Introduction to Natural Language Processing

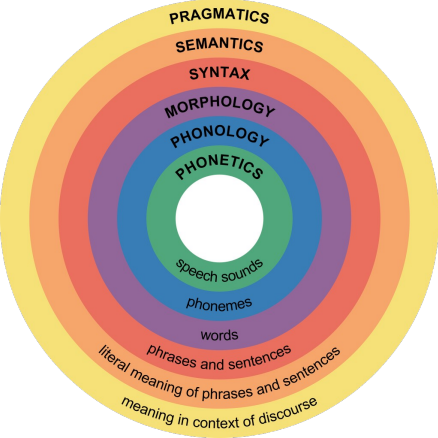
Text preprocessing & word representations

Ana Sabina Uban

auban@fmi.unibuc.ro

https://nlp.unibuc.ro/master\_en.html

+ some slides credits: Dan Jurafsky, James Martin, Chris Manning, Reda Bouadjenek, Pandu Nayak and Prabhakar Raghavan, Liviu Dinu

**Levels of linguistic analysis**

**Levels of linguistic analysis**

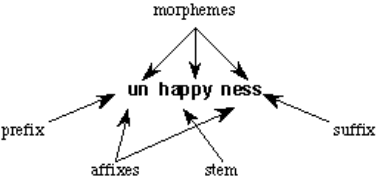
**Phonetic** or phonological level: deals with

pronunciation



**Levels of linguistic analysis**

**Morphological** level: deals with the smallest parts of words that carry meaning, and suffixes and prefixes.



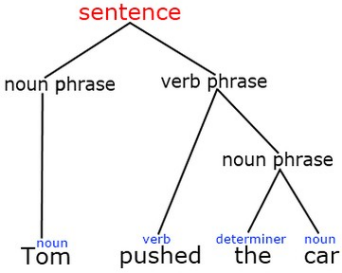
**Levels of linguistic analysis**

**Lexical** level: deals with words.



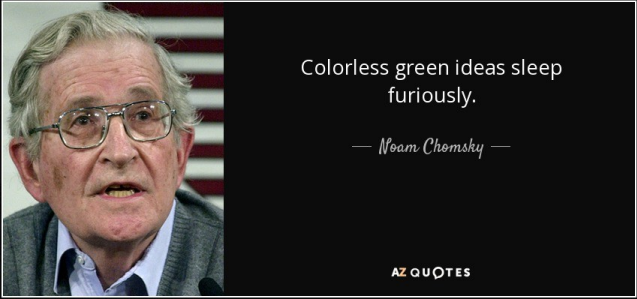
**Levels of linguistic analysis**

**Syntactic** level: deals with grammar and structure of sentences.



**Levels of linguistic analysis**

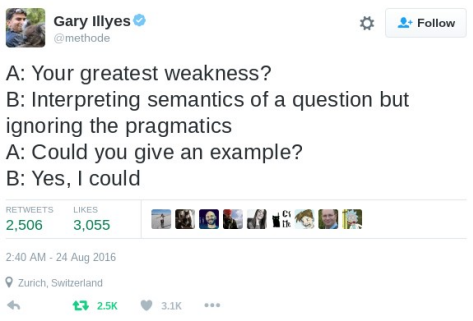
**Semantic** level: deals with the meaning of words and sentences.



**Levels of linguistic analysis**

**Pragmatic** level: deals with the knowledge that comes from the outside world, i.e., from outside the content of

the document.



**Natural Language Processing**

***Natural language processing*** *= the application of computational techniques to the analysis and synthesis of natural language and speech*

● Large variety of different tasks, useful as standalone applications or as steps in a more complex NLP pipeline

● Solutions can be algorithm/rule-based, can be trained classifiers (machine learning),

unsupervised information extraction etc

**Examples of NLP tasks**

● Syntactic parsing (input: text, output: syntactic structure)

● Sequence labelling: POS (part-of-speech) tagging, NER (named entity recognition) (input: text, output: labels for each token)

● Text classification (sentiment analysis, fake news detection, offensive language detection etc) ● Information retrieval (input: query, output: ranked documents)

● Topic modelling (input: text, output: detected topics)

● Language modelling & text generation: summarization, ChatGPT, machine translation (input: text?, output: text)

What do words mean?

