# IIS

Text Processing & NLP

# Outline

- Text Processing & NLP
- Bag of Words (BOW) & TF-IDF
- **Word Embeddings**

Repetition from PNAP

# Text Mining & NLP

- **Text mining**
  - The goal of text mining is to **discover relevant information in text** by transforming the text into data that can be used for further analysis.
  - Text mining accomplishes this through the use of a variety of analysis methodologies; **natural language processing (NLP)** is one of them.

- **Natural language processing (or NLP)** is a component of text mining that performs a special kind of linguistic analysis that essentially helps a machine “read” text.

# Knowledge discovery from text data

- IBM's Watson wins at Jeopardy! - 2011

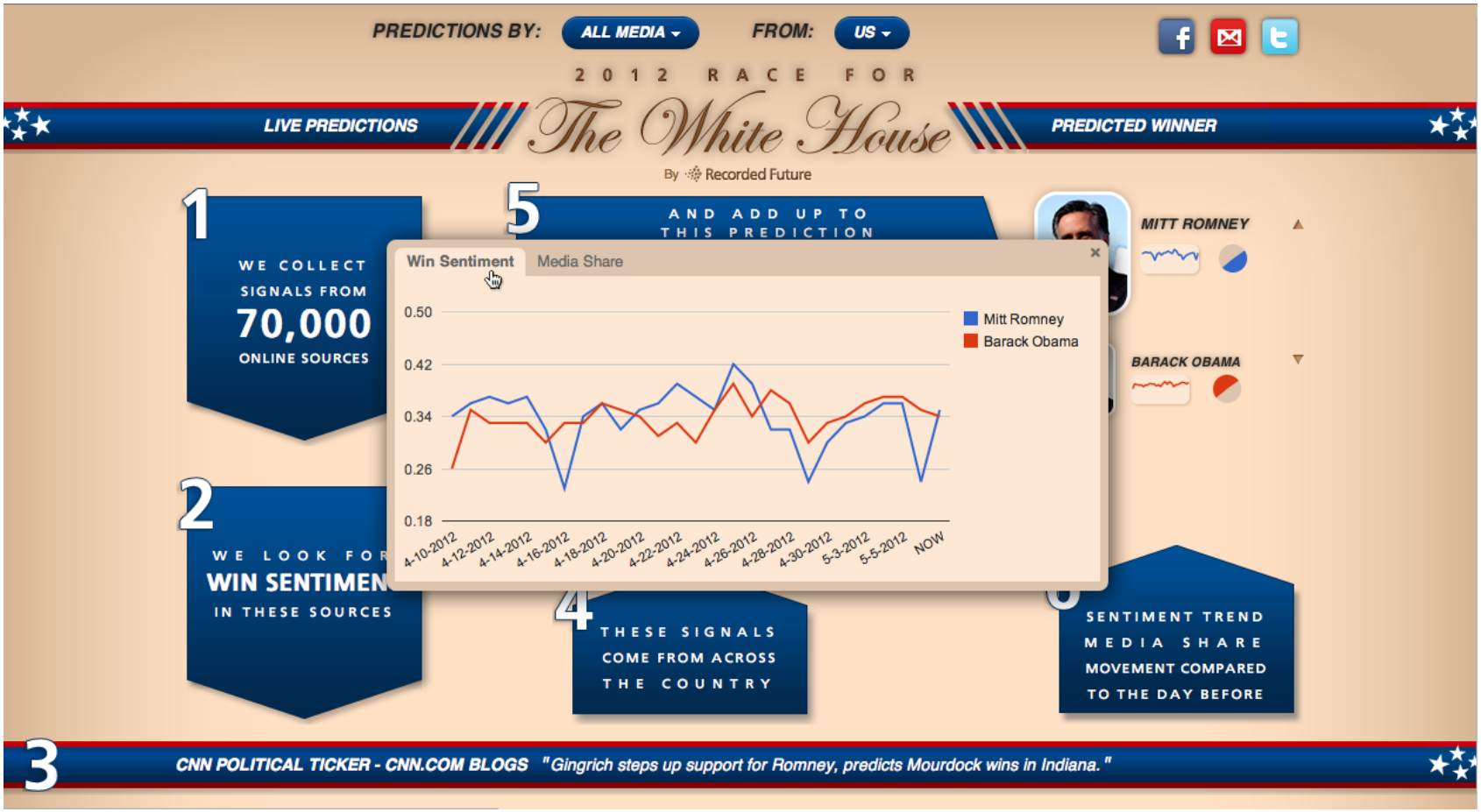
# What is inside Watson?

- *"Watson had access to 200 million pages of structured and unstructured content consuming four terabytes of disk storage including the full text of Wikipedia" – PC World*

- *"The sources of information for Watson include encyclopedias, dictionaries, thesauri, newswire articles, and literary works. Watson also used databases, taxonomies, and ontologies. Specifically, DBPedia, WordNet, and Yago were used." – AI Magazine*

