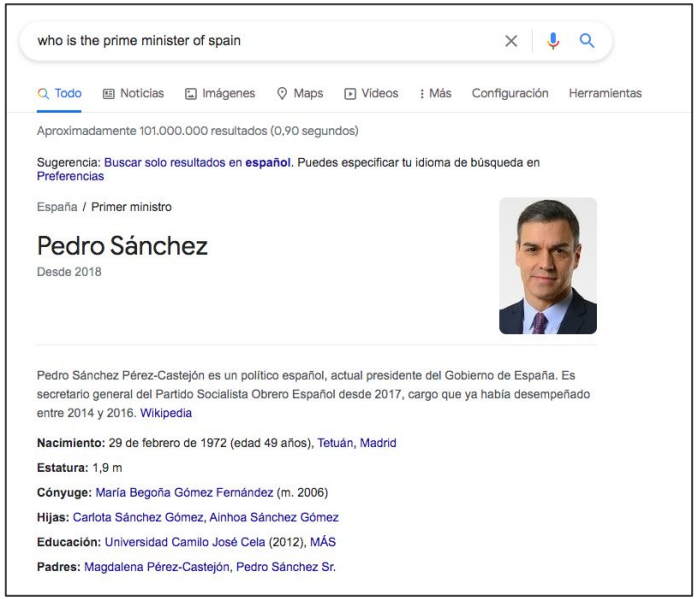
Applications of NLP

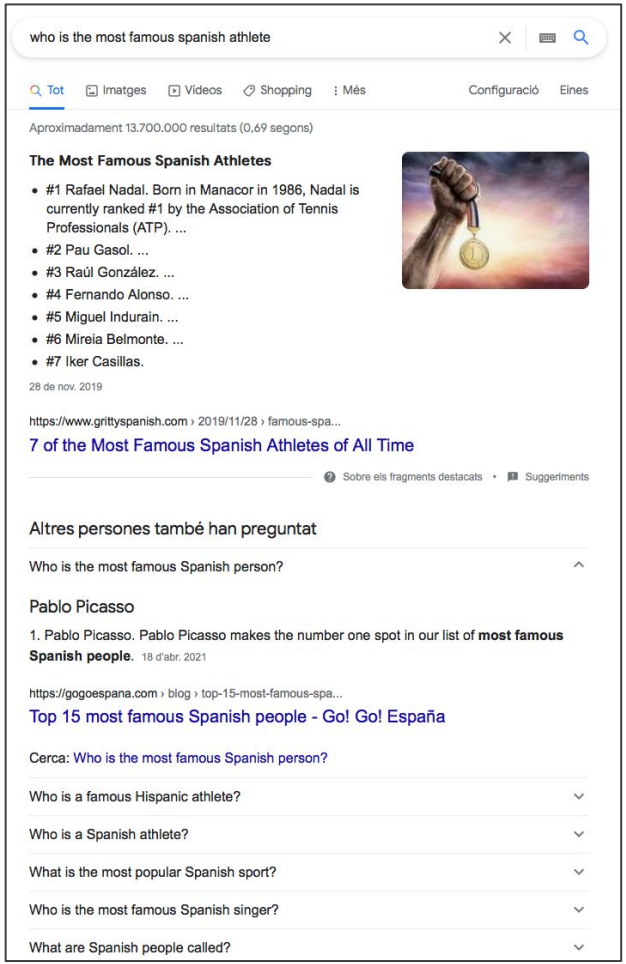
Customer service



Applications of NLP

Question answering

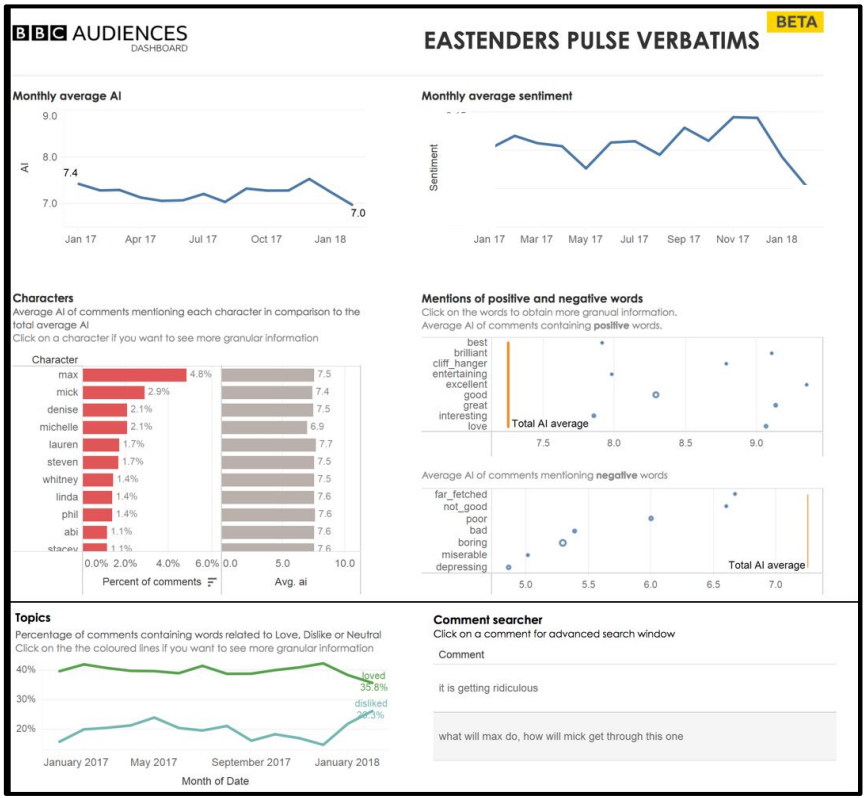




Applications of NLP

Media

BBC



Applications of NLP

Media

You might be interested as well in:

Content similarity recommenders

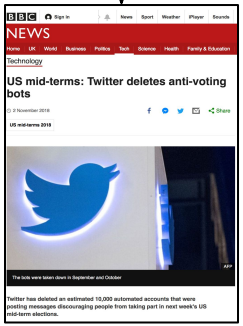
Applications of NLP

Media

You might be interested as well in:

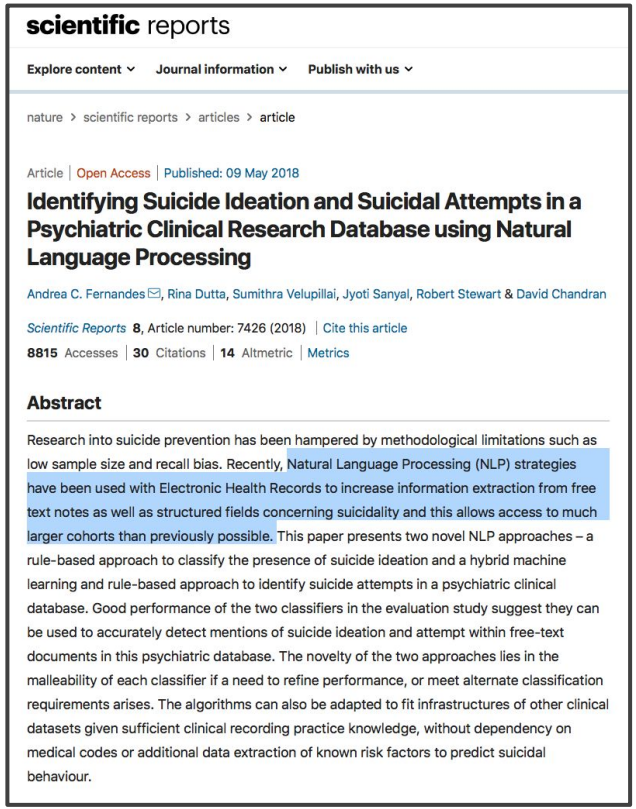
Content similarity recommenders

Which article is more similar?



Applications of NLP

Medicine and social science



scientific reports

Explore content | Journal information | Publish with us

nature › scientific reports › articles › article

Article | Open Access | Published: 09 May 2018

Identifying Suicide Ideation and Suicidal Attempts in a Psychiatric Clinical Research Database using Natural Language Processing

Andrea C. Fernandes, Rina Dutta, Sumithra Velupillai, Jyoti Sanyal, Robert Stewart & David Chandran

Scientific Reports **8**, Article number: 7426 (2018) | Cite this article

8815 Accesses | **30** Citations | **14** Altmetric | Metrics

Abstract

Research into suicide prevention has been hampered by methodological limitations such as low sample size and recall bias. Recently, Natural Language Processing (NLP) strategies have been used with Electronic Health Records to increase information extraction from free text notes as well as structured fields concerning suicidality and this allows access to much larger cohorts than previously possible. This paper presents two novel NLP approaches – a rule-based approach to classify the presence of suicide ideation and a hybrid machine learning and rule-based approach to identify suicide attempts in a psychiatric clinical database. Good performance of the two classifiers in the evaluation study suggest they can be used to accurately detect mentions of suicide ideation and attempt within free-text documents in this psychiatric database. The novelty of the two approaches lies in the malleability of each classifier if a need to refine performance, or meet alternate classification requirements arises. The algorithms can also be adapted to fit infrastructures of other clinical datasets given sufficient clinical recording practice knowledge, without dependency on medical codes or additional data extraction of known risk factors to predict suicidal behaviour.

Applications of NLP

Reducing response times to citizen legal questions across Africa

barefootlaw

[Data Science for Social Good. Chicago University]

Understanding Text Data to Help Disadvantaged Families

Understanding Text Data to Help Disadvantaged Families

[DataKind UK]

https://www.meetup.com/DataForGoodBCN/

Government

[Using AI to unleash the power of unstructured government data]

Improving forensics investigations

A growing number of government agencies are using NLP-based solutions to improve investigations in critical areas such as law enforcement, defense, and intelligence. The DoD's DEFT program referenced above uses NLP to uncover connections implicit in large text documents. Its objective is to improve the efficiency of defense analysts who investigate multiple documents to detect anomalies and causal relationships.[36]

The European Union's Horizon 2020 program launched an initiative called RED (Real-time Early Detection) Alert, aimed at countering terrorism by using NLP to monitor and analyze social media conversations. RED Alert is designed to provide early alerts of potential propaganda and signs of warfare by identifying online content posted by extremists. To comply with the General Data Protection Regulation (GDPR), this analysis uses homomorphic encryption, a method that allows mathematical operations to be performed on encrypted text without disturbing the original encryption.[37]

Enhancing policy analysis

The World Bank's Poverty and Equity Global Practice Group used LDA topic modeling to measure changes in policy priorities by examining presidential speeches in 10 Latin American countries and Spain from 1819 to 2016. Using LDA, the authors could identify the main topics for each document and indicate the variation in their significance across countries and over time. In Peru, for instance, topics on infrastructure and public services diminished in importance over time. With the help of topic modeling, the authors were able to establish, for each nation, a negative correlation between policy volatility and long-term growth.[41]

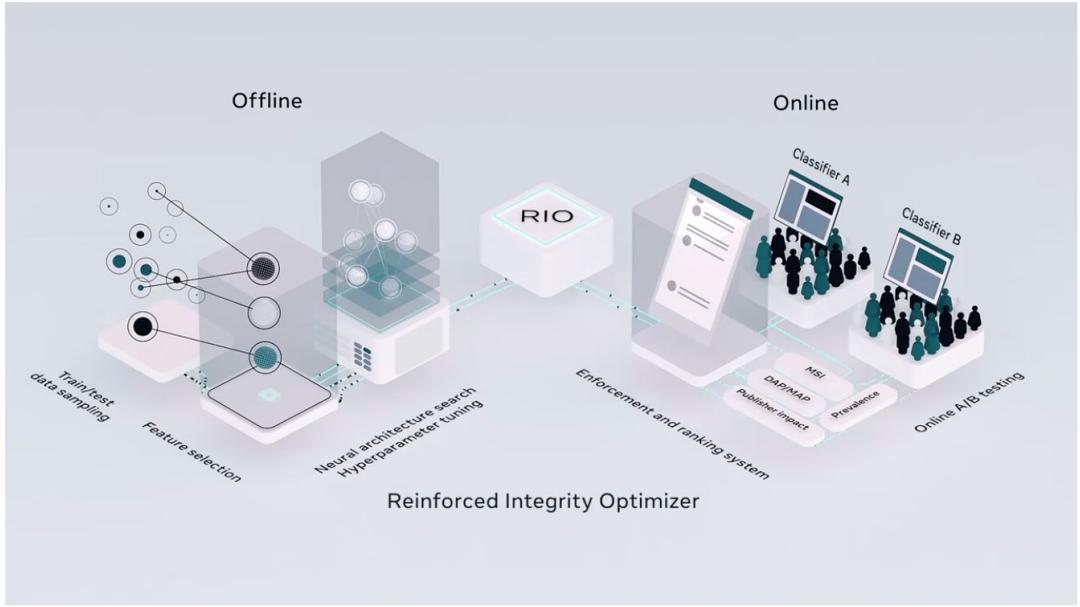
Improving predictions to aid decision-making

One of the most striking characteristics of NLP is its ability to facilitate better predictions, which can help agencies design preemptive measures. The police department of Durham, North Carolina, uses NLP in crimefighting by enabling the police to observe patterns and interrelations in criminal activities and identify pockets with a high incidence of crime, thus allowing for quicker interventions. This contributed to a 39 percent drop in violent crime in Durham from 2007 to 2014.[38]

NLP also is being used to combat child trafficking. About 75 percent of child trafficking involves online advertisements. DARPA, in collaboration with commercial technology experts, has developed a platform that monitors and draws connections among the dubious content of online advertisements. Virginia's Fairfax County Police Department and New Orleans's Homeland Security investigations both use this advanced software to identify high-risk web advertisements and detect code words used by traffickers.[39]

Applications of NLP

Hate Speech detection



[Facebook]

NLP is developing quickly

- **Widely available and easy-to-use NLP tools,** techniques, and APIs are now all-pervading in the industry. There has never been a better time to build quick NLP solutions.
- **Development of more interpretable and generalized approaches** has improved the baseline performance for even complex NLP tasks, such as open-domain conversational tasks and question answering, which were not practically feasible before.
- More and **more organizations**, including Google, Microsoft, and Amazon, **are investing heavily in more interactive consumer products, where language** is used as the primary medium of communication.
- **Increased availability of useful open source datasets**, along with standard benchmarks on them, has acted as a catalyst in this revolution, as opposed to being impeded by proprietary datasets only available to limited organizations and individuals.
- **The viability of NLP has moved beyond English or other major languages.** Datasets and language-specific models are being created for the less-frequently digitized languages too. A fruitful product that came out this effort was a near-perfect automatic machine translation tool available to all individuals with a smartphone.
- The **amount of data available plus the advances in deep learning** is 'opening' a new era in NLP

NLP capabilities

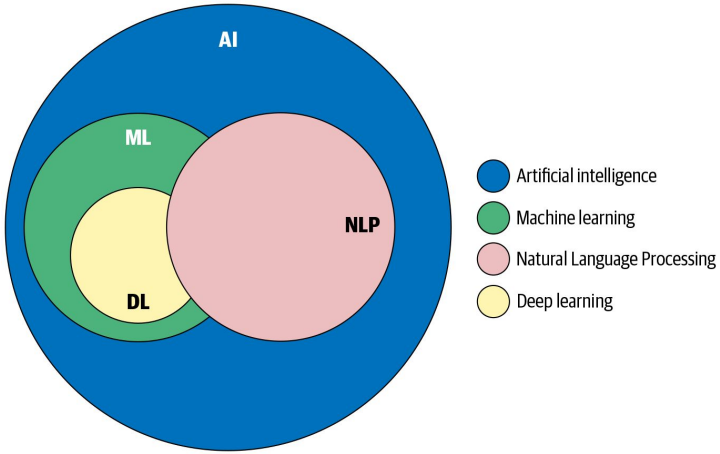
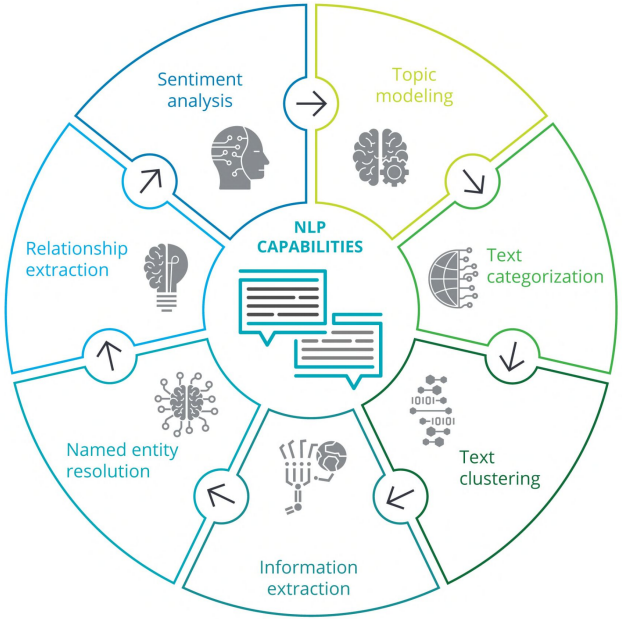


FIGURE 2

Key NLP capabilities



Source: Deloitte analysis.