**Lexical semantics**: the study of word meaning

**First**: What is a word, and how do we extract and represent it numerically?

**Processing text data**

Preprocessing &

Converting texts to numbers

12

**Main steps**

⚫Machine learning models need numerical inputs, so preprocessing is basically the process of taking a raw chunk of text and converting it into numbers.

⚫Therefore, for most applications, preprocessing can roughly be divided into the following 3 steps:

⚫**Step 1: Normalization (Cleaning)**

⚫**Step 2: Segmentation (Tokenization) => words!** ⚫**Step 3: Numericalization (Vocabulary mapping)** 13

**Definitions**

⚫**Normalization**: is where we clean the data in advance to remove unwanted inputs and to convert certain

characters/sequences into canonical forms.

⚫**Tokenization** is breaking a text chunk in smaller parts. Whether it is breaking paragraphs into sentences, sentences into words or words into characters.

⚫**Numericalization**: is where we convert the textual entities into numbers/ids so that we can feed them to our model. 14

**Main difficulty:**

● Language dependent!

● Domain dependent

● Application dependent

15

**Normalization (Cleaning)**

⚫ Refers to the process of cleaning up the input and mapping characters/words to a "canonical" form. ⚫**Standard steps:**

⚫Handling repeating characters (e.g. "gooooood" -> "good") ⚫Handling homoglyphs (e.g. "$tupid" -> "stupid")

⚫Mapping special inputs such as URLs, email addresses, and HTML tags to a canonical form (e.g. "http://www.foo.com/bar" - > "[URL]")

⚫Unicode normalization

**Normalization (Cleaning)**

⚫ Need to “normalize” terms

⚫ Information Retrieval: indexed text & query terms must have same form.

■ We want to match ***U.S.A.*** and ***USA***

⚫We implicitly define equivalence classes of terms ⚫e.g., deleting periods in a term

⚫e.g.: mapping all characters to their lowercase form

(Ciao=ciao).

17

**Normalization (Cleaning)**

⚫Applications like IR: reduce all letters to lower case

⚫Since users tend to use lower case

⚫Possible exception: upper case in mid-sentence? ⚫ e.g., ***General Motors***

⚫ ***Fed*** vs. ***fed***

⚫For sentiment analysis, machine translation, Information extraction

⚫Case is helpful (***US*** versus ***us*** is important)

**Segmentation /**

**Tokenization**

⚫Segmentation/Tokenization is probably the most complex part of the preprocessing pipeline.

⚫Naive tokenization algorithms,

⚫Rule-based tokenizers,

⚫modern (subword) tokenizers that are learned n t. 19

**Tokenizing on**

**Whitespace/Punctuation**

⚫The most naive form of tokenization.

⚫"I saw a girl with a telescope."

⚫This would be split into

⚫"I", "saw", "a", "girl", "with", "a", "telescope.“

⚫**Never** use purely whitespace tokenization in NLP!

⚫A slightly better approach is to tokenize based on punctuation like

⚫"I", "saw", "a", "girl", "with", "a", "telescope", "."20

**~~Sen~~tence**

**Segmentation**

⚫**What is a sentence?**

⚫The first answer to what is a sentence is something ending with a “.” or “!”

⚫!, ? are relatively unambiguous

⚫Period “.” is quite ambiguous

⚫ Sentence boundary

⚫ Abbreviations like Inc. or Dr.

⚫ Numbers like .02% or 4.3

⚫Build a binary classifier

⚫ Looks at a “.”

⚫ Decides EndOfSentence/NotEndOfSentence

⚫ Classifiers: hand-written rules, regular expressions, or machine-learning

**Lemmatization,**

**morphology?**

⚫Reduce inflections or variant forms to base form (a type of normalization)

⚫*am, are, is* → *be*

⚫*car, cars, car's*, *cars'* → *car*

⚫*the boy's cars are different colors* → *the boy car be different color*

⚫Lemmatization: have to find correct dictionary headword form

⚫Machine translation

**Why?**

⚫Many NLP tasks can benefit from lemmatization.

