DATA SCIENCE RETREAT®
SINCE 2014

Crash Course: Intro to NLP

David Nogueira

Senior Software Engineer | Researcher | Speaker
NLP | ML & AI | Voice Services | Innovation

linkedin.com/in/davidjacomenogueira

Plan

- Overview & context of NLP within ML
- Overview of NLP basic concepts (Bag-of-Words Model, Word embeddings versus one-hot encoders, Cosine similarity)
- Overview of popular NLP tasks
- State of the art and the challenges in modern NLP
- Lessons learned and software engineering tips

- Deep dive
  - Text/document classification
    - Sentiment Analysis (fast reading two SOTA papers + coding the neural network)
    - Topic Labelling (with the same code and minor tweaks, let’s tackle another NLP task)
  - Recommendation (let’s code movie recommendations)
  - Named Entity Recognition & Named Entity Linking (we will be reading two state-of-the-art literature papers)

Relevant python libraries

- numpy & scipy » processing large multidimensional arrays and matrices
- pandas » high-level data structures, read/write from csv, etc. Built-in methods for grouping, filtering, and combining data, as well as the time-series functionality
- nltk, spaCy, gensim » natural language processing toolkits
- scikit-learn » algorithms for many standard machine learning and data mining task: clustering, regression, classification, etc.
- pytorch & tensorflow & keras » popular frameworks for deep and machine learning: for training and running neural networks
- matplotlib & seaborn » visualization: libraries for 2D diagrams and graphs

Some NLP Datasets for your own NLP projects

Google multi-billion N-grams from Google books: https://aws.amazon.com/datasets/google-books-ngrams/ (2.2 TB of multilingual data)

Common Crawls Corpus: https://registry.opendata.aws/commoncrawl/ (Contains 2 billion web pages with metadata, updated monthly)

Google BigQuery Public Datasets: https://cloud.google.com/bigquery/public-data/ (Contains lots of interesting datasets, from Bay Area Bike Share Trips to Github Commits history)

Complete Dump of All Wikipedia Pages: https://en.wikipedia.org/wiki/Wikipedia:Database_download

216,930 Jeopardy Questions, Answers, and Other Data: https://lnkd.in/ejeCjuQ (Dataset description: https://lnkd.in/emesvXu)

1.1 Million property tax bills from New York City: https://lnkd.in/eVbjtZE

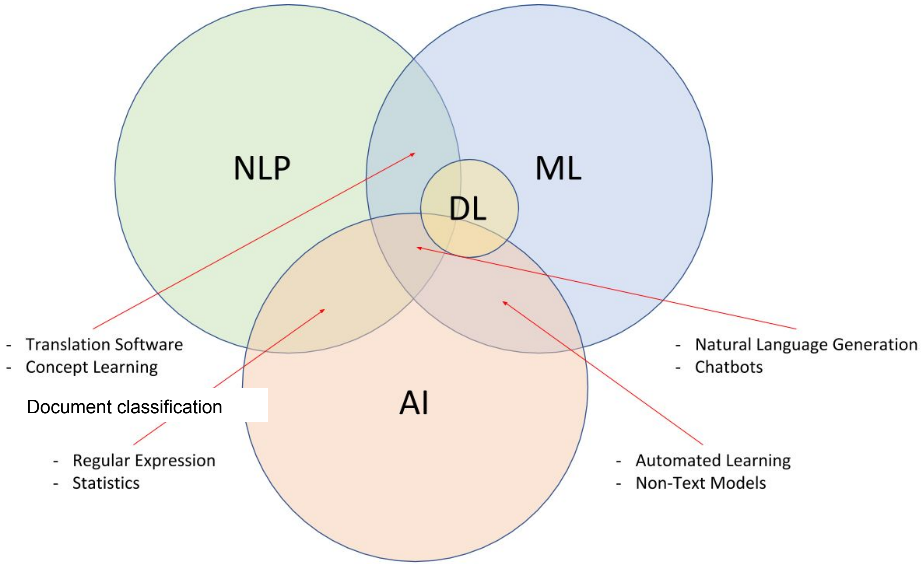
Enron Email Dataset: http://academictorrents.com/details/4697a6e1e7841602651b087d84f904d43590d4ff

Online News Dataset: http://academictorrents.com/details/95d3b03397a0bafd74a662fe13ba3550c13b7ce1

Global Entity Graph: https://blog.gdeltproject.org/announcing-the-global-entity-graph-geg-and-a-new-11-billion-entity-dataset/

Others: https://www.reddit.com/r/datasets/

Overview & context of NLP within ML



AI - Artificial Intelligence
Development of “intelligent” systems, that to exhibit intelligent behaviour.

ML - Machine Learning
Development of “intelligent” systems that perform specific tasks without using explicit instructions, relying on patterns and inference instead.

DL - Deep Learning
Development of “intelligent” systems that can learn from **huge amounts of** data and have many *layered artificial neural networks*.

NLP - Natural Language Processing
Development of “intelligent” systems that process/”understand” human language.

Natural Language Processing

“Natural language processing (NLP) is a subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data.”

wikipedia

In the early days, many language-processing systems were designed by hand-coding a set of rules: such as by writing grammars or devising heuristic rules for stemming.
Since the so-called "statistical revolution" in the late 1980s and mid 1990s, much natural language processing research has relied heavily on statistics/machine learning.

Major hurdles in NLP when coming from other areas

The major difference when approaching an NLP problem is that the input data is text, and not numeric.

- Text is sparse;
- Same words in different documents have different meaning (context);
- Different languages;
- For humans, communication is innate and trivial; machines don’t really understand human communication as we do.

NLP Tasks

- Text/document classification
  - Topic Labelling
  - Sentiment Analysis
- Recommendation
- Document Clustering
- Summarization
- Named Entity Recognition
- Named Entity Linking
- Machine Translation

Note: these tasks below, although related with human natural language, they’re not part of NLP field, as they don’t deal exclusively with text data:

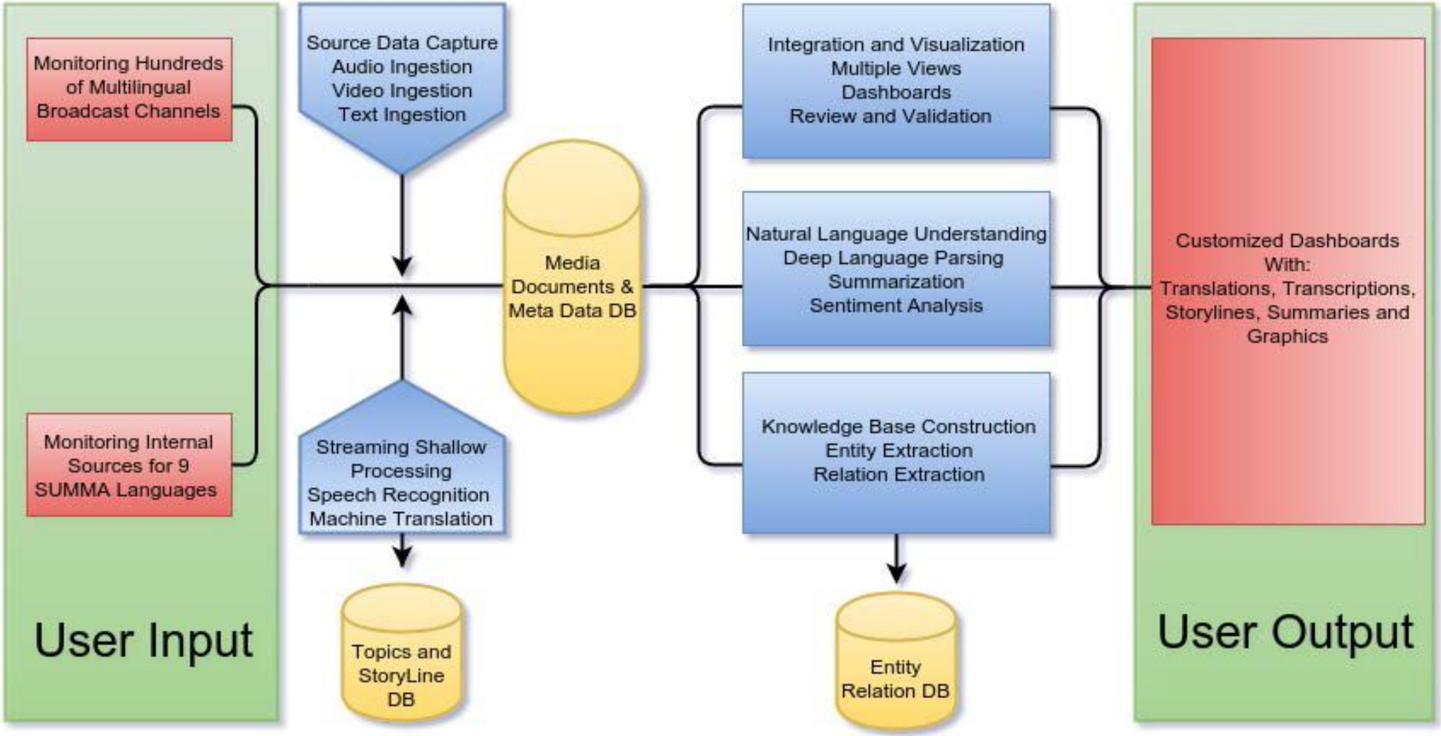
- Speech Recognition
- Speech Generation

NLP Pipelines: SUMMA

SUMMA encompassed 10 technology components:

- ASR: Speech recognition
- Meta: Metadata extraction from broadcast media
- MT: Machine translation
- CT: Streaming implementation of Storyline Clustering and Topic detection
- ETL: Entity Tagging & Linking
- KB: Knowledge Base Construction
- FC: Forecasting and Fact Checking
- SP: Story-level semantic parsing
- SH: Story highlight generation/summarisation

NLP Pipelines: SUMMA



Monitoring Hundreds of Multilingual Broadcast Channels

Monitoring Internal Sources for 9 SUMMA Languages

User Input

Source Data Capture
Audio Ingestion
Video Ingestion
Text Ingestion

Streaming Shallow Processing
Speech Recognition
Machine Translation

Topics and StoryLine DB

Media Documents & Meta Data DB

Integration and Visualization
Multiple Views
Dashboards
Review and Validation

Natural Language Understanding
Deep Language Parsing
Summarization
Sentiment Analysis

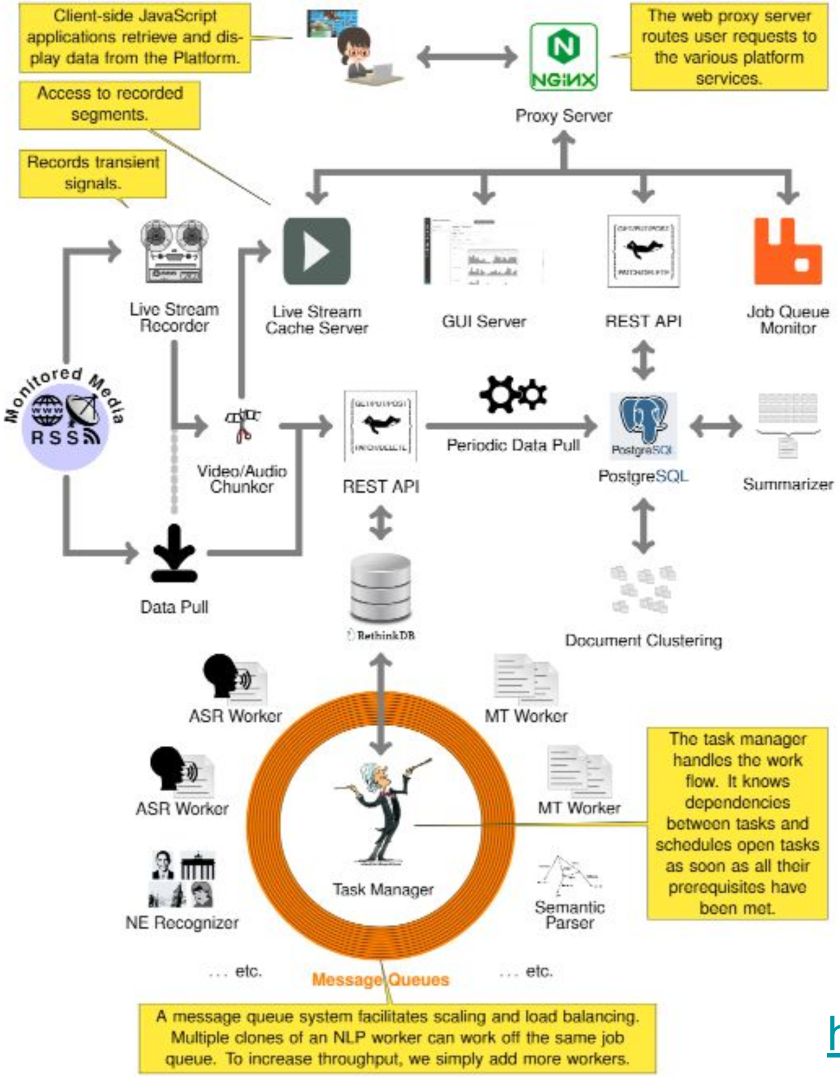
Knowledge Base Construction
Entity Extraction
Relation Extraction

Entity Relation DB

Customized Dashboards With: Translations, Transcriptions, Storylines, Summaries and Graphics

User Output

http://summa-project.eu



Client-side JavaScript applications retrieve and display data from the Platform.

The web proxy server routes user requests to the various platform services.

Access to recorded segments.

Records transient signals.

The task manager handles the work flow. It knows dependencies between tasks and schedules open tasks as soon as all their prerequisites have been met.

A message queue system facilitates scaling and load balancing. Multiple clones of an NLP worker can work off the same job queue. To increase throughput, we simply add more workers.

http://summa-project.eu

SUMMA Dissemination Posters

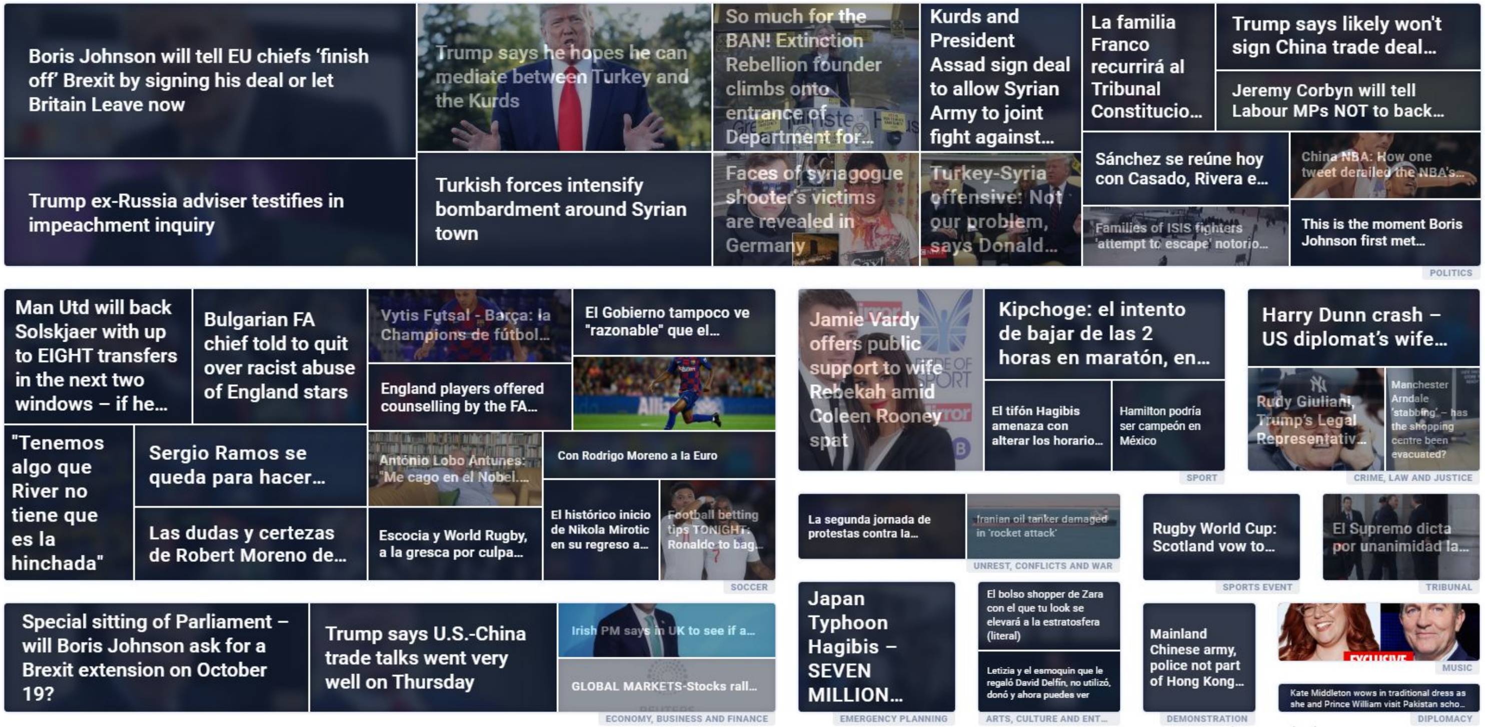
- Open Source SUMMA Platform
- Test Data Supplied by BBC
- Multilingual Topic Detection in News Data
- Summarization & Sentiment classification
- Punctuation Prediction
- Speech recognition
- Machine Translation
- Fact checking
- Entity Tagging & Linking
- Deutsche Welle SUMMA Dataflow
- Multilingual Clustering of Streaming News

NLP pipelines: INSIGHT

The INSIGHT project has as its main objective the validation with the market of a platform to support technology monitoring (*technology intelligence* or *technology watch)* and *horizon scanning* processes, which consist of the capture, analysis, selection and systematic dissemination of information to support the strategic decision making in a company or organization.