Natural Language Generation

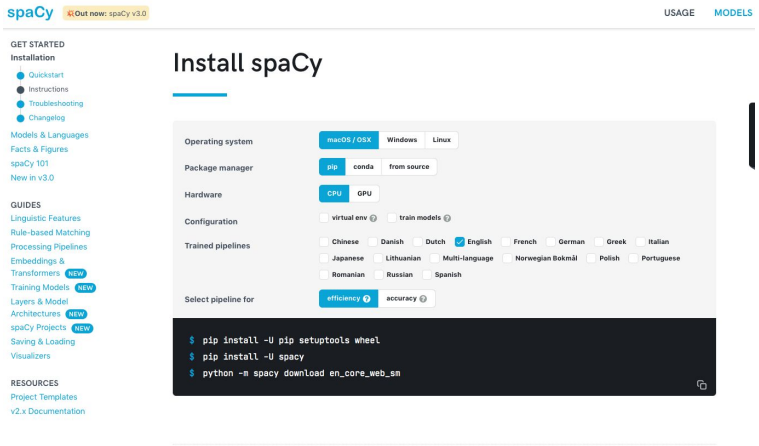
Question answering

Text Summarization

Machine Translation

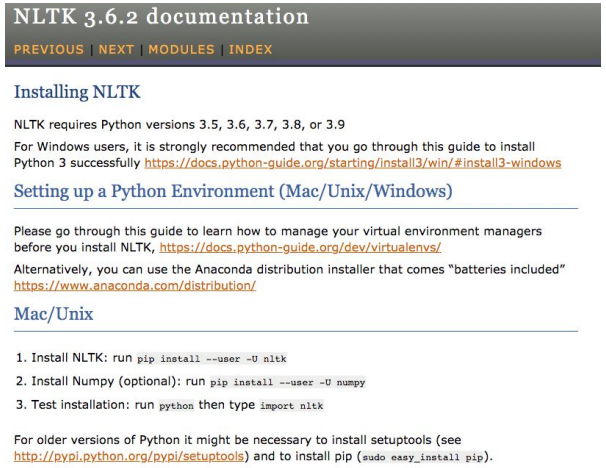
NLP Tools and Libraries

spaCy



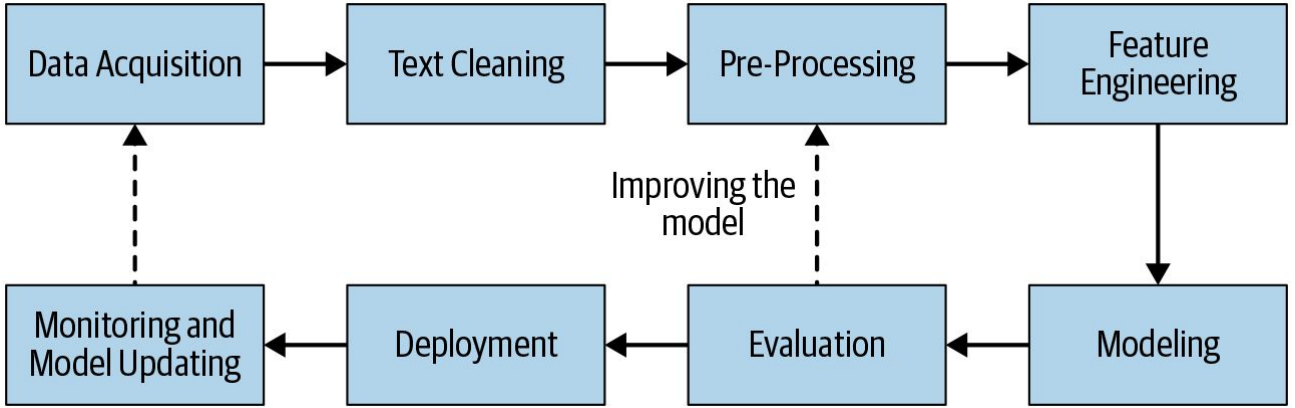
https://spacy.io/usage

NLTK



https://www.nltk.org/install.html

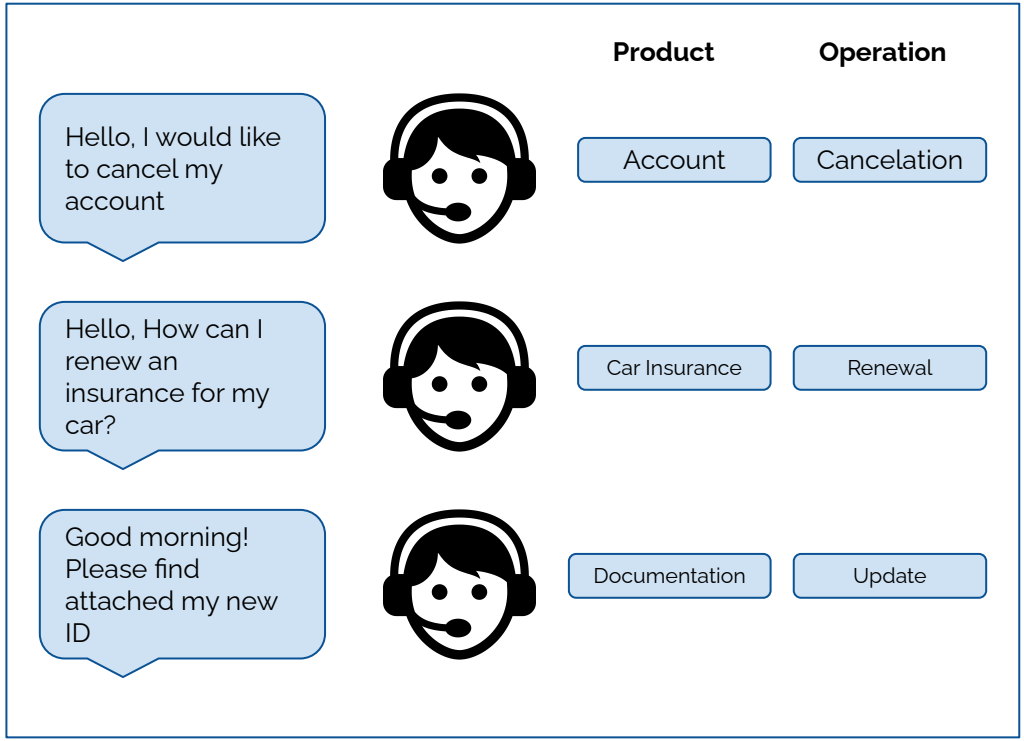
NLP Pipeline



NLP Pipeline

Data acquisition

- Use a public dataset
- Scrape data
- Product intervention / sensorization
- Data Augmentation -> refer to paper from EMNLP!
    - Back translation
    - Active learning



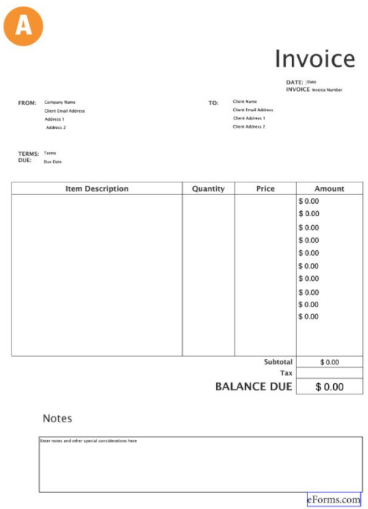
NLP Pipeline

Text extraction and cleaning

- Remove markdown (html)
- Extract text embedded with images
- Remove special characters

unidecode!

A



B

```
<p style=" text-align: justify;">
The book will be around 350 pages. It will be accompanied by a code
repository containing several Jupyter notebooks for all the chapters to
give a walk-through and explain the code in detail. The code base is in
Python and various machine learning and natural language processing
libraries. The book assumes that the readers have a good grasp of
programming but no theoretical and practical knowledge of NLP.
</p>

    <p><br>
    </p>


<a
href="https://www.oreilly.com/library/view/practical-natural-language/97
81492054047/">
<img src="book-link-1.png" alt="Book Link">
</a>

<h1 style="color: #e74c3c;">Commonly Asked Questions</h1>
<ul>
<li> Can I contribute to the book?
<p>The book is accompanied by open source Jupyter notebooks and demo
applications. If you are a great ML or front-end engineer looking to
build something meaningful you can apply by filling <a
href="https://goo.gl/forms/do6NcW251iX26ajk1">this form</a>. Also refer
to the next question.
```

```
from bs4 import BeautifulSoup
from urllib.request import urlopen
myurl = "https://stackoverflow.com/questions/415511/   \
  how-to-get-the-current-time-in-python"
html = urlopen(myurl).read()
soupified = BeautifulSoup(html, "html.parser")
question = soupified.find("div", {"class": "question"})
questiontext = question.find("div", {"class": "post-text"})
print("Question: \n", questiontext.get_text().strip())
answer = soupified.find("div", {"class": "answer"})
answertext = answer.find("div", {"class": "post-text"})


    print("Best answer: \n", answertext.get_text().strip())
```

NLP Pipeline

Text preprocessing

Depends highly on the use case

- Language detection (polyglot)
- Text normalization
  - Convert all text to lowercase or uppercase
  - Convert digits to text
  - Normalize date format
- Remove punctuation, digits
- **Remove stop words**
- Sentence segmentation
- Stemming / lemmatization
- Word tokenization
- Bigram / trigram..
- Advanced processing
  - POS tagging, entity recognition

Sentence segmentation

```
text = u"This is first sentence. Second sentence. Third sentence."
text_sentences = nlp(text)
for sentence in text_sentences.sents:
   print(sentence.text)
```

```
This is first sentence.
Second sentence.
Third sentence.
```

NLP Pipeline

Text preprocessing

Depends highly on the use case

- Language detection (polyglot)
- Text normalization
  - Convert all text to lowercase or uppercase
  - Convert digits to text
  - Normalize date format
- Remove punctuation, digits
- **Remove stop words**
- Sentence segmentation
- Stemming / lemmatization
- Word tokenization
- Bigram / trigram..
- Advanced processing
  - POS tagging, entity recognition

Stop words

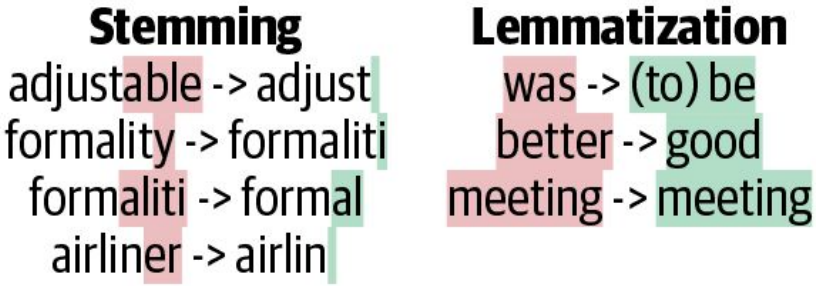
> *Generally, the most common words used in a text are "the", "is", "in", "for", "where", "when", "to", "at" etc.*

NLP Pipeline

Text preprocessing

Depends highly on the use case

- Language detection (polyglot)
- Text normalization
  - Convert all text to lowercase or uppercase
  - Convert digits to text
  - Normalize date format
- Remove punctuation, digits
- Remove stop words
- Sentence segmentation
- **Stemming / lemmatization**
- Word tokenization
- Bigram / trigram..
- Advanced processing
  - POS tagging, entity recognition




```
nlp = spacy.load( 'en_core_web_sm' )
doc = nlp(u 'I am Clara and this is the NLP class' )
#for token in doc:
#    print(token.text, token.lemma_, token.is_stop)

for token in doc:
    print(token.text, token.lemma_)
```

```
(u'I', u'-PRON-', True)
(u'am', u'be', True)
(u'Clara', u'Clara', False)
(u'and', u'and', True)
(u'this', u'this', True)
(u'is', u'be', True)
(u'the', u'the', True)
(u'NLP', u'NLP', False)
(u'class', u'class', False)
```

NLP Pipeline

Text preprocessing

Depends highly on the use case

- Language detection (polyglot)
- Text normalization
  - Convert all text to lowercase or uppercase
  - Convert digits to text
  - Normalize date format
- Remove punctuation, digits
- Remove stop words
- Sentence segmentation
- Stemming / lemmatization
- Word tokenization
- Bigram / trigram..
- **Advanced processing**
  - POS tagging, entity recognition

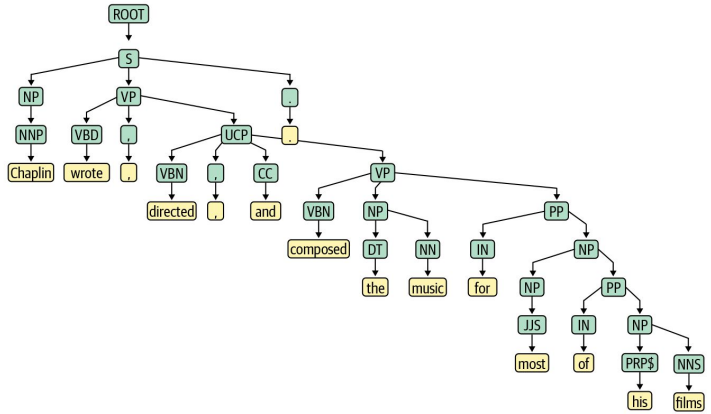
Input

Chaplin wrote, directed, and composed the music for most of his films.

Tokenization with Lemmatization

POS Tagging

Parse Tree



NLP Pipeline

Text preprocessing

Depends highly on the use case

- Language detection (polyglot)
- Text normalization
  - Convert all text to lowercase or uppercase
  - Convert digits to text
  - Normalize date format
- Remove punctuation, digits
- Remove stop words
- Sentence segmentation
- Stemming / lemmatization
- Word tokenization
- Bigram / trigram..
- **Advanced processing**
  - POS tagging, entity recognition

```
nlp = spacy.load('en_core_web_sm')

doc = nlp(u'I am Clara and this is the NLP class')

for token in doc:
    print(token.text, token.lemma_, token.is_stop, token.pos_)
```

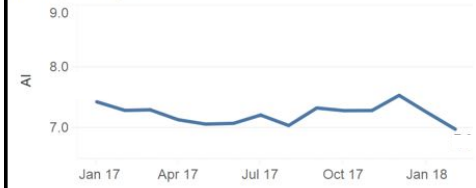
```
(u'I', u'-PRON-', True, u'PRON')
(u'am', u'be', True, u'VERB')
(u'Clara', u'Clara', False, u'PROPN')
(u'and', u'and', True, u'CCONJ')
(u'this', u'this', True, u'DET')
(u'is', u'be', True, u'VERB')
(u'the', u'the', True, u'DET')
(u'NLP', u'NLP', False, u'PROPN')
(u'class', u'class', False, u'NOUN')
```

BBC AUDIENCES DASHBOARD

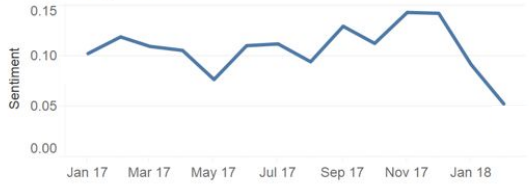
EASTENDERS PULSE VERBATIMS

BETA

Monthly average AI



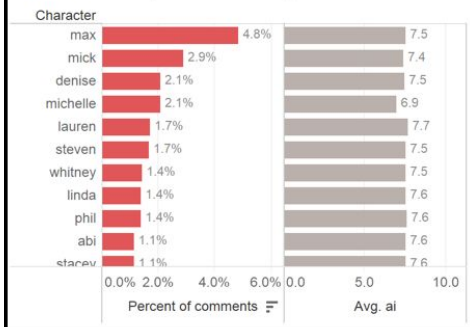
Monthly average sentiment



Characters

Average AI of comments mentioning each character in comparison to the total average AI

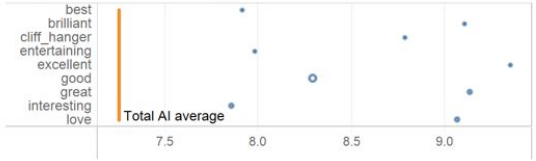
Click on a character if you want to see more granular information



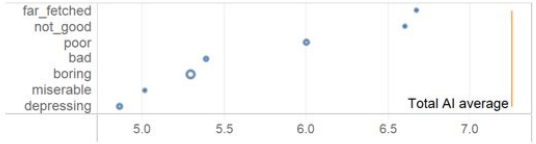
Mentions of positive and negative words

Click on the words to obtain more granual information.

Average AI of comments containing **positive** words.



