Text Mining

Jose Martinez Heras

26/04/2018

Resources

Watch the video of this lecture

https://dlmultimedia.esa.int/download/public/videos/2048/04/013/4804_013_AR_EN.mp4

Watch the practical exercise video

https://dlmultimedia.esa.int/download/public/videos/2048/04/012/4804_012_AR_EN.mp4

Get presentation and additional resources on

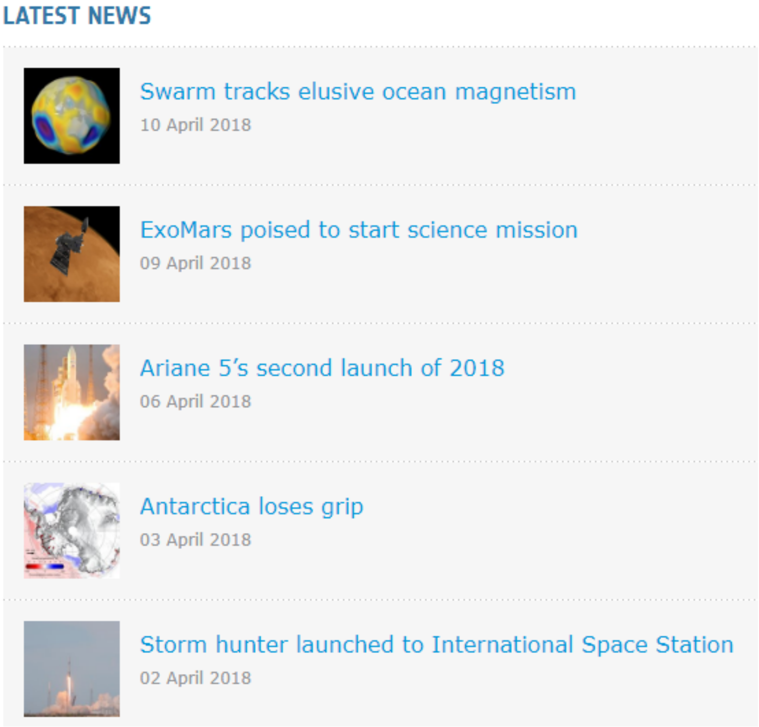
https://github.com/jmartinezheras/2018-MachineLearning-Lectures-ESA

Outline for Text Mining

Session 6: Text Mining

- Some Text Mining applications
- Text Representation
- Document Search
- Topic Extraction
- Machine Learning with Text: Text Mining
- Word Embeddings

- Hands – on: predict the number of views on ESA News articles

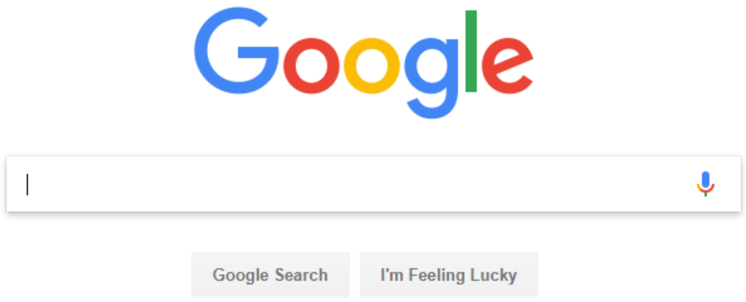


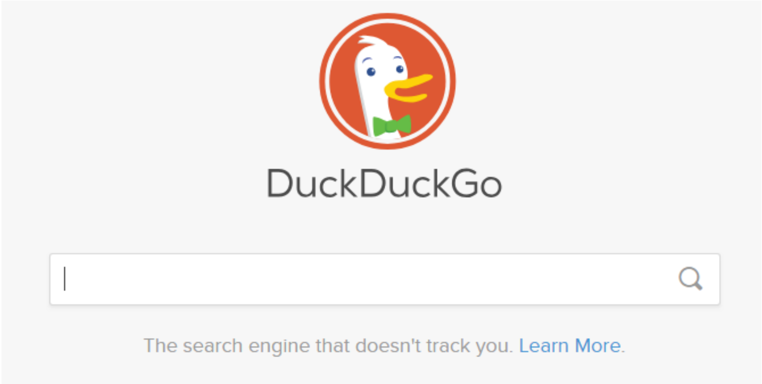
http://www.esa.int/Our_Activities/Space_News