For companies, the purpose of technology monitoring is to support strategic decision making regarding the adoption of new technologies or processes or the creation of new products or services. In the case of public entities, where this activity is commonly known as *horizon scanning*, the objective is to define public policies.

NLP pipelines: INSIGHT



Boris Johnson will tell EU chiefs 'finish off' Brexit by signing his deal or let Britain Leave now

Trump ex-Russia adviser testifies in impeachment inquiry

Trump says he hopes he can mediate between Turkey and the Kurds

Turkish forces intensify bombardment around Syrian town

So much for the BAN! Extinction Rebellion founder climbs onto entrance of Department for...

Faces of synagogue shooter's victims are revealed in Germany

Kurds and President Assad sign deal to allow Syrian Army to joint fight against...

Turkey-Syria offensive: Not our problem, says Donald...

La familia Franco recurrirá al Tribunal Constitucio...

Trump says likely won't sign China trade deal...

Jeremy Corbyn will tell Labour MPs NOT to back...

Sánchez se reúne hoy con Casado, Rivera e...

China NBA: How one tweet derailed the NBA's...

Families of ISIS fighters 'attempt to escape' notorio...

This is the moment Boris Johnson first met...

POLITICS

Man Utd will back Solskjaer with up to EIGHT transfers in the next two windows – if he...

Bulgarian FA chief told to quit over racist abuse of England stars

Vytis Futsal - Barça: la Champions de fútbol...

El Gobierno tampoco ve "razonable" que el...

England players offered counselling by the FA...

"Tenemos algo que River no tiene que es la hinchada"

Sergio Ramos se queda para hacer...

Las dudas y certezas de Robert Moreno de...

António Lobo Antunes: "Me cago en el Nobel...

Con Rodrigo Moreno a la Euro

Escocia y World Rugby, a la gresca por culpa...

El histórico inicio de Nikola Mirotic en su regreso a...

Football betting tips TONIGHT: Ronaldo to bag...

SOCCER

Jamie Vardy offers public support to wife Rebekah amid Coleen Rooney spat

Kipchoge: el intento de bajar de las 2 horas en maratón, en...

El tifón Hagibis amenaza con alterar los horario...

Hamilton podría ser campeón en México

SPORT

Harry Dunn crash – US diplomat's wife...

Rudy Giuliani, Trump's Legal Representativ...

Manchester Arndale 'stabbing' – has the shopping centre been evacuated?

CRIME, LAW AND JUSTICE

La segunda jornada de protestas contra la...

Iranian oil tanker damaged in 'rocket attack'

UNREST, CONFLICTS AND WAR

Rugby World Cup: Scotland vow to...

SPORTS EVENT

El Supremo dicta por unanimidad la...

TRIBUNAL

Special sitting of Parliament – will Boris Johnson ask for a Brexit extension on October 19?

Trump says U.S.-China trade talks went very well on Thursday

Irish PM says in UK to see if a...

GLOBAL MARKETS-Stocks rall...

ECONOMY, BUSINESS AND FINANCE

Japan Typhoon Hagibis – SEVEN MILLION...

EMERGENCY PLANNING

El bolso shopper de Zara con el que tu look se elevará a la estratosfera (literal)

Letizia y el esmoquin que le regaló David Delfín, no utilizó, donó y ahora puedes ver

ARTS, CULTURE AND ENT...

Mainland Chinese army, police not part of Hong Kong...

DEMONSTRATION

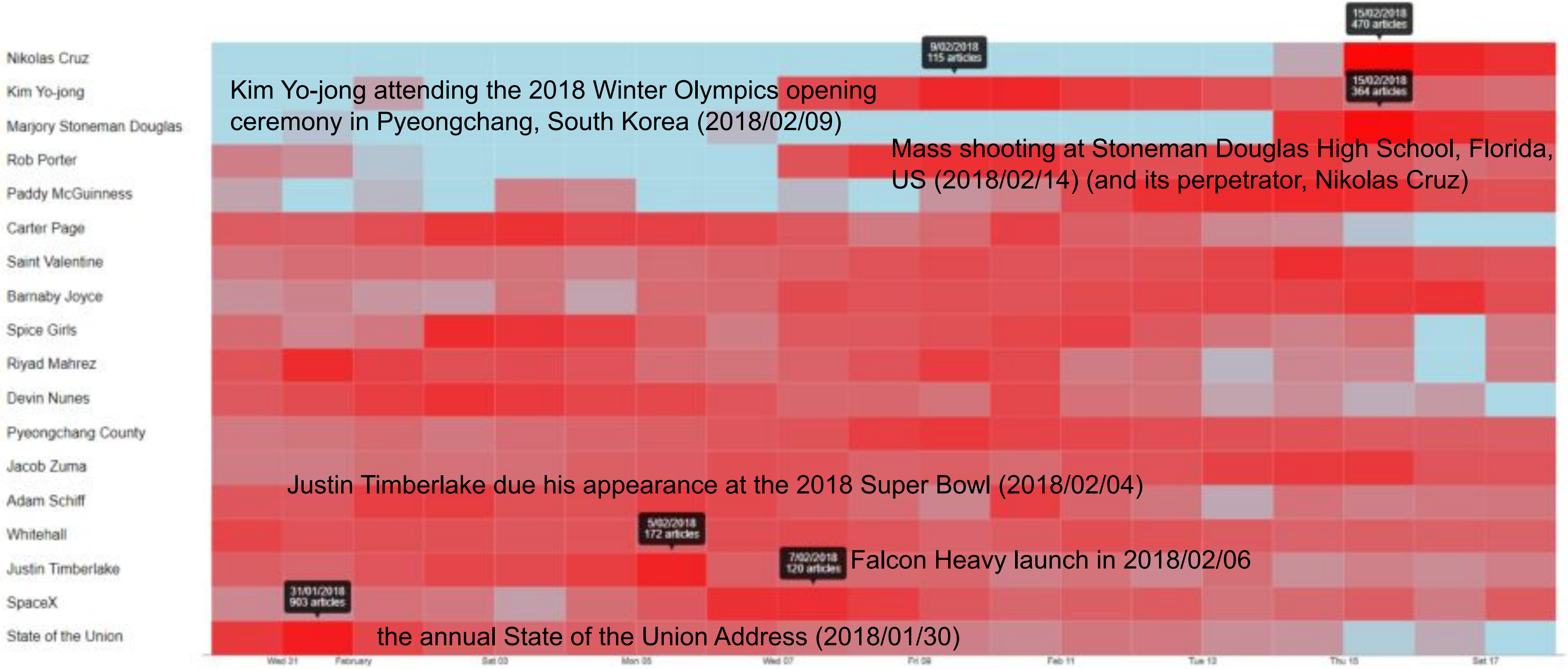
MUSIC

Kate Middleton wows in traditional dress as she and Prince William visit Pakistan scho...

DIPLOMACY

http://staging.insight.priberam.com

NLP Pipelines: SUMMA



Kim Yo-jong attending the 2018 Winter Olympics opening ceremony in Pyeongchang, South Korea (2018/02/09)

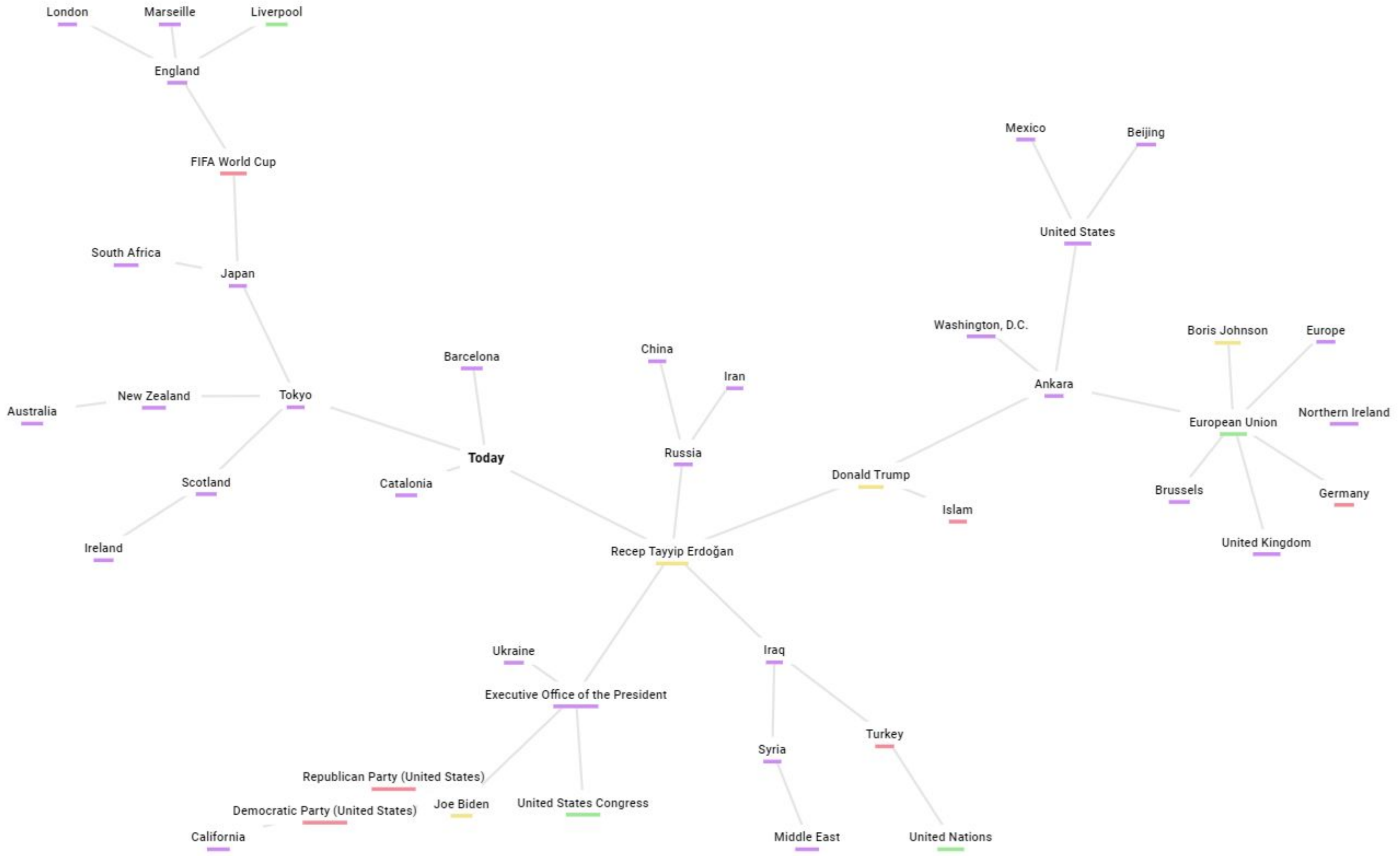
Mass shooting at Stoneman Douglas High School, Florida, US (2018/02/14) (and its perpetrator, Nikolas Cruz)

Justin Timberlake due his appearance at the 2018 Super Bowl (2018/02/04)

Falcon Heavy launch in 2018/02/06

the annual State of the Union Address (2018/01/30)

NLP pipelines: INSIGHT



break

What are Shared Tasks? What’s “state-of-the-art”?

Shared tasks are competitions organized to tackle specific problems, with a particular dataset. Typically, the availability of data is a major issue, and shared-tasks have a dual goal:

- offering a good dataset for researchers to train and test their models;
  - for example, without shared-tasks, would be hard to have access to authentic de-identified clinical records
  - getting access to large amounts of annotated data, which would be expensive to annotate (normally, sites like ‘Mechanical Turk’ are used)
- putting different researchers and teams competing to solve a task, and then being able to compare the effectiveness of the different methods and models against that particular task/dataset.

They are normally sponsored by several institutions, like big tech companies, institutes or universities, which provide the data, the annotation, experience or infrastructure.

What’s trending in NLP from the business perspective?

- Chatbots
  - E-commerce virtual agents (help with FAQs)

The creation of super agents is a growing market (potential for huge staff reduction).

- Virtual Assistants
  - Amazon Alexa
  - Google Assistant

Voice is the new “touch”. Adoption of voice technologies has seen a bigger rate than when touch screens / apps were firstly introduced.
An interesting read is the Reuters Institute report on the subject:
“The Future of Voice and the Implications for News”
https://reutersinstitute.politics.ox.ac.uk/sites/default/files/2018-11/Newman%20-%20Future%20of%20Voice%20FINAL_2.pdf

Voice ecosystem

This is not just about smart speakers

Amazon Echo

Apple Homepod

Google Home

Microsoft Cortana

Google Home Hub, Facebook Portal, Amazon Show

Bose headphones

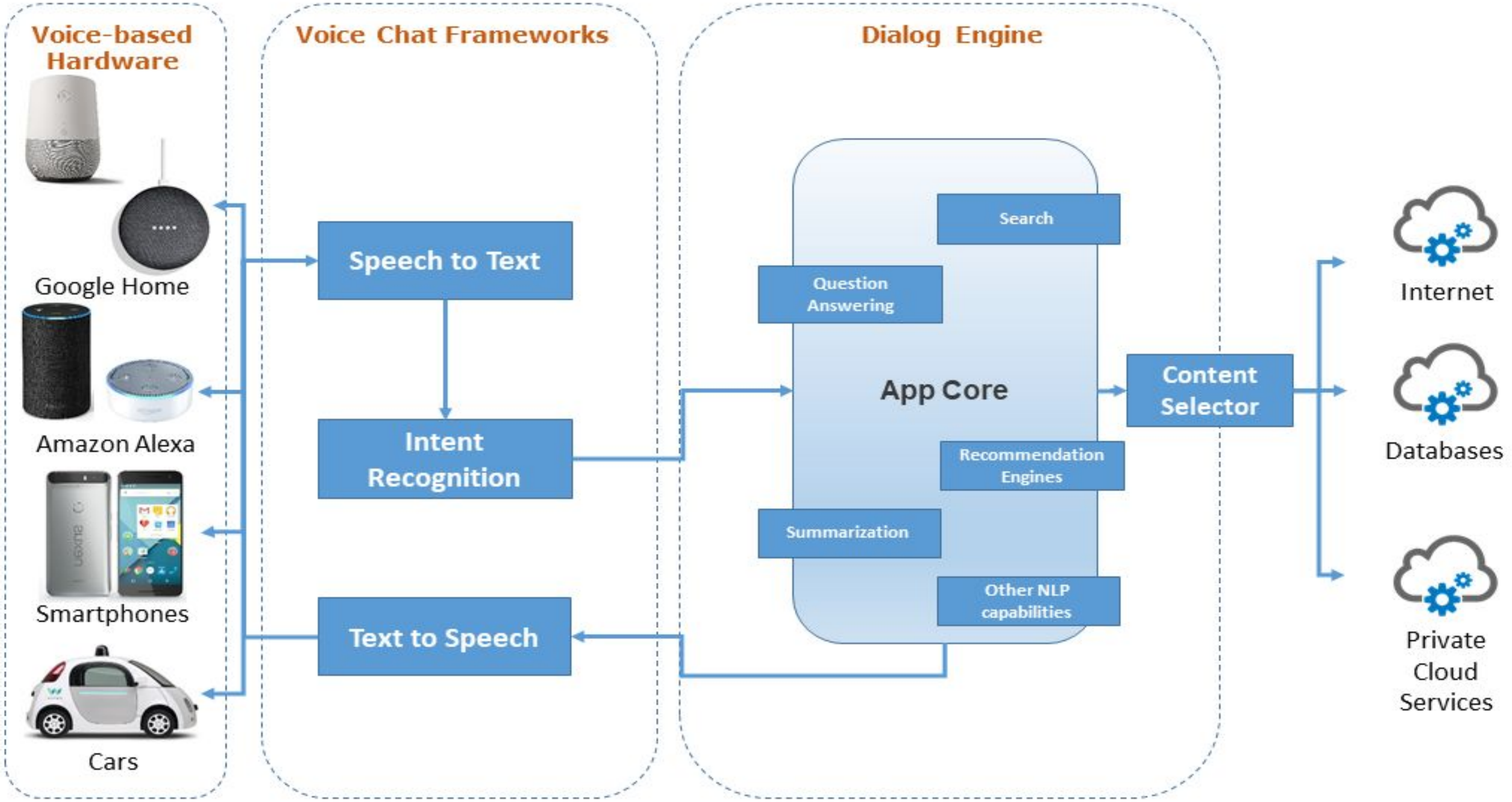
BMW Assistant

Echo Auto

Alexa microwave

From “The Future of Voice and the Implications for News”

Standard Voice Assistant Platform Architecture



Current challenges in NLP

1) Natural Language Understanding

2) NLP in low-resource languages

3) Reasoning in multiple document scenarios

4) Datasets, problems and evaluation

5) Abstractive text summarization

Top 4 problems as identified by several NLP experts.
Survey by Sebastian Ruder, 2019

Current challenges in NLP

http://ruder.io/4-biggest-open-problems-in-nlp/

Challenge #1: Natural Language Understanding

“I think the biggest open problems are all related to natural language understanding. [...] **we should develop systems that read and understand text the way a person does**, by forming a representation of the world of the text, with the agents, objects, settings, and the relationships, goals, desires, and beliefs of the agents, and everything else that humans create to understand a piece of text. Until we can do that, all of our progress is in improving our systems’ ability to do pattern matching.”