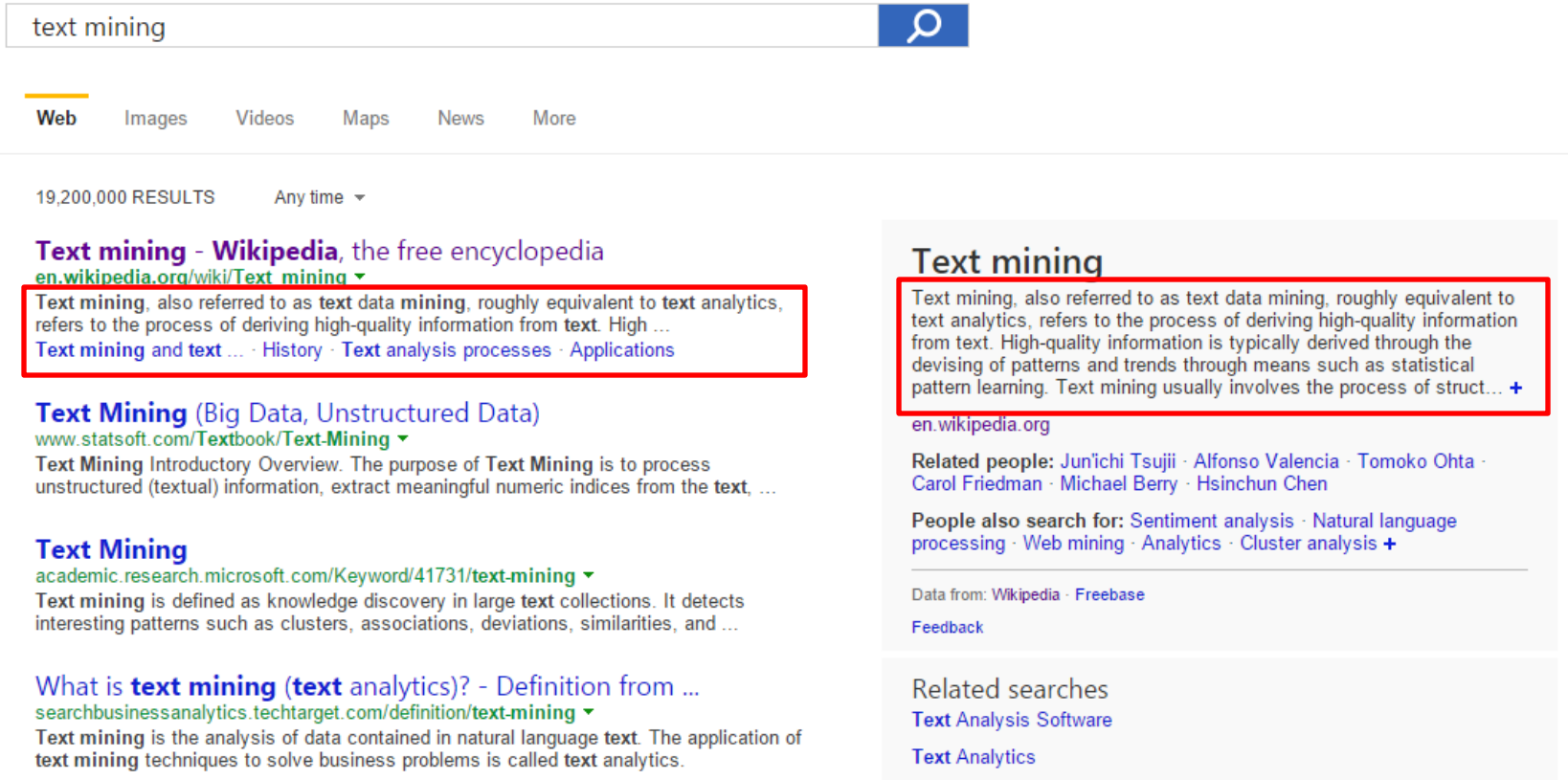
# NLP around us

- Sentiment analysis



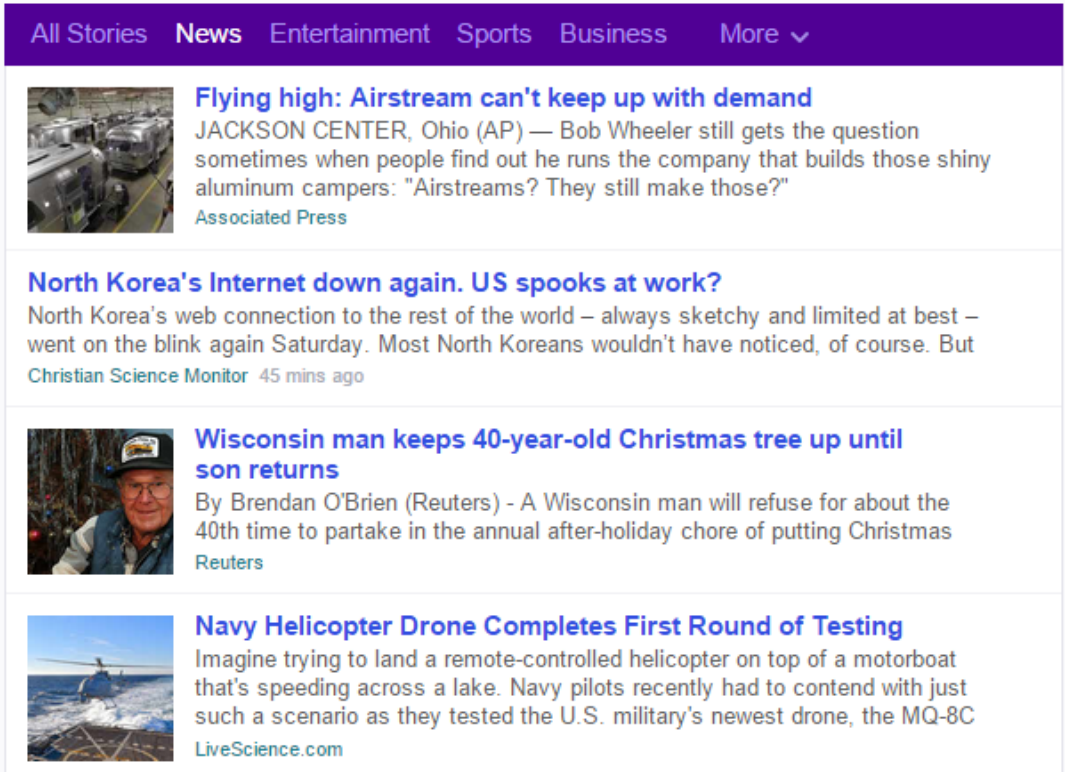
# NLP around us

- Document summarization



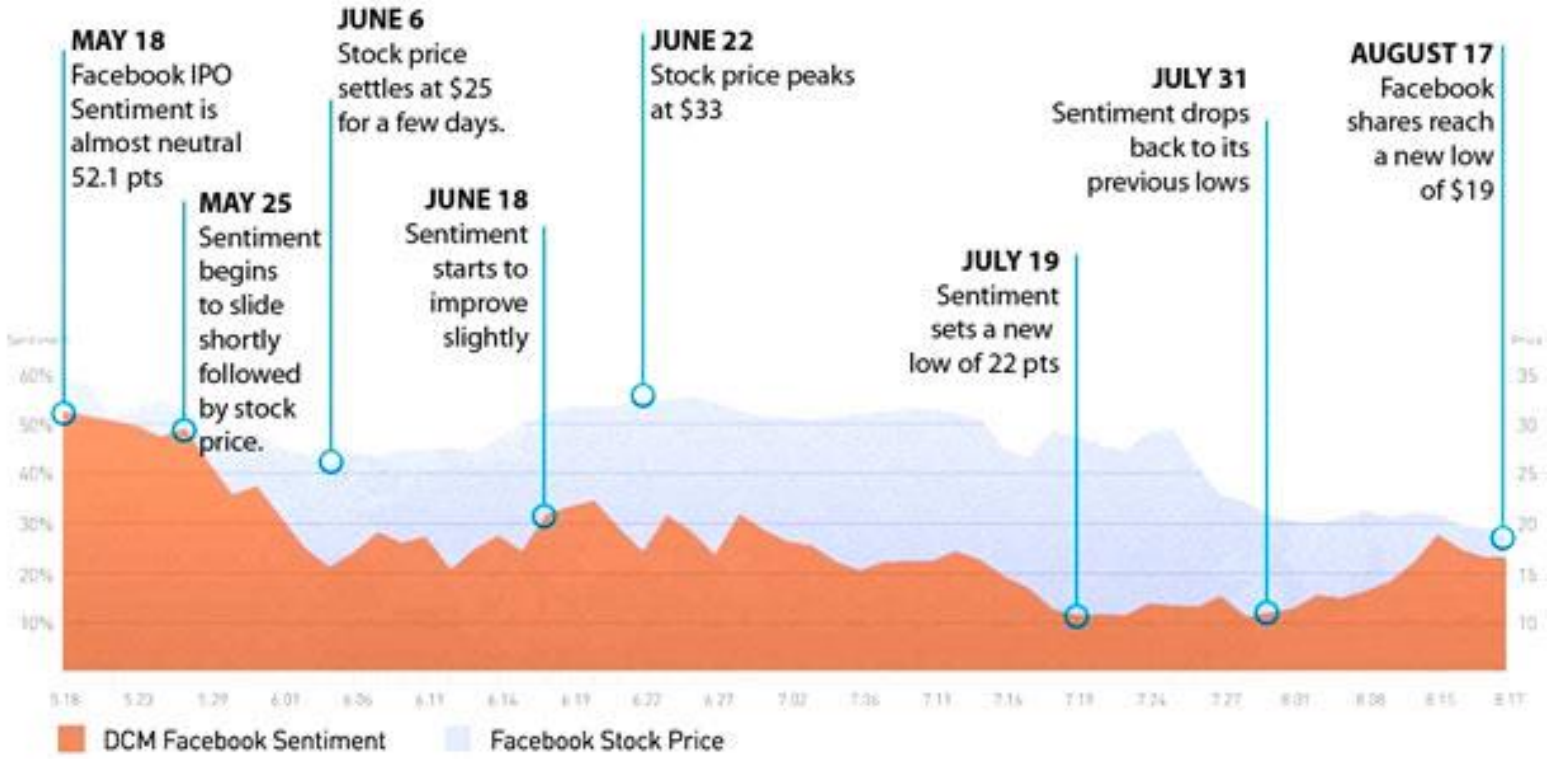
# NLP around us

- News recommendation



# NLP around us

- Text analytics in financial services



# Challenges in text processing

- Data collection is “free text”
  - Data is not well-organized
    - Semi-structured or unstructured
  - Natural language text contains ambiguities on many levels
    - Lexical, syntactic, semantic, and pragmatic
  - Learning techniques for processing text typically need annotated training examples
    - Expensive to acquire at scale

# Text processing problems

- Lexical semantics and word senses
  - Identifying which sense of a word (i.e. meaning) is used in a sentence, when the word has multiple meanings

Bass: fish

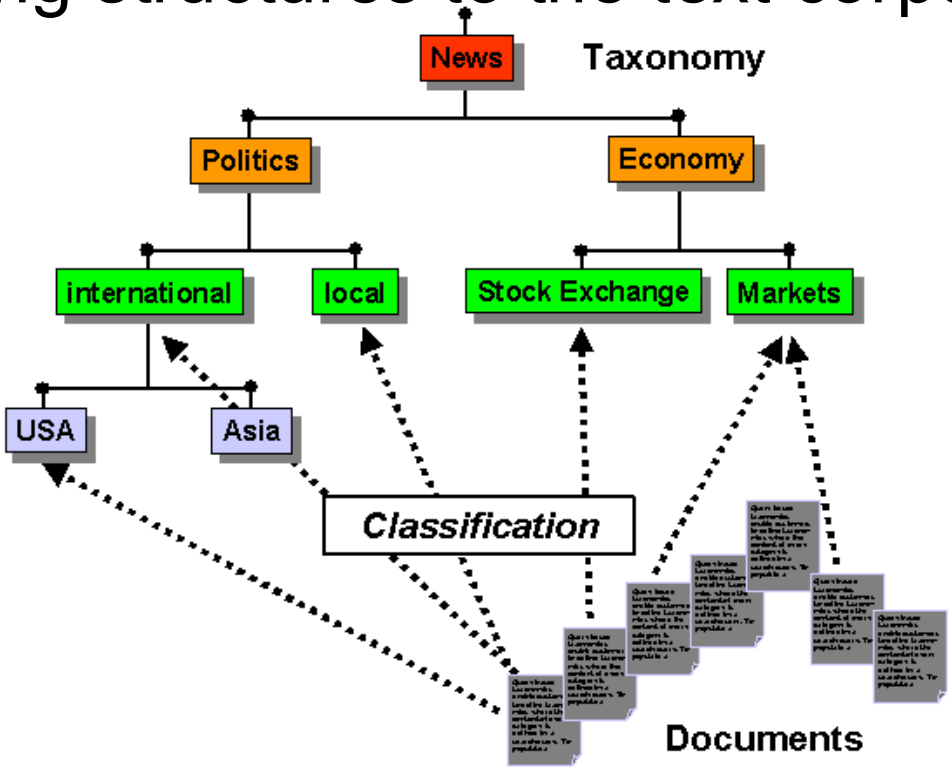
???


Bass: instrument

**Homonyms**

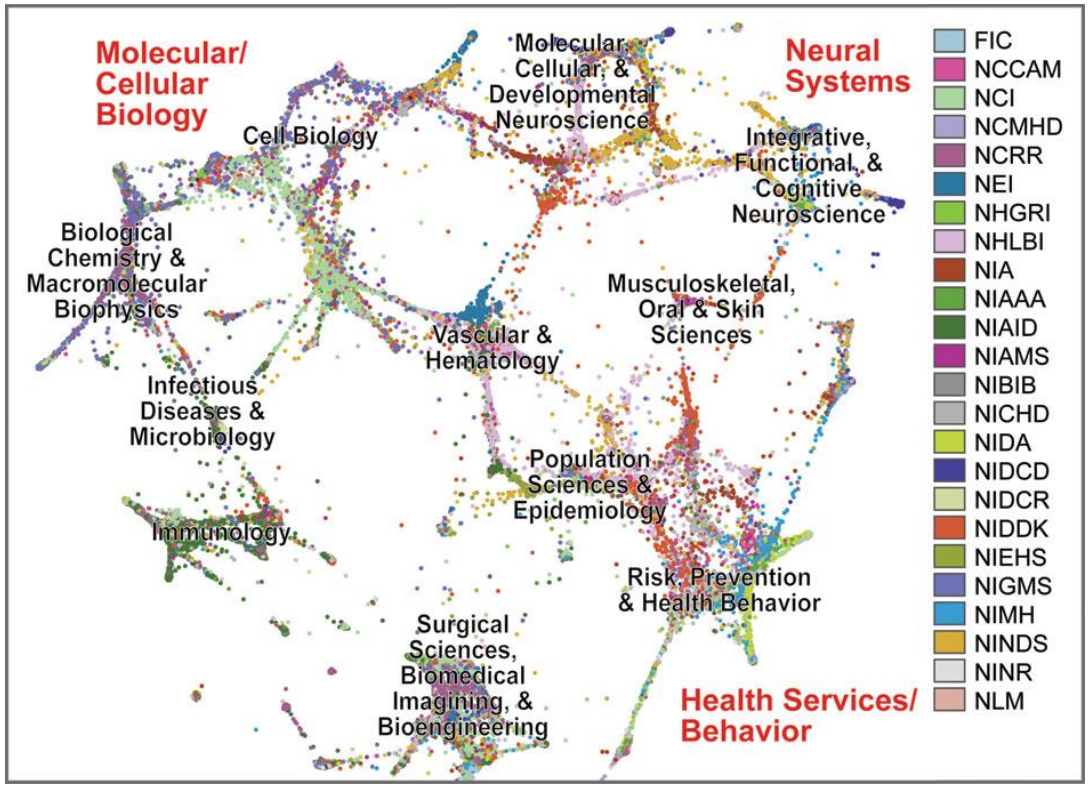
# Text processing problems

- Document categorization
  - Adding structures to the text corpus



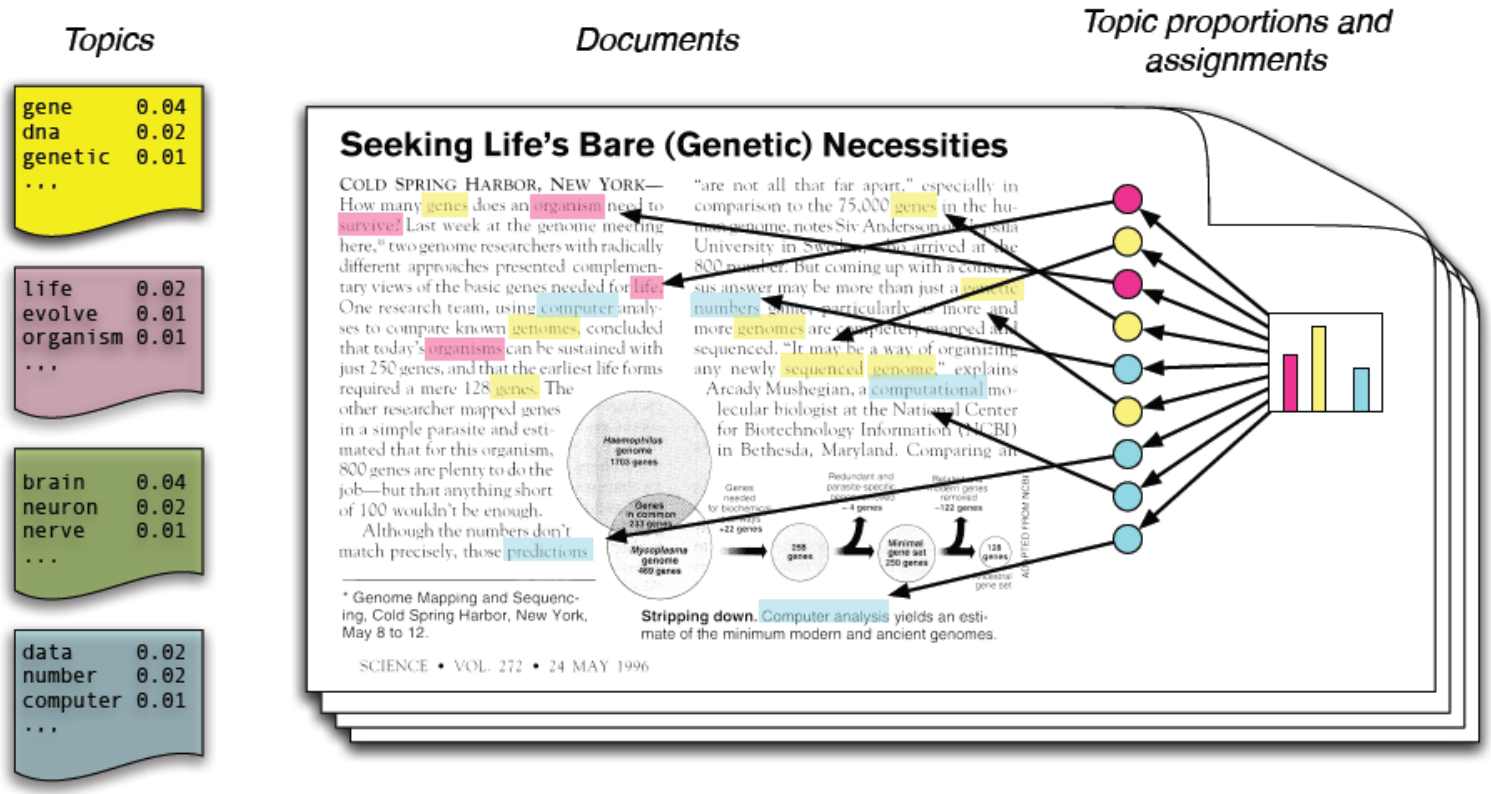
# Text processing problems

- Text clustering
  - Identifying structures in the text corpus



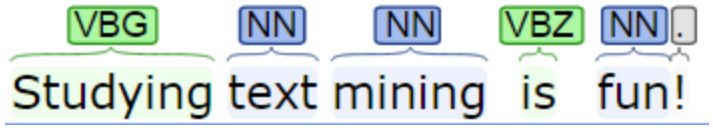
# Text processing problems

- Topic modeling
  - Identifying structures in the text corpus

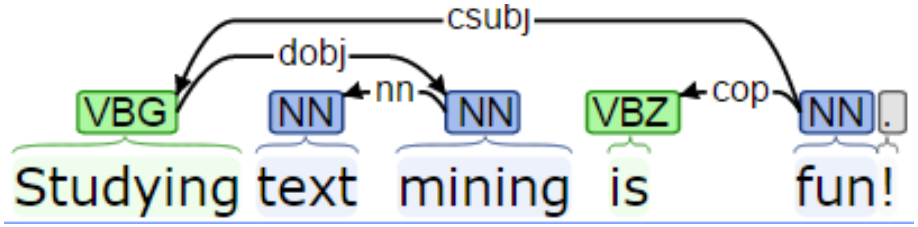


# NLP examples

- Tokenization
  - “Studying text mining is fun!” -> “studying” + “text” + “mining” + “is” + “fun” + “!”
- Part-of-speech tagging
  - “Studying text mining is fun!” ->