Applications – Spam Filtering

Spam Filter

Applications - Search





search.esa.int

Applications – Sentiment Analysis

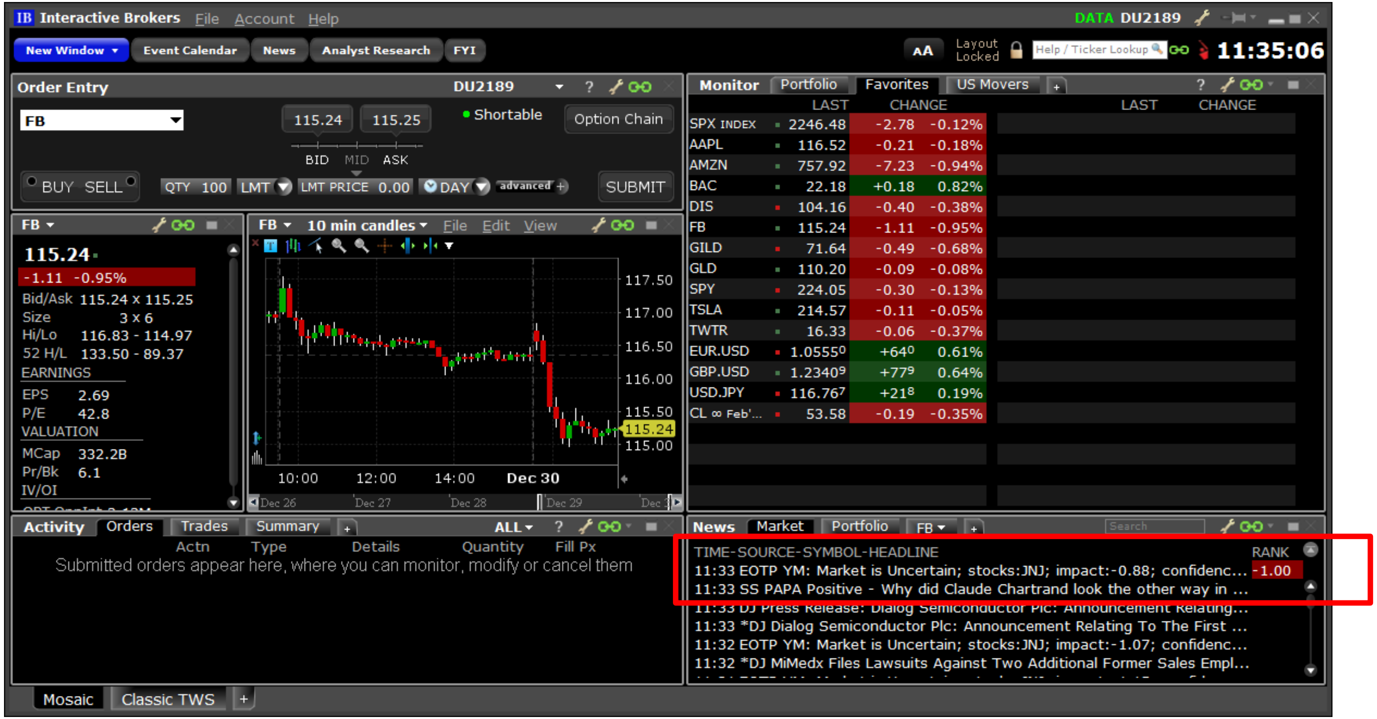


Image Credit: https://www.interactivebrokers.com/en/index.php?f=1235

Applications – Image Captioning

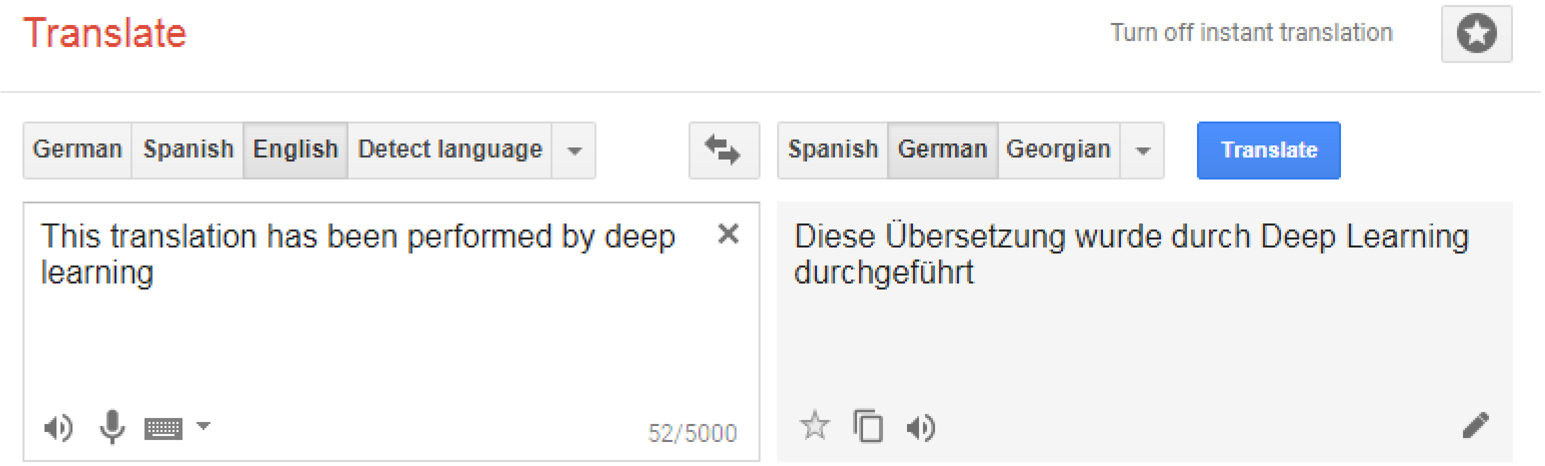
"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

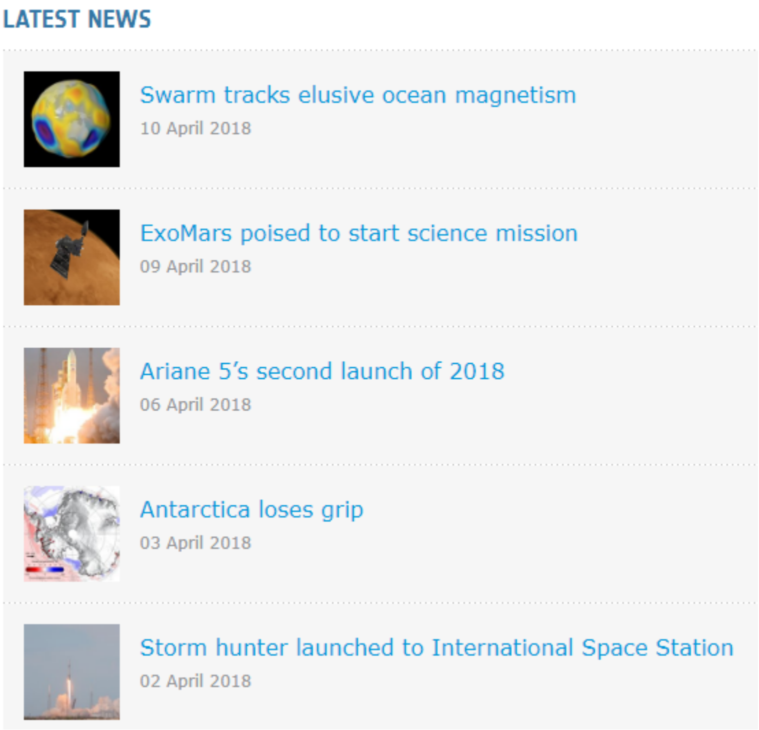
Image credit: https://cs.stanford.edu/people/karpathy/deepimagesent/

Applications – Language Translation



Applications - Prediction

Predict if an article will receive a high number of views



http://www.esa.int/Our_Activities/Space_News

Text Representation – Bag of Words

Let’s use 2 documents for a running example:

(1) John likes to watch movies. Mary likes movies too.

(2) John also likes to watch football games.