– Kevin Gimpel

Challenge #2: NLP in low-resource languages

“Dealing with low-data settings (low-resource languages, dialects (including social media text "dialects"), domains, etc.). This is not a completely "open" problem in that there are already a lot of promising ideas out there; **but we still don't have a universal solution to this universal problem**.”

– Karen Livescu

Challenge #3: Reasoning in multi-doc scenarios

“Representing large contexts efficiently. **Our current models are mostly based on recurrent neural networks, which cannot represent longer contexts well.** […] The stream of work on graph-inspired RNNs is potentially promising, though has only seen modest improvements and has not been widely adopted due to them being much less straightforward to train than a vanilla RNN.”

– Isabelle Augenstein

Challenge #4: Datasets, problems and evaluation

“Perhaps the biggest problem is to **properly define the problems themselves.** And by properly defining a problem, I mean building **datasets and evaluation** procedures that are appropriate to measure our progress towards concrete goals. Things would be easier if we could reduce everything to Kaggle style competitions!”

– Mikel Artetxe

Challenge #5: Abstractive text summarization

While extractive text summarization is relatively “solved”, human-level quality on abstractive text summarization is still very much an open problem.

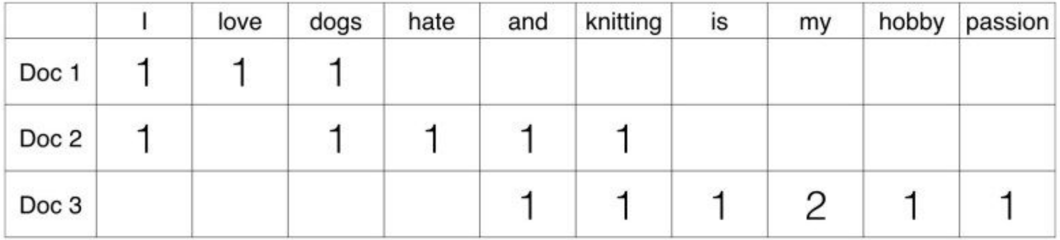
It’s relatively difficult to allow the model to generate text with words that may come from outside the documents being summarized, without potentially introducing inaccurate information in the output text.

Bag-of-Words Model

The bag-of-words model is a simplifying representation used in NLP & IR (Natural Language Processing and Information Retrieval).

A text (can either be a sentence or a document) is represented as a set of its words (multiset), disregarding grammar and word order, but keeping multiplicity.

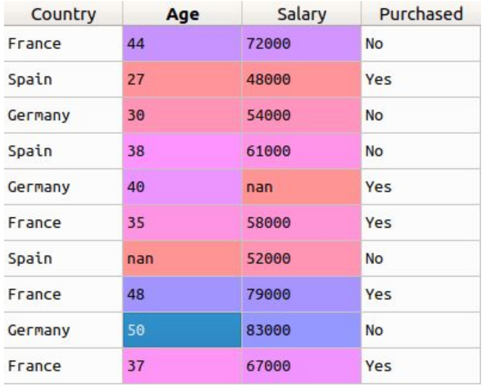
> *A multiset, unlike a set, allows for multiple instances for each of its elements (i.e., it keeps their multiplicity).*



	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

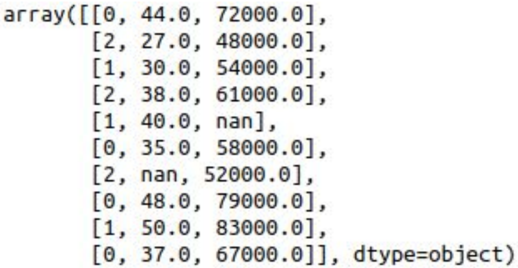
This is a one-hot encoding

Label Encoder vs One-hot Encoder



Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40	nan	Yes
France	35	58000	Yes
Spain	nan	52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
x[:, 0] = labelencoder.fit_transform(x[:, 0])
```

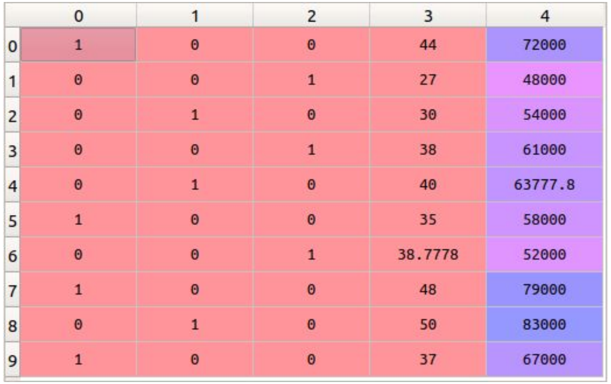


```
array([[0, 44.0, 72000.0],
       [2, 27.0, 48000.0],
       [1, 30.0, 54000.0],
       [2, 38.0, 61000.0],
       [1, 40.0, nan],
       [0, 35.0, 58000.0],
       [2, nan, 52000.0],
       [0, 48.0, 79000.0],
       [1, 50.0, 83000.0],
       [0, 37.0, 67000.0]], dtype=object)
```

The problem here is, since there are different numbers in the same column, the model will misunderstand the data to be in some kind of order, 0 < 1 < 2.

But this isn't the case at all.
To overcome this problem, we use One-hot encoding.