- Dependency parsing
  - “Studying text mining is fun!” ->




https://cs.nyu.edu/grishman/jet/guide/PennPOS.html

- Machine learning techniques
  - Supervised methods
    - Naïve Bayes, k Nearest Neighbors, Logistic Regression
  - Unsupervised methods
    - K-Means, hierarchical clustering, topic models
  - Semi-supervised methods
    - Expectation Maximization

# Text processing in the era of Big Data

- Huge in size
  - Google processes 5.13B queries/day (2013)
  - Twitter receives 340M tweets/day (2012)
  - Facebook has 2.5 PB of user data + 15 TB/day (4/2009)
  - eBay has 6.5 PB of user data + 50 TB/day (5/2009)
- 80% data is unstructured (IBM, 2010)

# NLP Pipeline

Levels of text representation

- Character (character n-grams and sequences)
- Words (stop-words, stemming, lemmatization)
- Phrases (word n-grams, proximity features)
- Part-of-speech tags
- Taxonomies / thesauri
- Vector-space model

# Levels of text representation

- **Character (character n-grams and sequences)**
- Words (stop-words, stemming, lemmatization)
- Phrases (word n-grams, proximity features)
- Part-of-speech tags
- Taxonomies / thesauri
- BOW and TF-IDF

# Character level

- Character level representation of a text consists from sequences of characters…
  - …a document is represented by a frequency distribution of sequences
  - Usually we deal with contiguous strings…
  - …each character sequence of length 1, 2, 3, … represent a feature with its frequency

# Good and bad sides

- Representation has several important strengths:
  - …it is very robust since avoids language morphology
    - (useful for e.g. language identification)
  - …it captures simple patterns on character level
    - (useful for e.g. spam detection, copy detection)
  - …because of redundancy in text data it could be used for many analytic tasks
    - (learning, clustering, search)
    - It is used as a basis for “string kernels” in combination with SVM for capturing complex character sequence patterns
- …for deeper semantic tasks, the representation is too weak

# Levels of text representation

- Character (character n-grams and sequences)
- **Words (stop-words, stemming, lemmatization)**

  **Tokenization**
  **Stemming**
  **Lemmatization/Normalization**
- Phrases (word n-grams, proximity features)
- Part-of-speech tags
- Taxonomies / thesauri
- Vector-space model

# Word level

- The most common representation of text used for many techniques
  - …there are many tokenization software packages which split text into the words
- Important to know:
  - Word is well defined unit in western languages – e.g. Chinese has different notion of semantic unit

# Words Properties

- Relations among word surface forms and their senses:
  - **Homonomy**: same form, but different meaning (e.g. bank: river bank, financial institution)
  - **Polysemy**: same form, related meaning (e.g. bank: blood bank, financial institution)
  - **Synonymy**: different form, same meaning (e.g. singer, vocalist)
  - **Hyponymy**: one word denotes a subclass of an another (e.g. breakfast, meal)

- Word frequencies in texts have **power distribution**:
  - …small number of very frequent words
  - …big number of low frequency words

# Tokenization

- Break a stream of text into meaningful units
  - Tokens: words, phrases, symbols
    - **Input:** It’s not straight-forward to perform so-called “tokenization.”
    - **Output(1):** 'It’s', 'not', 'straight-forward', 'to', 'perform', 'so-called', '“tokenization.”'
    - **Output(2):** 'It', '’', 's', 'not', 'straight', '-', 'forward, 'to', 'perform', 'so', '-', 'called', ‘“', 'tokenization', '.', ”’‘
  - Definition depends on language, corpus, or even context

# Tokenization

- Solutions
  - Regular expressions
    - [\w]+: so-called -> ‘so’, ‘called’
    - [\S]+: It’s -> ‘It’s’ instead of ‘It’, ‘’s’
  - Statistical methods
    - Explore rich features to decide where the boundary of a word is
      - Apache OpenNLP (http://opennlp.apache.org/)
      - Stanford NLP Parser (http://nlp.stanford.edu/software/lex-parser.shtml)
    - Online Demo
      - Stanford (http://nlp.stanford.edu:8080/parser/index.jsp)
      - UIUC (http://cogcomp.cs.illinois.edu/curator/demo/index.html)

# Stopwords

- Useless words for document analysis
  - Not all words are informative
  - Remove such words to reduce vocabulary size
  - No universal definition
  - Risk: break the original meaning and structure of text
    - E.g., this is not a good option -> option

      to be or not to be -> null

# Stopwords

| Nouns | Verbs | Adjectives | Prepositions | Others |
|---|---|---|---|---|
| 1. time | 1. be | 1. good | 1. to | 1. the |
| 2. person | 2. have | 2. new | 2. of | 2. and |
| 3. year | 3. do | 3. first | 3. in | 3. a |
| 4. way | 4. say | 4. last | 4. for | 4. that |
| 5. day | 5. get | 5. long | 5. on | 5. I |
| 6. thing | 6. make | 6. great | 6. with | 6. it |
| 7. man | 7. go | 7. little | 7. at | 7. not |
| 8. world | 8. know | 8. own | 8. by | 8. he |
| 9. life | 9. take | 9. other | 9. from | 9. as |
| 10. hand | 10. see | 10. old | 10. up | 10. you |
| 11. part | 11. come | 11. right | 11. about | 11. this |
| 12. child | 12. think | 12. big | 12. into | 12. but |
| 13. eye | 13. look | 13. high | 13. over | 13. his |
| 14. woman | 14. want | 14. different | 14. after | 14. they |
| 15. place | 15. give | 15. small | 15. beneath | 15. her |
| 16. work | 16. use | 16. large | 16. under | 16. she |
| 17. week | 17. find | 17. next | 17. above | 17. or |
| 18. case | 18. tell | 18. early | | 18. an |
| 19. point | 19. ask | 19. young | | 19. will |
| 20. government | 20. work | 20. important | | 20. my |
| 21. company | 21. seem | 21. few | | 21. one |
| 22. number | 22. feel | 22. public | | 22. all |
| 23. group | 23. try | 23. bad | | 23. would |
| 24. problem | 24. leave | 24. same | | 24. there |
| 25. fact | 25. call | 25. able | | 25. their |

The OEC: Facts about the language

# Stemming

- Reduce inflected or derived words to their root form
  - Plurals, adverbs, inflected word forms
    - E.g., ladies -> lady, referring -> refer, forgotten -> forget
  - Bridge the vocabulary gap
  - Solutions (for English)
    - Porter stemmer: patterns of vowel-consonant sequence
    - Krovetz stemmer: morphological rules
  - Risk: lose precise meaning of the word
    - E.g., lay -> lie (a false statement? or be in a horizontal position?)

# Normalization/Lemmatization

- Convert different forms of a word to a normalized form in the vocabulary
  - U.S.A. -> USA, St. Louis -> Saint Louis
- Solution
  - Rule-based
    - Delete periods and hyphens
    - All in lower cases
  - Dictionary-based
    - Construct equivalent class
      - Car -> “automobile, vehicle”
      - Mobile phone -> “cellphone”

# Levels of text representation

- Character (character n-grams and sequences)
- Words (stop-words, stemming, lemmatization)
- **Phrases (word n-grams, proximity features)**
- Part-of-speech tags
- Taxonomies / thesauri
- Vector-space model

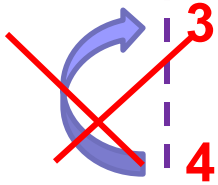
# N-grams

- N-grams: a contiguous sequence of N tokens from a given piece of text
  - E.g., '*Text mining is to identify useful information.*'
  - Bigrams: '*text_mining*', '*mining_is*', '*is_to*', '*to_identify*', '*identify_useful*', '*useful_information*', '*information_.*'
- Pros: capture local dependency and order
- Cons: a purely statistical view, increase the vocabulary size $O(V^N)$

# Text Processing process

D1: '*Text mining is to identify useful information.*'

1. **Tokenization:**
   D1: *'Text', 'mining', 'is', 'to', 'identify', 'useful', 'information', '.'*
2. **Stemming/normalization:**
   D1: *'text', 'mine', 'is', 'to', 'identify', 'use', 'inform', '.'*
3. **N-gram construction:**
   D1: *'text-mine', 'mine-is', 'is-to', 'to-identify', 'identify-use', 'use-inform', 'inform-*
4. **Stopword/controlled vocabulary filtering:**
   D1: *'text-mine', 'to-identify', 'identify-use', 'use-inform'*

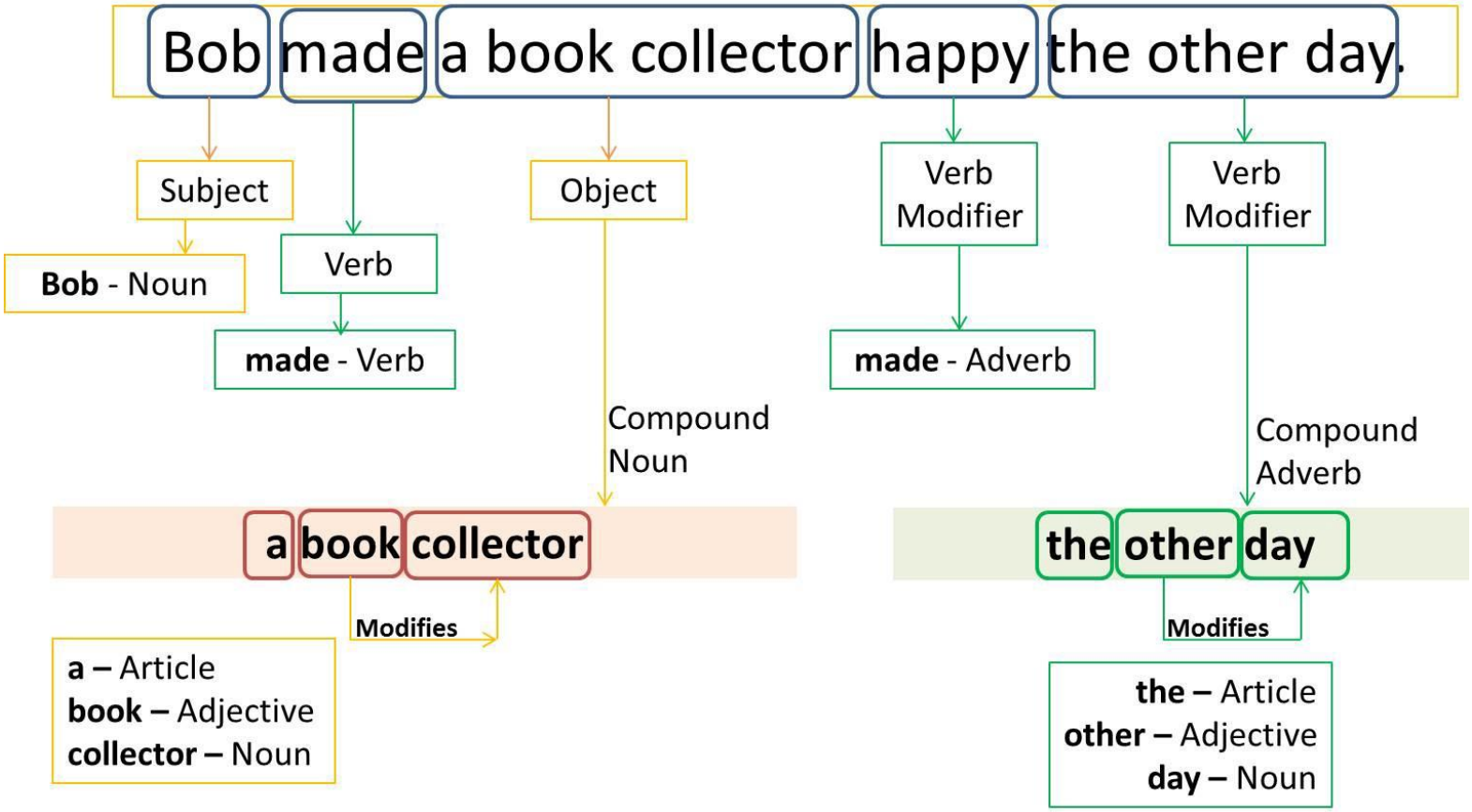


# Levels of text representation

- Character (character n-grams and sequences)
- Words (stop-words, stemming, lemmatization)
- Phrases (word n-grams, proximity features)
- **Part-of-speech tags**
- Taxonomies / thesauri
- Vector-space model