Extract words, remove punctuation

(1) John, likes, to, watch, movies, Mary, likes, movies, too

(2) John, also, likes, to, watch, football, games

Text Representation – Bag of Words

(1) John, likes, to, watch, movies, Mary, likes, movies, too

(2) John, also, likes, to, watch, football, games

List all the words in an arbitrary order (without repetition)

John, likes, to, watch, movies, Mary, too, also, football, games

Count how many times each word appear on each document

(1) [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]

(2) [1, 1, 1, 1, 0, 0, 0, 1, 1, 1]

Text Representation – Bag of Words

	John	likes	to	watch	movies	Mary	too	also	football	games
(1)	1	2	1	1	2	1	1	0	0	0
(2)	1	1	1	1	0	0	0	1	1	1

Each document is transformed in a vector of n-dimensions

n is the number of different words considered

The word order is not considered

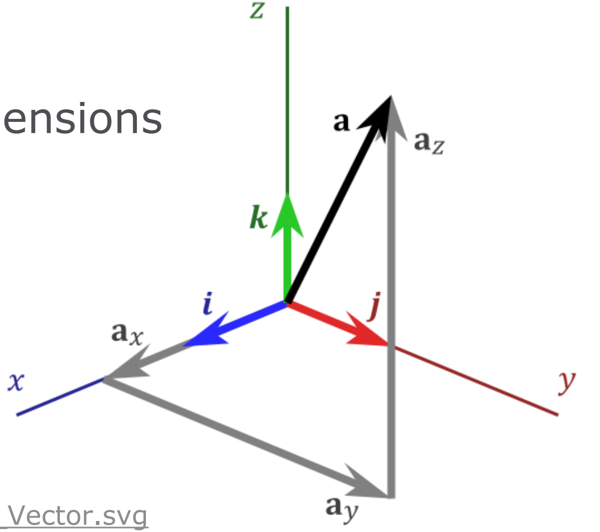
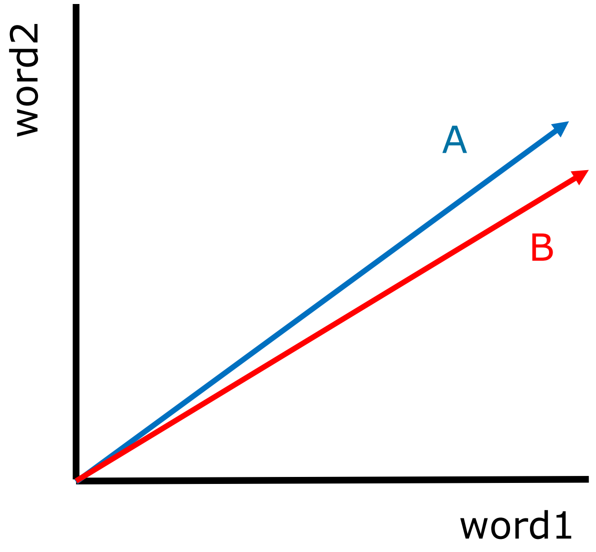
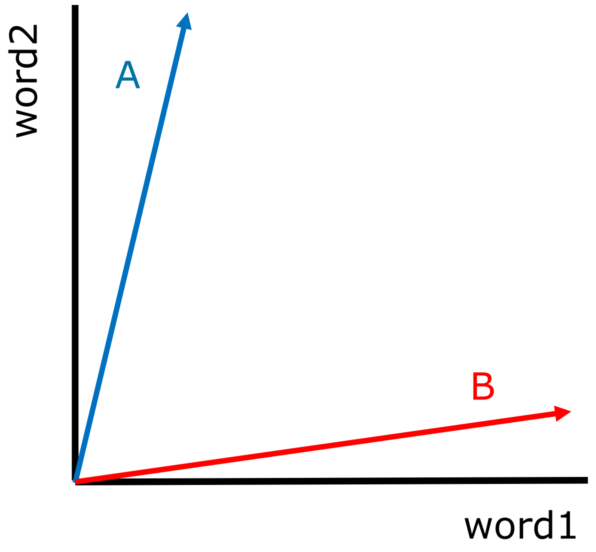


Image credit: https://commons.wikimedia.org/wiki/File:3D_Vector.svg

Document Similarity – 2D (2 words) intuition

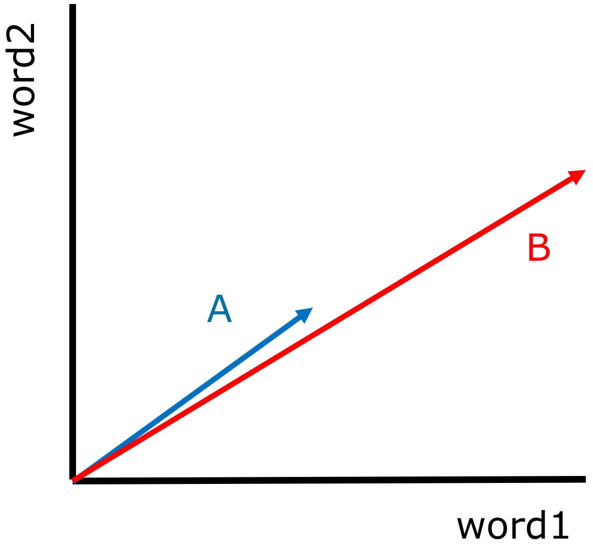


Similar documents



Different documents

Document Similarity – 2D (2 words) intuition

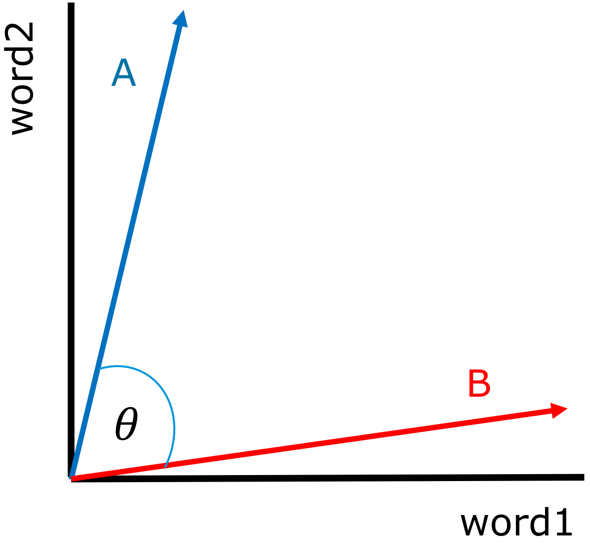


Similar or different documents?