```
from sklearn.preprocessing import OneHotEncoder
onehotencoder = OneHotEncoder(categorical_features = [0])
x = onehotencoder.fit_transform(x).toarray()
```



	0	1	2	3	4
0	1	0	0	44	72000
1	0	0	1	27	48000
2	0	1	0	30	54000
3	0	0	1	38	61000
4	0	1	0	40	63777.8
5	1	0	0	35	58000
6	0	0	1	38.7778	52000
7	1	0	0	48	79000
8	0	1	0	50	83000
9	1	0	0	37	67000

Example from **Kirill Eremenko** and **Hadelin de Ponteves** (Udemy)

TF-IDF

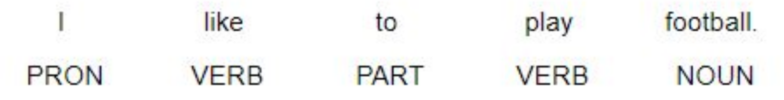
One popular pre-processing step is called term frequency-inverse document frequency. Not every word provides the same information, as some of them occur in every document and they are not salient (i.e., they do not convey special meaning to a document).

Therefore, in a document representation, each word is weighted with its frequency in the document, and inversely proportional to its frequency in a large collection of documents (corpus).

Text tagging as additional features

Part-of-Speech Tagging (POS):

- A part of speech (PoS or POS) is a category of words (lexical items) that have similar grammatical properties
- Words that are assigned to the same part of speech generally display similar syntactic behavior
  - they play similar roles within the grammatical structure of sentences
- Typical POS are: noun, verb, adjective, adverb, pronoun, preposition, conjunction, interjection, and sometimes numeral, article, or determiner.



I	like	to	play	football.
PRON	VERB	PART	VERB	NOUN

Text tagging as additional features

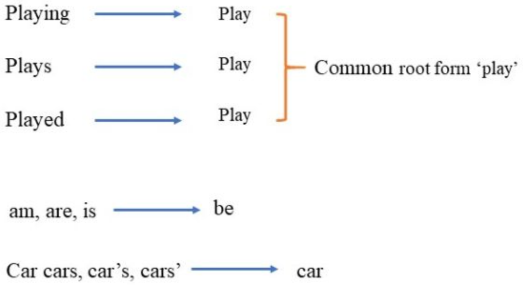
Named Entity Recognition (NER):

Detecting named entities like people, organizations or places (countries, cities, etc.), provides extra features that carry more information about the meaning of the sentence/document, than the other words

Stemming / Lemmatization:

Stemming and lemmatization are text normalization techniques:

- Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the language.
- Lemmatization, unlike stemming, reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called lemma. A lemma (plural lemmas or lemmata) is the canonical form, dictionary form, or citation form of a set of words.



Coding break

https://github.com/davidalbertonogueira/NLP-tutorials/blob/master/code/SpacyPOSandNER.ipynb

Word embeddings versus one-hot encoders

- It’s not enough to map just a word to a number (index in a vector).
    - That word’s representation would be a vector of zeros with a one in its index.
- With one-hot encoding, words with the same meaning (synonyms) are orthogonal to each other.
- Words are encoded from a pre-defined and finite set of possible words.
    - that finite set is the size of the vocabulary.

Contrary to BoW, which uses all the corpus words (or a large top n, most frequent words), word embeddings solve the high dimensionality problem by representing words in a smaller vector space, normally around 200 or 300 dimensions.

Word embeddings

- Word embeddings rationale: words that appear in the same context, surrounded by the same words, must have a similar meaning.
  - word’s representations are vectors of a few hundred dimensions, of floating points.
  - similar words/synonyms have similar vectors (small euclidean distance).

Skip-gram vs CBOW

CBOW is learning to predict the word by the context.

The skip-gram model is designed to predict the context.

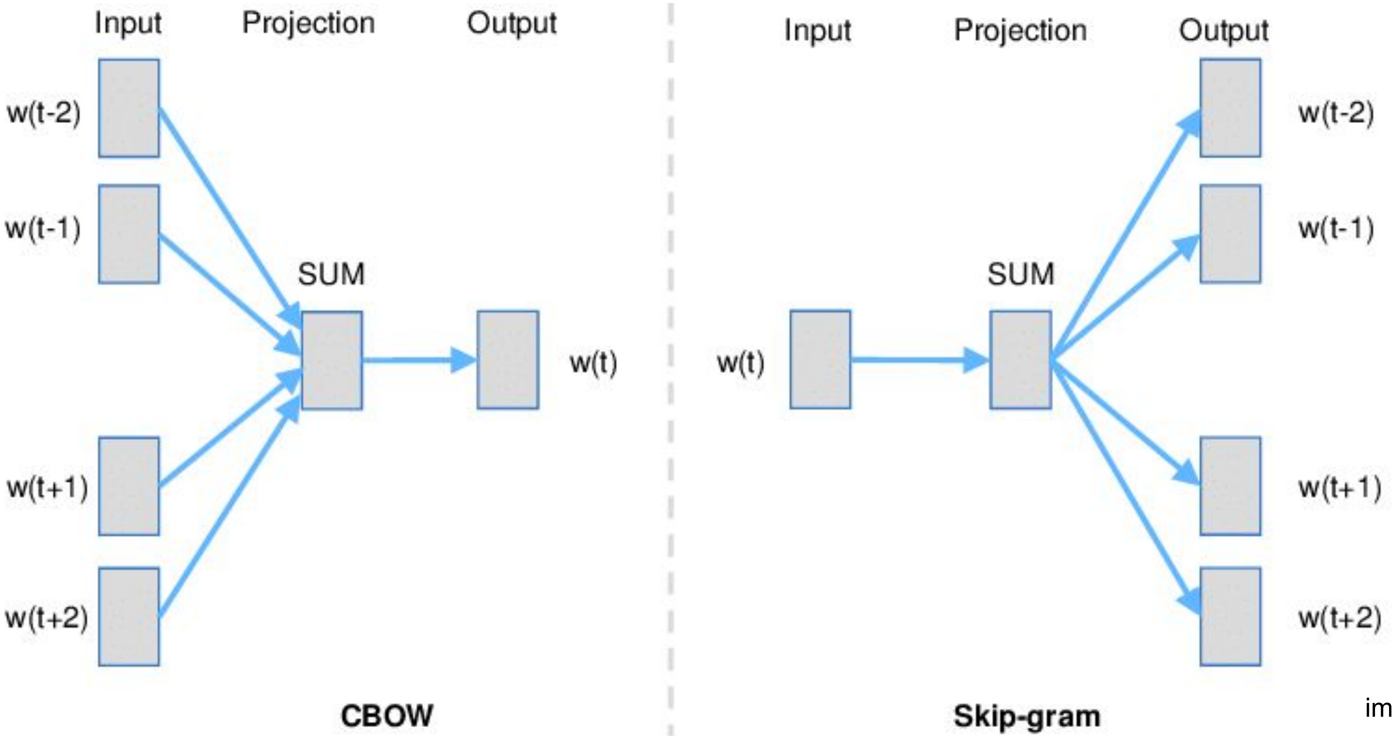
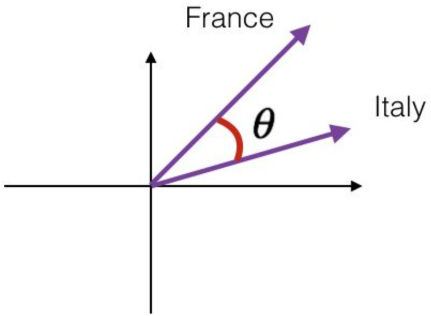


image by Daniel Braun

Cosine similarity

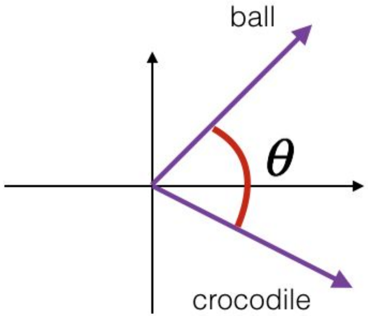
Metric for measuring distance when the magnitude of the vectors does not matter. Its valid both for sparse (BoW) and dense vectors (embeddings). If the vectors are word counts, the fact that it doesn't take into account magnitude, means that it works well comparing documents of uneven lengths.



France and Italy are quite similar

θ is close to 0°

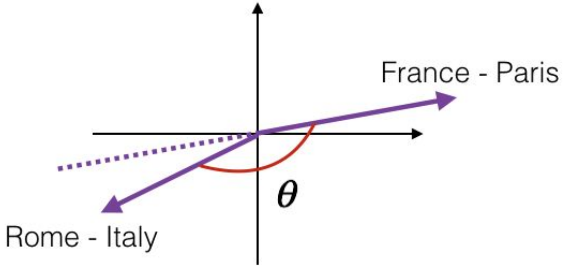
$\cos(\theta) \approx 1$



ball and crocodile are not similar

θ is close to 90°

$\cos(\theta) \approx 0$



the two vectors are similar but opposite
the first one encodes (city - country)
while the second one encodes (country - city)

θ is close to 180°

$\cos(\theta) \approx -1$

Popular pre-trained word embedding models

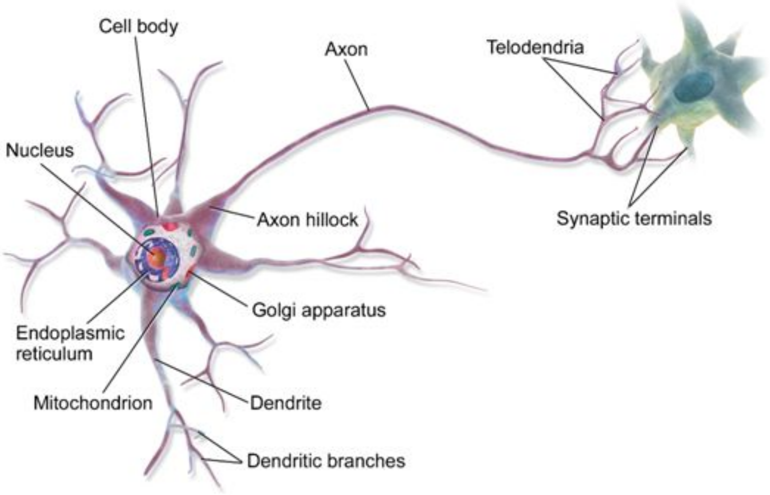
- Word2Vec (by Google)

- GloVe (by Stanford)

- fastText (by Facebook)

Intro to neural networks

- First artificial neuron produced 70 years ago
- Objective: mimic brain cells

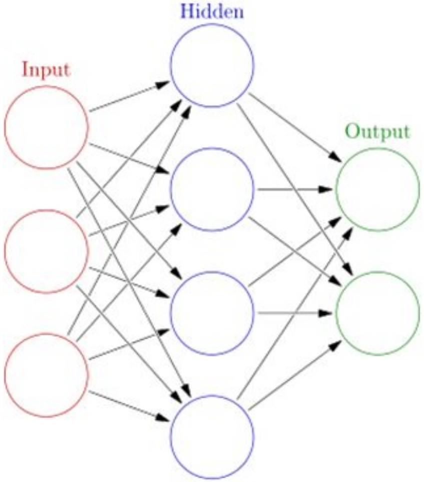


Neural networks: definition

Neural networks are composed of:

- nodes (which mimic neurons) connected together through
- edges (which mimic synapses).

Form a network that mimics a biological brain neural network.



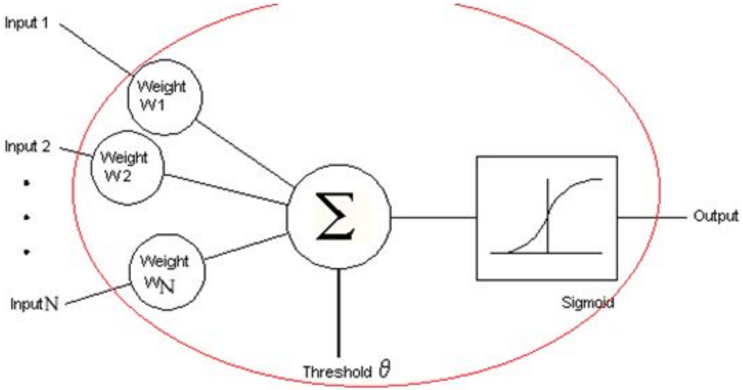
Building a neural network: basic block

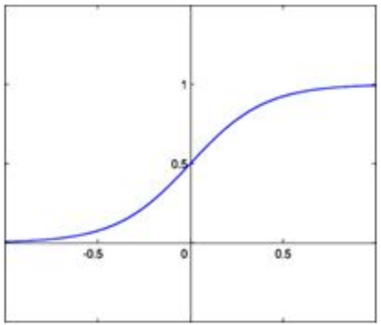
- The basic block of a neural network:

1. Multiple input variables
2. Each input is multiplied by a weight value
3. Summation of such products

   input1 * W1 + input2 * W2 + … + inputN * WN = summation

4. They pass through a sigmoid/threshold function (activation function) -> [0;1]
5. Output is obtained



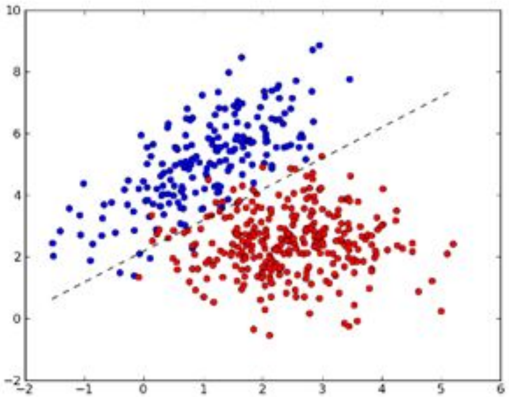


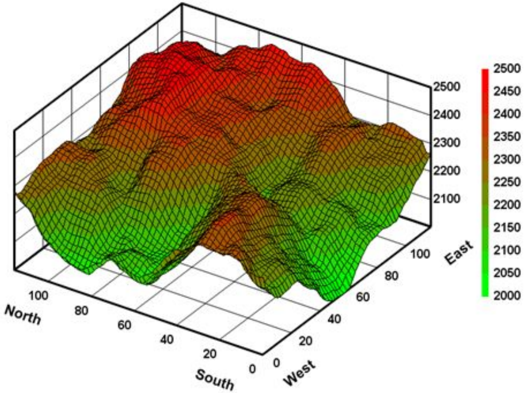
Neural networks: Applications

A single node can perform linear classification and regression:

Ax + by + cz = K (linear equation)

A single-node can only be used to represent linearly separable functions.





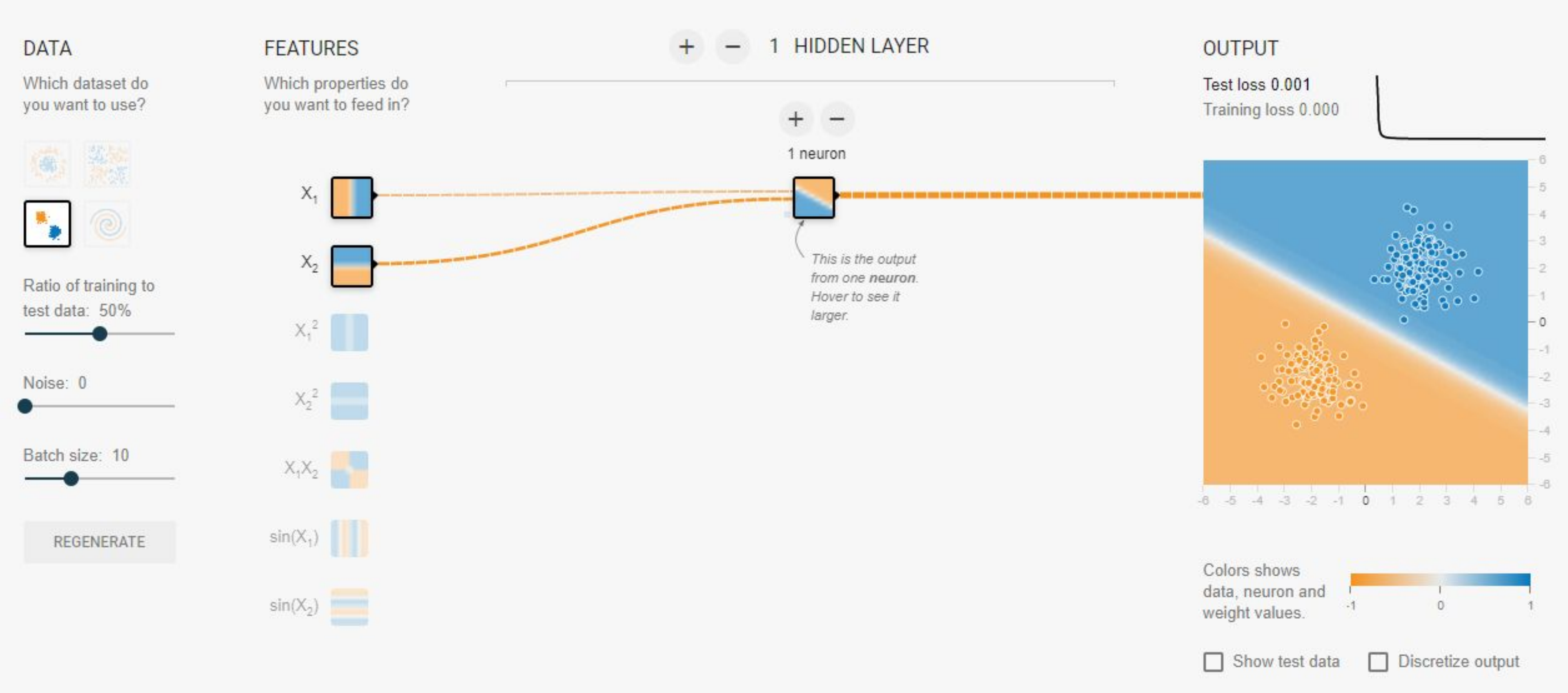
Adding multiple nodes allow us to solve non-linear problems:

it can draw shapes in high-dimensional space and overcome the limitation of linear separability.

Playing with Neural Networks

https://playground.tensorflow.org

Playing with Neural Networks



DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

1 HIDDEN LAYER

1 neuron

*This is the output from one **neuron**. Hover to see it larger.*

OUTPUT

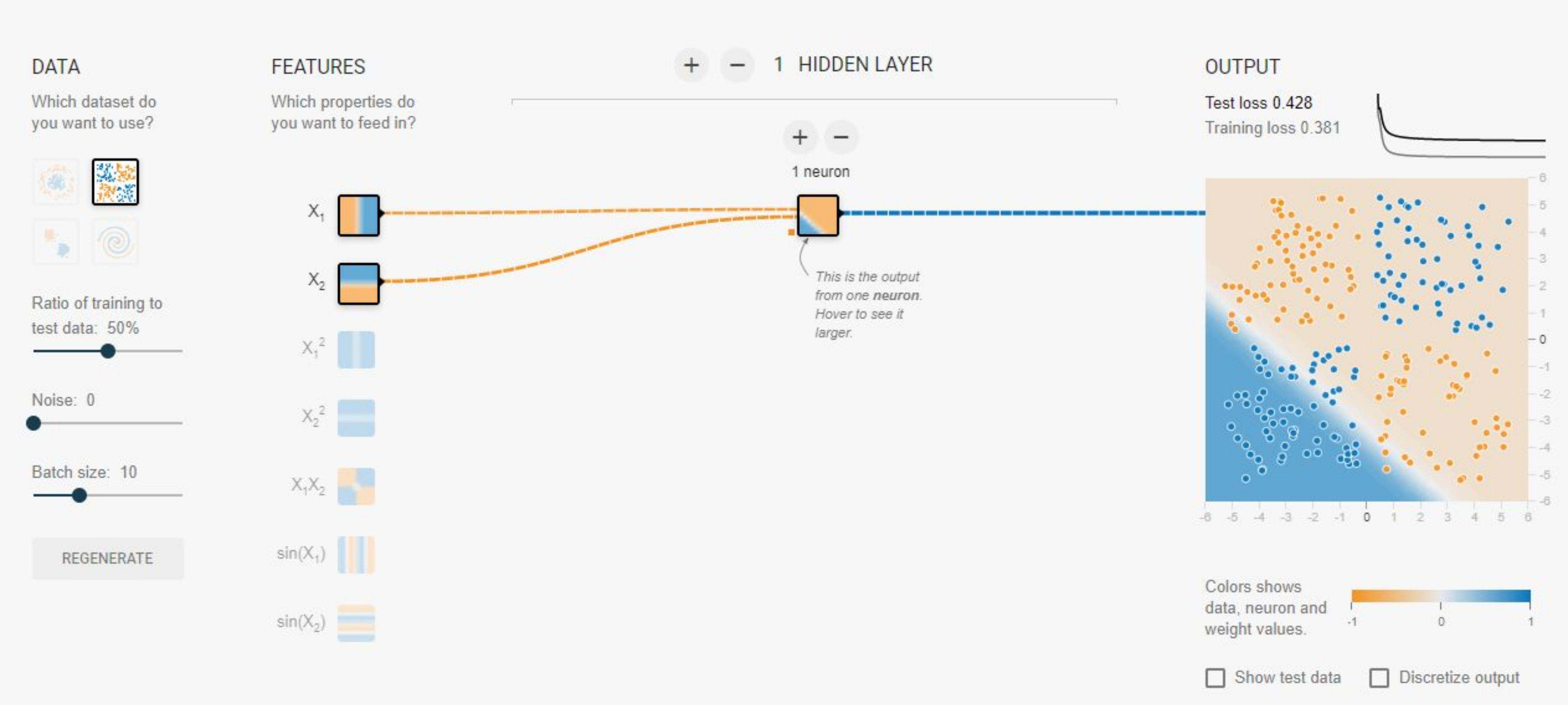
Test loss 0.001

Training loss 0.000

Colors shows data, neuron and weight values.

□ Show test data □ Discretize output

Playing with Neural Networks



DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

+ − 1 HIDDEN LAYER

+ − 1 neuron

*This is the output from one **neuron**. Hover to see it larger.*

OUTPUT

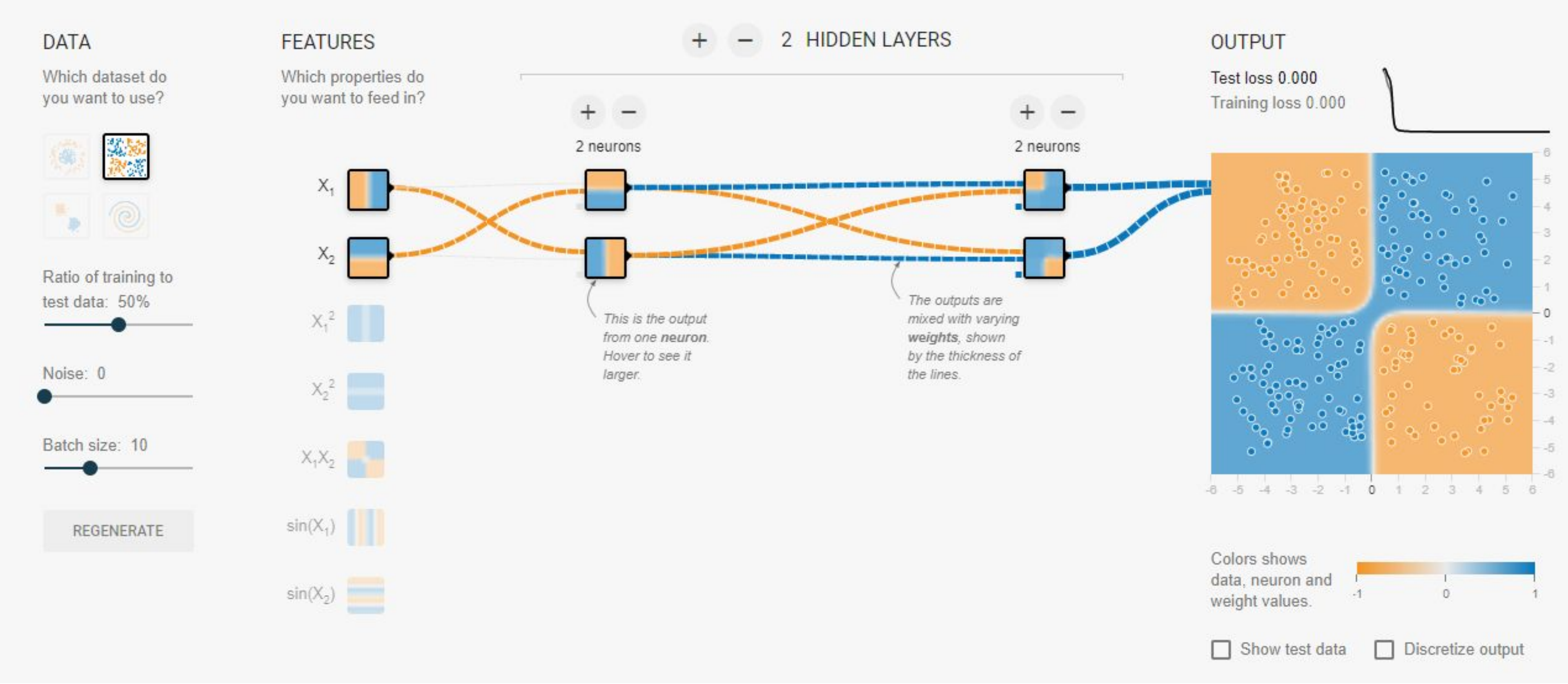
Test loss 0.428

Training loss 0.381

Colors shows data, neuron and weight values.

□ Show test data □ Discretize output

Playing with Neural Networks



DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

2 HIDDEN LAYERS

2 neurons

2 neurons

*This is the output from one **neuron**. Hover to see it larger.*

*The outputs are mixed with varying **weights**, shown by the thickness of the lines.*

OUTPUT

Test loss 0.000

Training loss 0.000

Colors shows data, neuron and weight values.

□ Show test data □ Discretize output

Backpropagation

$$L(t, y) = (t - y)^2 = E,$$

For each neuron:

$$o_j = \varphi(\text{net}_j) = \varphi\left(\sum_{k=1}^{n} w_{kj} o_k\right),$$

φ is the non-linear and differentiable activation function

$$\varphi(z) = \frac{1}{1 + e^{-z}} \qquad \frac{d\varphi(z)}{dz} = \varphi(z)(1 - \varphi(z))$$

Using the chain rule:

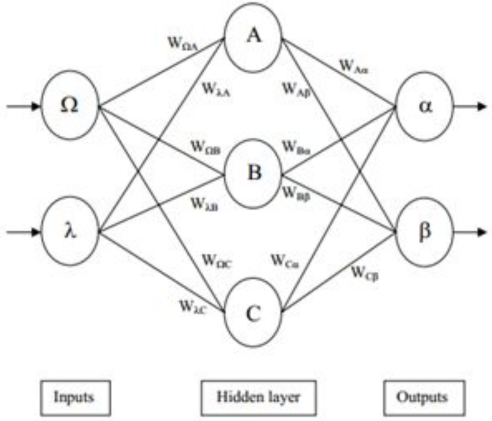
$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \text{net}_j}\frac{\partial \text{net}_j}{\partial w_{ij}}$$

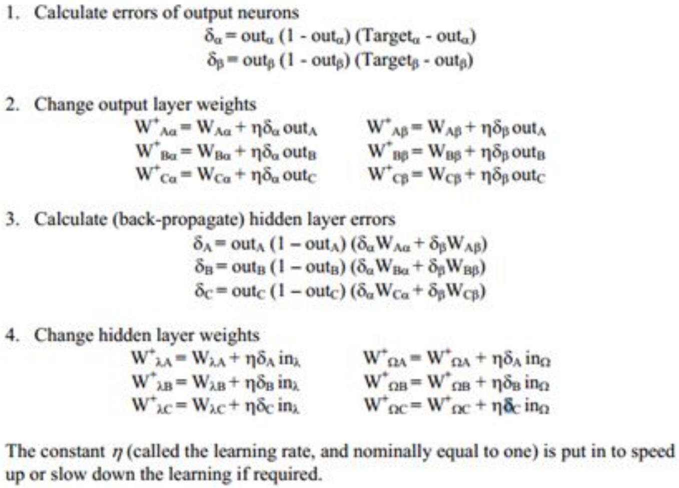
$$\frac{\partial \text{net}_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}\left(\sum_{k=1}^{n} w_{kj} o_k\right) = \frac{\partial}{\partial w_{ij}} w_{ij} o_i = o_i.$$

$$\frac{\partial o_j}{\partial \text{net}_j} = \frac{\partial}{\partial \text{net}_j}\varphi(\text{net}_j) = \varphi(\text{net}_j)(1 - \varphi(\text{net}_j))$$

$$\frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial y} = \frac{\partial}{\partial y}\frac{1}{2}(t - y)^2 = y - t \qquad \frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial y}$$

$$\delta_j = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \text{net}_j} = \begin{cases} (o_j - t_j)o_j(1 - o_j) & \text{if } j \text{ is an output neuron,} \\ (\sum_{\ell \in L} w_{j\ell}\delta_\ell)o_j(1 - o_j) & \text{if } j \text{ is an inner neuron.} \end{cases}$$





1. Calculate errors of output neurons

$$\delta_\alpha = \text{out}_\alpha\ (1 - \text{out}_\alpha)\ (\text{Target}_\alpha - \text{out}_\alpha)$$
$$\delta_\beta = \text{out}_\beta\ (1 - \text{out}_\beta)\ (\text{Target}_\beta - \text{out}_\beta)$$

2. Change output layer weights

$$W^{+}_{A\alpha} = W_{A\alpha} + \eta\delta_\alpha\,\text{out}_A \qquad W^{+}_{A\beta} = W_{A\beta} + \eta\delta_\beta\,\text{out}_A$$
$$W^{+}_{B\alpha} = W_{B\alpha} + \eta\delta_\alpha\,\text{out}_B \qquad W^{+}_{B\beta} = W_{B\beta} + \eta\delta_\beta\,\text{out}_B$$
$$W^{+}_{C\alpha} = W_{C\alpha} + \eta\delta_\alpha\,\text{out}_C \qquad W^{+}_{C\beta} = W_{C\beta} + \eta\delta_\beta\,\text{out}_C$$

3. Calculate (back-propagate) hidden layer errors

$$\delta_A = \text{out}_A\ (1 - \text{out}_A)\ (\delta_\alpha W_{A\alpha} + \delta_\beta W_{A\beta})$$
$$\delta_B = \text{out}_B\ (1 - \text{out}_B)\ (\delta_\alpha W_{B\alpha} + \delta_\beta W_{B\beta})$$
$$\delta_C = \text{out}_C\ (1 - \text{out}_C)\ (\delta_\alpha W_{C\alpha} + \delta_\beta W_{C\beta})$$

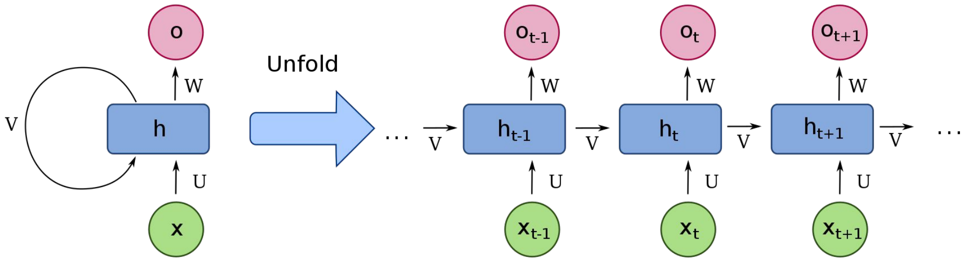
4. Change hidden layer weights

$$W^{+}_{\lambda A} = W_{\lambda A} + \eta\delta_A\,\text{in}_\lambda \qquad W^{+}_{\Omega A} = W^{+}_{\Omega A} + \eta\delta_A\,\text{in}_\Omega$$