# Part of Speech tagging

- By introducing part-of-speech tags we introduce word-types enabling to differentiate words functions
  - Nouns, Verbs, Adjectives, …
- Part-of-Speech taggers are usually learned by HMM algorithm on manually tagged data


Bob made a book collector happy the other day.
Subject
Bob - Noun
Verb
made - Verb
Object
Verb Modifier
made - Adverb
Verb Modifier
Compound Noun
Compound Adverb
a book collector
the other day
Modifies
Modifies
a – Article
book – Adjective
collector – Noun
the – Article
other – Adjective
day – Noun

| part of speech | function or "job" | example words | example sentences |
|---|---|---|---|
| Verb | action or state | (to) be, have, do, like, work, sing, can, must | EnglishClub **is** a web site. I **like** EnglishClub. |
| Noun | thing or person | pen, dog, work, music, town, London, teacher, John | This is my **dog**. He lives in my **house**. We live in **London**. |
| Adjective | describes a noun | good, big, red, well, interesting | My dogs are **big**. I like **big** dogs. |
| Determiner | limits or "determines" a noun | a/an, the, 2, some, many | I have **two** dogs and **some** rabbits. |
| Adverb | describes a verb, adjective or adverb | quickly, silently, well, badly, very, really | My dog eats **quickly**. When he is **very** hungry, he eats **really** quickly. |
| Pronoun | replaces a noun | I, you, he, she, some | Tara is Indian. **She** is beautiful. |
| Preposition | links a noun to another word | to, at, after, on, but | We went **to** school **on** Monday. |
| Conjunction | joins clauses or sentences or words | and, but, when | I like dogs **and** I like cats. I like cats **and** dogs. I like dogs **but** I don't like cats. |
| Interjection | short exclamation, sometimes inserted into a sentence | oh!, ouch!, hi!, well | **Ouch**! That hurts! **Hi**! How are you? **Well**, I don't know. |

https://www.englishclub.com/grammar/parts-of-speech.htm

# Levels of text representation

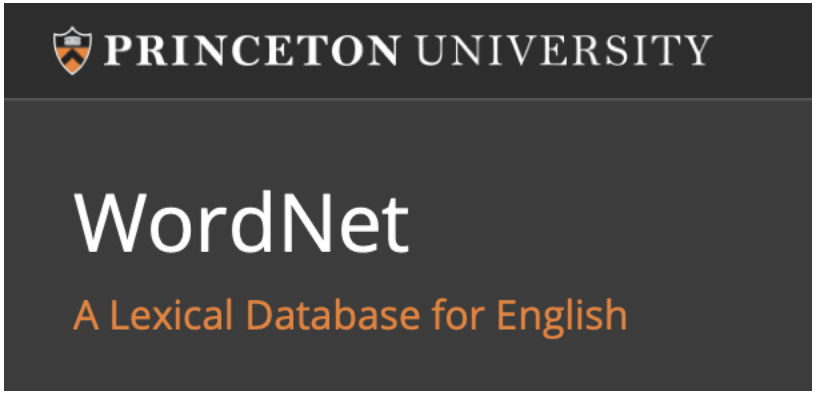
- Character (character n-grams and sequences)
- Words (stop-words, stemming, lemmatization)
- Phrases (word n-grams, proximity features)
- Part-of-speech tags
- **Taxonomies / thesauri**
- Vector-space model

# Taxonomies/thesaurus level

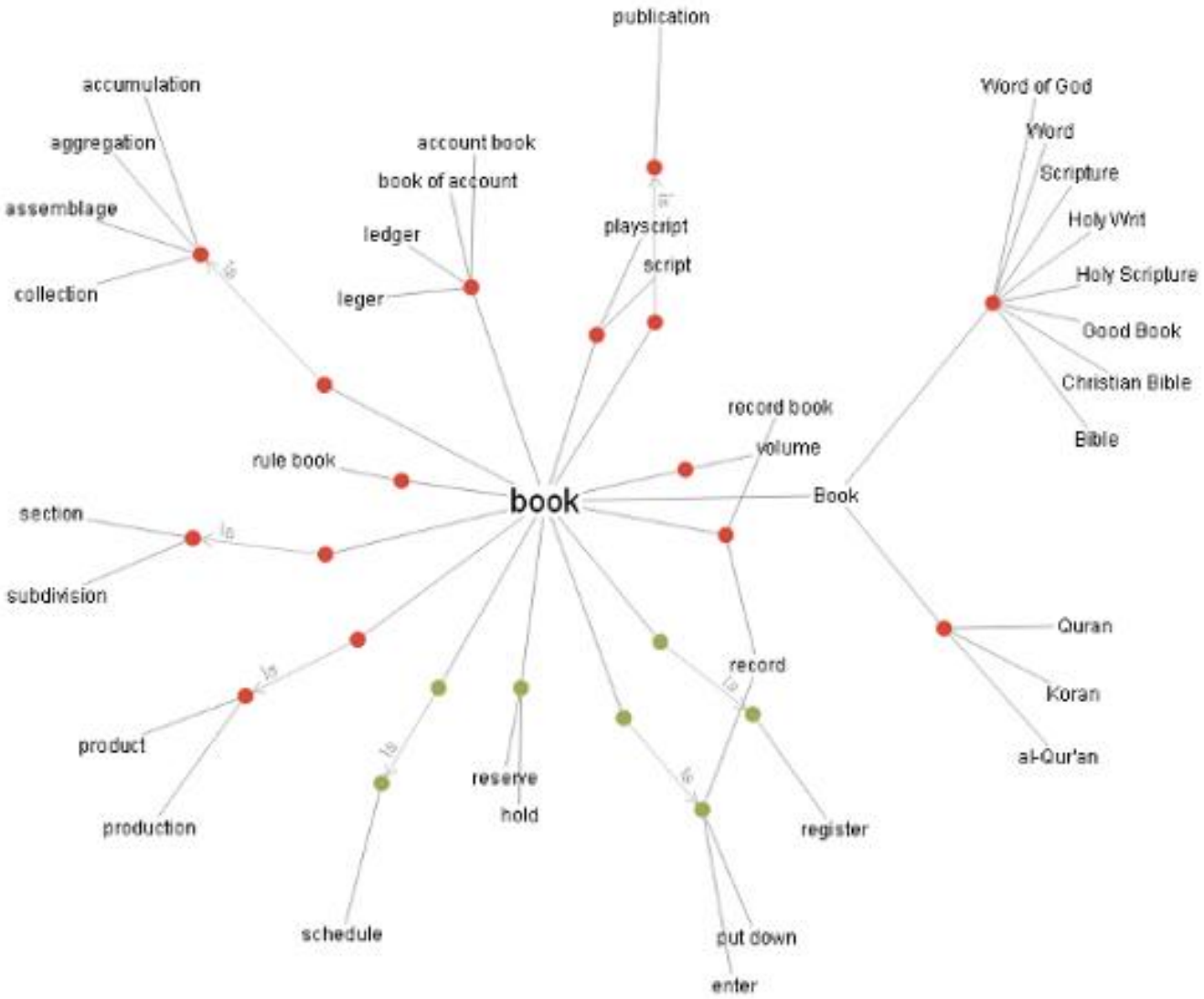
- Thesaurus has a main function to connect different surface word forms with the same meaning into one sense (synonyms)
  - …additionally we often use hypernym relation to relate general-to-specific word senses
  - …by using synonyms and hypernym relation we compact the feature vectors
- The most commonly used general thesaurus is WordNet which exists in many languages (e.g. EuroWordNet)
  - http://www.illc.uva.nl/EuroWordNet/

# WordNet

- WordNet is a lexical database of semantic relations between words in more than 200 languages.
- WordNet links words into semantic relations including synonyms, hyponyms, and meronyms.



# WordNet


publication
accumulation
aggregation
assemblage
collection
account book
book of account
ledger
leger
playscript
script
Word of God
Word
Scripture
Holy Writ
Holy Scripture
Good Book
Christian Bible
Bible
record book
volume
rule book
book
Book
section
subdivision
Quran
Koran
al-Qur'an
record
product
production
reserve
hold
register
schedule
put down
enter

# Resources

- http://www.cs.virginia.edu/~hw5x/Course/TextMining-2018Spring/_site/lectures/

- Text, Web and Multimedia Mining  - MPS Jozef Stefan, Slovenia

# Bag of Words and TF-IDF

# Levels of text representation

- Character (character n-grams and sequences)
- Words (stop-words, stemming, lemmatization)
- Phrases (word n-grams, proximity features)
- Part-of-speech tags
- Taxonomies / thesauri
- **Bag of Words (BOW) model**
- Word Embeddings

# How to represent a document

## University of Virginia

From Wikipedia, the free encyclopedia

The **University of Virginia** (**UVA** or **U.Va.**), often referred to as simply **Virginia**, is a public research university in Charlottesville, Virginia. UVA is known for its historic foundations, student-run honor code, and secret societies.

Its initial Board of Visitors included U.S. Presidents Thomas Jefferson, James Madison, and James Monroe. President Monroe was the sitting President of the United States at the time of the founding; Jefferson and Madison were the first two rectors. UVA was established in 1819, with its Academical Village and original courses of study conceived and designed entirely by Jefferson. UNESCO designated it a World Heritage Site in 1987, an honor shared with nearby Monticello.[4]

The first university of the American South elected to the Association of American Universities in 1904, UVA is classified as *Very High Research Activity* in the Carnegie Classification. The university is affiliated with 7 Nobel Laureates, and has produced 7 NASA astronauts, 7 Marshall Scholars, 4 Churchill Scholars, 29 Truman Scholars, and 50 Rhodes Scholars, the most of any state-affiliated institution in the U.S.[5][6][7] Supported in part by the Commonwealth, it receives far more funding from private sources than public, and its students come from all 50 states and 147 countries.[2][8][9] It also operates a small liberal arts branch campus in the far southwestern corner of the state.

# Bag-of-Words (BOW) representation

- Algorithm that counts how many times a word appears in a document.
- Those word counts allow to compare documents and find their similarities for applications like search, document classification, and topic modeling.
- BOW is a method for preparing text for input in a deep-learning net.