Similar but different length

Document Similarity

Let's quantify similarity



$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

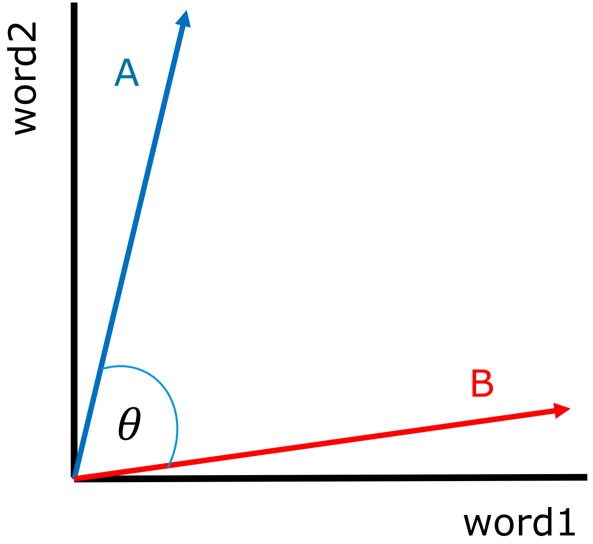
$$similarity = [0, 1] = 1\ most\ similar$$

$\theta = 70°$ $\cos(70°) = 0.34$

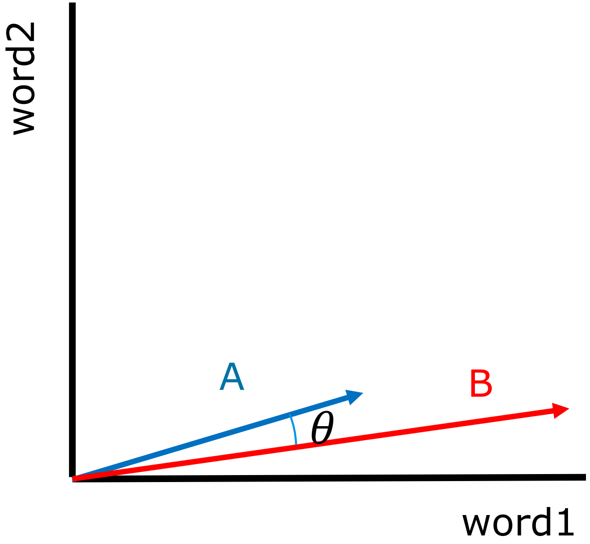
Document Similarity

Let’s quantify similarity

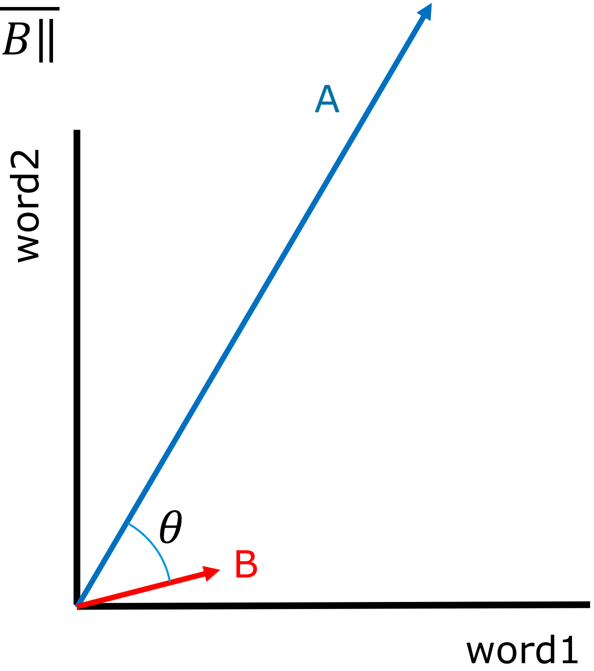
$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



$$\cos(70°) = 0.34$$



$$\cos(10°) = 0.98$$



$$\cos(45°) = 0.71$$

Document Search

Goal: find the documents which are most similar to your query

- Compute the pairwise cosine similarity between the query and all documents
- Return the top-10 documents that rank higher

It still needs some tweaks to get relevant matches – let’s discuss them

Getting more relevant matches

Reduce the number of irrelevant dimensions

- Remove punctuation, lowercase
- Stop-words
  - me, my, myself, we, our, … with, about, when, where, might, could …
- Stemming / Lemmatization
  - child → child
  - children → child

Getting more relevant matches

(1) John likes to watch movies. Mary likes movies too.

(2) John also likes to watch football games.

tf	John	likes	to	watch	movies	Mary	too	also	football	games
(1)	1	2	1	1	2	1	1	0	0	0
(2)	1	1	1	1	0	0	0	1	1	1

tf	john	like	watch	movie	mary	football	game
(1)	1	2	1	2	1	0	0
(2)	1	1	1	0	0	1	1

Getting more relevant matches

Highlight important words within our document set

(1) John likes to watch movies. Mary likes movies too.

(2) John also likes to watch football games.

tf	john	like	watch	movie	mary	football	game
(1)	1	2	1	2	1	0	0
(2)	1	1	1	0	0	1	1

Term Frequency

Getting more relevant matches

Highlight important words within our document set

(1) John likes to watch movies. Mary likes movies too.

(2) John also likes to watch football games.

tf	john	like	watch	movie	mary	football	game
(1)	1	2	1	2	1	0	0
(2)	1	1	1	0	0	1	1

Term Frequency

df	john	like	watch	movie	mary	football	game
DF	2	2	2	1	1	1	1

Document Frequency

Getting more relevant matches - tfidf

Highlight important words within our document set with *tfidf*

$$\frac{term\ frequency}{document\ frequency} = \frac{tf}{df} = tf \cdot idf = tfidf$$

tf	john	like	watch	movie	mary	football	game
(1)	1	2	1	2	1	0	0
(2)	1	1	1	0	0	1	1

Term Frequency

df	john	like	watch	movie	mary	football	game
DF	2	2	2	1	1	1	1

Document Frequency

Getting more relevant matches - tfidf

Highlight important words within our document set with *tfidf*

tfidf	john	like	watch	movie	mary	football	game
(1)	0.5	1	0.5	2	1	0	0
(2)	0.5	0.5	0.5	0	0	1	1

tfidf

tf	john	like	watch	movie	mary	football	game
(1)	1	2	1	2	1	0	0
(2)	1	1	1	0	0	1	1

Term Frequency

df	john	like	watch	movie	mary	football	game
DF	2	2	2	1	1	1	1

Document Frequency

Getting more relevant matches - tfidf

Highlight important words within our document set with *tfidf*

$$\frac{term\ frequency}{document\ frequency} = \frac{tf}{df} = tf \cdot idf = tfidf \qquad tfidf = tf \cdot \left(1 + \log\left(\frac{1+n_d}{1+df}\right)\right)$$