$$W^{+}_{\lambda B} = W_{\lambda B} + \eta\delta_B\,\text{in}_\lambda \qquad W^{+}_{\Omega B} = W^{+}_{\Omega B} + \eta\delta_B\,\text{in}_\Omega$$
$$W^{+}_{\lambda C} = W_{\lambda C} + \eta\delta_C\,\text{in}_\lambda \qquad W^{+}_{\Omega C} = W^{+}_{\Omega C} + \eta\delta_C\,\text{in}_\Omega$$

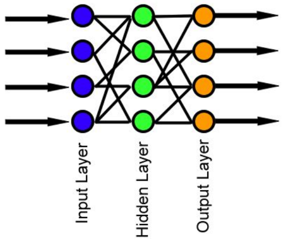
The constant η (called the learning rate, and nominally equal to one) is put in to speed up or slow down the learning if required.

RNNs, LSTMs and CNNs

RNNs are a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence.

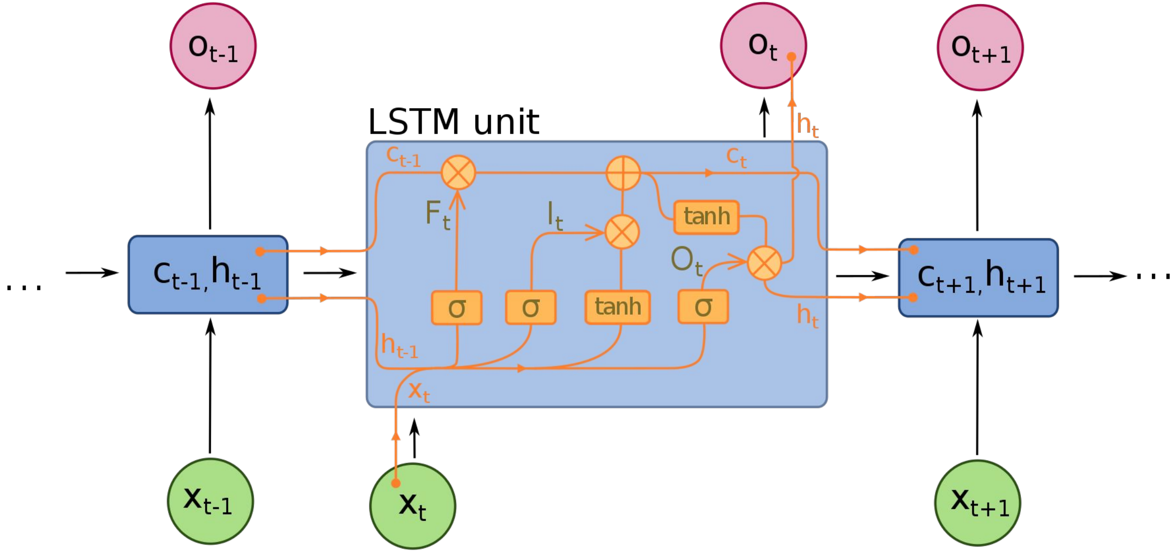


Contrary to the feed forward networks, shown previously, in which information always moves one direction. it never goes backwards.



RNNs, LSTMs and CNNs

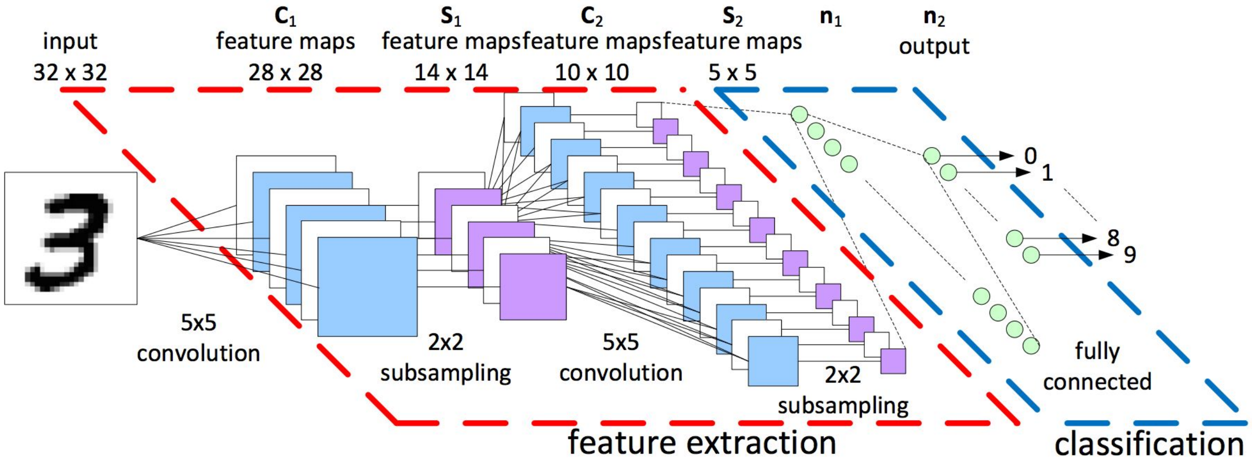
Long short-term memory (LSTM) is a variant of the RNNs.
LSTM prevents backpropagated errors from vanishing or exploding (see *vanishing gradient problem*). Instead, errors can flow backwards through unlimited numbers of virtual layers unfolded in space.



RNNs, LSTMs and CNNs

CNNs are a class of deep neural networks, popularized by their effectiveness in the image processing domain.

Convolutional networks are neural networks that use mathematical operation convolution in place of general matrix multiplication in at least one of their layers.



Transformers

Equivalent to LSTMs and CNNs

Paper: Attention Is All You Need

https://arxiv.org/pdf/1706.03762.pdf

http://nlp.seas.harvard.edu/2018/04/03/attention.html

https://github.com/davidalbertonogueira/annotated-transformer/

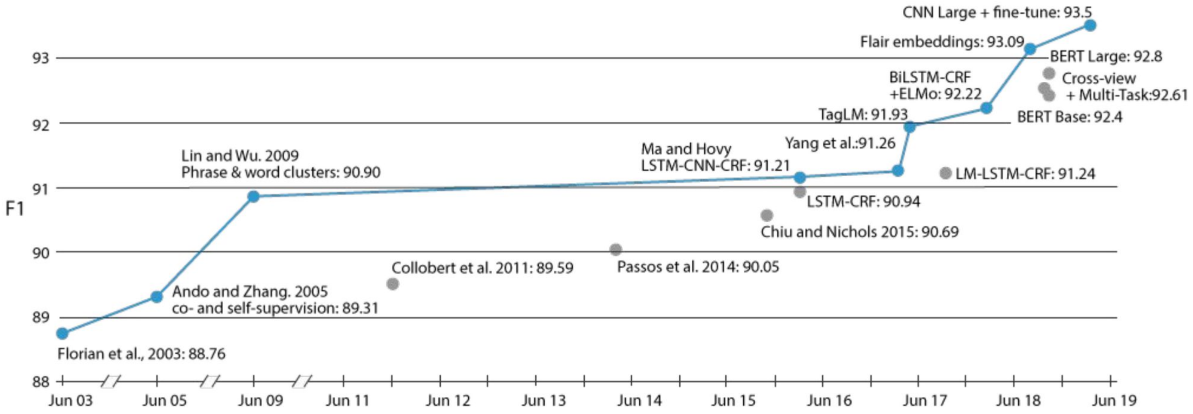
State-of-art Language Models

- ELMo (from AI2) : https://github.com/allenai/bilm-tf
- BERT (from Google) : https://github.com/google-research/bert
- GPT (from OpenAI) : https://github.com/openai/finetune-transformer-lm
- GPT-2 (from OpenAI) : https://blog.openai.com/better-language-models/
- Transformer-XL (from Google/CMU) : https://github.com/kimiyoung/transformer-xl
- XLNet (from Google/CMU) : https://github.com/zihangdai/xlnet/
- XLM (from Facebook) : https://github.com/facebookresearch/XLM/
- RoBERTa (from Facebook) : https://github.com/pytorch/fairseq/tree/master/examples/roberta
- DistilBERT (from HuggingFace): https://github.com/huggingface/transformers/tree/master/examples/distillation

The State of Transfer Learning in NLP

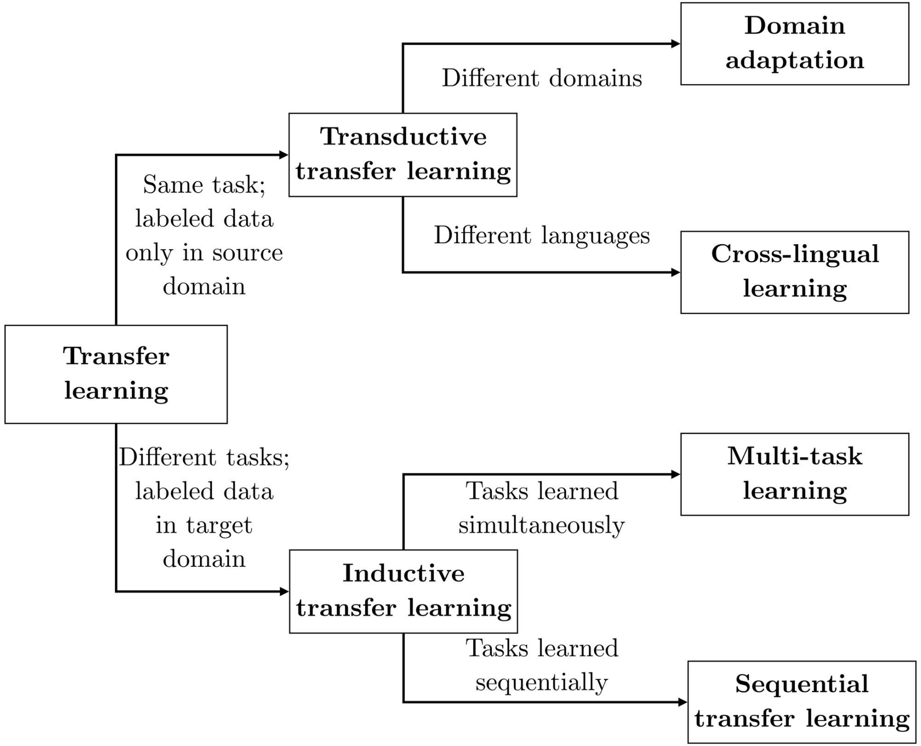
In the span of little more than a year, transfer learning in the form of pretrained language models has become *the norm* in NLP.

They’ve set new state-of-the-art scores in a large array of NLP tasks.



Performance on Named Entity Recognition (NER) on CoNLL-2003 (English). Many improvements last year.

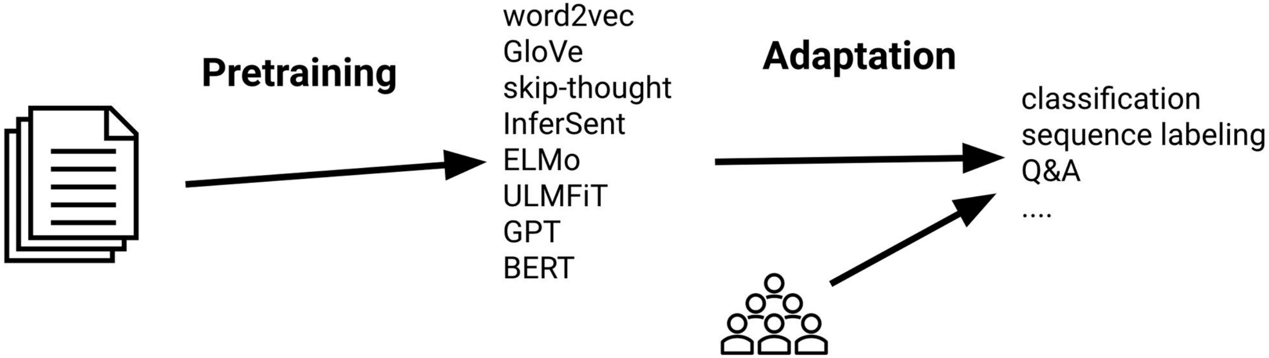
Taxonomy for transfer learning in NLP



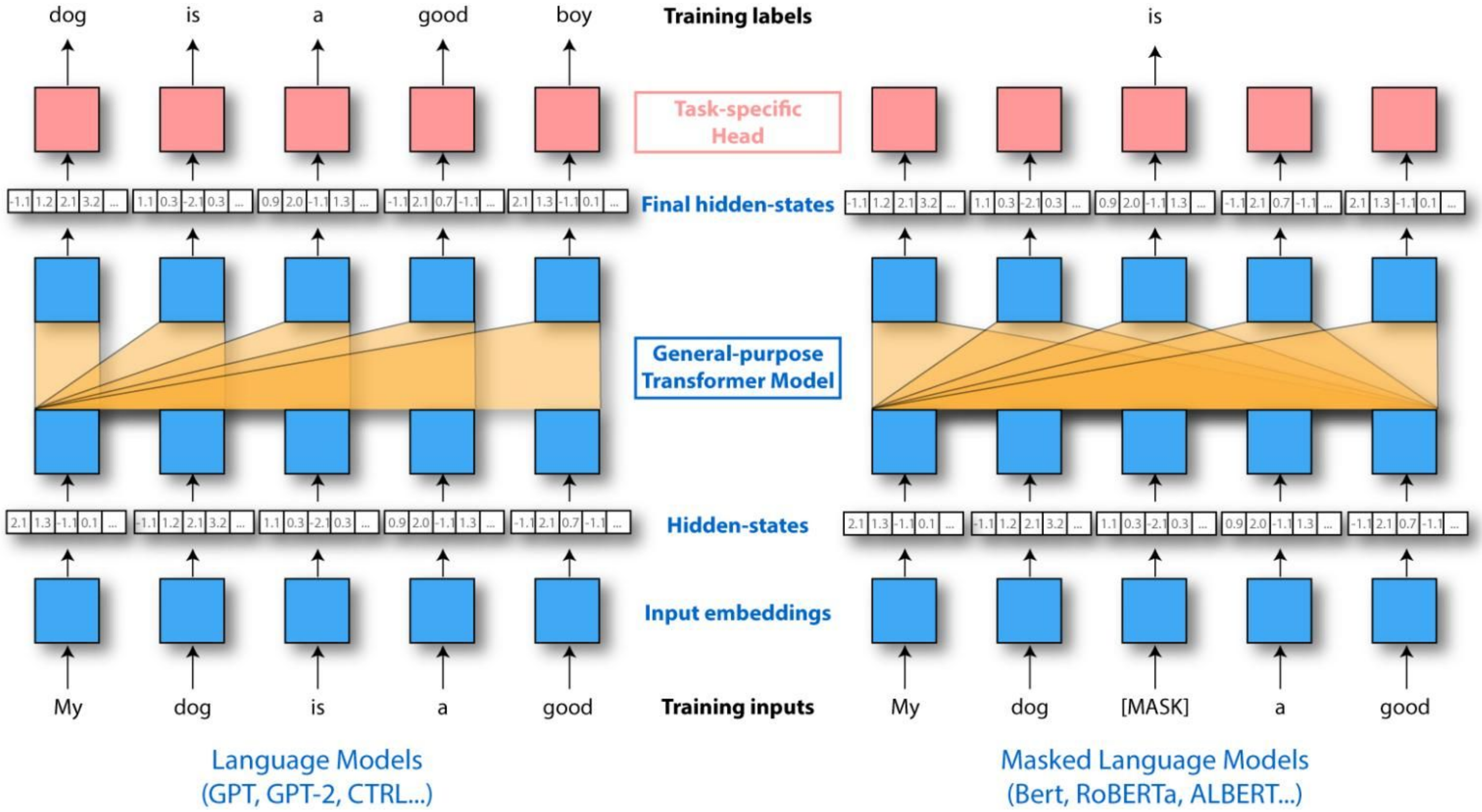
Transfer learning
Same task; labeled data only in source domain
Transductive transfer learning
Different domains
Domain adaptation
Different languages
Cross-lingual learning
Different tasks; labeled data in target domain
Inductive transfer learning
Tasks learned simultaneously
Multi-task learning
Tasks learned sequentially
Sequential transfer learning

Ruder, 2019

General procedure of sequential transfer learning



Masked vs Non-masked Language Models



break

Deep dive: NLP Tasks

- Text/document classification
    - Topic Labelling
    - Sentiment Analysis
- Recommendation systems
- Named Entity Recognition
- Named Entity Linking

Text/document classification

Text classification is a task in which we are interested in detecting certain patterns in the text (documents), allowing us to predict labels/categories.

Popular examples are spam filtering (spam / not-spam), article topic labelling (Politics, Economy, Sports, etc.), etc..

This task is solved with classification algorithms: like Naive Bayes, SVMs and Neural Networks.

Text/Document Classification

- Process of assigning a set of predefined tags or categories to the text according to its content.
- Unstructured data in the form of text is everywhere:
  - emails,
  - chats,
  - web pages,
  - social media,
  - support tickets,
  - survey responses.
- broad applications such as:
  - sentiment analysis (CRM tools);
  - topic labeling (News articles);
  - spam detection (Gmail/Outlook);
  - intent detection (Amazon Alexa/Google Assistant);
  - inappropriate comment detection (discussion forums).

Text/Document Classification

Three different types of text classification systems:

- Rule-based systems
- Machine Learning based systems
- Hybrid systems

Text/Document Classification: Rule-based systems

- One approach is define/characterize each tag/category with a list of words:
    - Topic labelling:
        - Donald Trump, Hillary Clinton, Putin, European Union, Congress, ... -> politics
        - Cristiano Ronaldo, Juventus, football, referee, game, FIFA ... -> sports
    - Sentiment Analysis:
        - love, enjoy, good, amazing, fantastic, recommend, ... -> Positive
        - hate, not, bad, horrible, expensive, overpriced, overhyped, ... -> Negative
- Prediction would be the category/tag with the highest count of occurrences

Text/Document Classification: Rule-based systems

- Disadvantages:
    - require deep knowledge of the domain.
    - time-consuming, since generating rules for a complex system can be quite challenging and usually requires a lot of analysis and testing.
    - difficult to maintain and don’t scale well given that adding new rules can affect the results of the pre-existing rules.

Text/Document Classification: Machine Learning based systems

- Instead of relying on manually crafted rules, Machine Learning based systems learn to make classifications based on past observations (pre-labeled examples as training data).
- Each input text needs to be transformed to a numerical representation - a feature set (in the form of a vector).
    - one of the most frequently used approaches is the bag of words.

Text/Document Classification: Machine Learning based systems

- Some of the most popular machine learning algorithms for creating text classification models include:
  - the naive bayes family of algorithms,
    - based on the probability theory and Bayes' Theorem, to predict the tag of a text
    - it allow us to compute the conditional probabilities of occurrence of two events based on the probabilities of occurrence of each individual event.

$$p(C_k \mid x_1, \dots, x_n) \qquad p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})} \qquad \text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}} \qquad p(C_k \mid x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

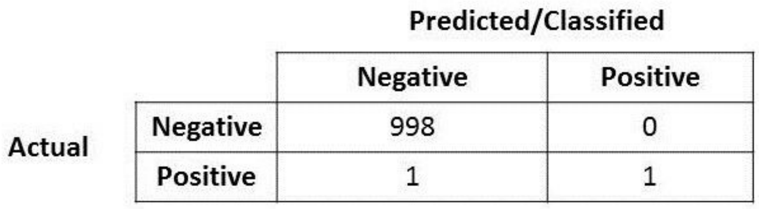
  - support vector machines (SVMs), and
  - neural networks

Coding break

https://github.com/davidalbertonogueira/NLP-tutorials/blob/master/code/TextClassification.ipynb

Data analysis and metric selection

Before starting coding, analyse the data. Quite often, the data will be unbalanced. Check for other particularities.
In spam / online fraud detection, the majority of the e-mails/transactions are not spam/fraudulent. A system that always say that they’re valid, would be right ~99.9% of the time (accuracy).
It matters the metric you’re trying to maximize (may not always be accuracy).
In spam/fraud detection, we want to minimize False Negatives, while not incurring in too many False Positives.



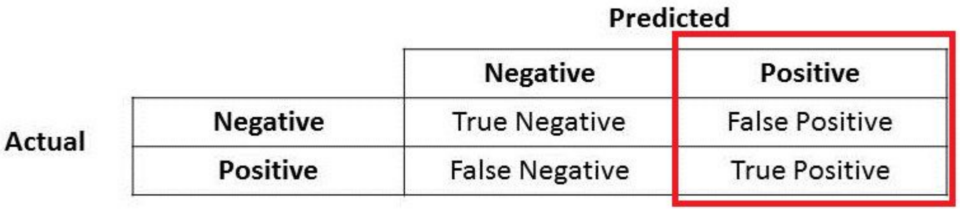
		Predicted/Classified	
		Negative	Positive
Actual	Negative	998	0
	Positive	1	1

Accuracy = 99.9%

This is a confusion matrix.
It shows the positive and negative data points, and the positive and negative predictions.

Good summary: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Precision and Recall

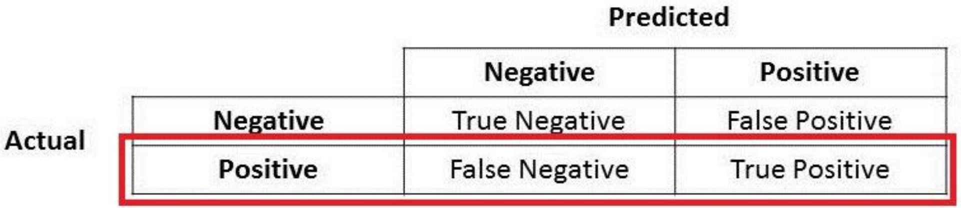


		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive. Precision is a good measure to determine, when the costs of False Positive is high.



		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

Recall considers how many of the Actual Positives the model correctly identifies as being Positive (True Positive). Recall should be the metric when there’s a high cost associated with False Negatives.

Precision penalizes FP, while recall penalizes FN.
A model that never predicts spam/fraud will have zero recall and undefined precision, whereas a model that always predicts fraud will have 100% recall but a very low precision — due to a high number of false positives.

F1-score

F1-score is very popular metric, which is a function of Precision and Recall.

$$F1 = 2 \text{ x } \frac{Precision * Recall}{Precision + Recall}$$

Accuracy, as seen before, can be largely influenced by a large number of True Negatives (which are not the focus).

False Negative and False Positive usually have business costs (tangible & intangible).

F1 Score is a better suited metric if:

- we need to seek a balance between Precision and Recall
- there is an uneven class distribution (large number of Actual Negatives).

Classification metrics and scores

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, classification_report, confusion_matrix
print(f1_score(predictions_NB, Test_Y, average="macro"))
print(precision_score(predictions_NB, Test_Y, average="macro"))
print(recall_score(predictions_NB, Test_Y, average="macro"))
```