# Bag-of-Words (BOW) representation

- Term as the basis for vector space
  - Doc1: Text mining is to identify useful information.
  - Doc2: Useful information is mined from text.
  - Doc3: Apple is delicious.

| | text | information | identify | mining | mined | is | useful | to | from | apple | delicious |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Doc1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| Doc2 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| Doc3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |

# Bag-of-Words (BOW) representation

```
import pandas as pd

import numpy as np

from sklearn.feature_extraction.text import CountVectorizer

data = ['It was the best of times','it was the worst of times','it was the age of wisdom','it was the age of foolishness']

data_vectorizer = CountVectorizer(stop_words='english')

data_feature = data_vectorizer.fit_transform(data)
```

```
data_frame=pd.DataFrame(data = data_feature.todense(), columns=data_vectorizer.get_feature_names())
data_frame
```

| | age | best | foolishness | times | wisdom | worst |
|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 1 | 0 | 0 |
| **1** | 0 | 0 | 0 | 1 | 0 | 1 |
| **2** | 1 | 0 | 0 | 0 | 1 | 0 |
| **3** | 1 | 0 | 1 | 0 | 0 | 0 |

# TF-IDF

Having a set of documents, represent each as a feature vector:

1. divide text into units (eg., words), remove punctuation, (remove stop-words, stemming,…)
2. each unit becomes a feature having numeric weight as its value (eg., number of occurrences in the text - referred to as term frequency or TF)

Commonly used weight is TFIDF:

$$TFIDF(w) = tf(w) * \log\left(\frac{N}{df(w)}\right)$$

- tf(w) – term frequency (no. of occurrences of word *w* in document)
- df(w) – document frequency (no. of documents containing word *w*)
- N – no. of all documents

# Term frequency

- Idea: a term is more important if it occurs more frequently in a document
- TF Formulas
  - $\text{tf}(t, d)$ is the frequency count of term $t$ in doc $d$

# Document frequency

- document frequency (no. of documents containing word w)
- Idea: a term is more discriminative if it occurs only in fewer documents

# Inverse document frequency

- Solution
  - Assign higher weights to the rare terms
  - Formula
    - $IDF(t) = 1 + \log(\frac{N}{df(t)})$

Non-linear scaling

Total number of docs in collection

Number of docs containing term $t$

  - A corpus-specific property
    - Independent of a single document

# Why document frequency

- How about total term frequency?
  - $ttf(t) = \sum_d c(t, d)$

Table 1. Example total term frequency v.s. document frequency in Reuters-RCV1 collection.

| Word | ttf | df |
|---|---|---|
| try | 10422 | 8760 |
| insurance | 10440 | 3997 |

  - Cannot recognize words frequently occurring in a subset of documents

# TF-IDF weighting

- Combining TF and IDF
  - Common in doc → high tf → high weight
  - Rare in collection→ high idf→ high weight
  - $w(t,d) = TF(t,d) \times IDF(t)$
- Most well-known document representation schema in IR! (G Salton et al. 1983)

*"Salton was perhaps the leading computer scientist working in the field of information retrieval during his time."* - wikipedia

Gerard Salton Award – highest achievement award in IR

# TF-IDF

```
import pandas as pd

import numpy as np

from sklearn.feature_extraction.text import TfidfVectorizer

data = ['It was the best of times','it was the worst of times','it was the age of wisdom','it was the age of foolishness']

tfidf_vectorizer = TfidfVectorizer(stop_words='english')

tfidf_feature = tfidf_vectorizer.fit_transform(data)
```

```
dataframe=pd.DataFrame(data=tfidf_feature.todense(), columns=tfidf_vectorizer.get_feature_names())
dataframe
```

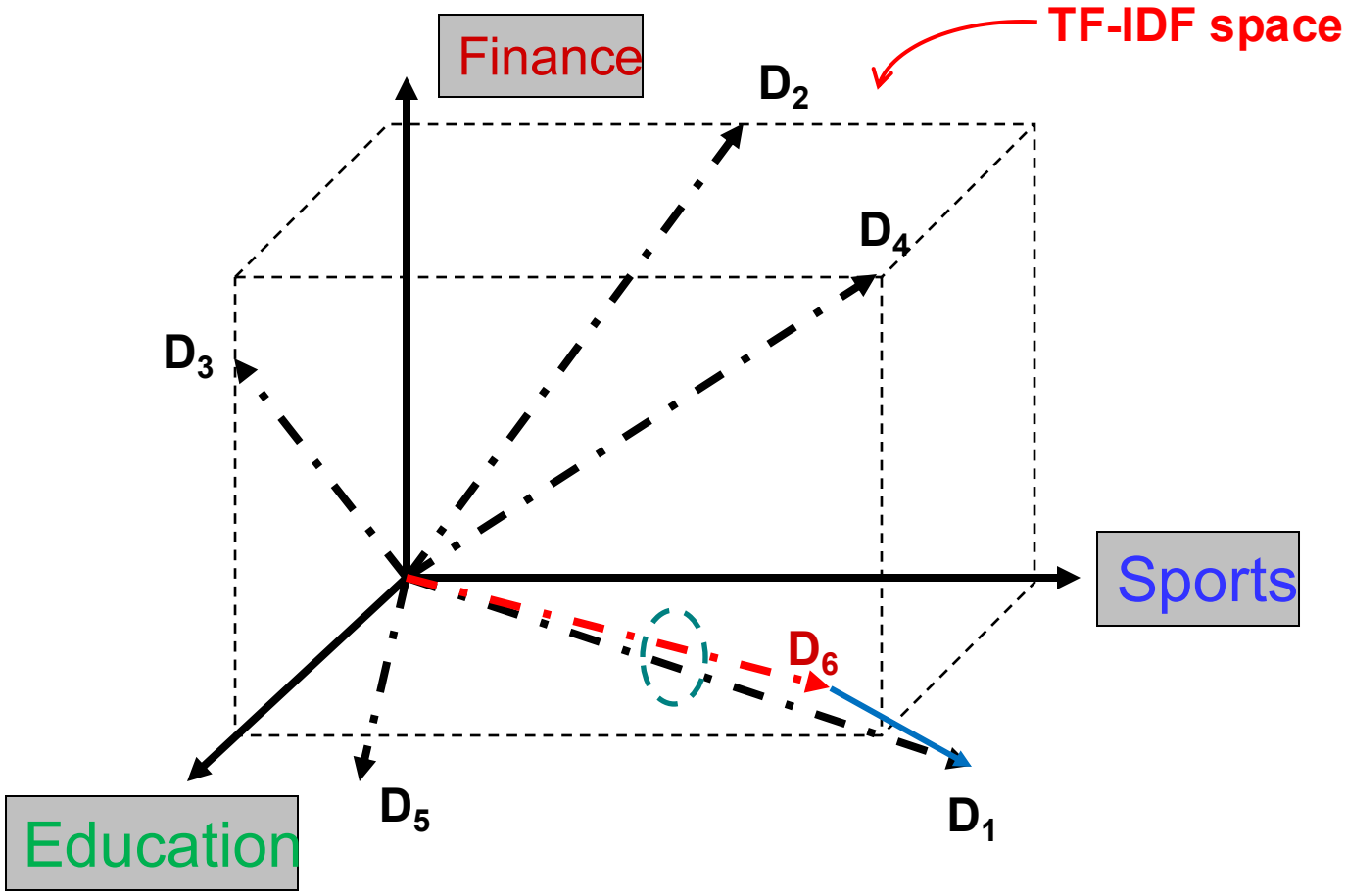
| | age | best | foolishness | times | wisdom | worst |
|---|---|---|---|---|---|---|
| **0** | 0.00000 | 0.930324 | 0.000000 | 0.366739 | 0.000000 | 0.000000 |
| **1** | 0.00000 | 0.000000 | 0.000000 | 0.619130 | 0.000000 | 0.785288 |
| **2** | 0.61913 | 0.000000 | 0.000000 | 0.000000 | 0.785288 | 0.000000 |
| **3** | 0.61913 | 0.000000 | 0.785288 | 0.000000 | 0.000000 | 0.000000 |

# Similarity Measure

- Once we have the weights, the vector (TF-IDF) we need a similarity measure between the vectors.

# How to define a good similarity metric?

- Euclidean distance?

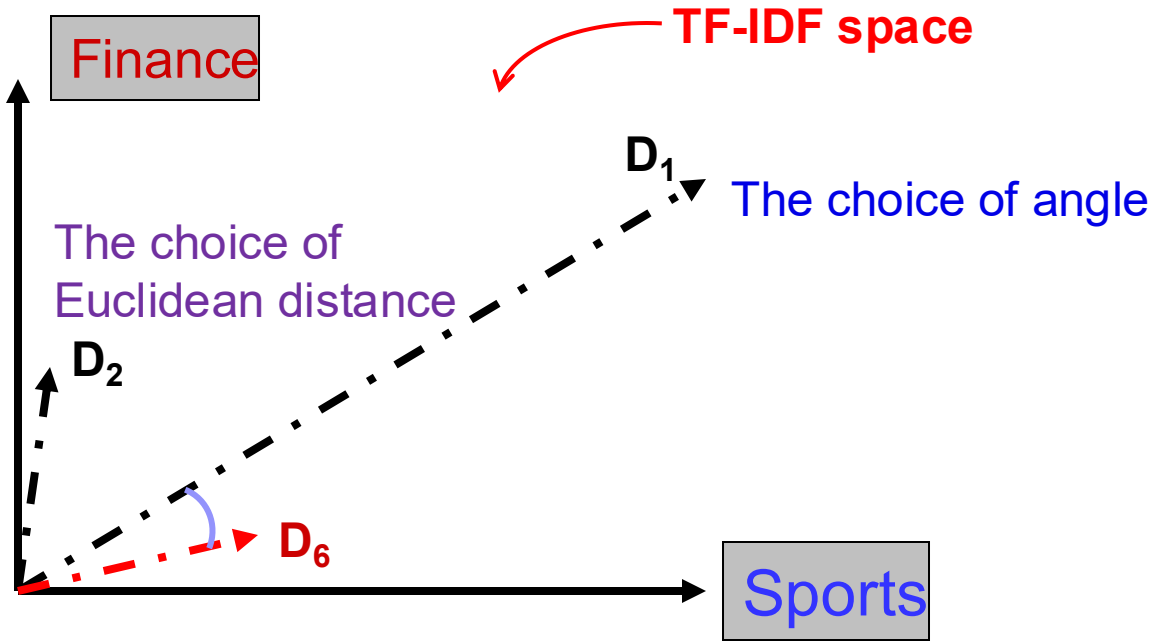


# How to define a good similarity metric?

- Euclidean distance
  - $dist(d_i, d_j) = \sqrt{\sum_{t \in V} [tf(t, d_i) idf(t) - tf(t, d_j) idf(t)]^2}$
  - Longer documents will be penalized by the extra words
  - We care more about how these two vectors are overlapped

# From distance to angle

- Angle: how vectors are overlapped
  - Cosine similarity – projection of one vector onto another



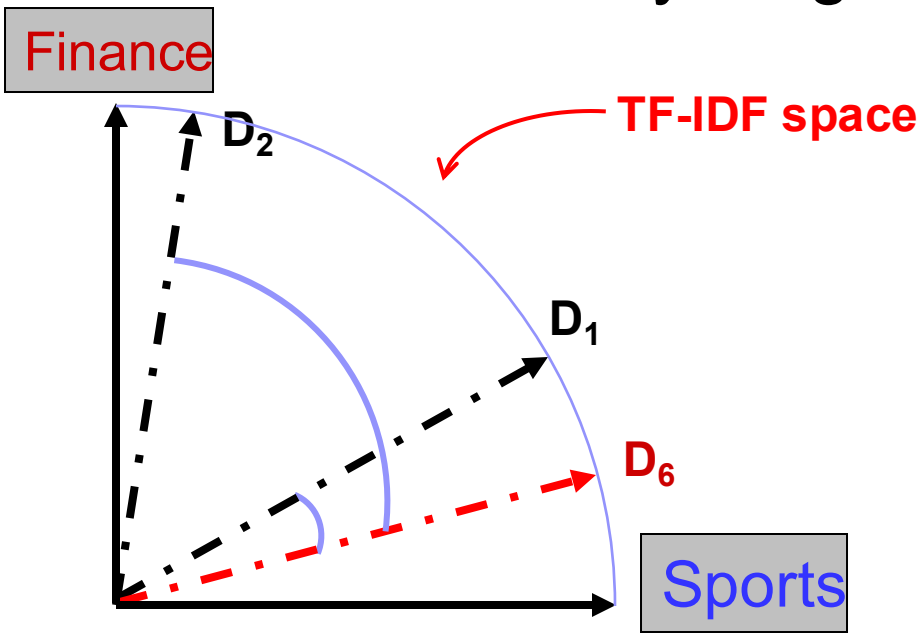
# Cosine similarity

- Angle between two vectors
  - $cosine(d_i, d_j) = \frac{V_{d_i}^T V_{d_j}}{|V_{d_i}|_2 \times |V_{d_j}|_2}$

TF-IDF vector

Unit vector

  - Documents are normalized by length



# Cosine similarity

- Each document is represented as a vector of weights D = <x>
- Cosine similarity (dot product) is the most widely used similarity measure between two document vectors
  - …calculates cosine of the angle between document vectors
  - …efficient to calculate (sum of products of intersecting words)
  - …similarity value between 0 (different) and 1 (the same)

$$Sim(D_1, D_2) = \frac{\sum_i x_{1i} x_{2i}}{\sqrt{\sum_j x_j^2} \sqrt{\sum_k x_k^2}}$$

# Advantages of BOW and TF-IDF model

- Empirically effective!
- Intuitive
- Easy to implement
- Well-studied/mostly evaluated
- Still widely used
- Warning: many variants of TF-IDF!