⚫Examples:

⚫ Topic modelling looks at word distribution in a document. ⚫By normalizing words to a common form, we can get better results in ML models (In word embeddings removing inflected wordforms can improve downstream NLP tasks.)

⚫For information retrieval (IR), lemmatization helps with query expansion so that suitable matches are returned even if there's not an exact word match.

⚫In document clustering, it's useful to reduce the number of tokens. It also helps in machine translation.

⚫The decision to use lemmas is application dependent; we should use lemmas only if they show better performance.

**Lemmatization.**

**How?**

⚫Use predefined dictionaries:

○ https://cst.dk/download/cstlemma/

○ https://www.clarin.si/repository/xmlui/handle/11356 /1041

○ WordNet (accessed directly through python libraries: NLTK, Spacy - see lab)

26

**Stemming**

⚫Reduce terms to their stems in information retrieval

⚫*Stemming* is crude chopping of affixes ⚫language dependent

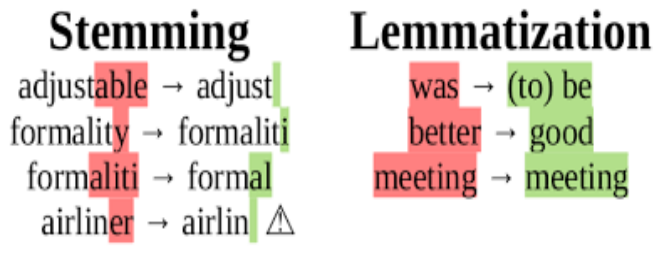
⚫e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

*for example compressed and compression are both*

*accepted as equivalent to*

*compress*.

*for exampl compress and compress ar both accept as equival to compress*

**Lemmatization vs stemming** 

28

**Porter’s algorithm. The most common English stemmer**

Step 1a

Step 2 (for long stems)

sses → ss caresses → caress ies → i ponies → poni ss → ss caress → caress s → ø cats → cat

Step 1b

(\*v\*)ing → ø walking → walk

ational→ ate relational→ relate izer→ ize digitizer → digitize

ator→ ate operator → operate

…

sing → sing

Step 3 (for longer stems)

(\*v\*)ed → ø plastered → plaster …

al → ø revival → reviv able → ø adjustable → adjust ate → ø activate → activ …

**Romanian Snowball**

https://snowballstem.org/algorithms/romanian/stemmer.html

⚫ ocol

ocolea

ocolesc

ocoleşte

ocoleşti

ocoli

ocolim

ocolind

ocolire

ocolişuri

ocolit

ocolită

ocoliţi

ocolul

ocoluri

ocolurile

⚫ ocol

ocol

ocol

ocol

ocol

ocol

ocol

ocol

ocol

ocolişur

ocol

ocol

ocol

ocol

ocolur

ocolur

30

**Choosing Normalization Steps**

⚫What kinds of normalization should we actually apply?

⚫No clear answers to this, but:

⚫Always look at the data manually!

⚫Are you using a pretrained model (e.g. BERT)? ⚫If so, make sure your preprocessing steps match the preprocessing that is conducted for pretraining. This can be more difficult than it seems, since papers don't always report all their preprocessing steps.

In the end, depends on the data, the application and the 31

**How many words?**

*they lay back on the San Francisco grass and looked at the stars and their*

⚫**Type**: an element of the vocabulary.

⚫**Token**: an instance of that type in running text. ⚫How many?

⚫15 tokens (or 14)

⚫13 types (or 12) (or 11?)