tf	john	like	watch	movie	mary	football	game
(1)	1	2	1	2	1	0	0
(2)	1	1	1	0	0	1	1

Term Frequency

df	john	like	watch	movie	mary	football	game
DF	2	2	2	1	1	1	1

Document Frequency

Document Search

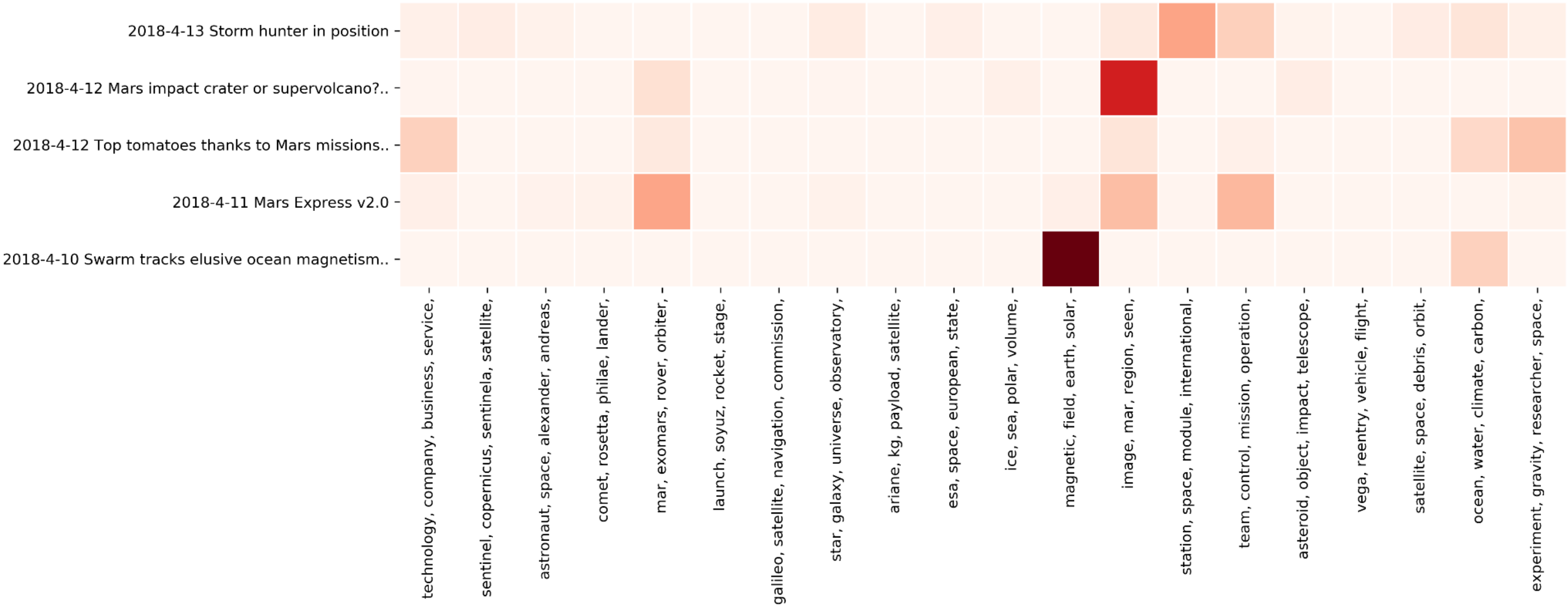
Goal: find the documents which are most similar to your query

- Remove punctuation, lowercase, stop-words, stemming of your documents
- *tfidf* your documents

- Remove punctuation, lowercase, stop-words, stemming of the query
- *tfidf* the query

- Compute the pairwise cosine similarity between the query and all documents
- Return the top-10 documents that rank higher

Topic Extraction

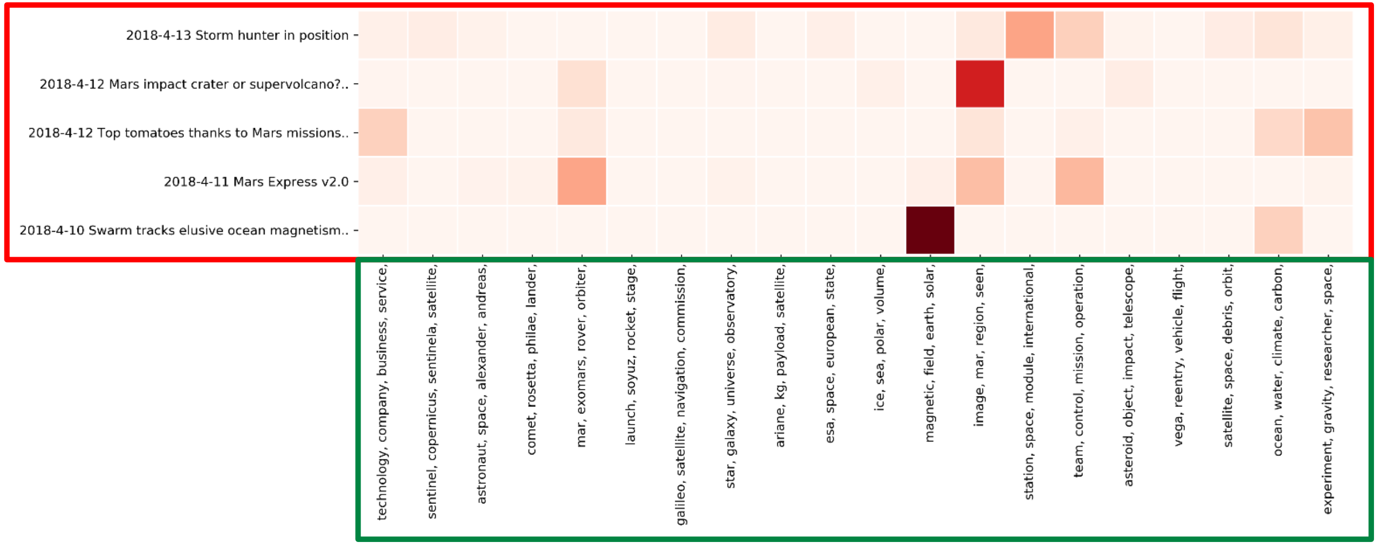


Topic Extraction

Matrix Factorization

$$Tfidf \approx Coefficients \times Features$$

$$Tfidf_{n_{docs} \times n_{terms}} \approx \boxed{Coefficients_{n_{docs} \times n_{topics}}} \times \boxed{Features_{n_{topics} \times n_{terms}}}$$



Machine Learning with Text

- In previous lectures we have discussed about:
  - Regression
  - Support Vector Machines
  - Decision Trees / Random Forests
  - Neural Networks / Deep Learning
  - Anomaly Detection

- To use Machine Learning with Text data …
  - Transform text to numeric (e.g. tfidf, topics, embeddings)
  - Do Machine Learning as you already know
    - e.g. predict the ESA News article popularity