Average AI of comments mentioning **negative** words



Topics

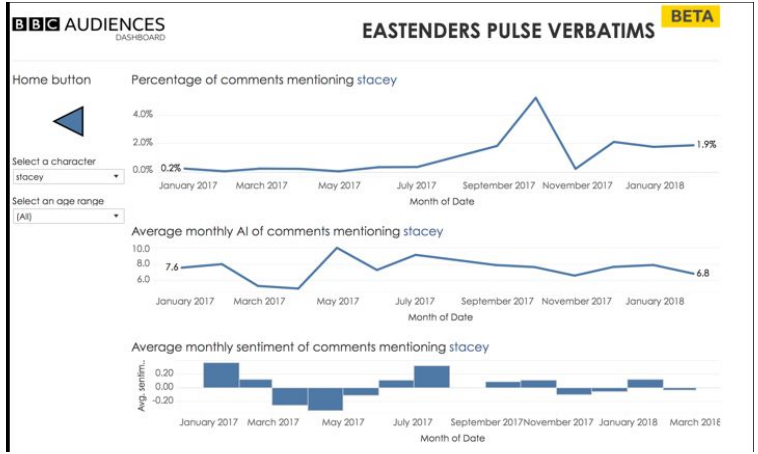
Percentage of comments containing words related to Love, Dislike or Neutral

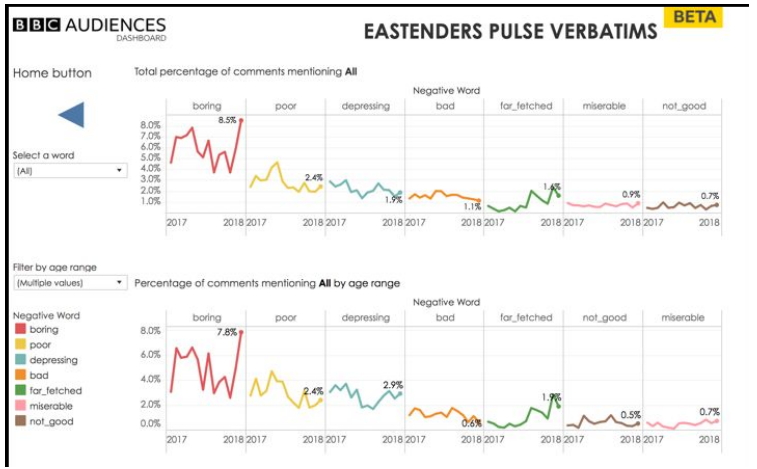
Click on the the coloured lines if you want to see more granular information

Comment searcher

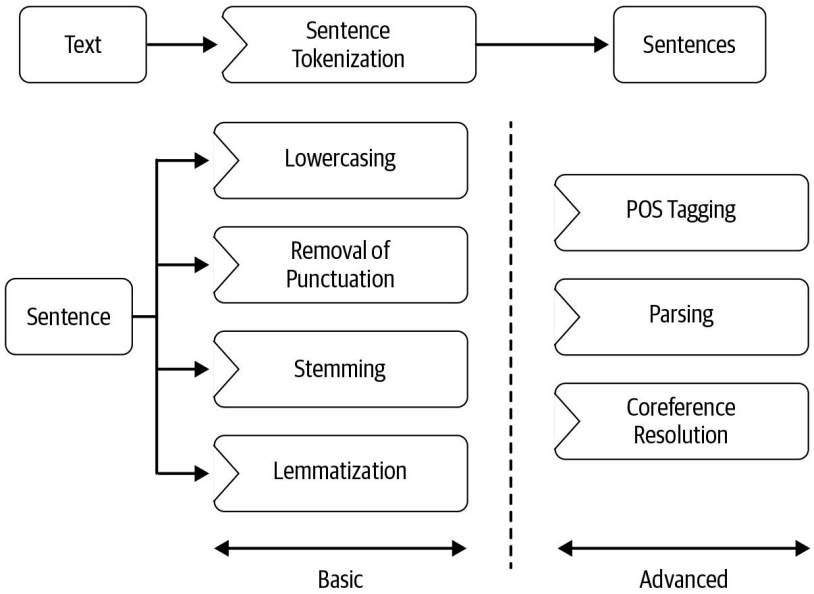
Click on a comment for advanced search window

Comment
it is getting ridiculous
what will max do, how will mick get through this one



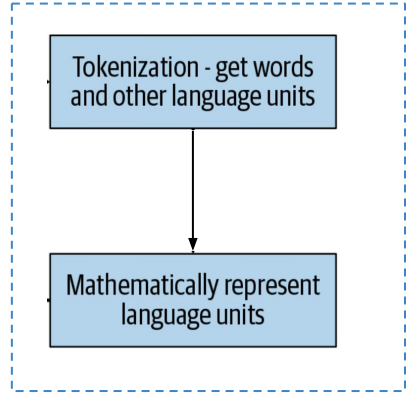


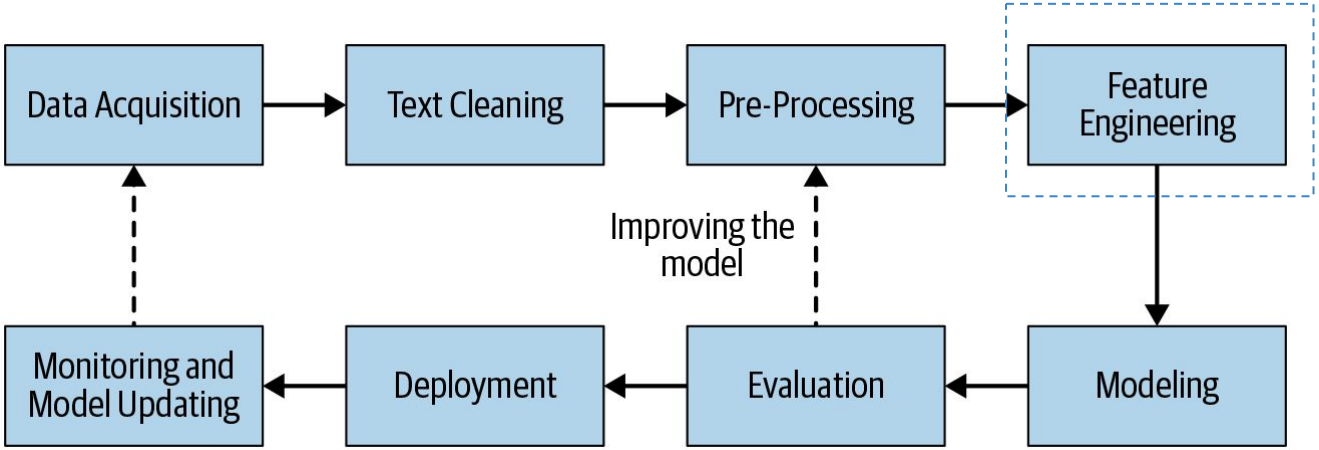
NLP Pipeline



NLP Pipeline

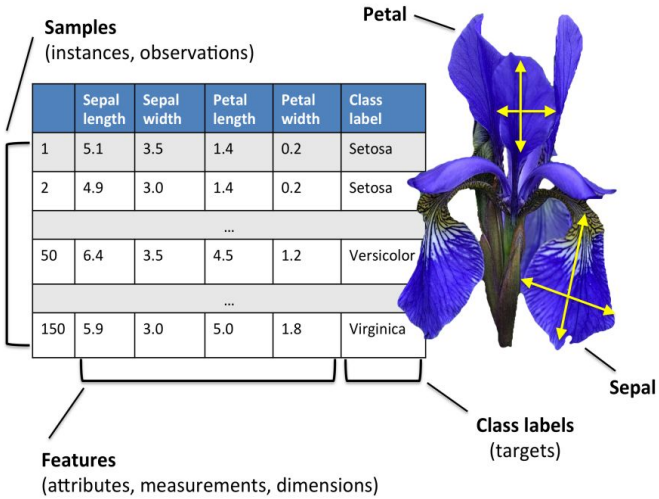
Feature engineering or text representation





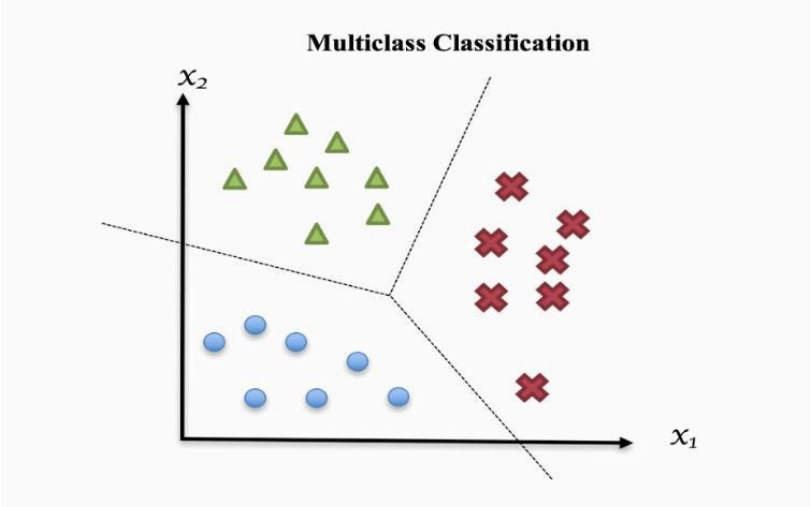
NLP Pipeline

Feature engineering or text representation

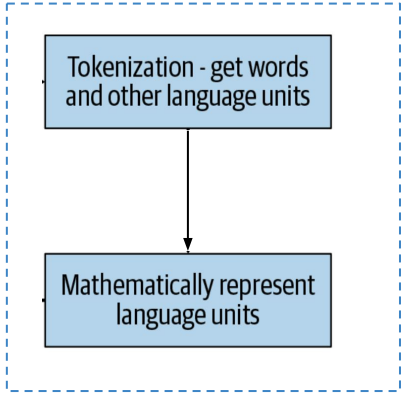


	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Classical input dataset for a ML algorithm



How do we train a model with text?
How do we translate text into a mathematical representation?



Bag of words approach (BoW)

John likes to watch movies. Mary likes movies too.

Mary also likes to watch football games.


```
{"John":1,"likes":3,"to":2,"watch":2,"movies":2,"Mary":2,"too":1,"also":1,"football":1,"games":1}
```


Revisar
binary=true

	the	red	dog	cat	eats	food
1. the red dog →	1	1	1	0	0	0
2. cat eats dog →	0	0	1	1	1	0
3. dog eats food →	0	0	1	0	1	1
4. red cat eats →	0	1	0	1	1	0

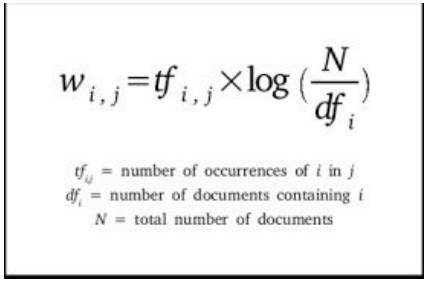
→ Vocabulary

```
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer(binary=True)
bow_rep_bin = count_vect.fit_transform(processed_docs)
temp = count_vect.transform(["the red dog"])


print("Bow representation for 'the red dog':",
temp.toarray())
```

Hands on exercise...

How do we train a model with text?
How do we translate text into a mathematical representation?

Bag of words approach (BoW) and **TF-IDF**



- Vectors instead of binary values have real values
- Penalises very frequent words

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer()
bow_rep_tfidf = tfidf.fit_transform(processed_docs)
print(tfidf.idf_) #IDF for all words in the vocabulary
print(tfidf.get_feature_names()) #All words in the vocabulary.

temp = tfidf.transform(["dog and man are friends"])


print("Tfidf representation for 'dog and man are friends'\n",
temp.toarray())
```

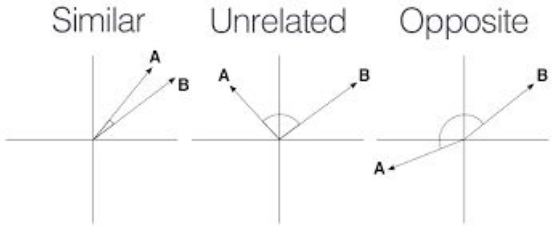
How do we train a model with text?
How do we translate text into a mathematical representation?

Bag of words approach (BoW) and **TF-IDF**

Vectorizing texts allows to make mathematical calculations with it

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Cosine similarity



How do we train a model with text?
How do we translate text into a mathematical representation?

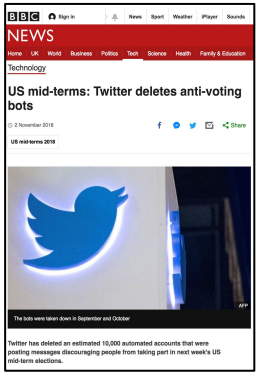
Bag of words approach (BoW) and **TF-IDF**

Vectorizing texts allows to make mathematical calculations with it

Database of articles

Article read
$(a_1,a_2a_3,a_4,..,a_n)$

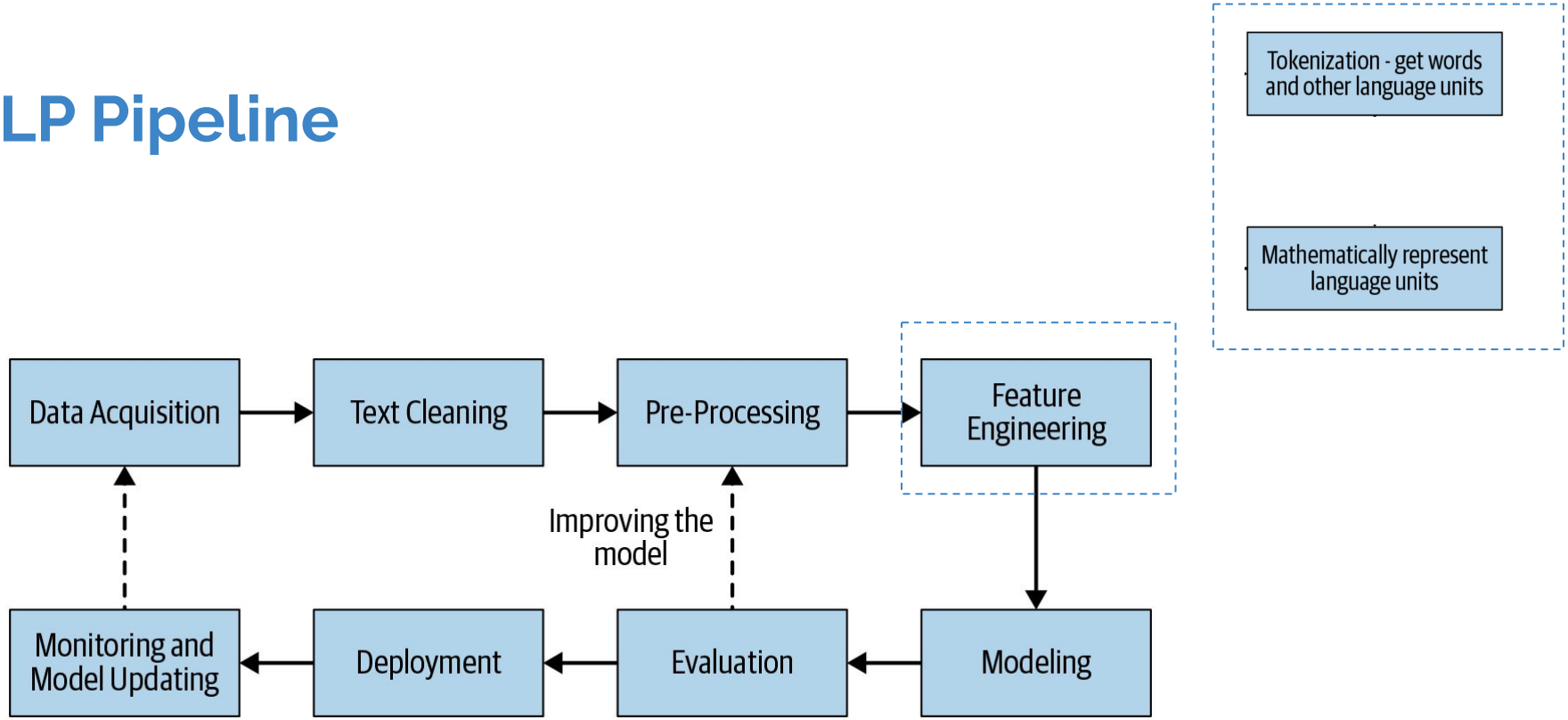
$(b_1,b_2b_3,b_4,..,b_n)$



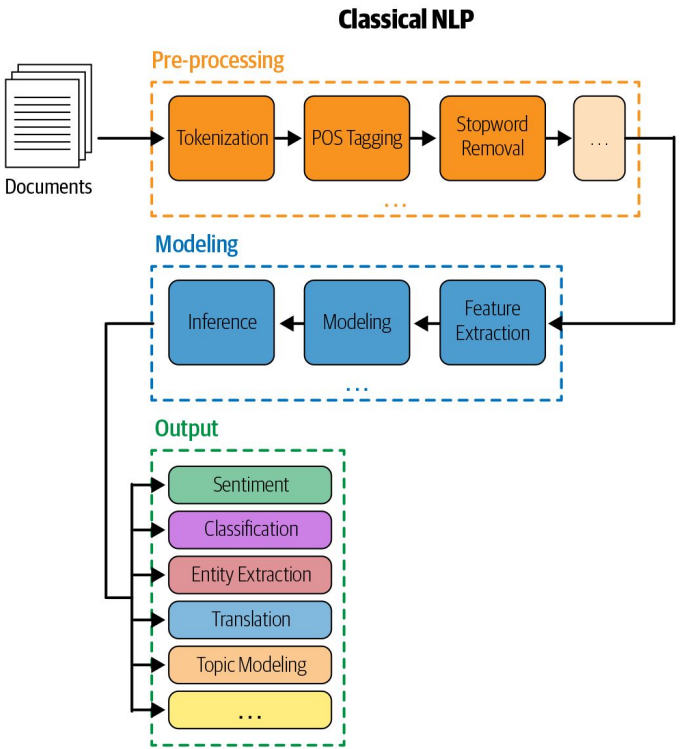
$(c_1,c_2c_3,c_4,..,c_n)$

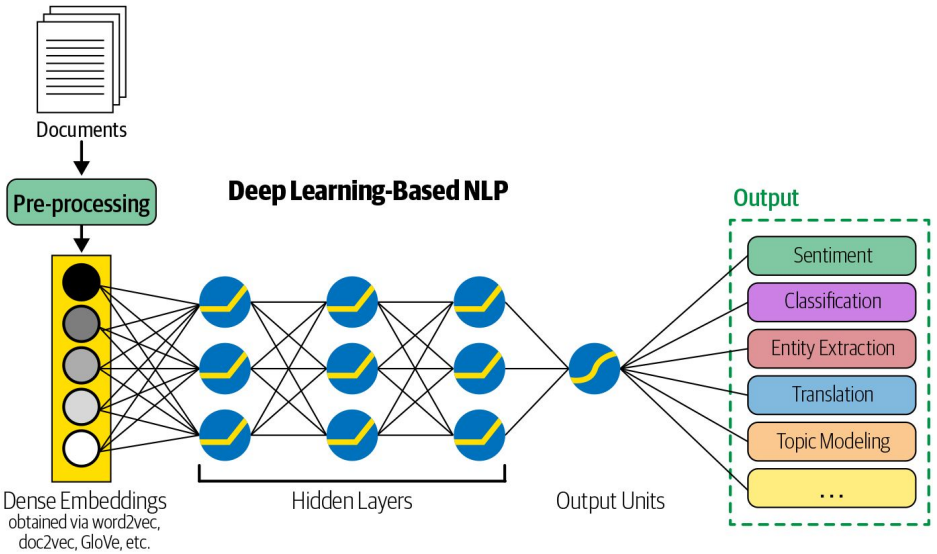
$(d_1,d_2d_3,d_4,..,d_n)$

NLP Pipeline



NLP Pipeline



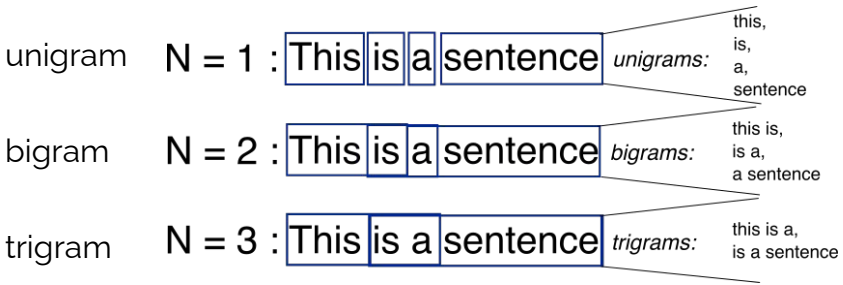


How do we train a model with text?
How do we translate text into a mathematical representation?

Bag of words approach (BoW)

+ Transforms text into vectors
+ Simplicity
- Ignores order and context
- Size of the vector increases with the size of the vocabulary
- Does not capture the similarity between words that mean the same (i.e. run and ran)

We can also use bigrams, trigrams,...



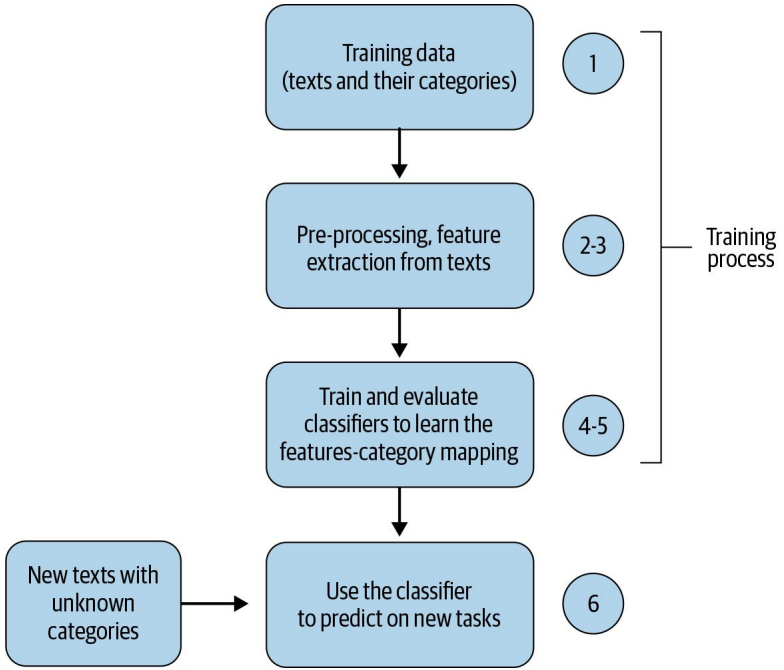