The parameter of average could be any of the following. However, for text classification problems you can choose either of Micro or Macro — If you have a imbalance class problem (i.e most of the records/data falls in one category) you can go with Micro.

'binary':Only report results for the class specified by pos_label. This is applicable only if targets (y_{true,pred}) are binary.

'micro':Calculate metrics globally by counting the total true positives, false negatives and false positives.

'macro': Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

'weighted': Calculate metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters ‘macro’ to account for label imbalance; it can result in an F-score that is not between precision and recall.

'samples': Calculate metrics for each instance, and find their average (only meaningful for multilabel classification where this differs from accuracy_score).

Sentiment Analysis

Another common type of text classification is **sentiment analysis**:

- goal: identify the polarity/sentiment of a chunk of text, i.e., the type of opinion it expresses.
- can take the form of:
    - a binary like/dislike rating,
    - positive, neutral and negative rating,
    - or a more granular set of options, such as a star rating from 1 to 5.
- widely used in the context of Customer relationship management (CRM), for automatic evaluation of reviews and survey responses, and social media:
    - analyzing Twitter posts to determine if people liked a particular movie, or
    - extrapolating the general public's opinion of a new brand of Nike shoes from Walmart reviews or
    - analysing current president's support based on replies to his Twitter account.

Popular substasks in Sentiment Analysis

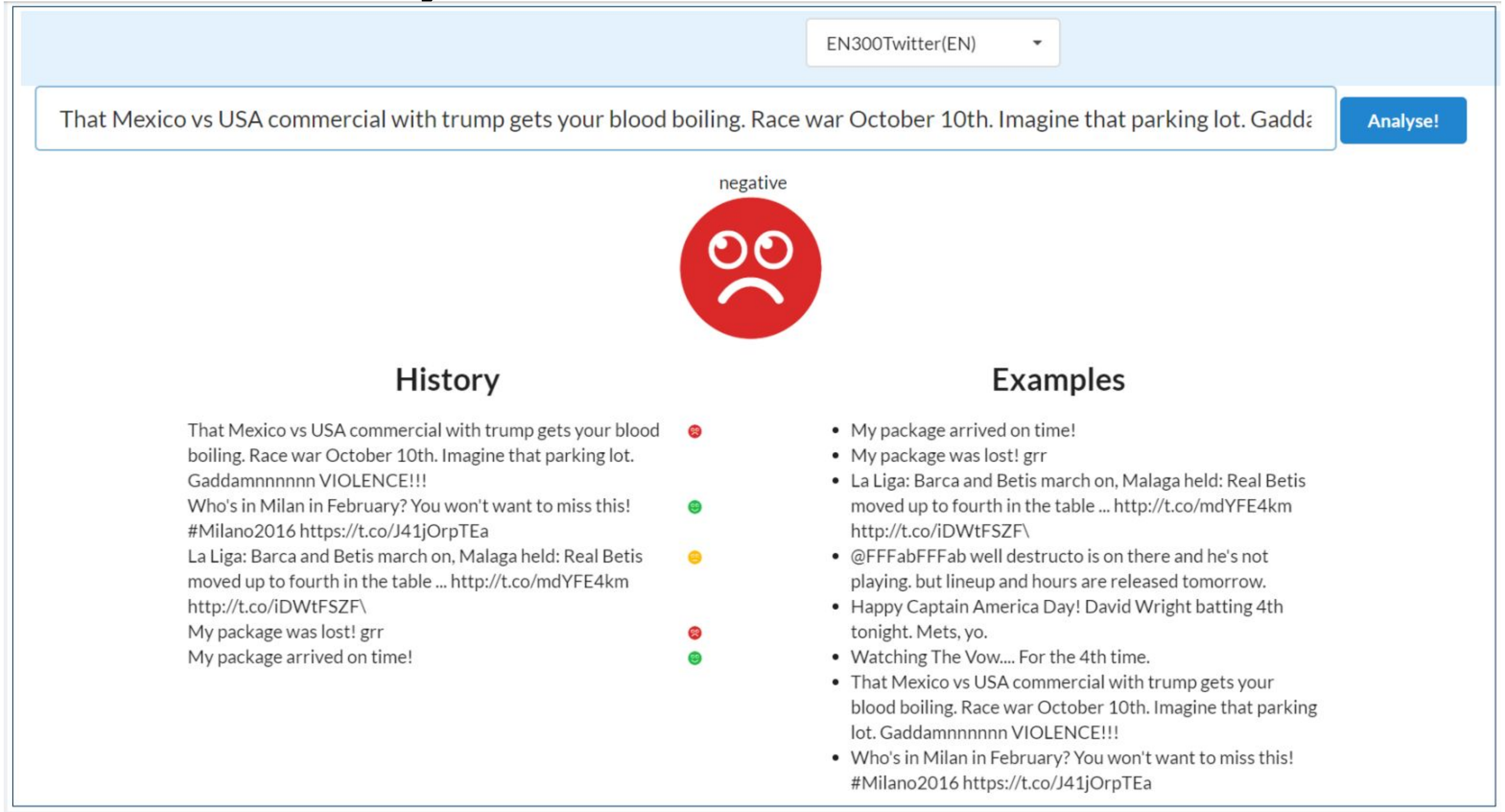
Popular substasks in Sentiment Analysis are:

- Message Polarity Classification: Given a message, classify whether the overall contextual polarity of the message is of positive, negative, or neutral sentiment.
- Topic-Based or Entity-Based Message Polarity Classification: Given a message and a topic or entity, classify the message towards that topic or entity.