# Disadvantages of BOW and TF-IDF model

- Assumes term independence
- Lots of parameter tuning!

# Software

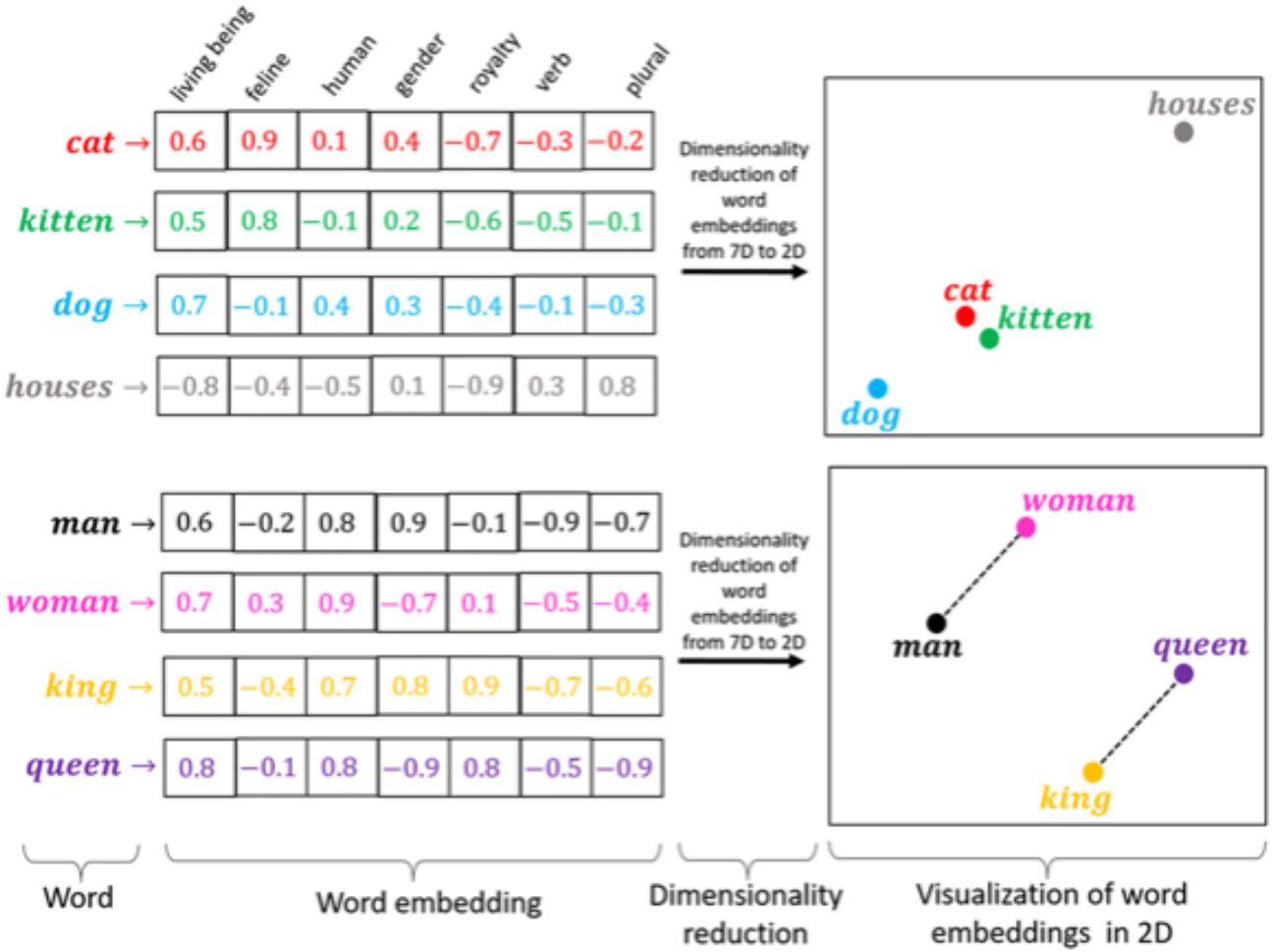
- Apache Lucene. Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java.
- Elasticsearch. Another high-performance, full-featured text search engine using Lucene.
- Gensim is a Python+NumPy framework for Vector Space modelling.
- Weka. Weka is a popular data mining package for Java including WordVectors and Bag Of Words models.
- Word2vec. Word2vec uses vector spaces for word embeddings.

# Levels of text representation

- Character (character n-grams and sequences)
- Words (stop-words, stemming, lemmatization)
- Phrases (word n-grams, proximity features)
- Part-of-speech tags
- Taxonomies / thesauri
- Vector-space model
- **Word Embeddings**

# Word Embeddings

# Word Embeddings




A **word embedding** is a learned **representation** for text where words that have the **same meaning** have a **similar representation**.

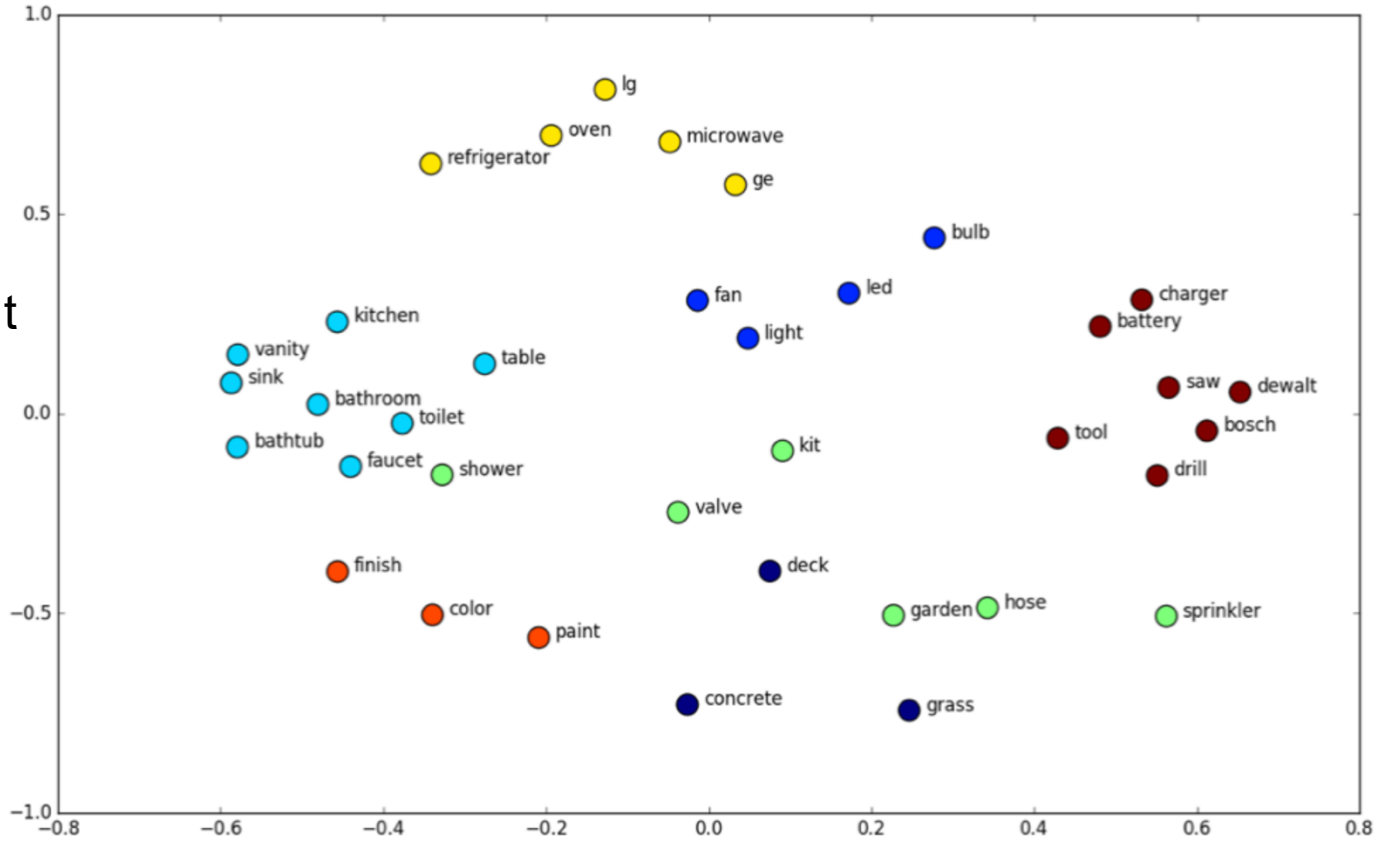
# Word Embeddings

- The collective name for a **set of language modeling** where **words or phrases** from the vocabulary are **mapped to vectors of real numbers**.
- Words are represented as **real-valued vectors** in a predefined vector space.
- Each word is **mapped to one vector** and the vector values are **learned** in a way that resembles a neural network (often used in deep learning)
- The vector is usually **tens or hundreds of dimensions**. This is contrasted to the thousands or millions of dimensions required for sparse word representations (vector-space model – TF-IDF)

# Word Embeddings

- **Related words are closer** (‘house’ and ‘home’) - similar n-dimensional vectors. **Dissimilar words** (‘house’ and ‘airplane’) are **further away**
- The ‘**meaning**’ of a word is reflected in its **embedding**, a model is then able to use this information to learn the **relationship between words**.

- The benefit is that a model trained on the word ‘**house**’ will be able to react to the word ‘**home**’ even if it had never seen that word in training.



# Word Embeddings

- **Learn specific embedding** for your problem
  Requires a large amount of text data to ensure that useful embeddings are learned, such as millions or billions of words.
  - **Learn it Standalone** – separate embedding model, which is saved and used as a part of another model for your task. (Use the same embedding in multiple models)
  - **Learn Jointly** – learned as part of a large task-specific model (sentiment analysis). This is a good approach if you only intend to use the embedding on one task.
- **Reuse pretrained Embedding** (word2vec and GloVe)
  - **Static** – the embeddings are used as they are provided. Good if the embeddings are similar to your problem domain
  - **Updated** – the pre-trained embedding is used to seed the model, but the embedding is updated jointly during the training of the model.