```
#n-gram vectorization example with count
vectorizer and uni, bi, trigrams

count_vect = CountVectorizer(ngram_range=(1,3))
```

Break 10 mins

NLP pipeline

Modeling

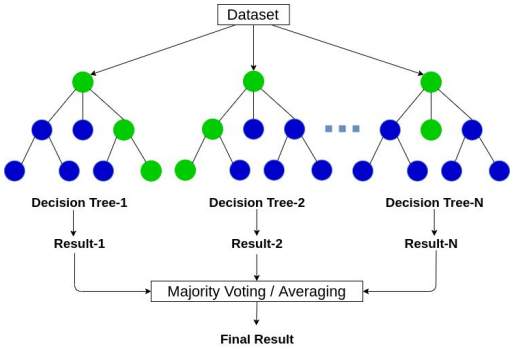
- **Types of problems**
  - Supervised learning problems
  - Unsupervised learning problems
  - Language generation problems
- Evaluation (metrics)
  - Supervised learning problems
  - Unsupervised learning problems
  - Generated language problems



NLP pipeline

Modeling

- **Types of problems**
  - Supervised learning problems - **classification**

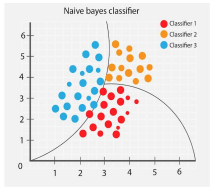


Random Forest

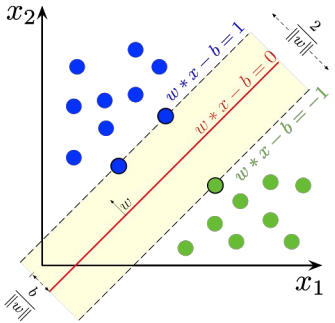
$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$



Naive Bayes



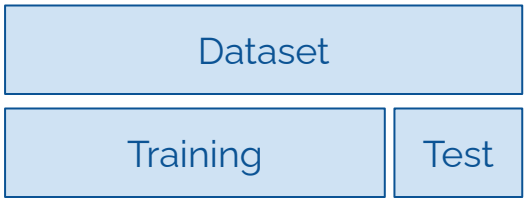
Support vector machines

Same models as in non-text problems

NLP Pipeline

Modeling

- **Evaluation**
  - Supervised learning problems



Regression	Classification
• Mean Absolute Error (MAE) • Root Mean Squared Error (RMSE) • R-Squared and Adjusted R-Squared	• Recall • Precision • F1-Score • Accuracy • Area Under the Curve (AUC)

Hands on exercise...

Quiz (15 min, 10-15 mins review collaboratively)

Introduction to Natural language processing

Session 2

Clara Higuera Cabañes, PhD
Barcelona Technology School, May 2021

Index

1. **Pending concepts from session 1 + unsupervised learning NLP (1h 30min)**
2. Unsupervised learning in NLP (30 min)
    a. Clustering
    b. Topic modeling
    c. Distance metrics
3. LDA applied
    a. Recommender system for BBC News with topic modeling
    b. Reading profile of users for segmentation
    c. Political knowledge gaps

Break 10 mins

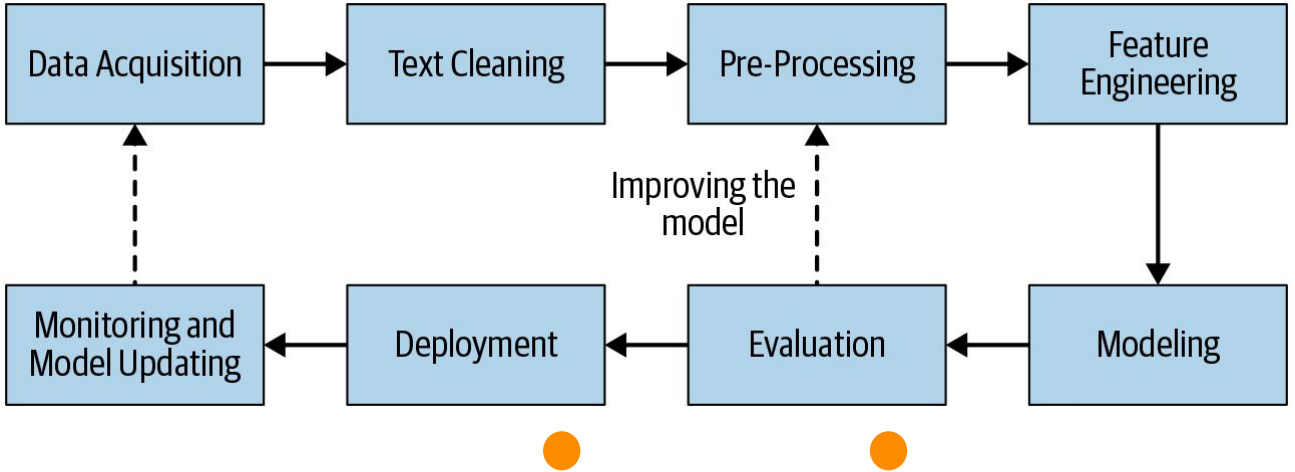
4. Hands on / live showcase nlp in action - LDA notebook (30 min)
5. **Deep learning (1h 30min)**
6. Word2vec - Embeddings (30 min)
    a. Notebook (15 min)

Break 10 min

7. Text classification with deep learning (15-20 min)
8. Natural language generation (20 min)

Quiz (20 min + 10 review)

Previously on NLP pipeline ...

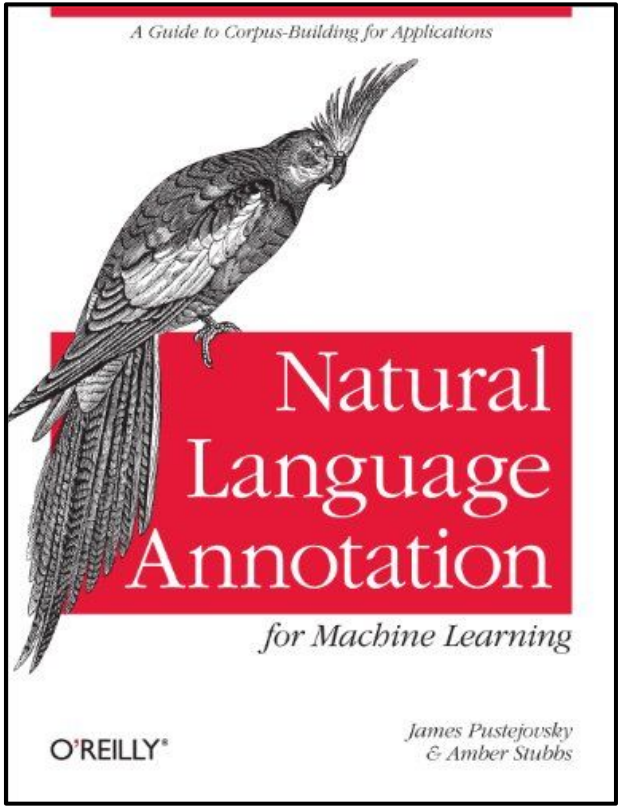


Concepts pending from session 1

NLP Pipeline

Modeling

- **Manual annotation**
  - **Needed when:**
    - Lack of data
    - NLP models validation



NLP Pipeline

Modeling

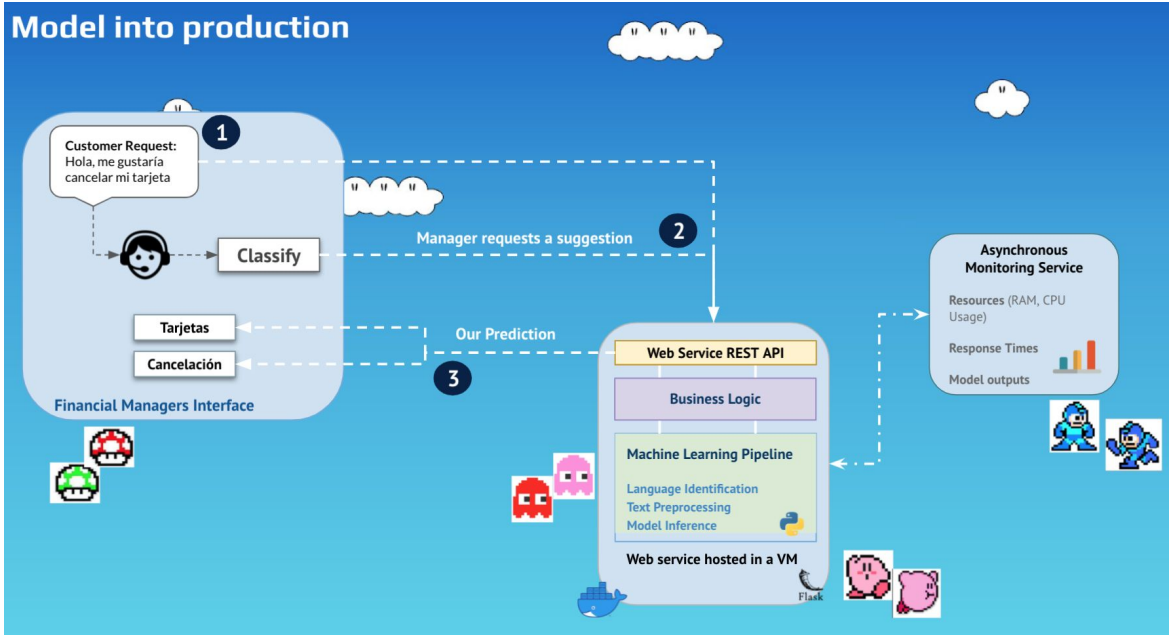
- **Manual annotation**
  - **Needed when:**
    - Lack of data
    - NLP models validation
  - **DIY / outsource**

Amazon Mechanical Turk

NLP Pipeline

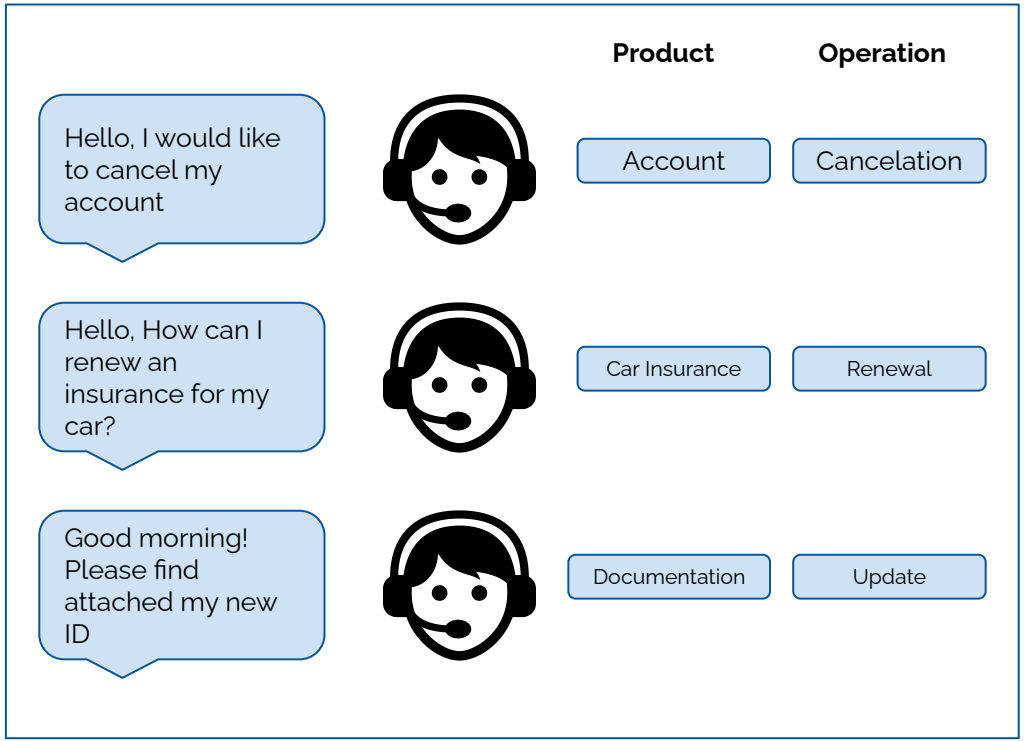
Post-modeling phases

- Deployment
- Monitoring
- Model updating



Real use case

Conversations classifier



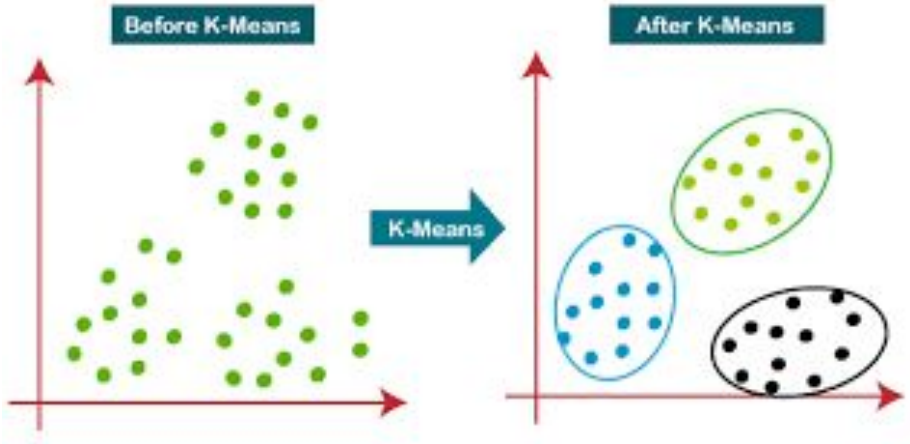
BBVA

Unsupervised learning in NLP

Goal: Find structure inherent to unlabeled data

Common techniques in NLP

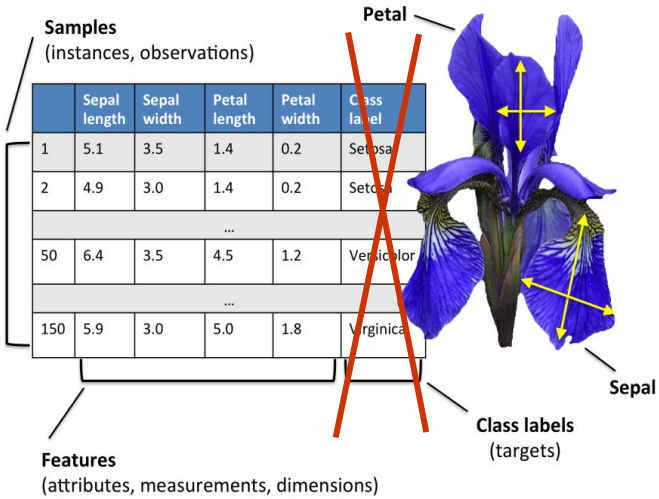
- Clustering
- Topic Modeling



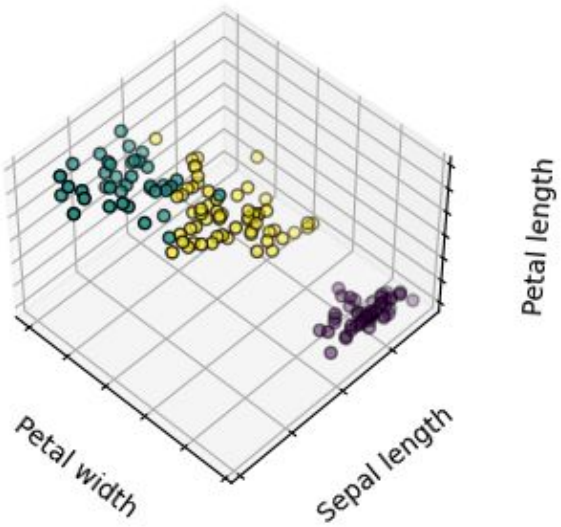
Unsupervised learning in NL

Common techniques in NLP

- Clustering
- Topic Modeling



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica



K-means

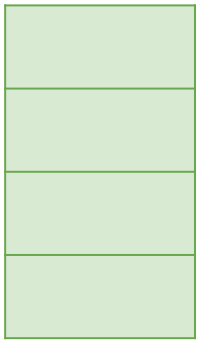
1. Input dataset in numerical representation
2. Select number of clusters
3. Initialize randomly centroid of each cluster
4. For each datapoint calculate distance to centroids and select minimum
5. Update centroid with new datapoint
6. Repeat step 4 until no data points are left

Unsupervised learning in NLP

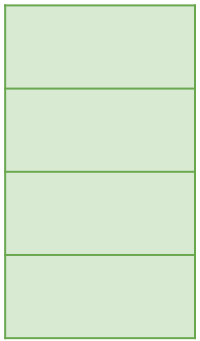
Common techniques in NLP

- Clustering

Client requests

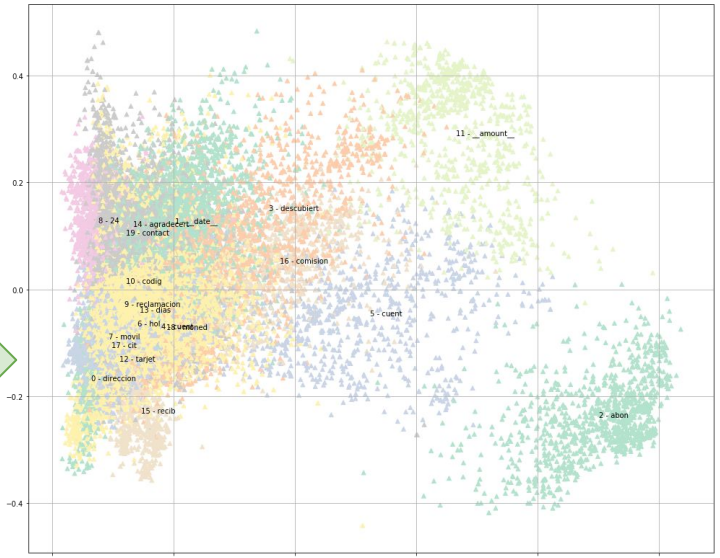


Financial advisor responses



Financial advisor responses
Cleaned, preprocessed, tokenized in tf-idf representation

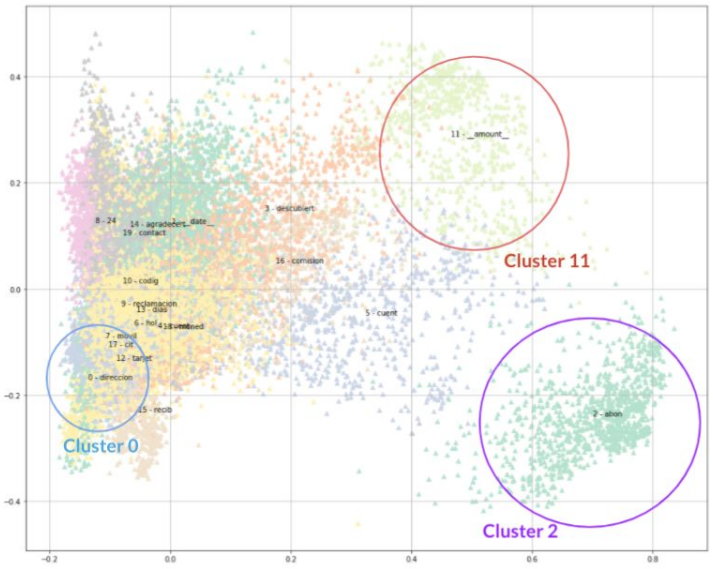
K-means



Unsupervised learning in NLP

Common techniques in NLP

- Clustering



Cluster 2: "Explicación de comisiones"

...se te han aplicado estos gastos al no cumplir los requisitos para estar dentro del PROGRAMA ADIÓS ***COMISIONES***, en tu caso en particular es debido a que no tienes ingresos en ***cuenta***. Para poder estar exento de ***comisiones*** de administración y mantenimiento, sería necesario cumplir con estos requisitos: ***Abono*** de nómina igual o superior a 600Â€ (mínimo 2 meses al ***cuatrimestre***), o ***Abono*** por Desempleo o Pensión igual o superior a 300Â€ (mínimo 2 meses al ***cuatrimestre***)....

...Las ***comisiones***, en tu caso son: - Administración: __AMOUNT__ (__AMOUNT__ por apunte contable en ***cuenta***). - Mantenimiento: __AMOUNT__ (__AMOUNT__ anuales). __NAME__, actualmente no cumples con el programa "Adiós ***comisiones***" en esta cuenta, pues, además de cumplir los requisitos de gastos ...

