Machine Learning for Natural Language Processing

Guillaume Gravier

guillaume.gravier@irisa.fr

ENSAI École nationale de la statistique et de l'analyse de l'information

Who's who

Guillaume Gravier

UMR IRISA

CNRS researcher

statistical modeling for audio, speech and language processing; machine learning for NLP; (multi)media analytics

What about you?

Outline of the course

Lectures (6 x 3h)

- Lecture #1: Introduction and representation of words
  Notions: morphology, tokens, lemmas, POS, word net, word embedding
  Hands-on: manipulate basic pipelines and visualize word embeddings
- Lecture #2: Representation of documents
  Notions: vocabulary, Zipf's curse, bag of words, Bayes
  Hands-on: basic tf-idf k-nn classifier
- Lecture #3: Language models
  Notions: ngrams, LSTM, bi-LSTM, language generation
  Hands-on: train a small LM and generate text
- Lecture #4: Transformers and large language models
  Notions: encoder/decoder, transformers, fine-tuning
  Hands-on: visualize word embeddings, document embeddings, fine-tune a LLM

Evaluation

- short quizz at the end of the last session

Pillow books

Daniel Jurafsky, James H. Martin. Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition, 2nd edition, Prentice-Hall, 2009. Re-edited in 2023.

Yoav Goldberg. Neural Network Methods for Natural Language Processing, 2017.

Links to online version available on moodle.

Some definitions of 'natural language'

A few definitions from `dictionary.com`:

1. a body of words and the systems for their use common to a people who are of the same community or nation, the same geographical area, or the same cultural tradition
2. communication by voice in the distinctively human manner, using arbitrary sounds in conventional ways with conventional meanings
3. the system of linguistic signs or symbols considered in the abstract (opposed to speech)
4. any set or system of such symbols as used in a more or less uniform fashion by a number of people, who are thus enabled to communicate intelligibly with one another
5. any system of formalized symbols, signs, sounds, gestures, or the like used or conceived as a means of communicating thought, emotion, etc.: the language of mathematics; sign language. the means of communication used by animals

Some definitions of ’natural language processing’

1. NLP aims at creating computer programs able to automatically process natural languages [Bouillon 98]
   - NLP ... but also computational linguistics
   - rather text data than languages
2. All research and development aiming at modeling and reproducing, using machines, the human ability to produce and understand linguistic utterances for communication purposes [Yvon 07]
3. 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 en]

Modeling of language competence and reproduction of language performance... but not necessarily in the same way as humans do

The ELIZA example – Joseph Weizenbaum, MIT, 1966

Automatic psychological analysis simulated:

[he says] I am depressed all the time

`I am sorry to hear you are depressed`

It's true that I am unhappy

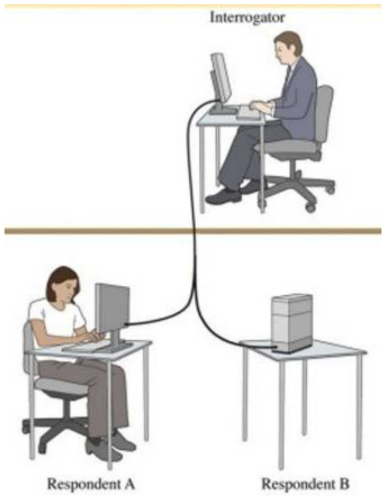
`Do you think coming here will help you not be unhappy`

I need help, that seems certain

`What would it mean if you got some help`

Perhaps I could learn to get along with my mother

`Tell me more about your family`



ELIZA passes the Turing test but in fact, no comprehension: merely keyword spotting and basic rules, e.g.,

```
s/.* all .*/IN WHAT WAY?/1
s/.*I am (depressed).*/I AM SORRY TO HEAR YOU ARE \1/1
s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/1
```