A popular Workshop with a specific task for Sentiment analysis is the SemEval (International Workshop on Semantic Evaluation)

2017 overview of such task (Task 4) can be reached at: http://www.aclweb.org/anthology/S17-2088

Sentiment Analysis



That Mexico vs USA commercial with trump gets your blood boiling. Race war October 10th. Imagine that parking lot. Gadd

negative

☹

History

- That Mexico vs USA commercial with trump gets your blood boiling. Race war October 10th. Imagine that parking lot. Gaddamnnnnnn VIOLENCE!!! ☹
- Who's in Milan in February? You won't want to miss this! #Milano2016 https://t.co/J41jOrpTEa ☺
- La Liga: Barca and Betis march on, Malaga held: Real Betis moved up to fourth in the table ... http://t.co/mdYFE4km http://t.co/iDWtFSZF\ 😐
- My package was lost! grr ☹
- My package arrived on time! ☺

Examples

- My package arrived on time!
- My package was lost! grr
- La Liga: Barca and Betis march on, Malaga held: Real Betis moved up to fourth in the table ... http://t.co/mdYFE4km http://t.co/iDWtFSZF\
- @FFFabFFFab well destructo is on there and he's not playing. but lineup and hours are released tomorrow.
- Happy Captain America Day! David Wright batting 4th tonight. Mets, yo.
- Watching The Vow.... For the 4th time.
- That Mexico vs USA commercial with trump gets your blood boiling. Race war October 10th. Imagine that parking lot. Gaddamnnnnnn VIOLENCE!!!
- Who's in Milan in February? You won't want to miss this! #Milano2016 https://t.co/J41jOrpTEa

Coding break

https://github.com/davidalbertonogueira/SentimentAnalysis

break

Recommendation systems

Systems that involve predicting user responses to options.

Examples:

1. Offering news articles to online newspaper readers, based on a prediction of reader interests.

2. Offering customers of an online retailer suggestions about what they might like to buy, based on their past history of purchases and/or product searches.

3. Recommending movies based on what you’ve watched previously and rated positively.

Recommendation systems

INPUT

- Explicitly: User rating
- Implicitly: monitoring user behavior, like movies watched, pages visited, apps downloaded

OUTPUT

- Rating prediction. Aims to fill the missing entries of the user-item rating matrix.
- Ranking prediction. Aims to produce a ranked list of n items to users.

Classic Recommendation systems categories

- **Content-Based** systems focus on properties of items. Similarity of items is determined by measuring the similarity in their properties.
- **Collaborative-Filtering** systems focus on the relationship between users and items. Similarity of items is determined by the similarity of the ratings of those items by the users who have rated both items.
- Hybrid systems.

Limitations:

- data sparsity
- cold-start problems

A Model for Recommendation Systems: Utility Matrix

Items

Users

	HP1	**HP2**	**HP3**	**TW**	**SW1**	**SW2**	**SW3**
A	4			5	1	?	
B	5	5	4				
C				2	4	5	
D		3			2	2	3

Sparse

- Users rate/interact with just a few items
- Each item is rated by only a few users

HP - Harry Potter
SW - Star Wars
TW- Twilight

Utility Matrix

NOTE:

It is not necessary to predict every blank entry in a utility matrix.
Rather, it is only necessary to discover some entries in each row that are likely to be high.
In most applications, the recommendation system does not offer users a ranking of all items, but rather suggests a few that the user should value highly.
It may not even be necessary to find all items with the highest expected ratings, but only to find a large subset of those with the highest ratings.

Content-based systems: item profiles

If the items are movies, relevant features could be:

1. The cast (set of actors).

2. The director.

3. The cinematographer.

4. The year of the movie (some viewers only watch the latest releases).

5. The movie genre (western, comedy, drama, action, etc.).

Content-based systems: item profiles

Each item is represented by a set of important characteristics / features.

If items are documents (articles), represent with sparse (sets of words - BOW) or dense (embeddings) vectors, as seen in Text Classification.

- Sparse representation: n words with the highest TF-IDF scores
  - pick n to be the same for all documents, or
  - n be a fixed percentage of the words in the document
  - pick all words whose TF-IDF scores are above a given threshold

To measure the similarity of two documents:

- Jaccard distance between the sets of words
- cosine distance between the vectors

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$$

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

Content-based systems: user profile

Average of the vector representation of rated/watched/… items

- Weighted average if we have ratings for each item

Estimating a user rating for unrated items

Having representations for the items and the users, for each user that we want to recommend new items

- search for similar users (get the top *n*) and then
- get the items that most of them liked (their average is the rating prediction for the initial user)

Better yet, we can use the user similarity (cosine similarity = 1 - cosine distance) to do a weighted average of their rating.

Note: Obviously, it goes without saying, that looping through the users is feasible in a short dataset, but in a real world scenario, you would want to index the users and perform the retrieval step there (looping through all the users has a *O(n)* complexity; search indexes have better complexity than that).

One such search engines is Facebook's Faiss: "Faiss is a library for efficient similarity search and clustering of dense vectors.", https://github.com/facebookresearch/faiss

Collaborative Filtering

CF projects users and items into a shared latent space, using a vector of latent features to represent a user or an item. The relationship between users and items is modelled as the inner product of their latent vectors.

Instead of using item features to determine similarity, it focus on user rating similarity:

- Each item is represented by its rating vector (ratings that users gave to the items), instead of a properties/features vector
- Each user is represented by its rating vector (ratings he gave to items), instead of averaging item feature vectors

Same similarity measures (Jaccard or cosine distance) apply.
Method previously described also applies.

Techniques

Locality-sensitive hashing & random-hyperplane:

- used to place item (profiles) in buckets

Hierarchical Clustering:

- Iteratively, cluster item profiles and user profiles to reduce matrix sparsity

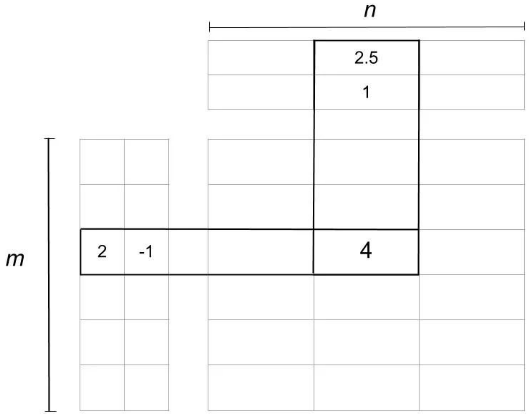
UV-Decomposition:

- estimate blank entries with the product of two matrices, U & V, computed minimizing RMSE: E = $\sum_{ij} (m_{ij} – \sum_k u_{ik} v_{kj})^2$
    - evaluate how well P = UV approximates M with RMSE (root-mean-square error).
    - Average $(m_{ij} – p_{ij})^2$ over all i and j and then take the square root.
- Similar concept to Singular value decomposition (SVD)

Dimensionality Reduction/Matrix Factorization

The core idea is to reduce/compress the huge but sparse utility/ratings matrix.

Reducing the matrix size leads to an improvement in terms of space and time.



When reducing the matrix, users and items get smaller representations (latent space).

The columns in the user matrix and the rows in the item matrix are called latent factors.

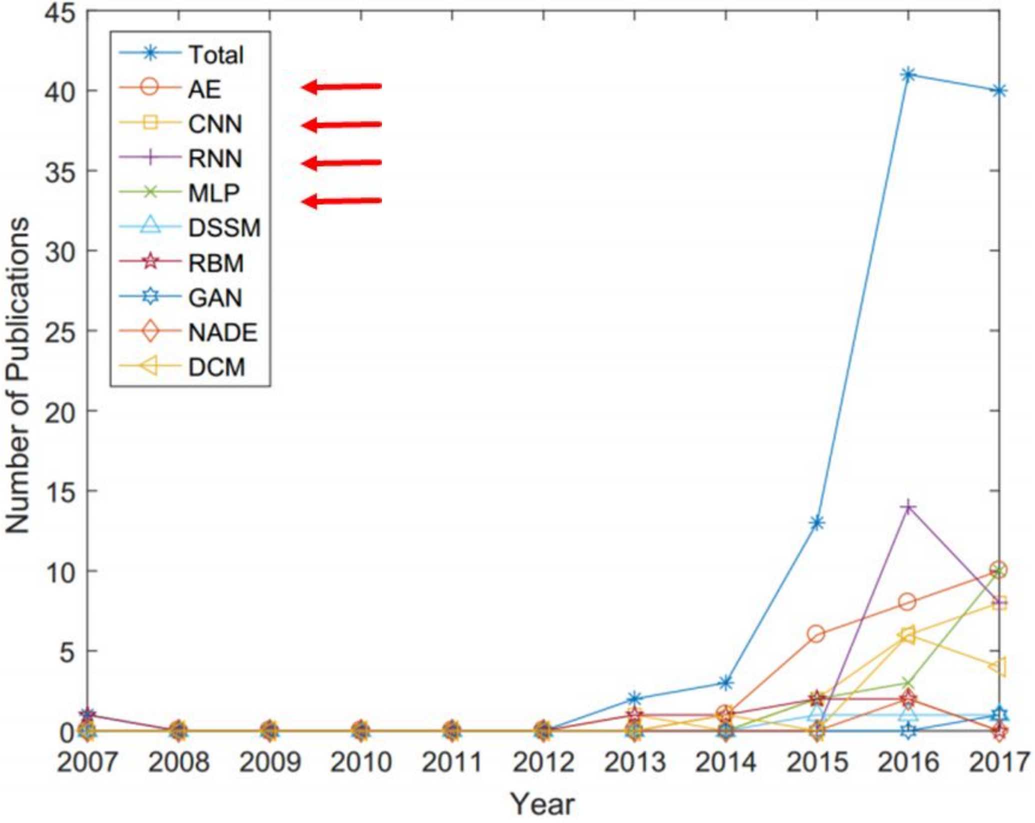
Coding break

https://github.com/davidalbertonogueira/NLP-tutorials/blob/master/code/Recommendation.ipynb

Deep learning techniques

- Multilayer Perceptron (MLP)
- Autoencoder (AE)
- Convolutional Neural Networks (CNN)
- Recurrent Neural Networks (RNN)
    - Long Short Term Memory (LSTM) and Gated Recurrent unit (GRU)
- Deep Semantic Similarity Model (DSSM)
- Restricted Boltzmann Machine (RBM)
- Generative Adversarial Networks (GAN)
- Composite models: combinations of these techniques

Number of publications



S. Zhang et al. (2017)

Multilayer Perceptron (MLP)

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua, Neural collaborative filtering, in Proceedings of the 26th International Conference on World Wide Web. International World Wide Web Conferences Steering Commitee, 2017. https://arxiv.org/pdf/1708.05031.pdf

They model the two-way interaction between users and items

- capture the non-linear relationship between users and items

$$\hat{r}_{ui} = f(U^T \cdot s_u^{user}, V^T \cdot s_i^{item} | U, V, \theta)$$

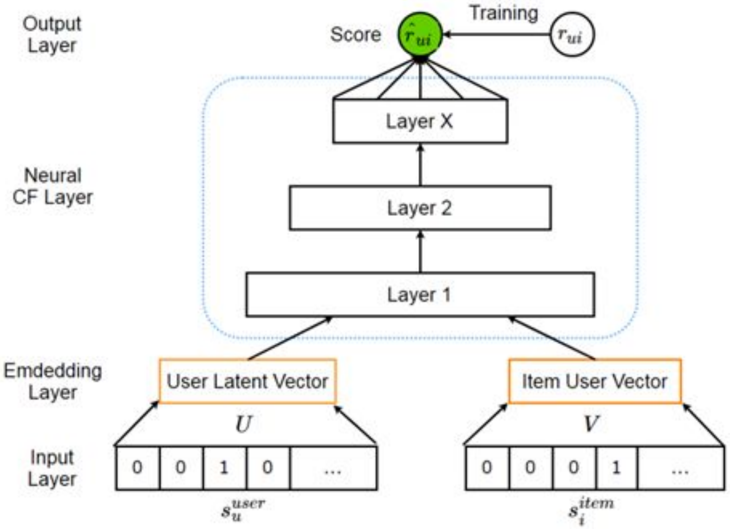
Su and Si denote side information (e.g. user and item profiles)

f (.) defines the multilayer perceptron, θ the parameters

Loss:

$$\mathcal{L} = \sum_{(u,i)\in \mathcal{O}\cup\mathcal{O}^-} w_{ui}(r_{ui} - \hat{r}_{ui})^2$$

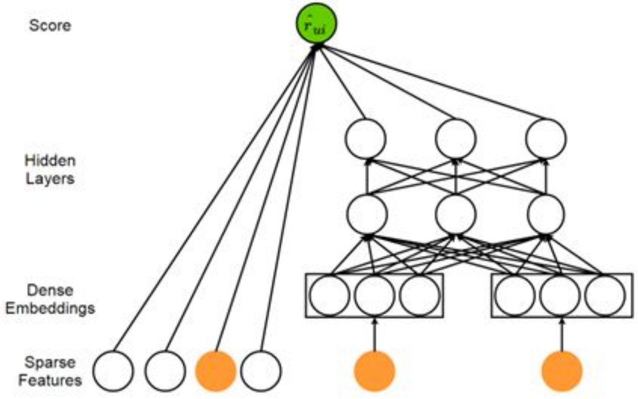
•Due to the large number of unobserved instances, NCF utilizes negative sampling to reduce the training data size



Neural Collaborative Filtering

Multilayer Perceptron (MLP)

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al, *Wide & deep learning for recommender systems*, in Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, ACM, 2016. https://arxiv.org/pdf/1606.07792.pdf



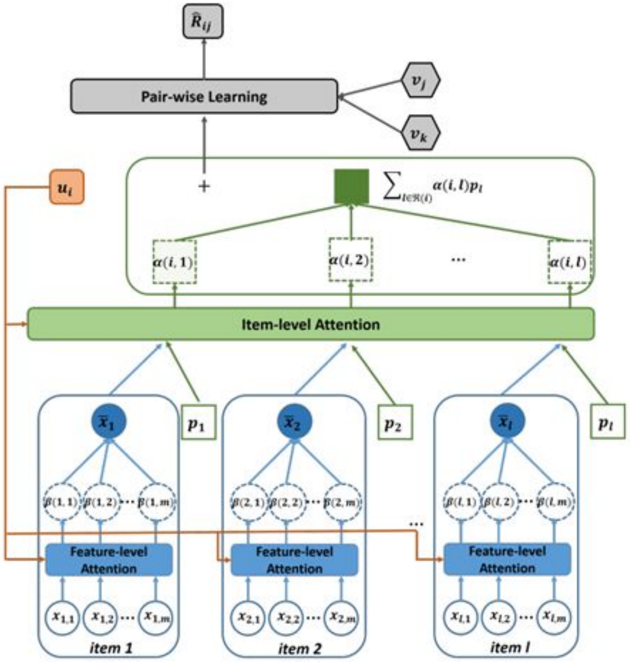
- This model can solve both regression and classification
- Initially introduced for App recommendation in Google play
- The wide learning component is a single layer perceptron which can also be viewed as a generalized linear model.
- The deep learning component is a multilayer perceptron.
- They claim it enables the recommender system to capture both memorization and generalization.
    - Memorization: wide learning component represents the capability of catching the direct features from historical data.
    - Generalization: deep learning component catches the generalization by producing more general and abstract representations.

$$P(\hat{r}_{ui} = 1|x) = \sigma(W_{wide}^{T}\{x, \phi(x)\} + W_{deep}^{T} a^{(l_f)} + bias)$$

Multilayer Perceptron (MLP) + Attention

Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua, Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention, in Proceedings of the
40th International ACM SIGIR conference on Research and Development in Information Retrieval, 2017.
https://ai.tencent.com/ailab/media/publications/Wei_Liu-Attentive_Collaborative_Filtering_Multimedia_Recommendation-SIGIR

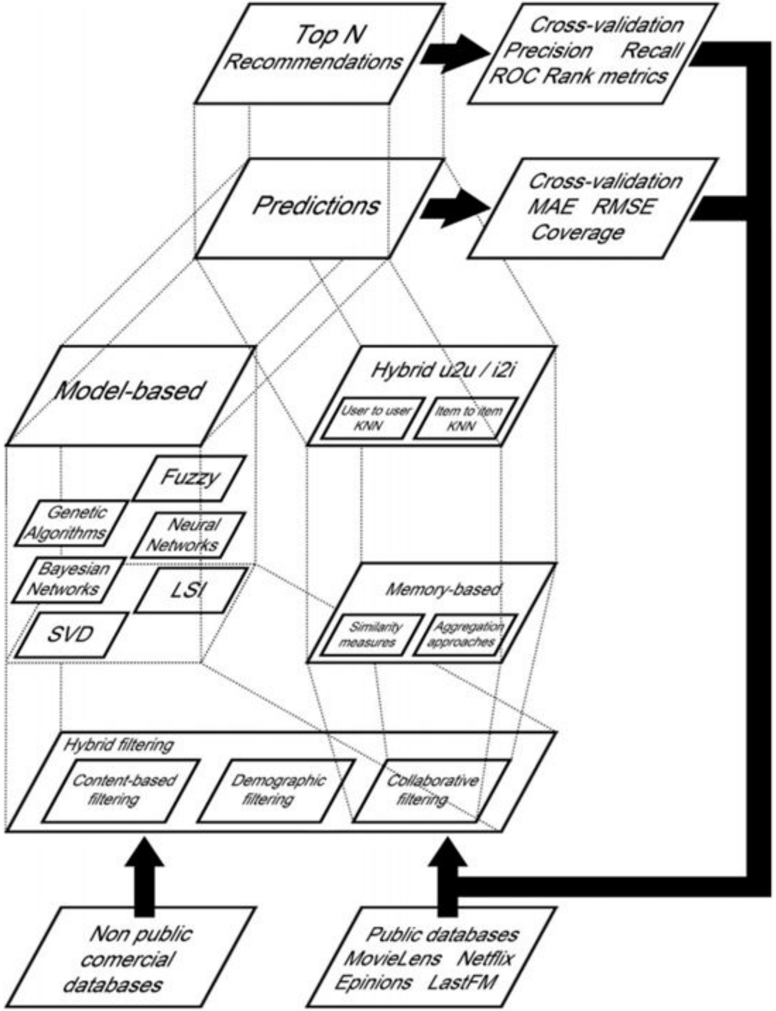
- They argue that, in multimedia recommendation, there exists item- and component-level implicitness which blurs the underlying users' preferences.
- A "like" of a photo or a "view" of a video does not provide any specific information about which parts of the photo/video the user is interested in (i.e., component-level).