Cluster 11: "Comisiones por descubierto"

...Comisión de ***descubierto*** tácito: __HOUR_OR_AMOUNT__ euros (3,6719% sobre el ***importe*** __NAME__ de ***descubierto*** en el periodo de liquidación) por el ***descubierto*** por *importe* de __AMOUNT__ producido el día __DATE__ ...

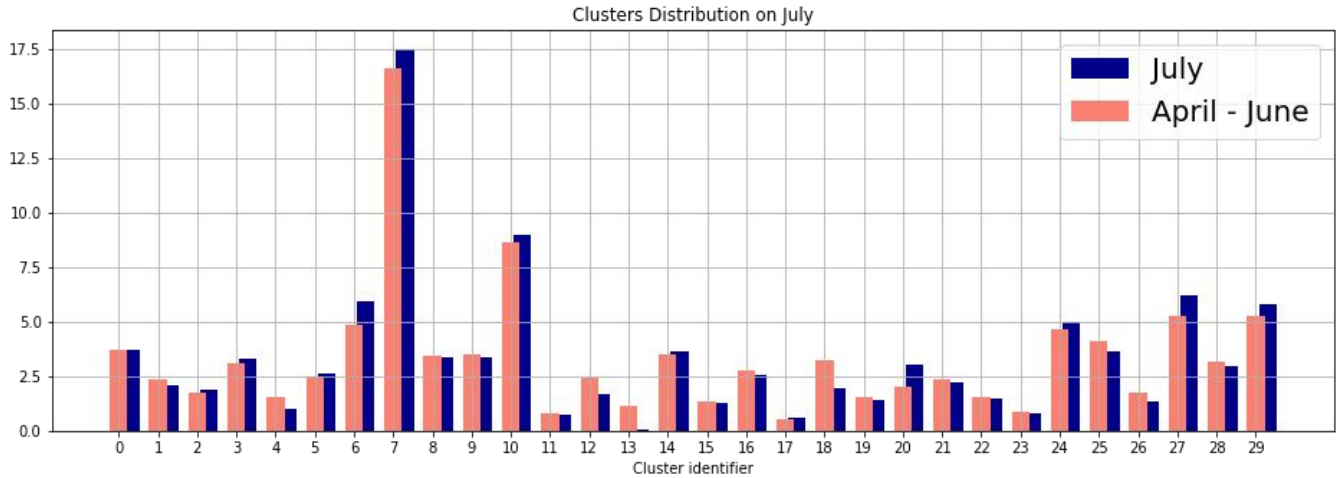
Cluster 0: "Cambio de domicilio fiscal"

... indicarte que puedes ***modificar*** la dirección postal de tus ***productos*** a través de la web ó la aplicación móvil de BBVA. A continuación te voy a indicar ambas rutas. __NAME__, para ***modificar*** el ***domicilio*** postal de tus ***productos***...

Unsupervised learning in NLP

Common techniques in NLP

- Clustering

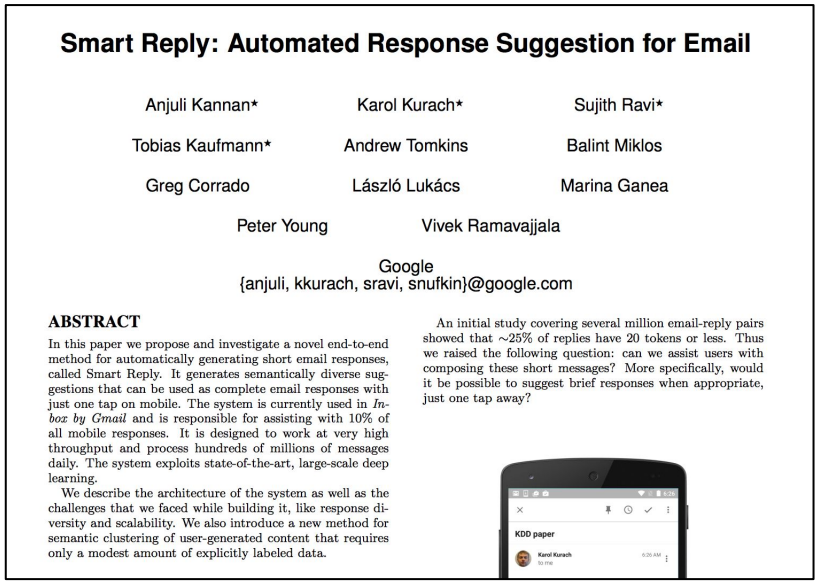


- Clustering model trained with April to June data
- July conversations clustered with the trained model

Unsupervised learning in NLP

Common techniques in NLP

- Clustering



Smart Reply: Automated Response Suggestion for Email

Anjuli Kannan* Karol Kurach* Sujith Ravi*

Tobias Kaufmann* Andrew Tomkins Balint Miklos

Greg Corrado László Lukács Marina Ganea

Peter Young Vivek Ramavajjala

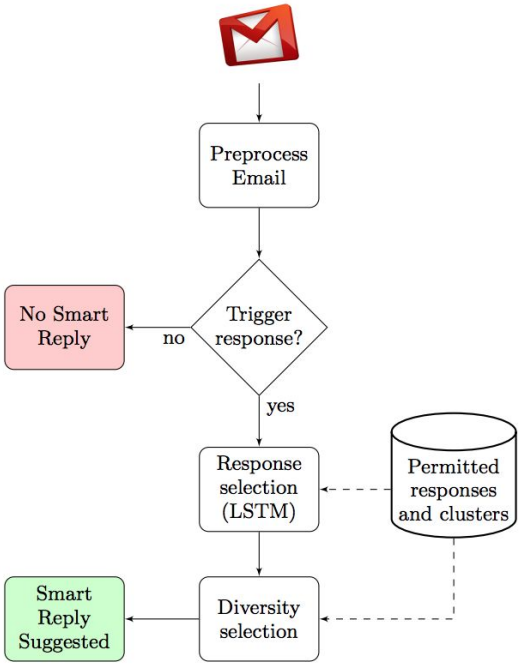
Google
{anjuli, kkurach, sravi, snufkin}@google.com

ABSTRACT

In this paper we propose and investigate a novel end-to-end method for automatically generating short email responses, called Smart Reply. It generates semantically diverse suggestions that can be used as complete email responses with just one tap on mobile. The system is currently used in *Inbox by Gmail* and is responsible for assisting with 10% of all mobile responses. It is designed to work at very high throughput and process hundreds of millions of messages daily. The system exploits state-of-the-art, large-scale deep learning.

We describe the architecture of the system as well as the challenges that we faced while building it, like response diversity and scalability. We also introduce a new method for semantic clustering of user-generated content that requires only a modest amount of explicitly labeled data.

An initial study covering several million email-reply pairs showed that ~25% of replies have 20 tokens or less. Thus we raised the following question: can we assist users with composing these short messages? More specifically, would it be possible to suggest brief responses when appropriate, just one tap away?

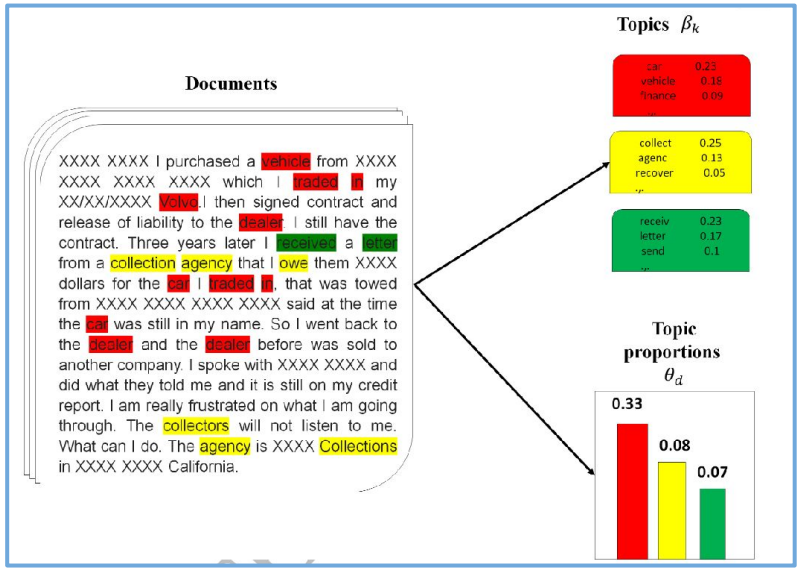


[Smart Reply: Automated Response Suggestion for Email]

Unsupervised learning in NLP

Common techniques in NLP

- Topic modeling
  - Use of a statistical model to discover the abstract "topics" that occur in a collection of documents.
  - Algorithms: **Latent Dirichlet Allocation**, Latent Semantic Allocation

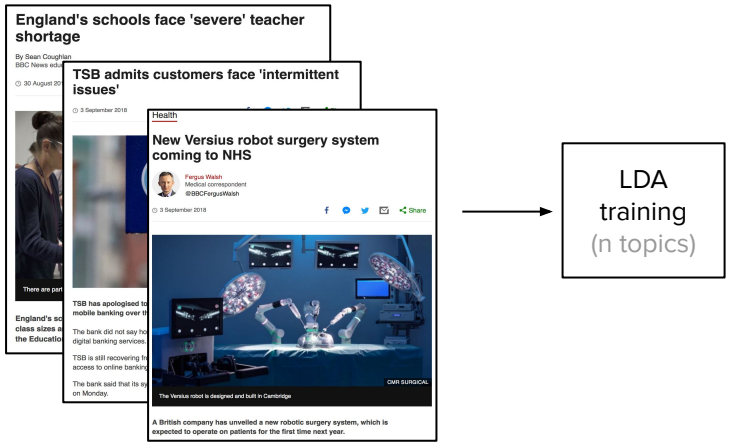


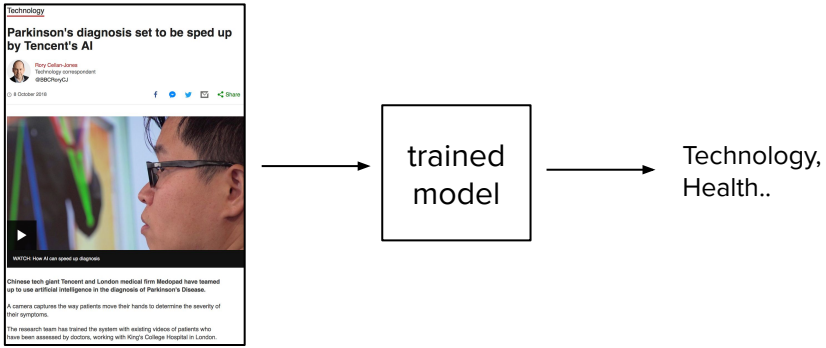
Unsupervised learning in NLP

Topic Modeling Latent Dirichlet Allocation (LDA)

- Generative probabilistic model
- Describes documents as collection of topics
- Applications
  - Document classification
  - Topic extraction

- Unsupervised: number of topics needed a priori
- Language agnostic
- Python using Gensim





Topic Modeling Latent Dirichlet Allocation (LDA)

Document 0	politics parliament politics parliament
Document 1	cell biology cell biology
Document 2	biology cell
Document 3	parliament politics
Document 4	politics biology

	biology	cell	parliament	politics
Document 0	0	0	2	2
Document 1	2	2	0	0
Document 2	1	1	0	0
Document 3	0	0	1	1
Document 4	1	0	0	1
	biology	**cell**	**parliament**	**politics**

	Topic 0	Topic 1
biology	0.570	0.001
cell	0.428	0.001
parliament	0.001	0.428
politics	0.001	0.570
	Topic 0	**Topic 1**

	Topic 0	Topic 1
Document 0	0.071	0.929
Document 1	0.929	0.071
Document 2	0.875	0.125
Document 3	0.125	0.875
Document 4	0.500	0.500
	Topic 0	**Topic 1**

David Lettier (2018)

https://lettier.com/projects/lda-topic-modeling/

Topic Modeling Latent Dirichlet Allocation (LDA)

Topics

Words

	1	2	3	4	5	6	....
brexit	0.6	0.3	0	0	0	0	
hospital	0	0	0.8	0.2	0	0	
...							

Matrix of topics

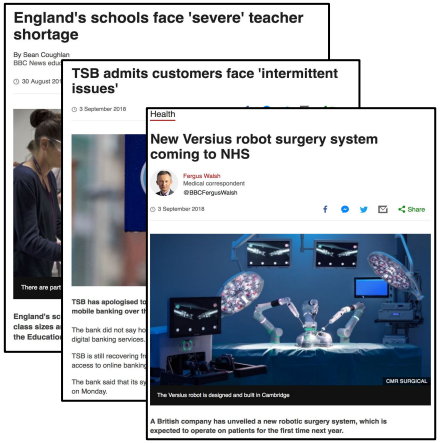
Topics

Docs	1	2	3	4	5	6	...
The Irish border Brexit backstop	0.7	0	0	0	0.1	0	
Scotland to get AI health research centre	0	0	0.9	0	0	0.1	
...							

Matrix of docs

Topic Modeling Latent Dirichlet Allocation (LDA)

Applications: content similarity recommenders



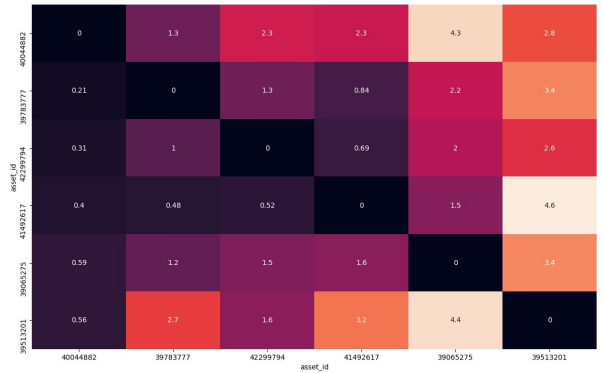
Dataset:
70.000 BBC articles published in 2017
English language