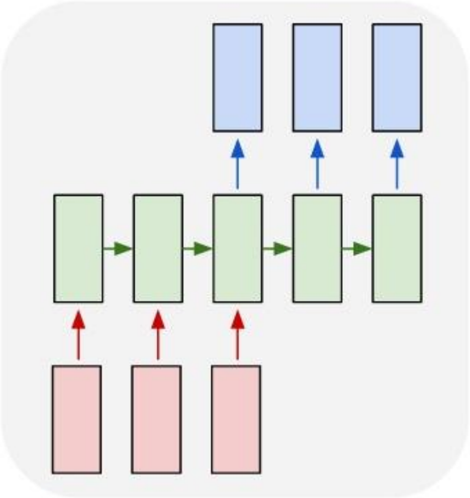
Another convention to encode words

One-hot-encoding

$$john = \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\end{bmatrix} \qquad like = \begin{bmatrix}0\\1\\0\\0\\0\\0\\0\end{bmatrix} \qquad watch = \begin{bmatrix}0\\0\\1\\0\\0\\0\\0\end{bmatrix} \qquad movie = \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix}$$

Input to Recurrent Neural Networks

many to many



Example: language translation

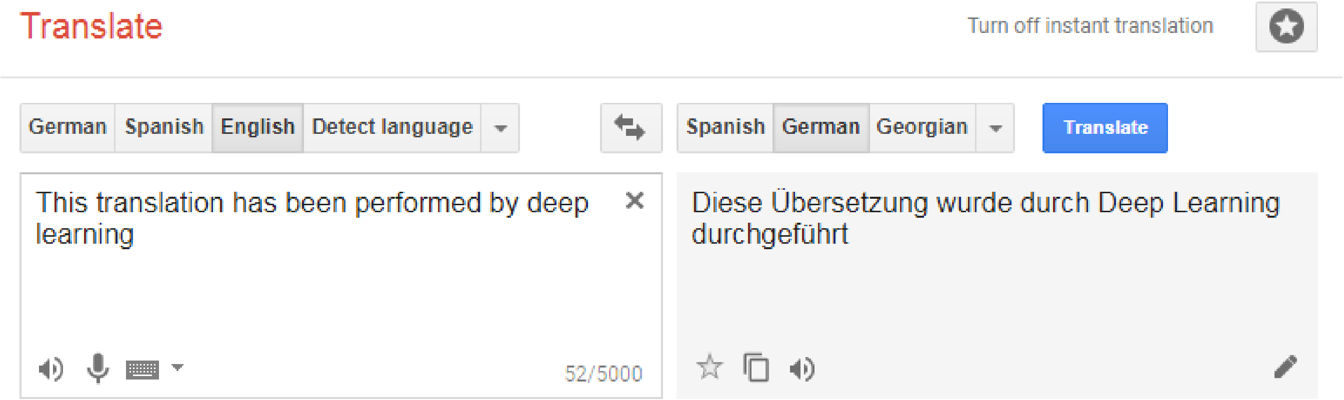


Image credit: https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Word Embedding

Issue: usually there are many terms (e.g. 1,000,000)

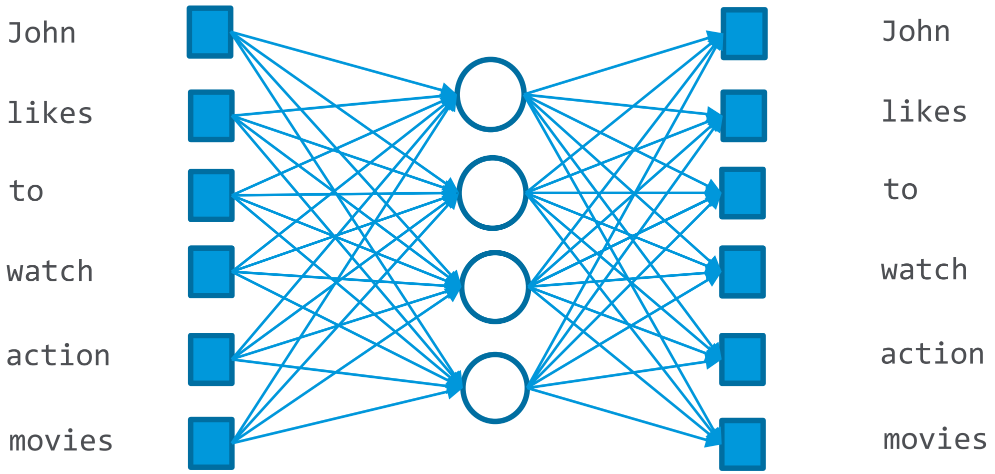
- Causes Machine Learning models to be complex, hard to train

$$watch = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad watch = \begin{bmatrix} 0.23 \\ -0.71 \\ 0.56 \\ 0.87 \\ -0.19 \end{bmatrix}$$

Smaller number of dimensions (e.g. 300)