⚫ In ML applications: we usually build a vocabulary of types after normalization, and keep the most frequent ones (discard very rare terms)

Church and Gale (1990): |V|

**How many words?** > O(N½)

| ***N*** = number of tokensSwitchboard ***V*** = vocabulary = set of typesphone  |*V*| is the size of the vocabularyconversations | **Tokens = N** 2.4 million | **Types = |V|** 20 thousand |
| --- | --- | --- |
| Shakespeare | 884,000 | 31 thousand |
| Google N  grams | 1 trillion | 13 million |

**Practical examples using Python**

Code along! 

https://colab.research.google.c

om/drive/1UosXW-s-yc-NBbaD

5cqE-uq4jvvlFdtN?usp=sharing

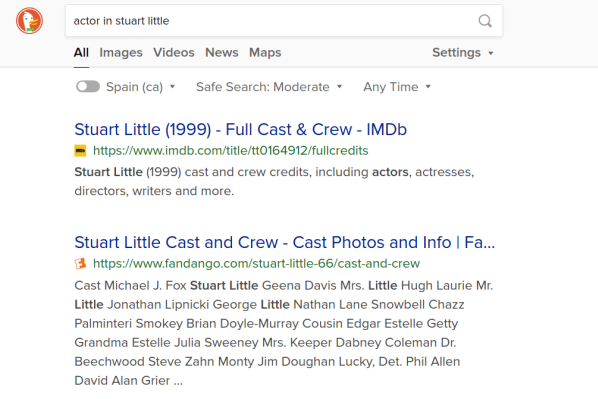
Pre-processing for NLP

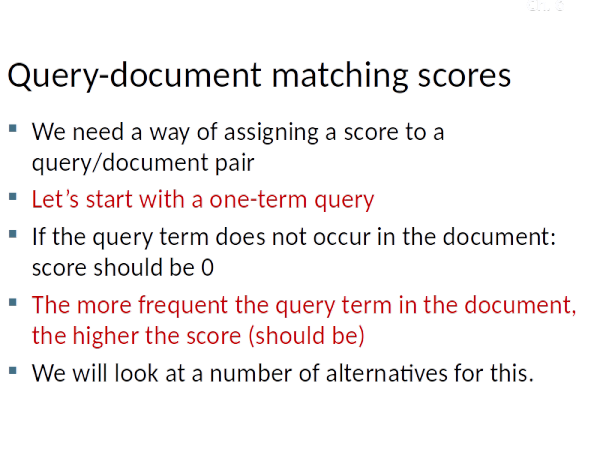
***Normalization***: is where we clean the data in advance to remove unwanted inputs and to convert certain characters/sequences into canonical forms.

***Tokenization*** *is breaking a text chunk in smaller parts. Whether it is breaking Paragraph in sentences, sentence into words or word in characters.*

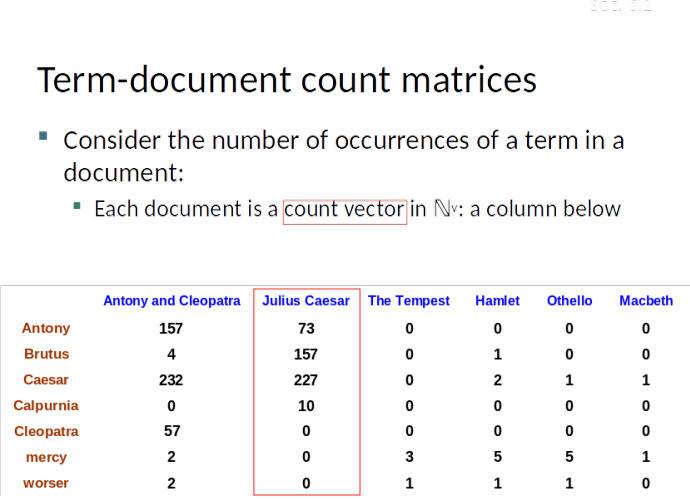
***Numericalization****:* is where we convert the textual entities into numbers/ids so that we can feed them to our model. 35

**Information Retrieval: finding relevant documents wrt a query**

****

****

****

****

****

****Source: https://dudeperf3ct.github.io/lstm/gru/nlp/2019/01/28/Force-of-LSTM-and-GRU/











**Query**: the movie that I liked the most

**Document1**: The cat is a domestic species of small carnivorous mammal. It is the only domesticated species in the family Felidae and is often referred to as the ...