KL = 6.74

Docs	1	2	3	4	5	6	...
The Irish border Brexit backstop	0.7	0	0.2	0	0.1	0	
Scotland to get AI health research centre	0	0	0.9	0	0	0.1	

Similarity measurement

- Discrete probability distributions
- Kullback-Leibler divergence or relative entropy
  - Information gain between distributions



Topic Modeling Latent Dirichlet Allocation (LDA)

Applications: reading user profiles for segmentation

International militant groups	women - gender pay gap	International relations upheaval	Trump travel ban	Miscellaneous - celtic text	Class A drugs	Companies news	Road closures and incidents	Hospital problems	Murder/arrests	...
0	1	2	3	4	5	6	7	8	9	...
attack	woman	president	university	belfast	drug	company	road	health	police	...
group	female	country	student	county	yorkshire	business	prison	hospital	man	...
kill	man	mr	ban	inquest	sheffield	firm	close	care	yearold	...
people	male	protest	law	ysgol	cannabis	pay	traffic	patient	arrest	...
military	gender	government	bill	qatar	supply	sell	m	nhs	attack	...
state	women	china	state	coroner	norfolk	m	crash	trust	woman	...
force	weinstein	state	new	primary	heroin	customer	bike	service	incident	...
paper	pay	iran	country	school	people	buy	reopen	treatment	find	...
muslim	equality	saudi	right	londonderry	find	sale	prisoner	staff	officer	...
violence	lee	chinese	immigration	st	suffolk	store	route	doctor	murder	...
report	work	israel	allow	paper	substance	money	near	cancer	hospital	...
army	harvey	minister	order	armagh	cocaine	share	cyclist	mental	street	...
soldier	singapore	opposition	people	cornwall	class	shop	driver	flight	road	...

Manual topic naming with BBC editors

Topic Modeling Latent Dirichlet Allocation (LDA)

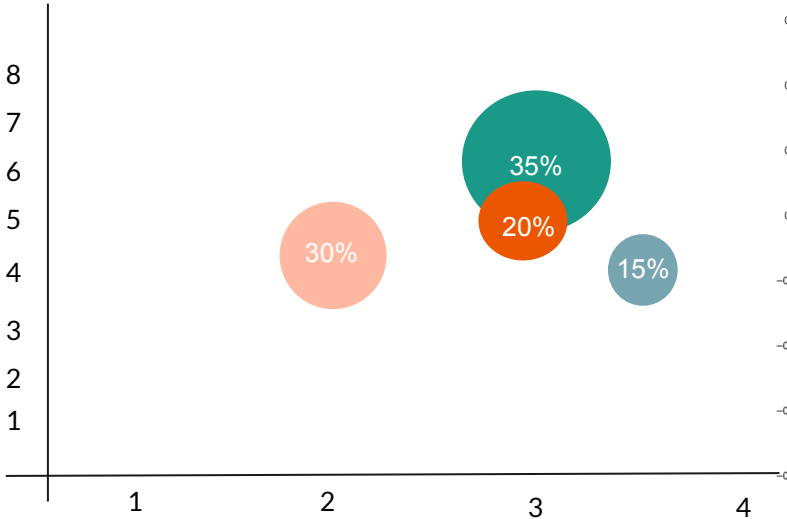
Applications: reading user profiles for segmentation

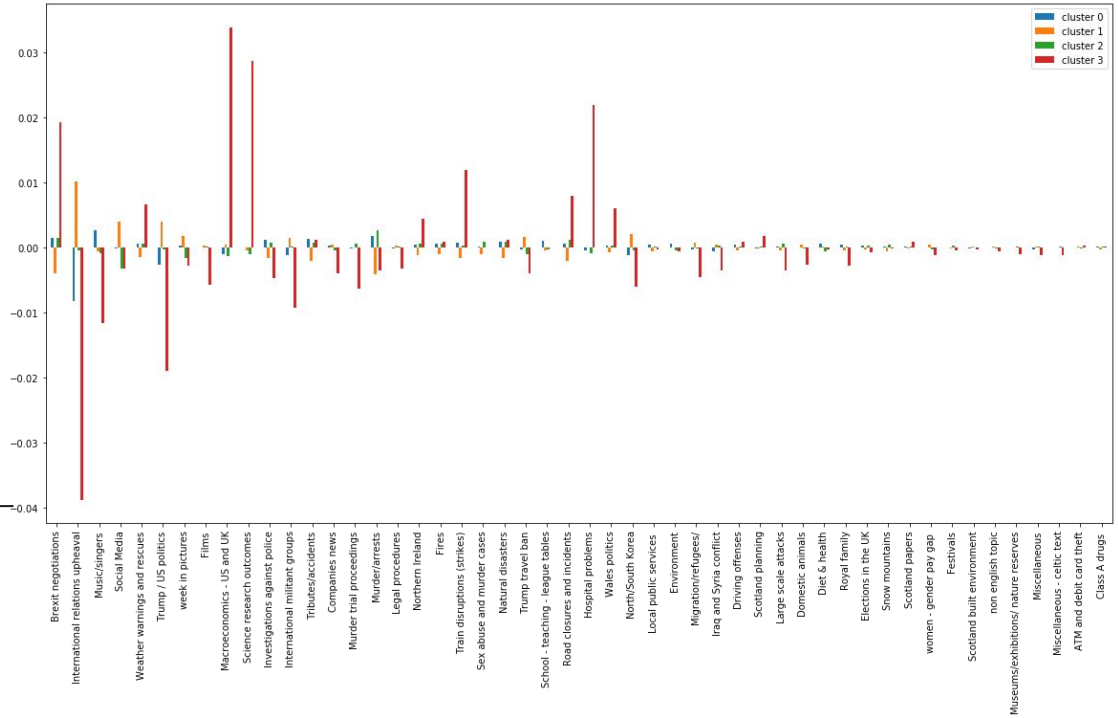
health	Scottish politics	education	local politics	brexit	crime	...
0.8	0	0.3	0	0	0.7	

health	Scottish politics	education	local politics	brexit	crime	...
0	0.6	0	0.5	0.6	0	

Topic Modeling Latent Dirichlet Allocation (LDA)

Applications: reading user profiles for segmentation





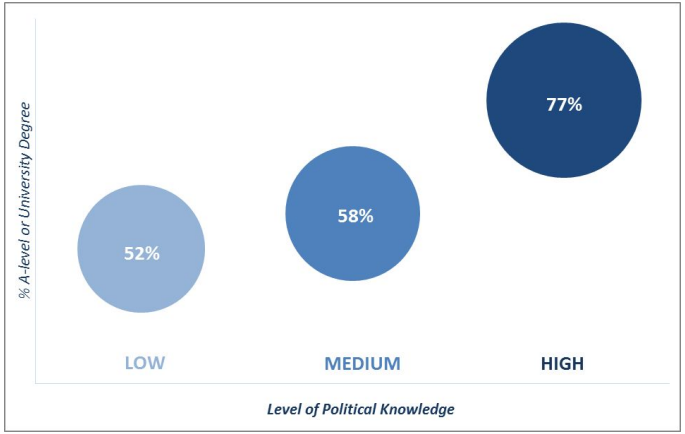
Topic Modeling Latent Dirichlet Allocation (LDA)

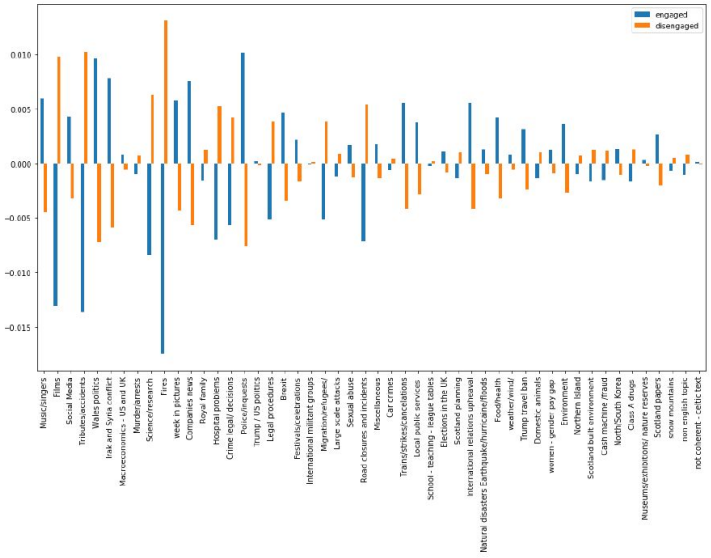
Applications: Political knowledge gaps

KNOWLEDGE GAPS AND THE ROLE OF THE BBC IN THE ERA OF FAKE NEWS

Marta Cantijoch
Nikki Soo
Tim Cowlishaw
Clara Higuera Cabañes

BBC R&D IRFS / BBC News Audience Engagement
University of Manchester Department of Politics

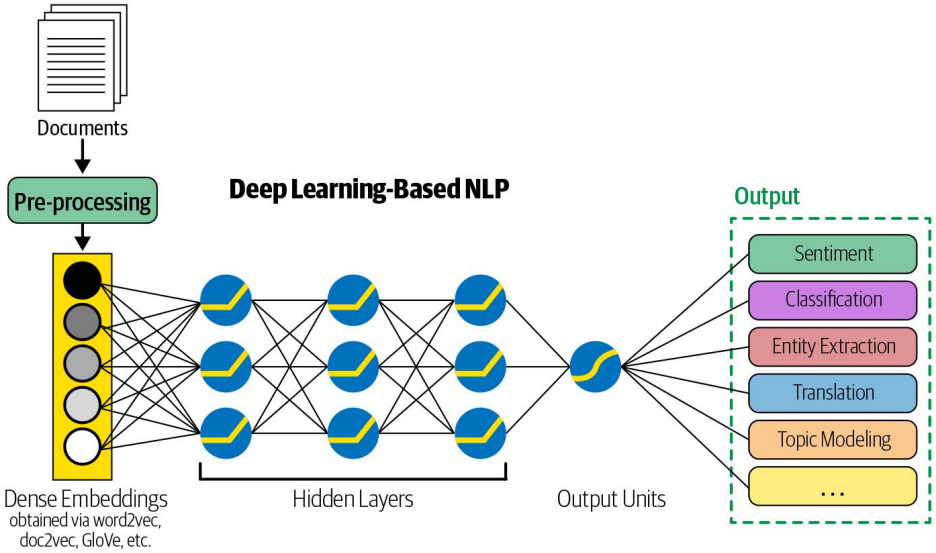




Hands on exercise...

NLP with deep learning

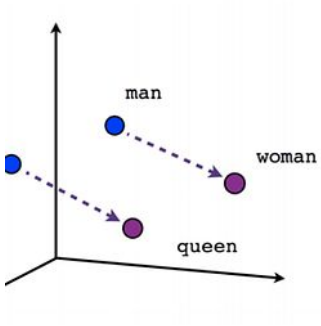
Preprocessing
Text representation
- Embeddings



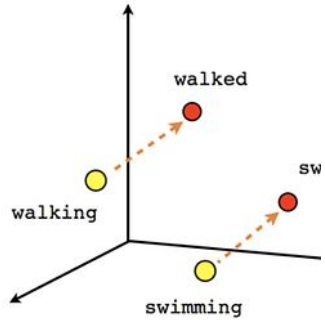
Word embeddings

Word embeddings are a **type of word representation** that allows words that **share similar context** to have a similar representation.

Use neural network architectures to create **dense, low-dimensional representations** of words and texts



Male-Female



Verb tense

Based on:

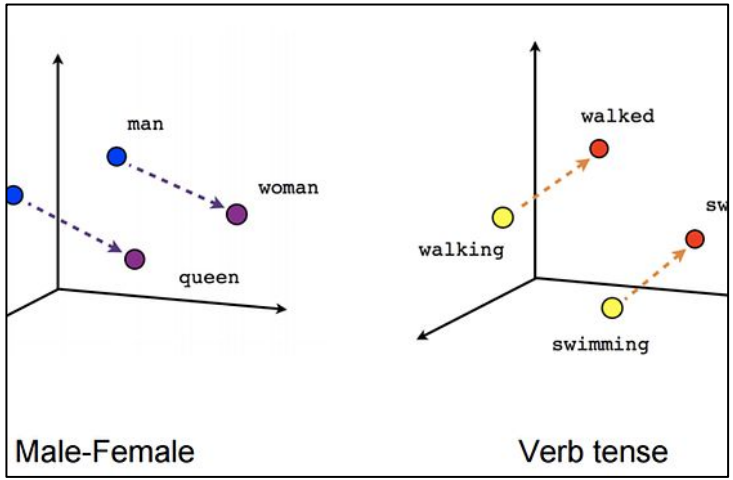
Distributional hypothesis
In linguistics, this hypothesizes that words that occur in similar contexts have similar meanings. For example, the English words “dog” and “cat” occur in similar contexts.

Thus, if two words often occur in similar context, then their corresponding representation vectors must also be close to each other.

Word embeddings

Neural network–based word representation model known as **“Word2vec,”** based on “distributional similarity,” can capture word analogy relationships such as

$$\text{King} - \text{Man} + \text{Woman} \approx \text{Queen}$$



- Captures context in vector representation vs BOW /tf-idf
- Fixed low dimensionality vectors vs BOW/tf-idf increasing vectors with increased vocabulary

Mikolov et al. 2013 Distributed Representations of Words and Phrases and their Compositionality

Word embeddings

1. Select a embedding dimension and a window size (hyperparameters)
2. For every word *w* in corpus, we start with a vector v_w initialized with random values.
   a. **CBOW model:** predicts the center word given the context words in which the center word appears.
   b. Continuous Skipgram model: predict the context words from the center word.

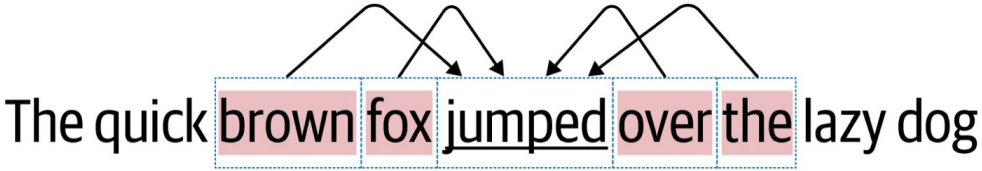


Figure 3-7. CBOW: given the context words, predict the center word

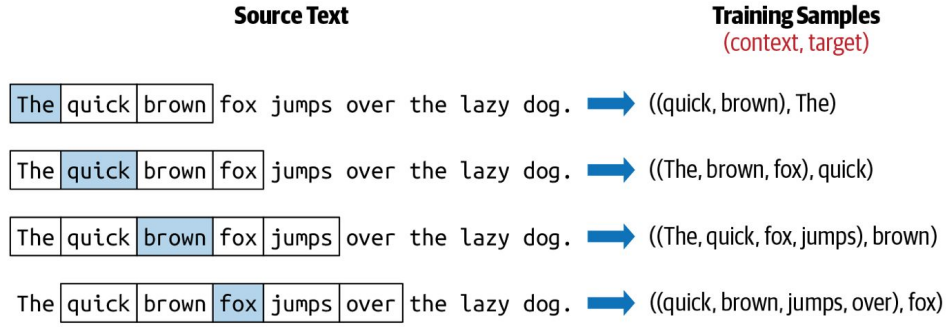


Figure 3-8. Preparing a dataset for CBOW

Word embeddings

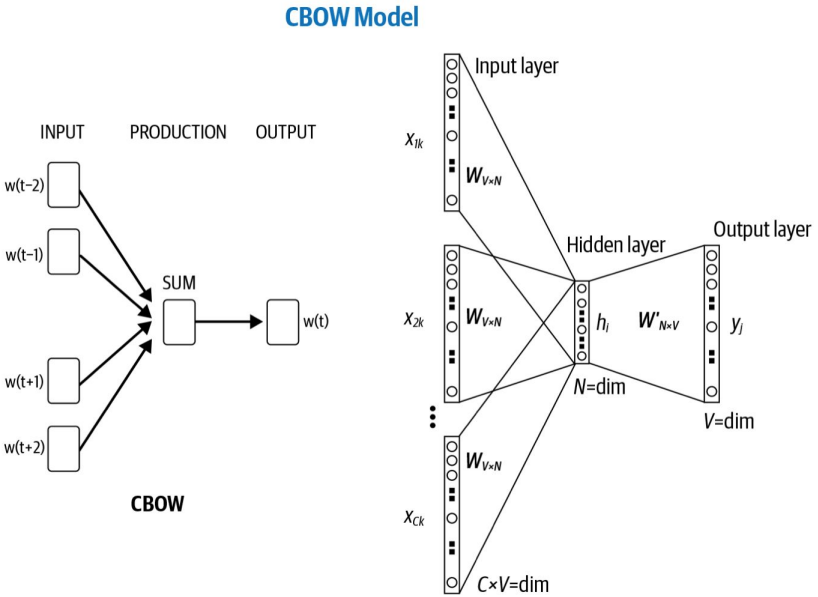


Figure 3-9. CBOW model [7]