$\hat{R}_{ij}$
Pair-wise Learning
v_j
v_k
u_i
+
$\sum_{l \in \mathcal{R}(i)} \alpha(i,l)p_l$
$\alpha(i,1)$
$\alpha(i,2)$
...
$\alpha(i,l)$
Item-level Attention
$\bar{x}_1$
p_1
$\bar{x}_2$
p_2
$\bar{x}_l$
p_l
$\beta(1,1)$ $\beta(1,2)$ ... $\beta(1,m)$
$\beta(2,1)$ $\beta(2,2)$ ... $\beta(2,m)$
$\beta(l,1)$ $\beta(l,2)$ ... $\beta(l,m)$
Feature-level Attention
$x_{1,1}$ $x_{1,2}$ ... $x_{1,m}$
$x_{2,1}$ $x_{2,2}$ ... $x_{2,m}$
$x_{l,1}$ $x_{l,2}$ ... $x_{l,m}$
item 1
item 2
item l

The item-level attention is used to select the most representative items to characterize users.

The component-level attention aims to capture the most informative features from multimedia auxiliary information for each user.

ACF

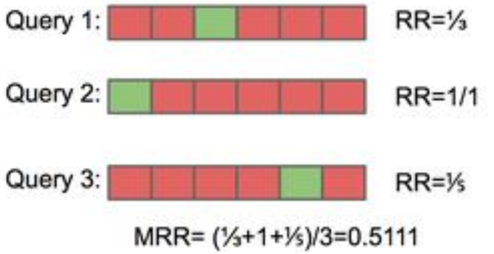

Top N Recommendations
Cross-validation Precision Recall ROC Rank metrics
Predictions
Cross-validation MAE RMSE Coverage
Model-based
Hybrid u2u / i2i
User to user KNN
Item to item KNN
Fuzzy
Genetic Algorithms
Neural Networks
Bayesian Networks
LSI
SVD
Memory-based
Similarity measures
Aggregation approaches
Hybrid filtering
Content-based filtering
Demographic filtering
Collaborative filtering
Non public comercial databases
Public databases MovieLens Netflix Epinions LastFM

Mean Reciprocal Rank

Statistic measure to evaluate the quality of a ranked list of possible responses to a query (ordered by probability of correctness).

The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer: 1 for first place, 1⁄2 for second place, 1⁄3 for third place, etc.. The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of queries Q:

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}.$$



MRR= (⅓+1+⅕)/3=0.5111

break

Named Entity Recognition & Linking

Named-entity recognition (NER) (also known as entity identification, entity chunking and entity extraction) is a subtask of information extraction that seeks to locate and classify named entity mentions in unstructured text into pre-defined categories such as the person names, organizations, places (locations, countries, cities, etc).

The task of entity linking, also called named entity linking (NEL) and named entity disambiguation (NED), aims at connecting detected mention occurrences in a document or multiple documents with the known entities they refer to, which are stored in a knowledge base (typically, the Wikipedia).

Named Entity Recognition

Bi-directional LSTM, Ma & Hovy

End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF, Ma & Hovy, 2016

CMU-NER (NER System Developed at CMU)

- Python

Transition-based NER/Stack-LSTM NER

Neural Architectures for Named Entity Recognition, Lample & Ballesteros, 2016

- C++ / Dynet
- Python / Pytorch

NER w/ End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF (Ma & Hovi, 2016)

Is considered a standard approach for training labeling/tagging systems

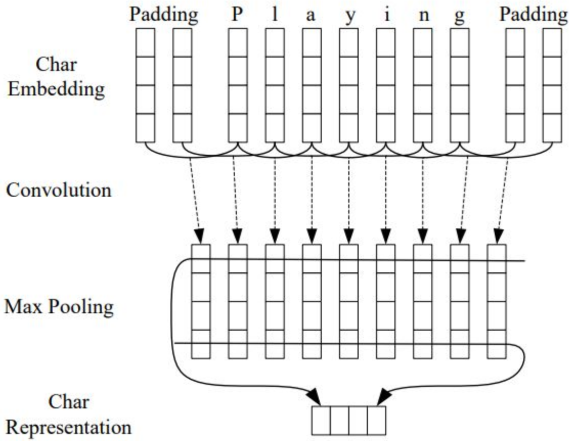


Figure 1: The convolution neural network for extracting character-level representations of words. Dashed arrows indicate a dropout layer applied before character embeddings are input to CNN.

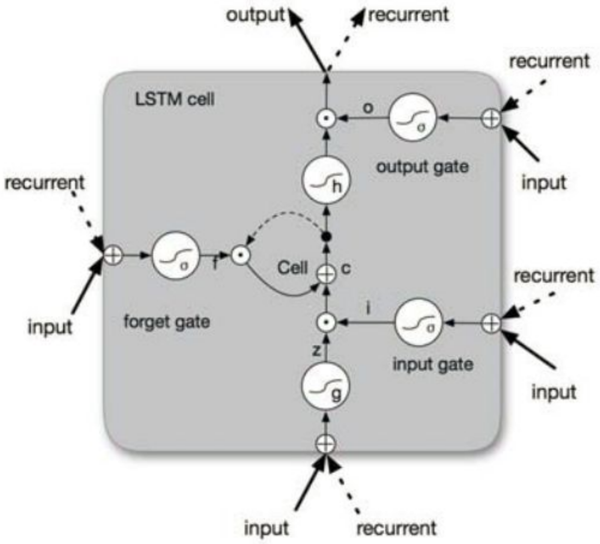


Figure 2: Schematic of LSTM unit.

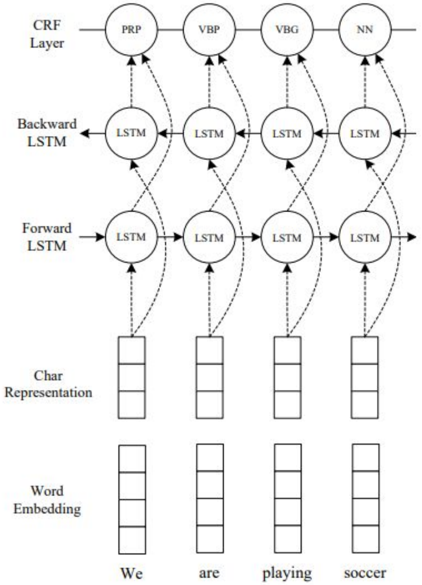


Figure 3: The main architecture of our neural network. The character representation for each word is computed by the CNN in Figure 1. Then the character representation vector is concatenated with the word embedding before feeding into the BLSTM network. Dashed arrows indicate dropout layers applied on both the input and output vectors of BLSTM.

Entity Linking

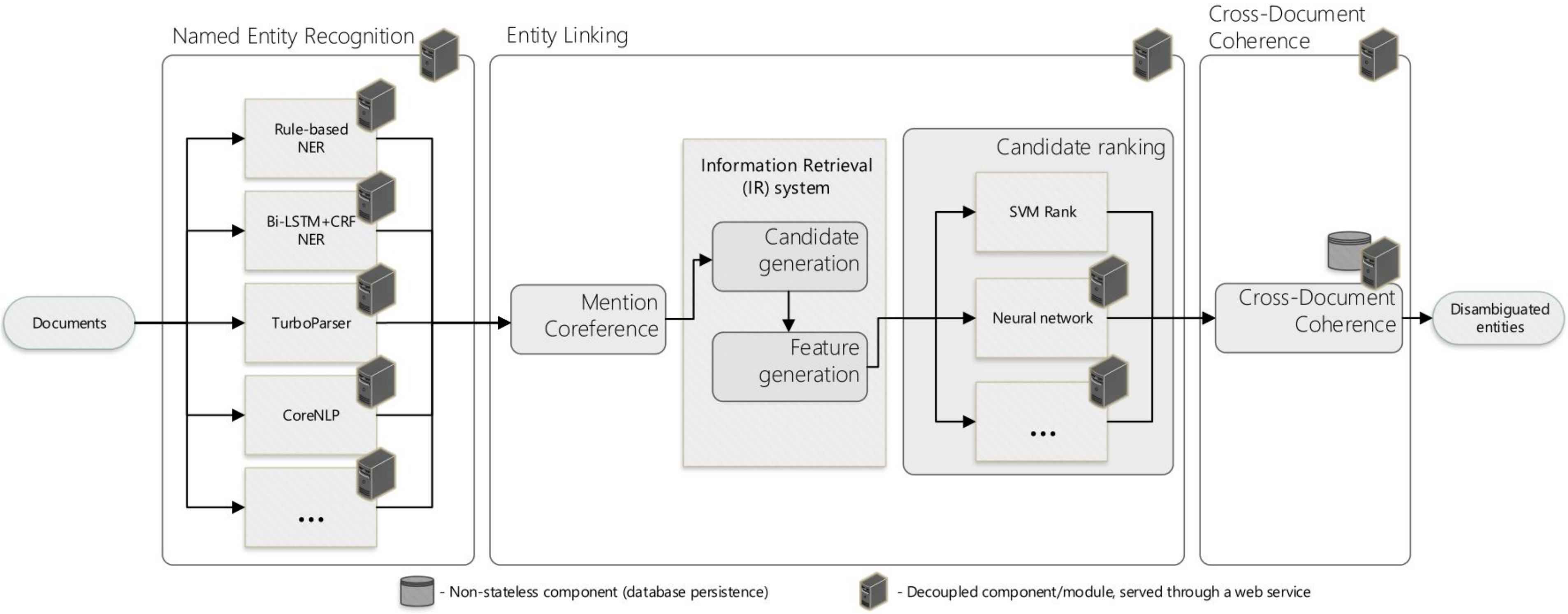
SUMMA at TAC Knowledge Base Population Task 2017

https://pdfs.semanticscholar.org/0ce1/d6a8e33b488d399bde1a1e20cef3b59f5f2d.pdf

SUMMA at TAC Knowledge Base Population Task 2016

https://tac.nist.gov/publications/2016/participant.papers/TAC2016.summa.proceedings.pdf

Entity Discovery and Linking system architecture



Entity Linking core

The entity linking core, is responsible for the entity disambiguation:

1) mention coreference and clustering,

2) candidate generation,

3) feature generation and

4) candidate ranking.

Entity Linking core

- mention coreference and clustering
  - cluster together mentions pertaining to the same entity using a high-precision substring matching approach
- candidate generation
  - using an information retrieval system, generate the set of all possible candidates for all mentions in the document, by performing search operations in its indexes, based on the anchors statistics in Wikipedia
- feature generation
  - nearest-neighbors features (obtained from the similarity between the document and the candidate description in the KB)
  - prior features
  - co-occurrence features (candidates get a co-occurrence score proportional to the number of
  - mentions they co-occur with in the KB)
  - coherence features (candidates get a score that weights the existence of links in the KB between them and the candidates of the other mentions)
  - words and character-level embeddings, for both the mention and the context

Pro-tips & lessons learned: Regarding versioning/git

- Use and abuse of branches. Every new test / feature, create a branch. Quite often, you'll need to update something in production and you need to get back to the master branch, add the fix, and then return to your development.
- Try to get used to using git add <filename> rather than git add -a. Check git status and git diff <remote-branch> git diff <filename> every time before adding and committing changes. You don't want to accidentally left in some changes you didn't mean to push.
- git tag your commits whenever you create a new model for production. You want to be able to trace back a model in production to the model code used to train it, as well as possible configurations. You should version your code, configurations files and models.

Pro-tips & lessons learned: python/pip

- Don’t to pip freeze > requirements.txt. It brings to your requirements dependencies that you do not need, but that were required by some of those libraries. If you want to update some of them in the future, they may change the versions of their sub-dependencies or stop requiring them altogether and you get stuck with them with no reason. It’s very common to have packages conflicts between different libraries having different version requirements. Create the least possible conflict, by instantiating the least number of them.
- Start using typing in python in the function arguments. It helps a lot detecting errors in which you are passing the wrong type of arguments to the functions, way before they may lead to a crash.

Pro-tips & lessons learned: Tackling a new task/problem

- During code development, use tqdm or some sort of print every 100 or 1000 iterations (i%1000) to keep track of your program speed. You don’t want to start a loop with a huge amount of data, only to find out after it finishes that the loop would take half an hour, and that you had a bug in the last iteration and all goes to waste (quite common in non-compiled languages like python).
- Therefore, always have a reduced dataset or input file. It will make your life easier to test the flow of the code and help in the early development of the project/task pipeline.
- If you have pre-processing to make (and it takes a considerable amount of time), store it in a file (either in a pickle - binary format), or in a human readable format (like csv or json).

Pro-tips & lessons learned: Tackling a new task/problem

- Always start with the most simple and straightforward approach. It may be simple, but it will give you a baseline score (the minimum reasonable value for your chosen metric). If you’re developing your algorithm, new neural network or similar, and your score is lower, it gives you a direct information that you have a problem in the pipeline (either a bug, an incorrect sign in the loss, incorrect input loading, incorrect data processing, etc.)
- Look for shared tasks and papers related to the problem/task you’re trying to solve. In some cases, you may have a direct match. It will give you a direction in terms of the challenge of the task, which algorithms are normally used (and which ones are currently giving the best results), the best pre processing approaches and what scores are reasonable to expect.

Lessons learned: from model training to production

- As soon as a trained model is deployed, in a real world, “live” scenario, it’s accuracy will start to go down. Best example is Entity Linking. The pipeline requires the indexation of a wikipedia dump, which means that new entities are not incorporated. As time goes by, the number of missing entities becomes higher and more noticeable.
  It’s important to think from the ground up in the full training/testing/deployment cycle. No step is done just “once” (even though you think so, it wouldn’t be that way in the future).

Lessons learned: from model training to production

- In many cases, the data you extract from the model in production is of greater value than the data used to train it. The way the users interact with your product (agreeing or rejecting your model's prediction; selecting not the first, but the third suggested article/product, etc…).
  If you're able to incorporate back that data in your model training, your model will become much better adapted to your customers' needs that if you just use your initial data.

Incorporating user feedback - example

- Imagine a product that feeds you articles according to your interests. The algorithm bootstrap may be just with document retrieval based on your tags (i.e., interests' keywords). As users receives recommended articles, they may accept them or mark them as irrelevant.
- An SVM model may be trained (per user and per topic) to predict if the user likes/reads the article, or if he will reject it. Those same SVM weights can then be propagated to the information retrieval algorithm to better rank possible articles, that will be feed to the SVM filter.
- As time goes by, the quality of the recommended articles will improve as the user interacts with the tool.

Further reading

- https://github.com/sebastianruder/NLP-progress (repo with current progress in each NLP task)
- https://ruder.io/ (blog from the same author with interesting articles about NLP)
- http://summa-project.eu/deliverables/ , http://summa-project.eu/publications-2/ (SUMMA output)
- Search for TAC KBP, NAACL, AAAI, EMNLP, ACL, NIPS, SemEval and read the top papers.

Bônus:
https://www.youtube.com/watch?v=wjqaz6m42wU (NIPS 2016 tutorial: "Nuts and bolts of building AI applications using Deep Learning" by Andrew Ng)

Discussion

Feel free to connect:

https://www.linkedin.com/in/davidjacomenogueira/

david_alberto_nogueira@hotmail.com