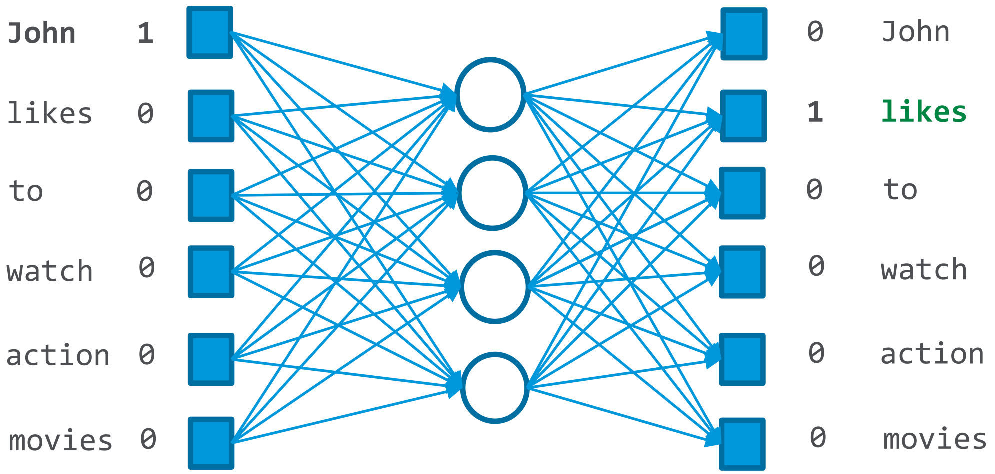
Word2vec

John likes to watch action movies



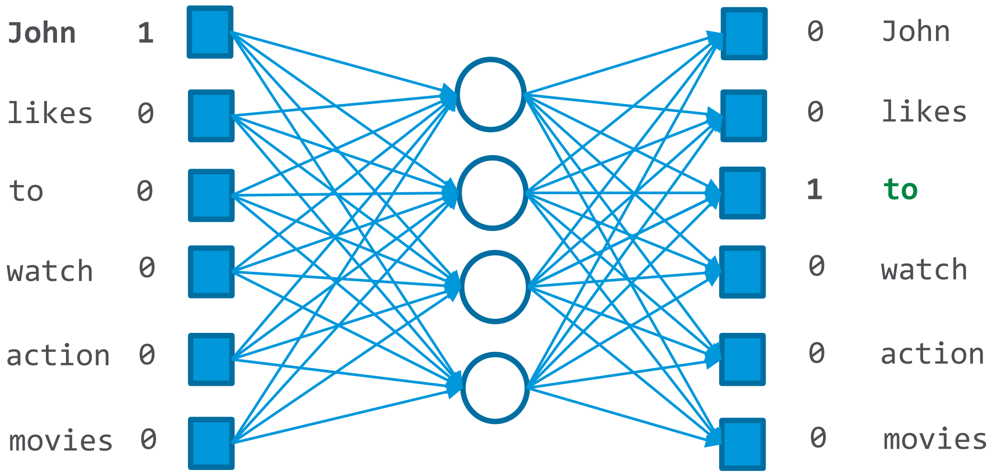
Word2vec

John **likes** to watch action movies



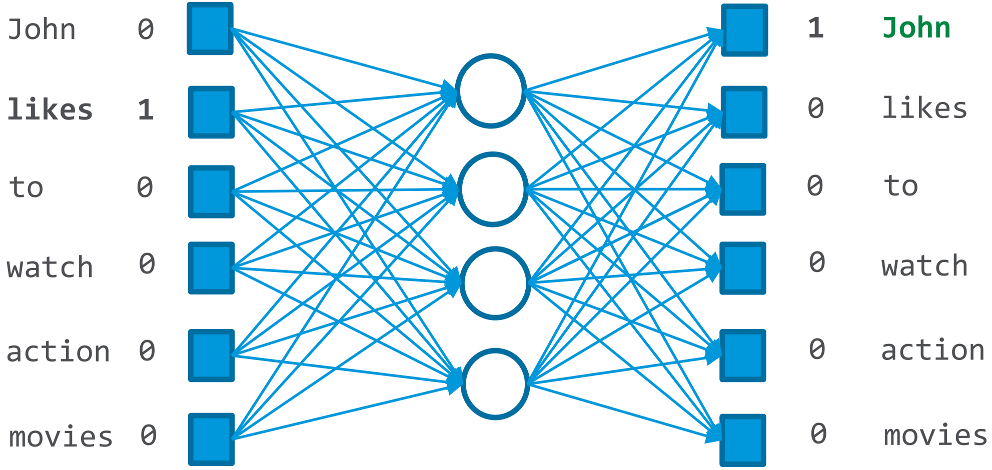
Word2vec

John likes **to** watch action movies



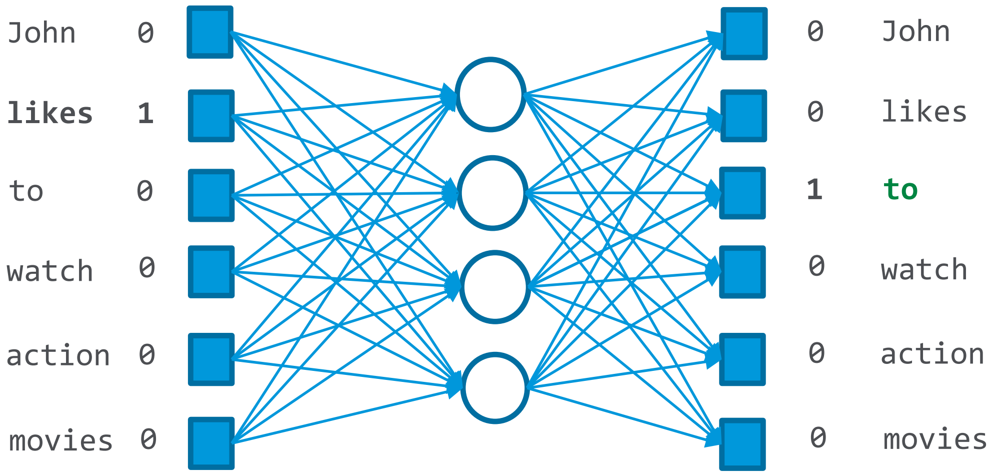
Word2vec

John likes to watch action movies



Word2vec

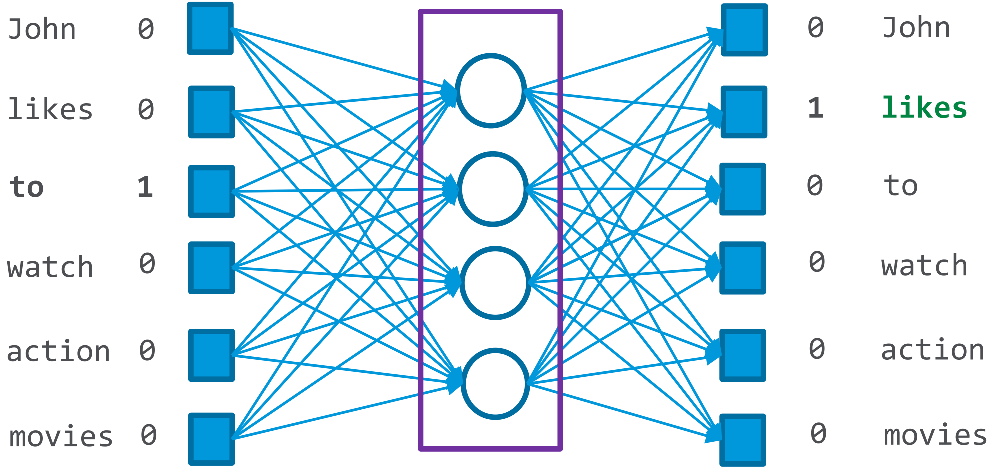
John **likes** to watch action movies



Word2vec

John **likes to** watch action movies

Etc. etc. etc.