```
[6] #What is the vector representation for a word?
    w2v_model['beautiful']
```

```
array([-0.01831055,  0.05566406, -0.01153564,  0.07275391,  0.15136719,
       -0.06176758,  0.20605469, -0.15332031, -0.05908203,  0.22851562,
       -0.06445312, -0.22851562, -0.09472656, -0.03344727,  0.24707031,
        0.05541992, -0.00921631,  0.1328125 , -0.15429688,  0.08105469,
       -0.07373047,  0.24316406,  0.12353516, -0.09277344,  0.08203125,
        0.06494141,  0.15722656,  0.11279297, -0.0612793 , -0.296875  ,
       -0.13378906,  0.234375  ,  0.09765625,  0.17773438,  0.06689453,
       -0.27539062,  0.06445312, -0.13867188, -0.08886719,  0.171875  ,
        0.07861328, -0.10058594,  0.23925781,  0.03808594,  0.18652344,
       -0.11279297,  0.22558594,  0.10986328, -0.11865234,  0.02026367,
        0.11376953,  0.09570312,  0.29492188,  0.08251953, -0.05444336,
       -0.0090332 , -0.0625    , -0.17578125, -0.08154297,  0.01062012,
       -0.04736328, -0.08544922, -0.19042969, -0.30273438,  0.07617188,
        0.125     , -0.05932617,  0.03833008, -0.03564453,  0.2421875 ,
        0.36132812,  0.04760742,  0.00631714, -0.03088379, -0.13964844,
        0.22558594, -0.06298828, -0.02636719,  0.1171875 ,  0.33398438,
       -0.07666016, -0.06689453,  0.04150391, -0.15136719, -0.22460938,
        0.03320312, -0.15332031,  0.07128906,  0.16992188,  0.11572266,
       -0.13085938,  0.12451172, -0.20410156,  0.04736328, -0.296875  ,
       -0.17480469,  0.00872803, -0.04638672,  0.10791016, -0.203125  ,
       -0.27539062,  0.2734375 ,  0.02563477, -0.11035156,  0.0625    ,
        0.1953125 ,  0.16015625, -0.13769531, -0.09863281, -0.1953125 ,
       -0.22851562,  0.25390625,  0.00915527, -0.03857422,  0.3984375 ,
       -0.1796875 ,  0.03833008, -0.24804688,  0.03515625,  0.03881836,
        0.03442383, -0.04101562,  0.20214844, -0.03015137, -0.09619141,
        0.11669922, -0.06738281,  0.0625    ,  0.10742188,  0.25585938,
       -0.21777344,  0.05639648, -0.0065918 ,  0.16113281,  0.11865234,
       -0.03088379, -0.11572266,  0.02685547,  0.03100586,  0.09863281,
        0.05883789,  0.00634766,  0.11914062,  0.07324219, -0.01586914,
        0.18457031,  0.05322266,  0.19824219, -0.22363281, -0.25195312,
        0.15039062,  0.22753906,  0.05737305,  0.16992188, -0.22558594,
        0.06494141,  0.11914062, -0.06640625, -0.10449219, -0.07226562,
       -0.16992188,  0.0625    ,  0.14648438,  0.27148438, -0.02172852,
       -0.12695312,  0.18457031, -0.27539062, -0.36523438, -0.03491211,
       -0.18554688,  0.23828125, -0.13867188,  0.00296021,  0.04272461,
        0.13867188,  0.12207031,  0.05957031, -0.22167969, -0.18945312,
        0.23242188, -0.28710938, -0.00866699, -0.16113281, -0.24316406,
        0.05712891, -0.06982422,  0.00053406, -0.10302734, -0.13378906,
       -0.16113281,  0.11621094,  0.31640625, -0.02697754, -0.01574707,
        0.11425781, -0.04174805,  0.05908203,  0.02661133, -0.08642578,
        0.140625  ,  0.09228516, -0.25195312, -0.31445312, -0.05688477,
        0.01031494,  0.0234375 , -0.02331543, -0.08056641,  0.01269531,
       -0.34179688,  0.17285156, -0.16015625,  0.07763672, -0.03088379,
        0.11962891,  0.11767578,  0.20117188, -0.01940918,  0.02172852,
        0.23046875,  0.28125   , -0.17675781,  0.02978516,  0.08740234,
       -0.06176758,  0.00939941, -0.09277344, -0.203125  ,  0.13085938,
       -0.13671875, -0.00500488, -0.04296875,  0.12988281,  0.3515625 ,
        0.0402832 , -0.12988281, -0.03173828,  0.28515625,  0.18261719,
        0.13867188, -0.16503906, -0.26171875, -0.04345703,  0.0100708 ,
        0.08740234,  0.00421143, -0.1328125 , -0.17578125, -0.04321289,
       -0.015625  ,  0.16894531,  0.25      ,  0.37109375,  0.19921875,
       -0.36132812, -0.10302734, -0.20800781, -0.20117188, -0.01519775,
       -0.12207031, -0.12011719, -0.07421875, -0.04345703,  0.14160156,
        0.15527344, -0.03027344, -0.09326172, -0.04589844,  0.16796875,
       -0.03027344,  0.09179688, -0.10058594,  0.20703125,  0.11376953,
       -0.12402344,  0.04003906,  0.06933594, -0.34570312,  0.03881836,
        0.16210938,  0.05761719, -0.12792969, -0.05810547,  0.0385742 ,
       -0.11328125, -0.1953125 , -0.28125   , -0.13183594,  0.15722656,
       -0.09765625,  0.09619141, -0.09960938, -0.00285339, -0.03637695,
        0.15429688,  0.06152344, -0.34570312,  0.11083984,  0.03344727],
      dtype=float32)
```

Figure 3-6. Vector representing the word "beautiful" in pre-trained Word2vec

Word embeddings

Ways of use

Pretrained models

- Word2vec in gensim
- Glove - Stanford
- FastText(Facebook)

Train your own model

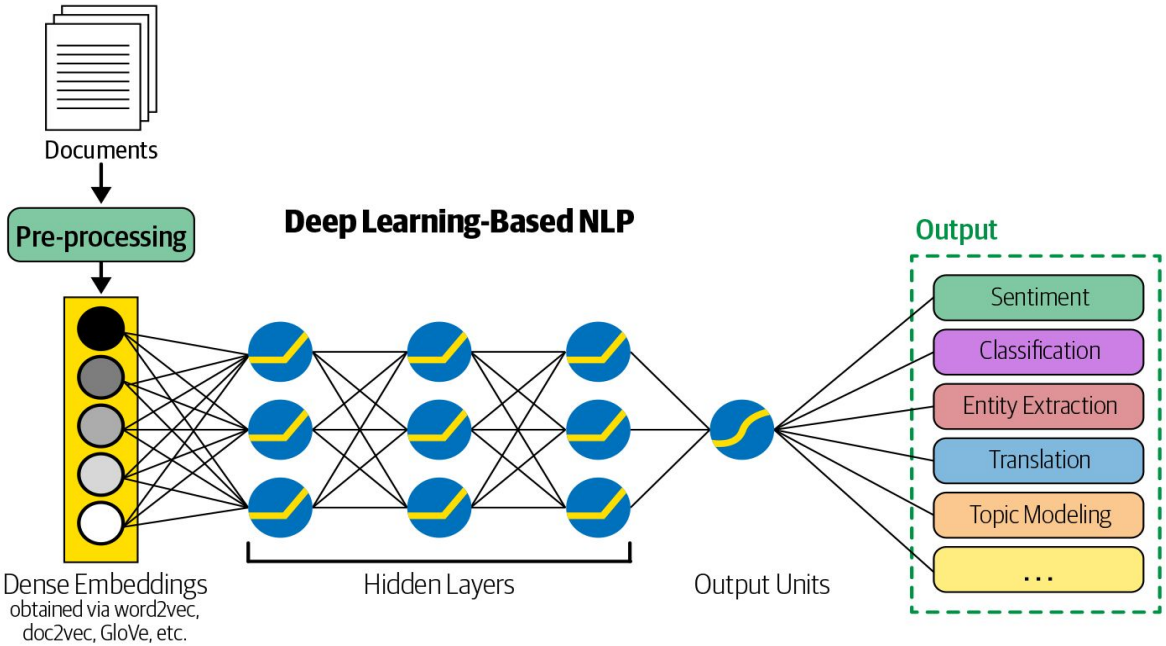
- Useful when your use case is context specific