**Document 2**: Stuart Little: best movie ever!



**Term Frequency Inverse Document Frequency** was introduced in a 1972 paper by Karen Spärck Jones — “*A statistical interpretation of term specificity and its application in retrieval*” �� (Cambridge, UK)



What do words mean?

**Lexical semantics**:

the study of word meaning

How do we represent word meaning to use them as:

- input to machine learning models - applications to other linguistics problems

Words, Lemmas, Senses, Definitions**sen**

**lemma definiti**

**se**

**on**

****

****

One word can have many senses

The same meaning can be expressed through different words

There are relations between senses

Relation: Synonymy

Synonyms have the same meaning in some or all contexts.

◦couch / sofa

◦big / large

◦automobile / car

◦vomit / throw up

◦Water / H20

Relation: Synonymy? 

Water/H20

Big/large

Brave/

courageous

Relation: Antonymy

Senses that are opposites with respect to one feature of meaning

Otherwise, they are very similar!

dark/light short/long fast/slow rise/fall hot/cold up/down in/out More formally: antonyms can

◦define a binary opposition

or be at opposite ends of a scale

◦ long/short, fast/slow

◦Be *reversives*:

◦ rise/fall, up/down

Similarity

Words with similar meanings. Not synonyms, but sharing some element of meaning

car, bicycle

cow, horse

Ask humans how

similar 2 words are

| **word1** | **word2** | **similarity** |
| --- | --- | --- |
| vanish | disappear | 9.8 |
| behave | obey | 7.3 |
| belief | impression | 5.95 |
| muscle | bone | 3.65 |
| modest | flexible | 0.98 |
| hole | agreement | 0.3 |

SimLex-999 dataset (Hill et

al., 2015)

More relations

Hypernymy / hyponymy (*tree* / *oak*) Meronymy / holonymy (*tree* / *branch*)

etc

**WordNet** : semantic network, curated, free

http://wordnetweb.princeton.edu/perl/webwn

Open Multilingual WordNet (OMW)

https://github.com/dumitrescustefan/RoWordNet







Sec. 9.2.2

From symbolic to distributed word representation

s

In machine learning vector space terms, this is a vectorone 1 and a lot of zeroes

[0 0 0 0 0 0 0 0 0 0 1 0 0 0

0] Deep learning people call this a “one-hot” representatiIt is a **localist** representation

Sec. 9.2.2

One-hot-encoding word

representation

sIts problems, e.g. for web search:

If • a usersearches for [Dell notebook battery size like], we to would match documents with “Dell laptop battery capacity”

But

size [0 0 0 0 0 0 0 0 0 1 0 0 0 0] T capacity [0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

Our queryand document vectors are **orthogonal** Thereis no natural notion of similaria set of one hot vectors

One-hot vectors, are ◦**long** (length |V|= 20,000 to 50,000)

◦**sparse** (most elements are zero)

Alternative: dense vectors

vectors which are

◦**short** (length 50-1000)

◦**dense** (most elements are non-zero)

“dense/continuous representations” “distributed representations“

“word embeddings”

84

Sparse versus dense vectors

Why dense vectors?

◦Short vectors may be easier to use as **features** in machine learning (fewer weights to tune)

◦Dense vectors may **generalize** better than storing explicit counts

◦They may do better at capturing synonymy: ◦*car* and *automobile* are synonyms; but are distinct dimensions

86

◦+ other semantic relationships (e.g. analogies) ◦

Capturing similarity

**Query**: *fast streams*

**Document**: *Dambovita is a very rapid river*

Orthogonal vectors: no common words (low similarity in tf-idf vector space) - but high semantic similarity!

Word similarity: cosine

distance



A solution via distributional similarity-based 

representations Idea: representing a word

by means of its neighbors /

context

“You shall know a word by the company i(J. R. Firth, 1957)

Philosophy: Ludwig 

Wittgenstein

“The meaning of a word is

defined by the way it is

”

running text as implicitly supervised training data!

• A word *s* near *apricot*

• Acts as gold ‘correct answer’ to the question

• “Is word *w* likely to show up near *apricot*?”

• No need for hand-labeled

supervision

• The idea comes from **neural language modeling**

• Bengio et al. (2003)