# Pre-trained - Word Embeddings

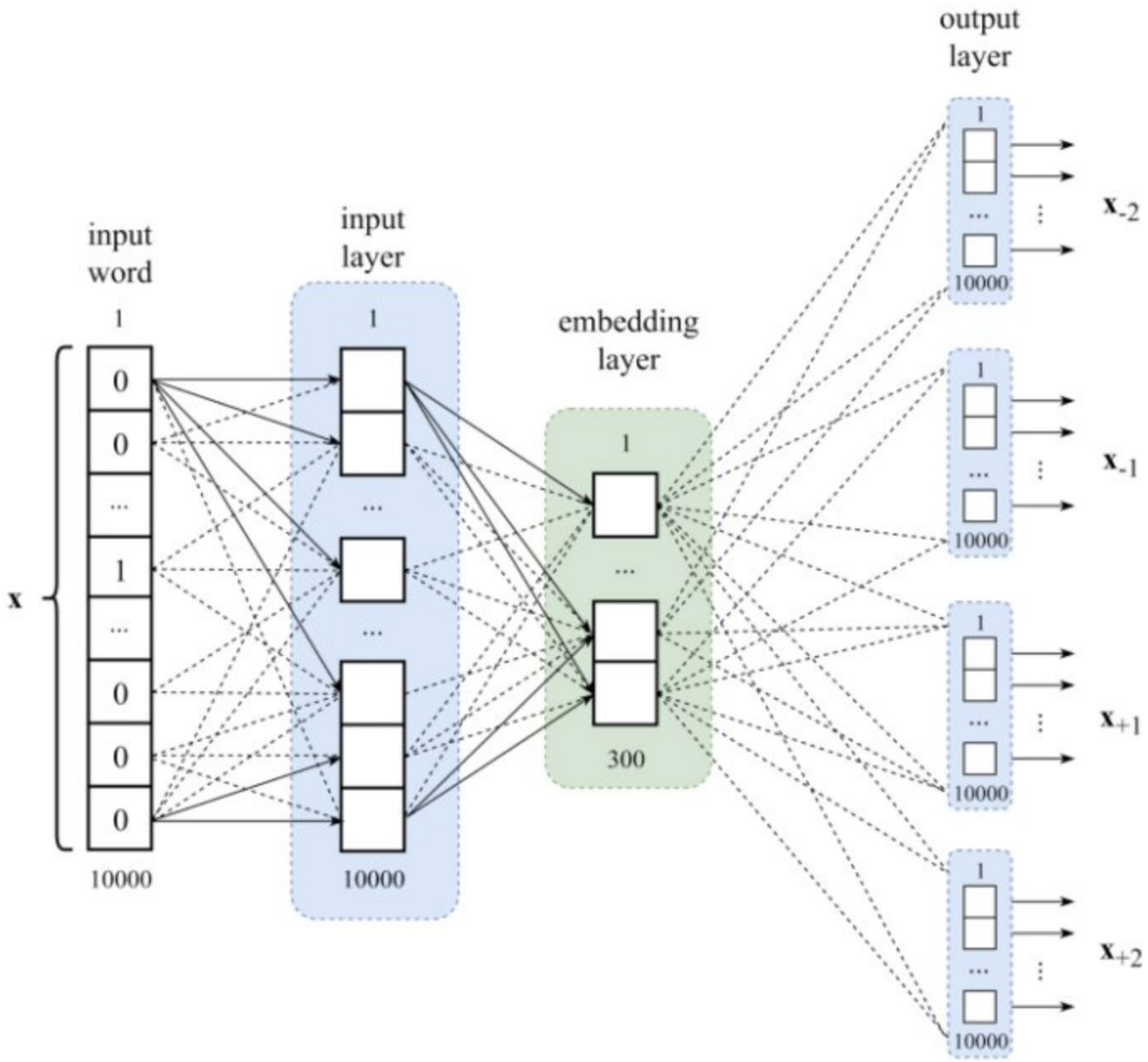
- **FastText**: https://fasttext.cc
  Library for efficient text classification and representation learning – by Facebook AI Research center – FAIR
- Trained on Wikipedia corpus
- Open-source, free, lightweight library that allows users to learn text representations and text classifiers.
- Models can later be reduced in size to even fit on mobile devices.
- Download pre-trained models for 157 different languages
- Macedonian:
  https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.mk.300.vec.gz

http://koneski.manu.edu.mk

# Word Embeddings – Approaches

- Embedding layer
- Word2Vec
- GloVe
- FastText
- BERT
- GPT

# Embedding Layer



# Embedding Layer

- Learned **jointly** with a neural network model on a specific NLP task (**document classification**)
- It requires that the text is **cleaned** and prepared such that **each word is one-hot encoded**.
- The **size of the vector space** is specified as part of the model, such as 50, 100, or 300 dimensions. The vectors are **initialized** with small **random numbers**.
- The **embedding layer** is used on the front end of a neural network and is fit in a supervised way using the Backpropagation algorithm.

# Word2Vec

- Word2vec is a group of models that are used to **produce word embeddings**.
- These models are shallow, **two-layer neural networks** that are trained to reconstruct linguistic contexts of words.
- Input is a **large corpus of text** and output is a vector (**several hundred dimensions**)
- Words that **share common contexts** in the corpus are located **close** to one another
- Created and published in 2013 by a team of researchers led by Tomas Mikolov at Google

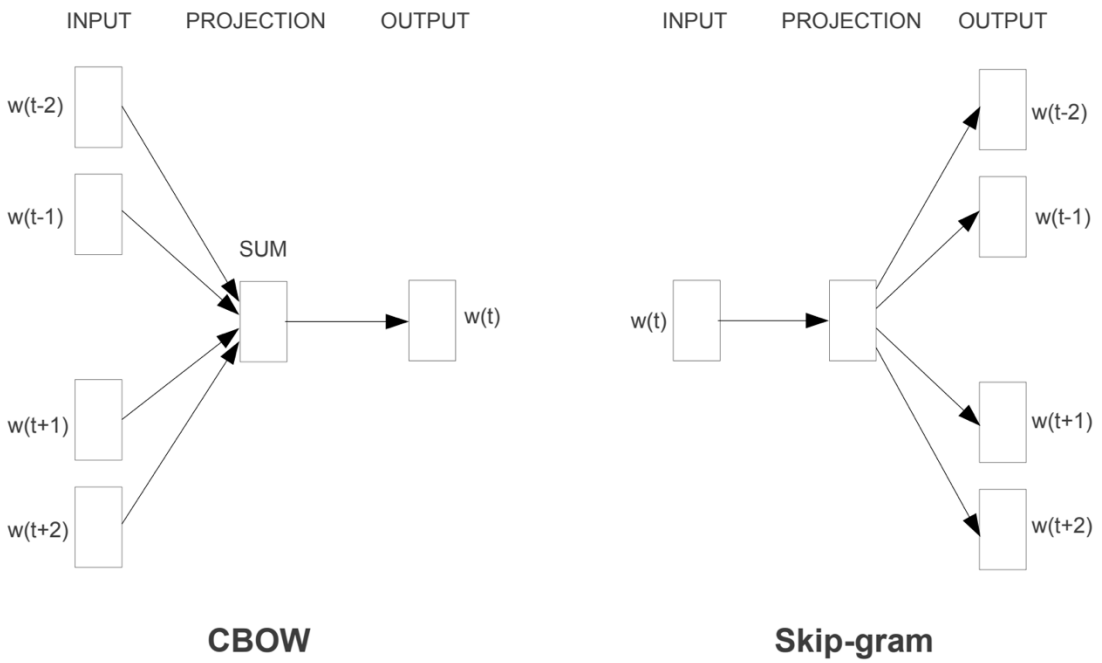
# Word2Vec - Architectures

## Two model architectures

- continuous bag-of-words (**CBOW**)
- continuous skip-gram. – **Skip-Gram**

Both models are focused on learning about words given their local usage **context**.

Where the **context** is defined by a **window of neighboring words**. This window is a configurable parameter of the model.

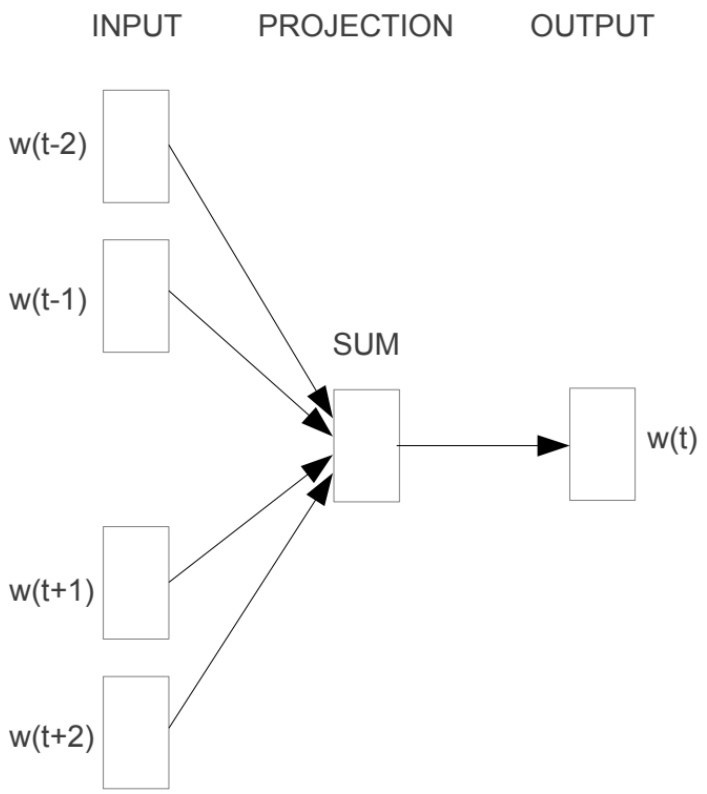



https://machinelearningmastery.com/what-are-word-embeddings/

# Word2Vec - **CBOW**

**CBOW**

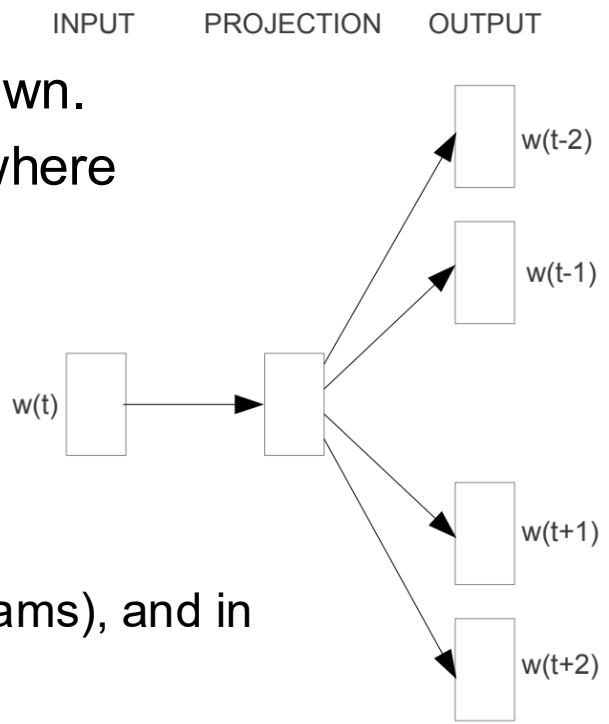
- The model predicts the current word from a window of surrounding context words.
- The order of context words does not influence prediction (bag-of-words assumption).




**CBOW**

# Word2Vec - **Skip-gram**

- The model uses the current word to predict the surrounding window of context words.
- Generalization of n-grams - may leave gaps that are skipped over.
  - **N-gram** is a consecutive subsequence of length n of some sequence of tokens w1 … wn.
  - **k-skip-n-gram** is a length-n subsequence where the words occur at distance at most k from each other.




For example, in the input text:

**the rain in Spain falls mainly on the plain**

the set of 1-skip-2-grams includes all the bigrams (2-grams), and in addition the subsequences

**the in, rain Spain, in falls, Spain mainly, falls on, mainly the, and on plain.**

# Word2Vec - **Skip-gram**

- CBOW is faster while skip-gram is slower but does a better job for infrequent words.

# GloVe

- Global Vectors for Word Representation (**GloVe**) –an **extension** to the word2vec method, developed by Pennington, et al. at **Stanford**.
- Rather than using a window to define local context, GloVe constructs an **explicit word-context** or **word co-occurrence matrix** using statistics across the whole text corpus.
- The result is a learning model that may result in **generally better word embeddings**.

https://nlp.stanford.edu/pubs/glove.pdf

# FastText

- **Facebook's** AI Research lab created **FastText**, which improves upon the Word2Vec model by **viewing words as assemblies of smaller character strings,** or character n-grams.
- This method enables the model to more effectively capture the intricacies of languages that have **complex word structures** and to **incorporate words not present in the original training data.**
- Consequently, FastText yields a more **adaptable** and comprehensive language model useful for a diverse set of machine-learning tasks.

# BERT

- **BERT** (Bidirectional Encoder Representations from Transformers) - Google AI.
- State-of-the-art results in a **wide variety of NLP tasks**, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.
- BERT's key technical innovation is applying the **bidirectional training of Transformer**, a popular attention model, to language modelling.
- This is in contrast to previous efforts which looked at a text sequence either from **left to right** or **combined** training.
- Have a deeper sense of language context and flow than single-direction language models.
- A novel technique named **Masked LM** (MLM) which allows **bidirectional training** in models in which it was previously impossible.

https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

# GPT - Generative Pre-trained Transformer

- Word embeddings are a **foundational component** in GPT (GPT-2, GPT-3, GPT-4). However, the architecture and approach are more advanced
  - In Word2Vec or GloVe, **each word is converted into a fixed vector** in a pre-defined space, which are then used as input to ML for tasks like classification, clustering, or seq-2-seq models for machine translation.
  - GPT models use a variant known as **"transformer embeddings",** which **not only embeds individual words but also considers the context**.
- GPT takes a **sequence of words (tokens) as input** and processes them through **multiple layers of transformer blocks**, which output a new sequence of vectors that represent not just the individual words, but also **their relationships** with all other words in the input sequence. Then used for NLP tasks
- GPT models use embeddings, they are far **more dynamic and context-aware** than traditional word embeddings. The embeddings in GPT models are part of a larger, more complex system designed to understand and generate human-like text based on the input it receives.