300-dimension embedding

Embedding properties

- Words that are close in the embedding space, are similar

```
w2v.most_similar('germany',)[:5]
```

```
[('german', 0.6809574365615845),
 ('europe', 0.6781216859817505),
 ('european', 0.650211036205291 7),
 ('sweden', 0.6384239196777344),
 ('switzerland', 0.6362128853797913)]
```

Embedding properties

- Vector Algebra seems to work:

  king – man + woman = queen (man is to king as woman is to … queen)

```
w2v.most_similar(positive=['king', 'woman'], negative=['man'])[:5]
```

```
[('queen', 0.7118192911148071),
 ('monarch', 0.6189674139022827),
 ('princess', 0.590243160724639 9),
 ('crown_prince', 0.5499460697174072),
 ('prince', 0.5377321243286133)]
```

Embedding properties

- Vector Algebra seems to work:

  smaller – small + big = bigger (small is to smaller as big is to … bigger)

```
w2v.most_similar(positive=['smaller', 'big'], negative=['small'])[:5]
```

```
[('bigger', 0.7836999893188477),
 ('larger', 0.5866796970367432),
 ('Bigger', 0.5707237720489502),
 ('biggest', 0.5240510702133179),
 ('splashier', 0.5107756853103638)]
```

Advantages of Word Embeddings

There are already pre-trained embeddings

word2vec

Tool for computing continuous distributed representations of words.

Introduction

This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research.

Pre-trained word and phrase vectors

We are publishing pre-trained vectors trained on part of Google News dataset (about 100 billion words). The model contains 300-dimensional vectors for 3 million words and phrases. The phrases were obtained using a simple data-driven approach described in [2]. The archive is available here: GoogleNews-vectors-negative300.bin.gz.

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning

Introduction

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Getting started (Code download)

- Download the code (licensed under the Apache License, Version 2.0)
- Unpack the files: unzip GloVe-1.2.zip
- Compile the source: cd GloVe-1.2 && make
- Run the demo script: ./demo.sh
- Consult the included README for further usage details, or ask a question
- The code is also available on GitHub

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

word2vec: https://code.google.com/archive/p/word2vec/

GloVe: https://nlp.stanford.edu/projects/glove/

Advantages of Word Embeddings

- use simpler models in Deep Learning (fewer inputs: 300 instead of 1,000,000)

- allows Machine Learning to recognize similar words
  - `river, water: 0.577`
  - `river, desert: 0.21`

- transfer learning
  - Queries with words that are not in your documents are now possible

Embedding humour

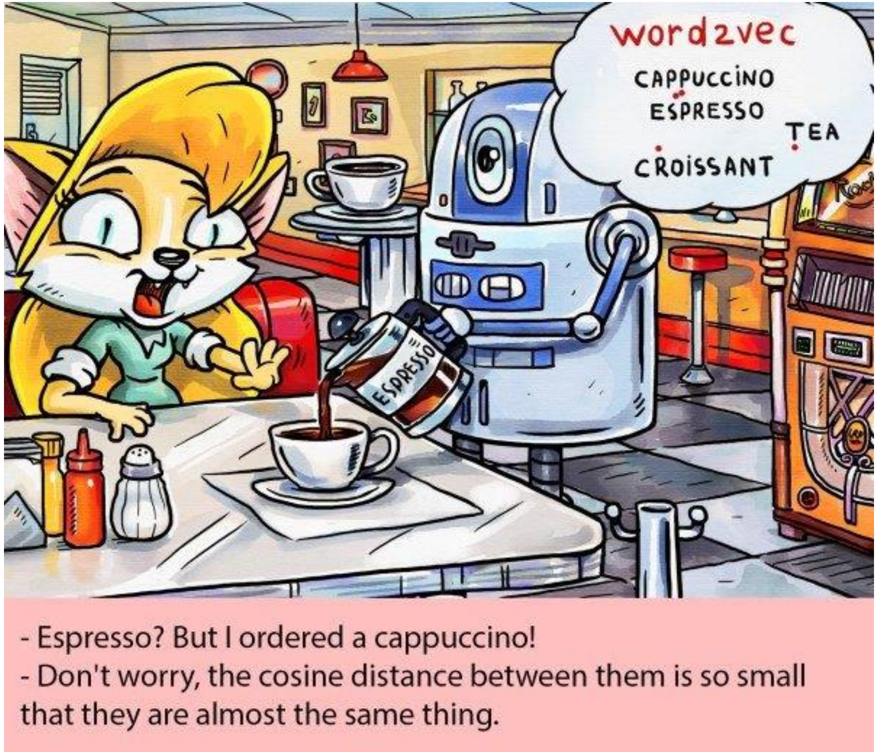


Image credit: https://twitter.com/MikeTamir/status/906357502899638272

Summary

- Some Text Mining applications
- Text Representation
- Document Search
- Topic Extraction
- Machine Learning with Text
- Word Embeddings

ARTS Text Mining – project pitch

ARTS = Anomaly Report Tracking System

- Mission "A" may have reported an anomaly (e.g. in the ground segment)
- Mission "B" may have faced (and solved) the same anomaly

- Mission "A" does not know about Mission "B" resolution

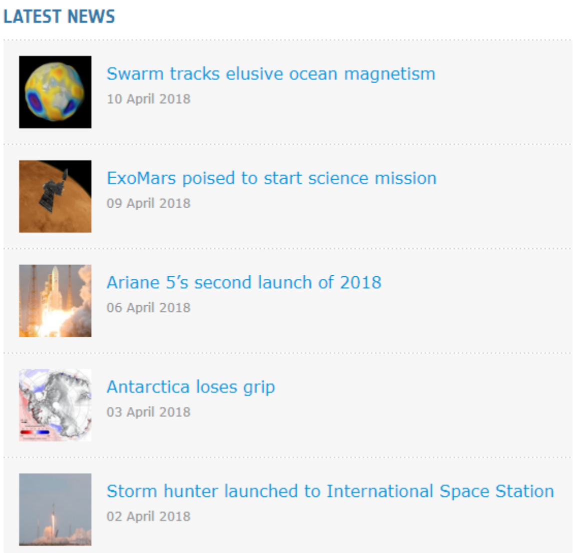
- **Text Mining** could automatically find these situations and contribute to the resolution of the anomaly for mission "A".

Materials: Slides, Code, Videos

Available on the Data Analytics ESA connect community

url: https://connect.esa.int/communities/community/data-analytics

Hands on: Text Mining on ESA News



http://www.esa.int/Our_Activities/Space_News

Resources

Watch the video of this lecture

https://dlmultimedia.esa.int/download/public/videos/2048/04/013/4804_013_AR_EN.mp4

Watch the practical exercise video

https://dlmultimedia.esa.int/download/public/videos/2048/04/012/4804_012_AR_EN.mp4

Get presentation and additional resources on

https://github.com/jmartinezheras/2018-MachineLearning-Lectures-ESA

Thank you

Data Analytics Team for Operations (DATO)

Jose Martinez Heras

LinkedIn: https://www.linkedin.com/in/josemartinezheras/