```
#Import a test data set provided in gensim to
train a model
from gensim.test.utils import common_texts
#Build the model, by selecting the parameters.
our_model = Word2Vec(common_texts, size=10,
window=5, min_count=1, workers=4)
```

Chapter 3. Text Representation. Practical Natural Language Processing. O´Reilly
Word2vec tutorial
CBOW model architecture - video

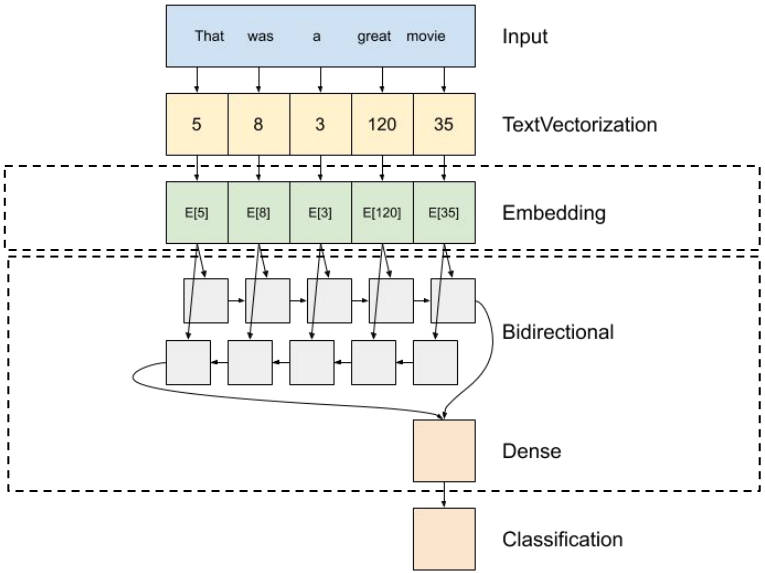
Hands on exercise...

NLP with deep learning



Text classification with deep learning

Text classification with deep learning



Pre-trained embedding / own embedding layer

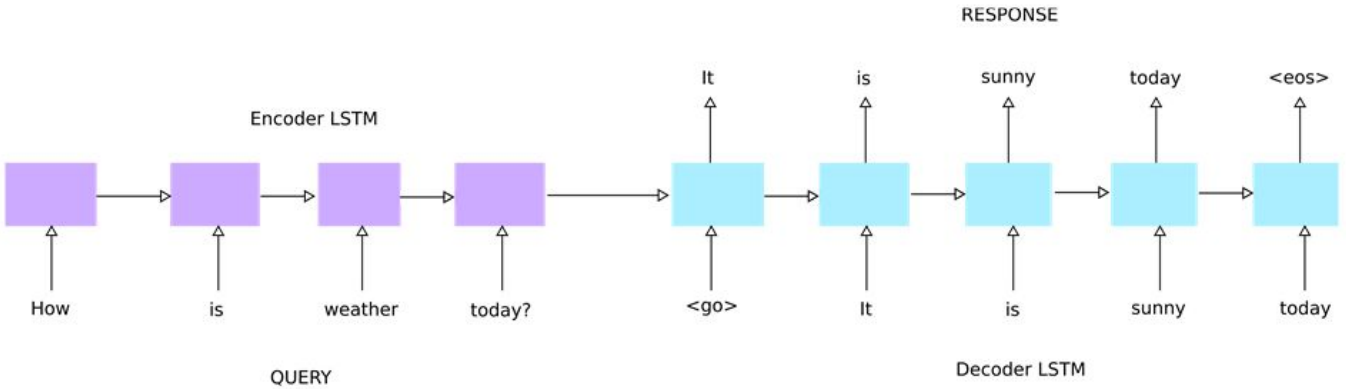
Different possible networks,
CNN, LSTM

[Practical example notebook]

Natural language generation

Applications

- Machine translation
- Summarization
- Question Answering



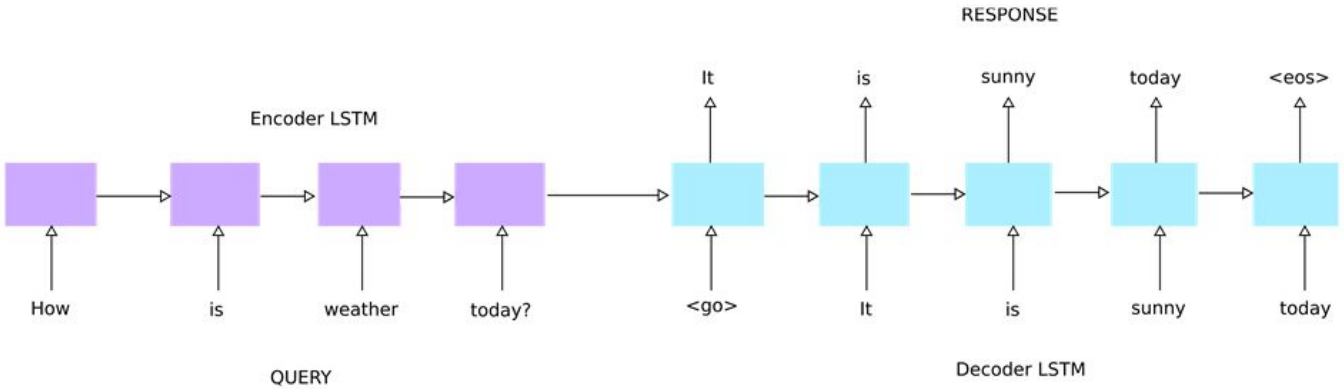
Question Answering with seq2seq

Natural language generation

Applications

- Machine translation
- Summarization
- Question Answering

[Video: Prof. John Kelleher explains neural machine translation]

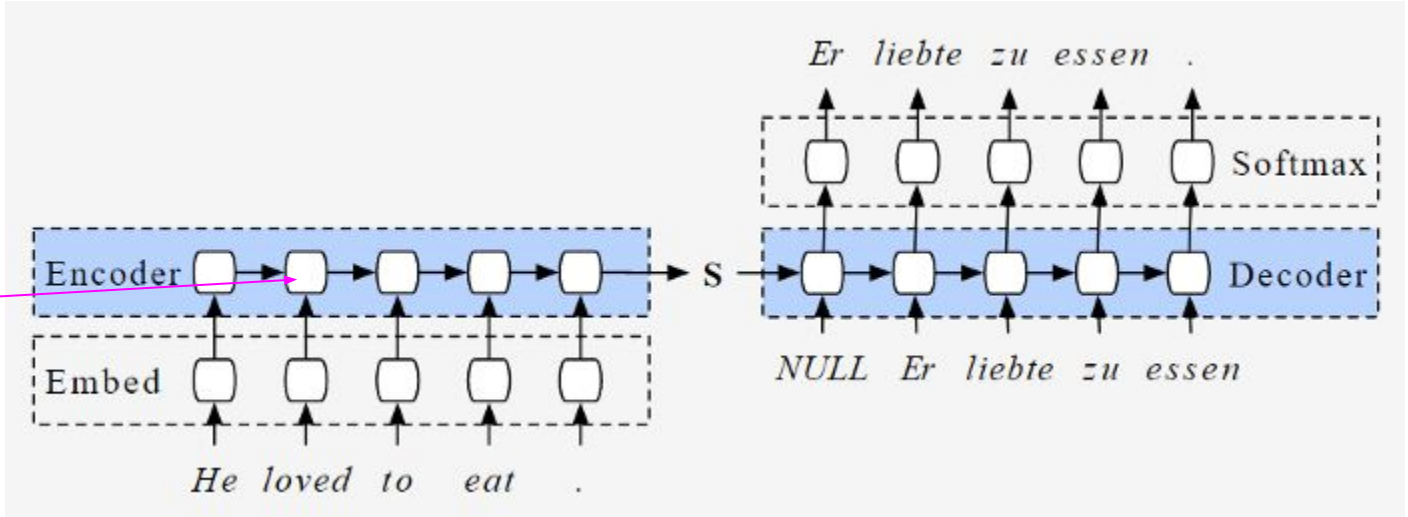


Question Answering with seq2seq

Natural language generation

seq2seq

hidden states: in training captur info about the current cell and precious one



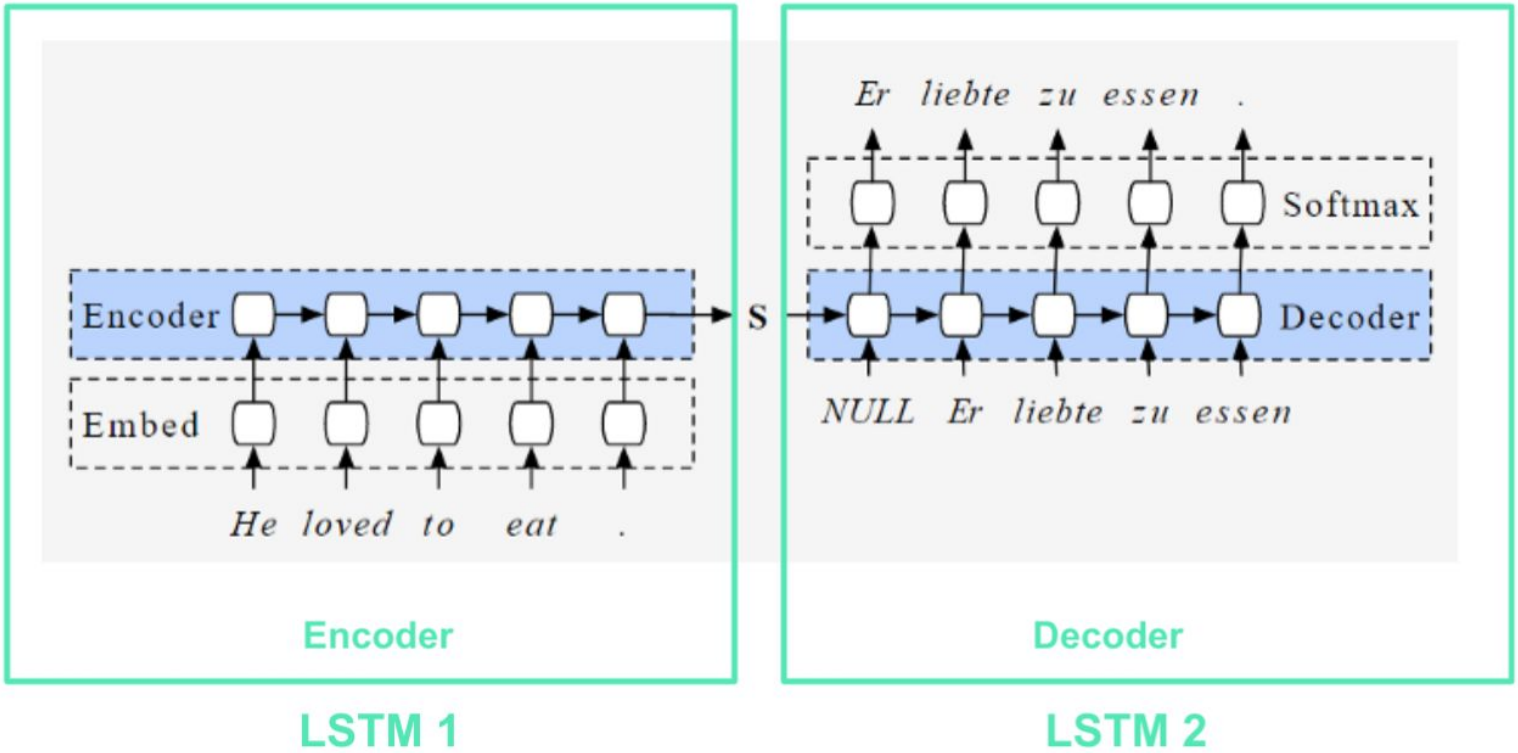
Encoder

Vector that represents entry sequence

Decoder

Natural language generation

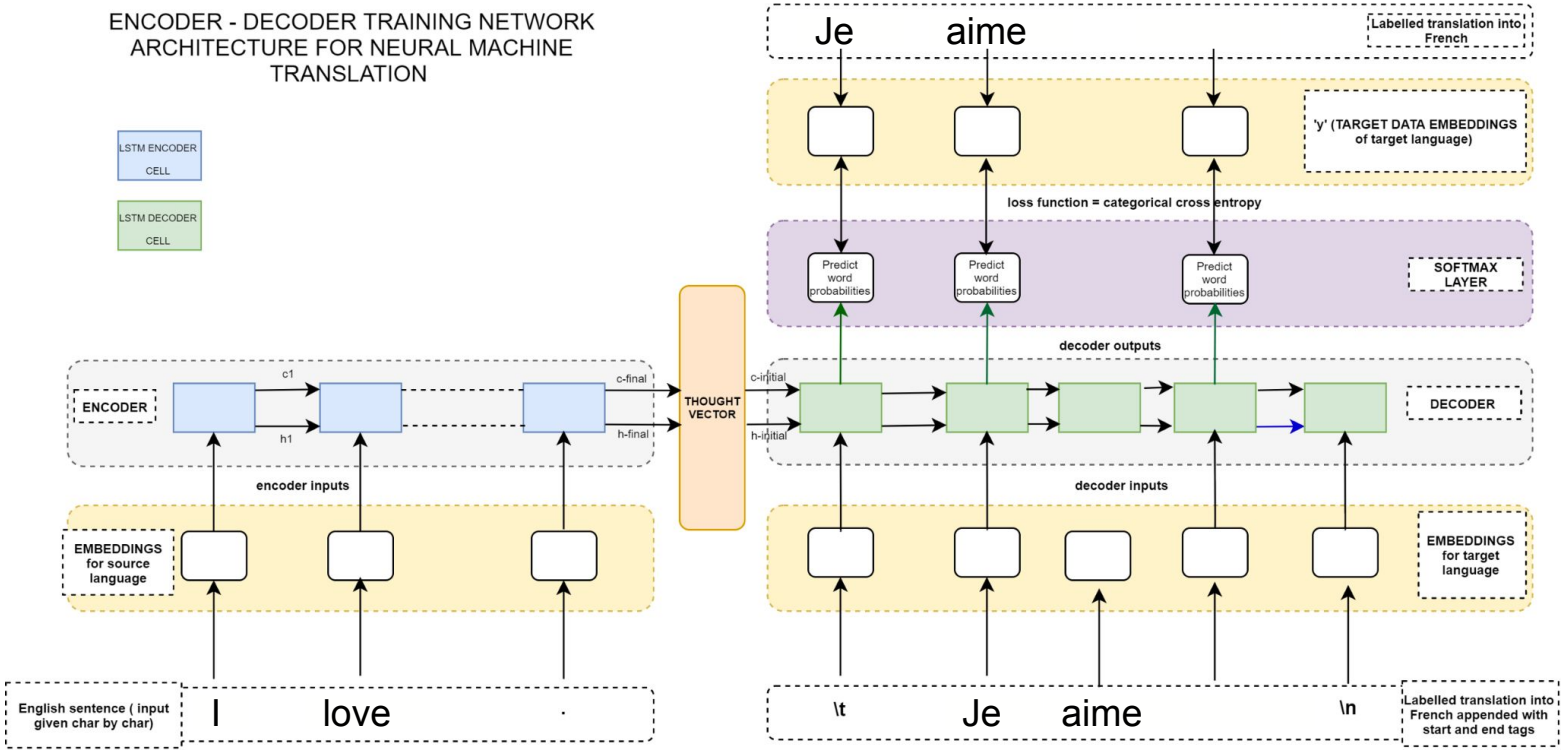
seq2seq



gif

Natural language generation

seq2seq



Training

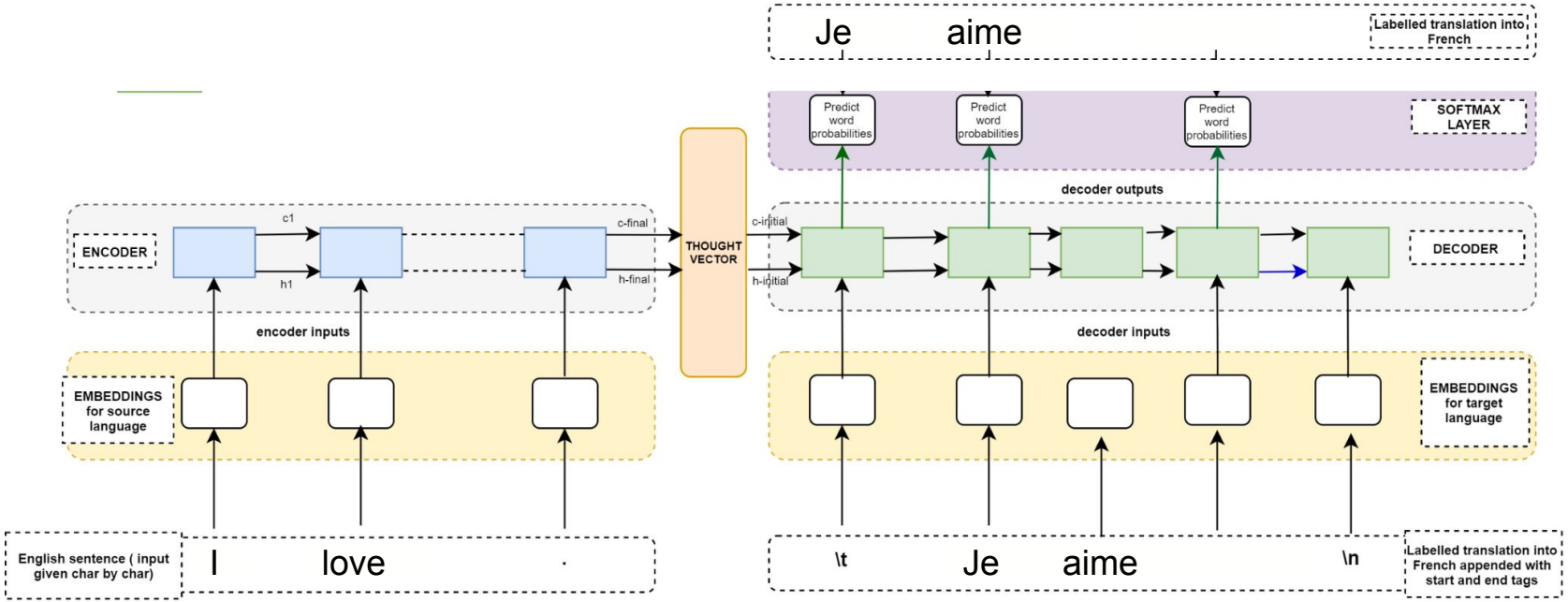
Parameters

- max_len_seq
- num epochs
- embedding_dim
- learning rate
- attention layer
- drop out
- ...

Natural language generation

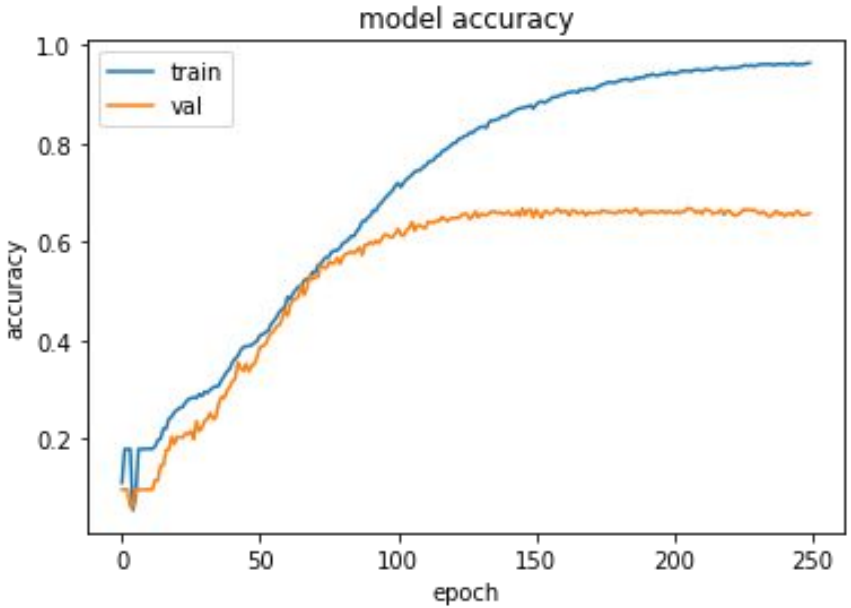
Inference

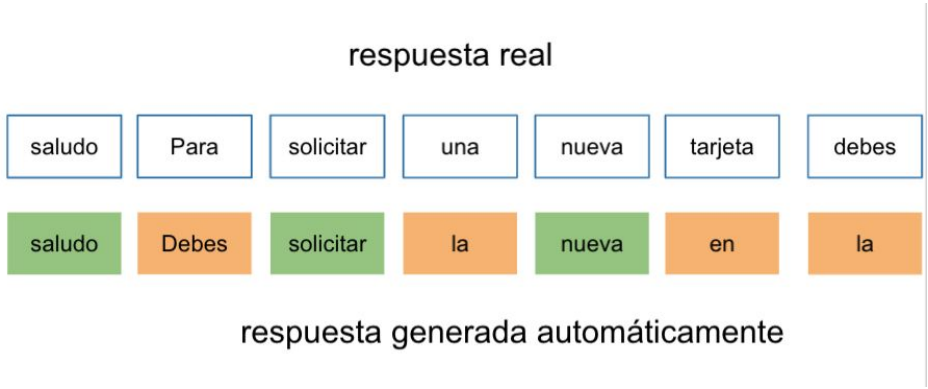
seq2seq



Natural language generation

NLG Evaluation





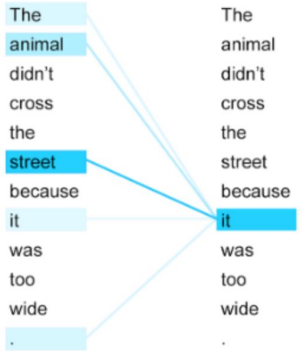
[Natural language generation metrics]
(bleu, rouge, f1…)

Natural language generation

Limitations of seq2seq

- Dealing with long-range dependencies is still challenging
- The sequential nature of the model architecture prevents parallelization.

> *"Self-attention, sometimes called intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence."*



The **Transformer in NLP** is a novel architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease.

Attention Is All You Need.

Understanding transformers

What´s left...

1. Transfer learning in NLP
2. Explainability in NLP
3. Bias and dangers in NLP

What´s left...

[Eli5 library]

1. Transfer learning in NLP
2. **Explainability in NLP**
3. Bias and dangers in NLP

y=jquery top features		**y=mysql** top features		**y=objective-c** top features		**y=php** top features	
Weight[?]	**Feature**	**Weight**[?]	**Feature**	**Weight**[?]	**Feature**	**Weight**[?]	**Feature**
+3.977	jquery	+4.897	mysql	+4.001	objective	+2.558	php
+2.060	ready	+1.316	million	+1.276	passcode	+1.607	ga
+1.166	jsfiddle	+1.158	isregistered	+1.268	monish	*… 23131 more positive …*	
+1.109	fadein	+0.962	restore	+1.262	cocoa	*… 128982 more negative …*	
+1.090	animate	+0.927	mdf	+1.241	zziplib	-1.271	iphone
… 24776 more positive …		+0.920	truncated	*… 35356 more positive …*		-1.278	java
… 127337 more negative …		+0.905	tablename	*… 116757 more negative …*		-1.282	objective
-1.029	blockquote	*… 32999 more positive …*		-1.155	who	-1.283	page_title_home
-1.035	employee	*… 119114 more negative …*		-1.182	comment	-1.339	void
-1.056	scope	-0.921	connections	-1.204	def	-1.340	flags
-1.081	python	-0.939	python	-1.206	stringbyappendingformat	-1.366	logout
-1.087	folder	-0.954	app	-1.223	jquery	-1.379	fields_by_id
-1.087	int	-0.992	web	-1.232	mysql	-1.398	javascript
-1.151	mean	-1.001	product_table	-1.243	selectedstring	-1.450	clicking
-1.164	sql	-1.033	2010	-1.282	textfielddidendediting	-1.566	rails
-1.192	javascript	-1.065	define	-1.422	rails	-1.600	net
-1.333	getelementbyid	-1.113	conn	-1.437	css	-1.617	angularjs
-1.502	rails	-1.124	net	-1.471	net	-1.705	footer
-1.650	angularjs	-1.161	name_table	-1.497	javascript	-1.711	uri
-1.930	ng	-1.642	android	-1.631	android	-2.092	python
-2.222	angular	-1.889	rails	-1.749	swift	-2.463	jquery
-3.738	<BIAS>	-3.892	<BIAS>	-2.530	<BIAS>	-3.317	<BIAS>

y=python (probability **0.005**, score **-7.792**) top features

Contribution[?]	**Feature**
-3.737	<BIAS>
-4.054	Highlighted in text (sum)

add element from a list of dictionary to another <pre><code>dict_1 = [{ incident_id : sd000001372596 first_call : t } { incident_id : sd000001372594 first_call : f } { incident_id : sd000001372598 first_call : f } { incident_id : sd000001372599 first_call : t } { incident_id : sd000001372602 first_call : f } { incident_id : sd000001372601 first_call : f } { incident_id : sd000001372605 first_call : f } { incident_id : sd000001372606 first_call : f } { incident_id : sd000001372607 first_call : f }] dict_2 = [{ incident_id : sd000001372605 date : 08-10-2016 00:54:13 } { incident_id : sd000001372606 date : 08-10-2016 00:57:20 } { incident_id : sd000001372607 date : 08-10-2016 01:00:25 } { incident_id : sd000001372598 date : 11-10-2016 10:57:34 } { incident_id : sd000001372602 date : 08-10-2016 10:44:34 } { incident_id : sd000001372601 date : 21-10-2016 22:30:49 } { incident_id : sd000001372594 date : 18-10-2016 14:53:34 }] </code></pre> i have two list of dictionaries with different length and i want to add the (dict_2 date) to the dict_1 according the incident_id

What´s left...

1. Transfer learning in NLP
2. Explainability in NLP
3. **Bias and dangers in NLP**

Bias in Natural Language Processing (NLP): A Dangerous But Fixable Problem

One of the biggest new issues that natural language processing (NLP) models face is the implicit biases that they learn.

https://towardsdatascience.com/bias-in-natural-language-processing-nlp-a-dangerous-but-fixable-problem-7d01a12cf0f7

References

Books

Practical Natural Language Processing. O´Reilly
Practical Natural Language Processing with Python with Case Studies from Industries Using Text Data at Scale

Generative Deep Learning. Teaching Machines to Paint, write Compose and Play. O´Reilly (Chapter 6 Language generation)

Interesting articles/posts

How AI is getting better at Detecting Hate Speech
The dark side of Guardian Comments

Scientific papers

Smart Reply: Automated Response Suggestion for Email