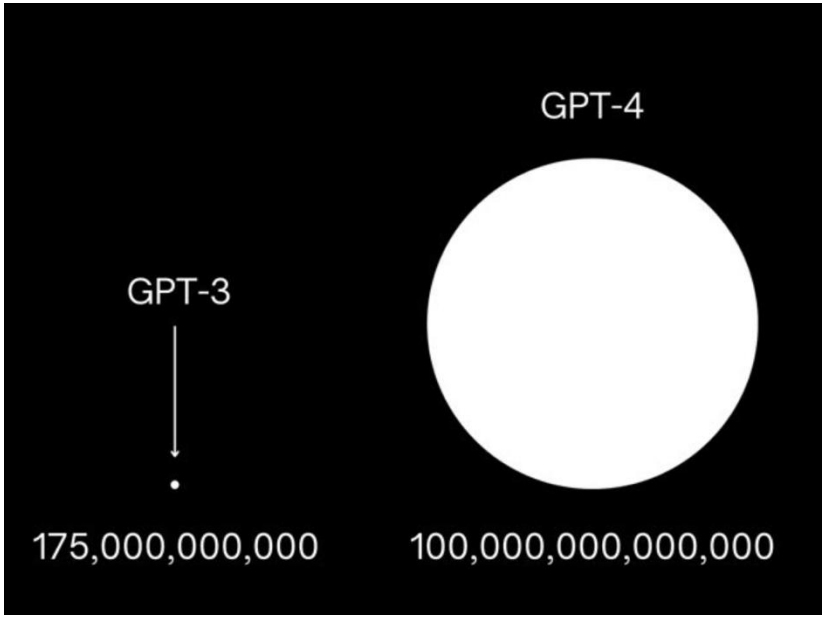
# GPT3

Generative Pre-trained Transformer 3 (GPT-3) - introduced in May 2020

- an autoregressive language model that uses deep learning to produce human-like text.
- It is the third-generation language prediction model in the GPT-n series (and the successor to GPT-2) created by OpenAI
- GPT-3's full version has a capacity of 175 billion machine learning parameters.
- Part of NLP Systems of pre-trained language representations
- GPT3 uses word embeddings as input (the input text is first transformed using word embeddings, then fed to the GPT3 architecture)

# GPT4

- In 2023
- 8 models with 220 billion parameters each, for a total of about 1.76 trillion parameters, connected by a Mixture of Experts (MoE).




- ChatGPT
  - A **chatbot** that uses the GPT language model as its backbone. ChatGPT is designed to have natural and engaging conversations with users

| | BERT | GPT |
|---|---|---|
| Task suitability | More suited to tasks like **sentiment analysis**, **question answering**, and **text classification**, | More suited in **text generation** especially natural-sounding text and completion challenges, , **summarization**, and **translation** |
| Training objective | Trained to predict **masked words** based **on the context** provided by other words. | Trained to predict the **next word** in a sentence given **previous words** |
| Context direction | **Bidirectional** – scans a sentence from left to right and right to left while making predictions - **deeper understanding** of context meaning | Unidirectional approach focused on **left-to-right processing** |
| Model architecture | A **single encoder** model that are pre-trained **jointly**, one for the masked language modeling task(MLM), and the other for the next sentence prediction(NSP) task. | Consists of a **stack of transformer blocks**, where each block has multiple **self-attention layers** and feedforward layers |

https://medium.com/@prudhvithtavva/bert-vs-gpt-a-tale-of-two-transformers-that-revolutionized-nlp-11fff8e61984#

# Word Embeddings vs TF-IDF

| Word Embedding | TF-IDF matrix |
| --- | --- |
| Multi dimensional vector which attempts to capture a words relationship to other words | Sparse matrix where each word maps to just a single value, captures no meaning |
| Often trained on large external corpus | Trained without external data |
| Must be applied to each word individually | Can be applied to each training document at once |
| More memory intensive | Less memory intensive |
| Ideal for problems involving a single word such as a word translation | Ideal for problems with many words and larger document files |

Word embedding caries much more information then a tf-idf column but comes at the cost of being more memory intensive and more difficult to apply.

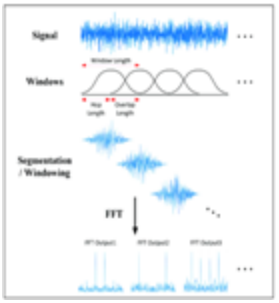
# Text to Speech

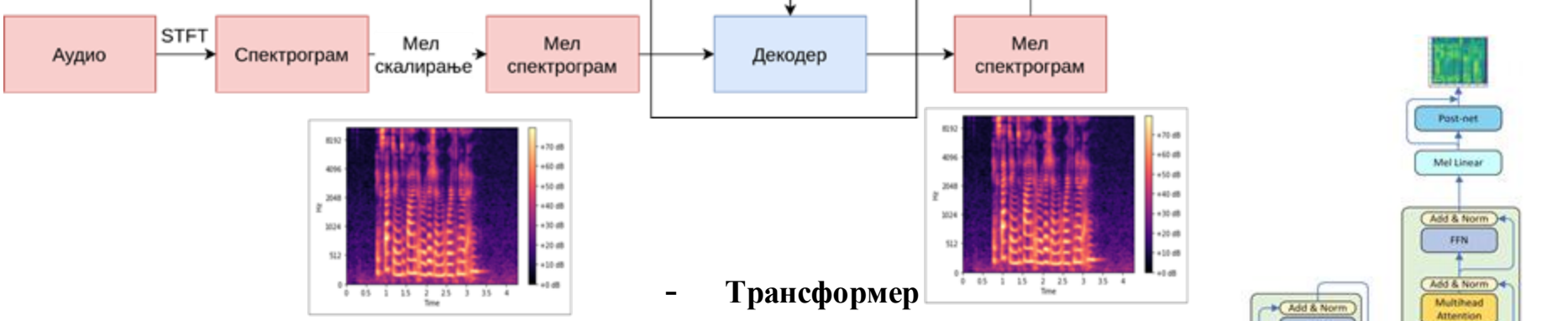
Душан Мерџановски - дипломска

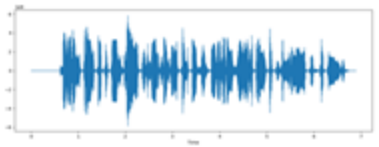
Здраво

Везилка

Ви благодарам







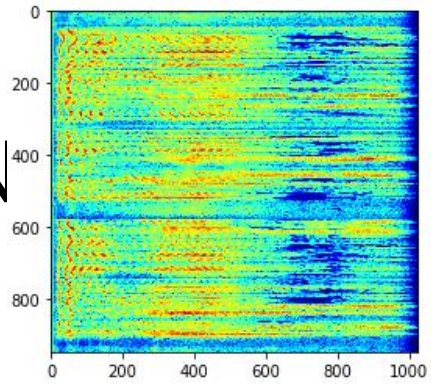


Инспирација е трудот „Neural Speech Synthesis with Transformer Network“

- **Трансформер**
  - Пред-процесирање на текст
  - Енкодер Предмрежа
  - Декодер Предмрежа
  - Слој за скалирано позиционо кодирање
  - Енкодер
  - Декодер
- **Постмрежа**

# Text to Speech for Macedonian

- Deep learning, CNNs and RN
- Proof of concept




*„Зеленооката девојка се насмевна, сепак се реши да ја прифати неговата понуда.“*

*„Сто пати ти кажав, дека на луѓе како него не треба да се верува.“*

*„Во шумата шумолеа зимзелените дрвја.“*

**Везилка**

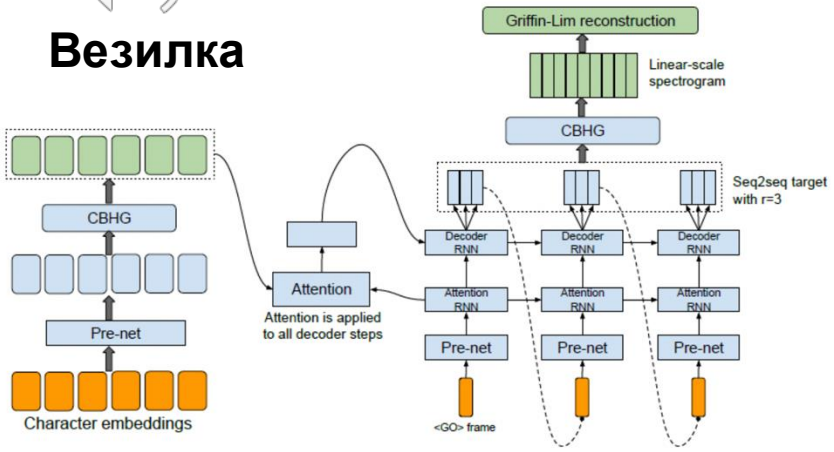


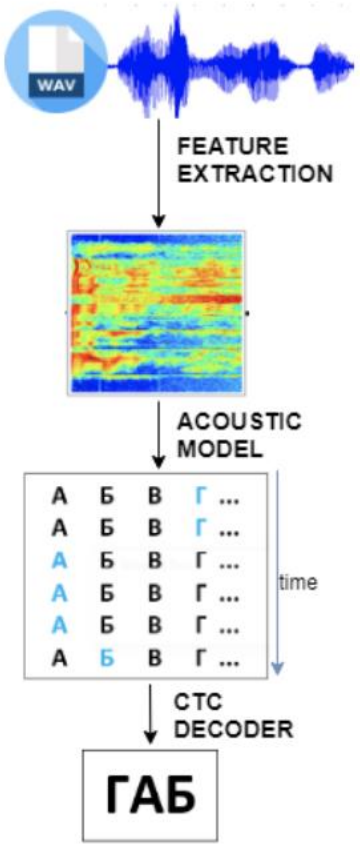

***Figure 1**: **Model architecture**. The model takes characters as input and outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesize speech.*

The CBHG (1-D convolution bank + highway network + bidirectional GRU) module adapted from Lee et al. (2016).

**Google. - Tacotron**

# MK Speech recognition

- Deep learning, RNNs




Connectionist Temporal Classification (CTC) objective function

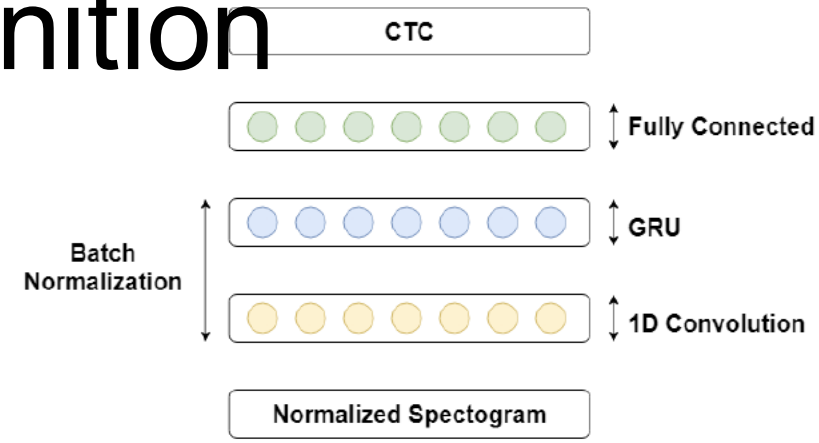



**Figure 3.** Network architecture – combination of convolutional layer and Gated Recurrent Units (GRUs), along with the CTC objective function

| True transcription | Predicted transcription |
|---|---|
| свири на виолина ми рече една вечер | сирина виоиина ли реч ердавечер |
| беше студено | еше студерно |
| зошто сакаш да ме излажеш | зошто сакашда ми зоажеш |
| ајде кажи ми | ајґле кажим |
| лекар сум | лека сум |
| збогум | зб оогум |
| и не знаев дека губам | и незноевекагу бан |
| зошто не си никола пилот ме праша | ошто не синикола пилот не праша |

**Table 2**. Sample experimental results - utterances with their true and predicted transcriptions

Marija Trajanoska, Hristijan Gjoreski. Towards end-to-end Speech Recognition in Macedonian Language. BALCANCOM 2019.

- https://cs230.stanford.edu/files/C5M2.pdf