See demo at `http://www.manifestation.com/neurotoys/eliza.php3`

The Google captioning example – Oriol Vinyals et al., 2014

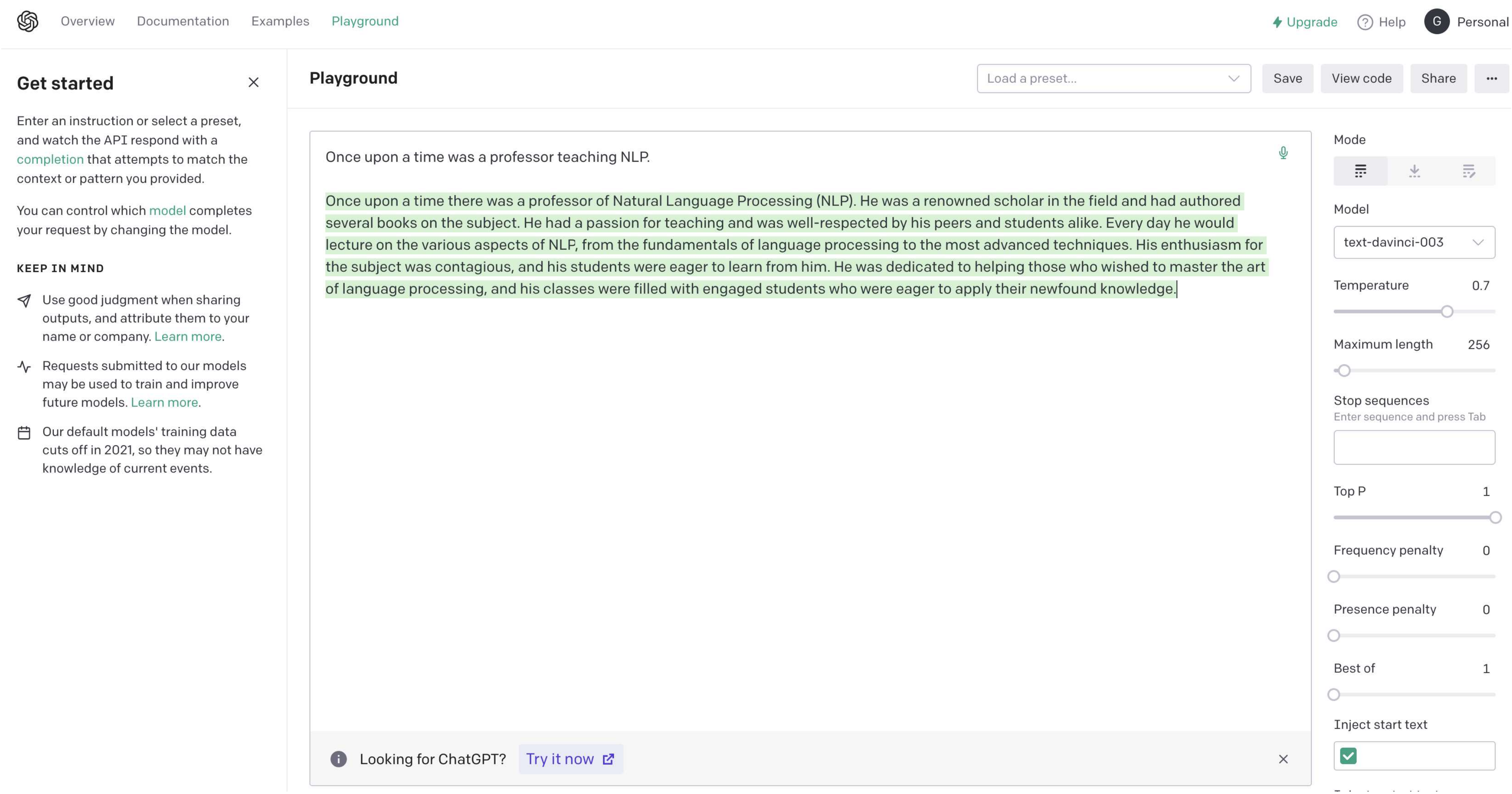
Automatic generation of captions from images ...

... with brute force machine learning behind the scene

ENSAI École nationale de la statistique et de l'analyse de l'information

The GPT-3 example – Tom Brown et al., 2020

Natural language generation from a prompt ...



here again with brute force machine learning behind the scene

Language is a complex multilevel process

Le président des antialcooliques mangeait une pomme avec un couteau.

(example borrowed from A. Vilnat)

Phonological: study of sounds in speech, organization of sounds

Graphical: segmentation of text into basic units (words?)

Lexical and morphological: identification of the lexical components (words?), of their properties

Syntactic: identification of higher level components and their organization (sentence level)

Semantic: constructing a representation of the meaning

Pragmatic: identifying the function of an utterance in its context

All levels are strongly intricated

- *J'ai lu un livre* $\longrightarrow$ gender determined by the determiner (syntactic)
- *Il ne s'agit pas de livres mais de lires* $\longrightarrow$ gender at the pragmatic level

Language is highly complex and ambiguous

- non structured data = no semantics brought by *a priori* structure
- graphical complexity and ambiguity
  - *etc*., SNCF, aujourd'hui, 1,23 %, 1939-45, Jean-Paul II, Washington, ...
  - park (v/n), rat (v/n/a) – in French, président (n/v), couvent (n/v)
- syntactic ambiguity
  - I saw the man with the binoculars. Look at that dog with one eye.
  - La belle ferme le voile. La petite brise la glace.
  - I saw a movie with Brad Pitt.
- semantic ambiguity
  - homonymy: park, bank, bright
  - polysemy: dish (plate *vs*. meal), glass (container *vs*. content)
- pragmatic ambiguity (co-references but not only)
  - A: Are you coming at Luke's party tonight?
  - B: I hear Ben will be there.

Also for different languages: tonal languages, aggutinative languages, etc.

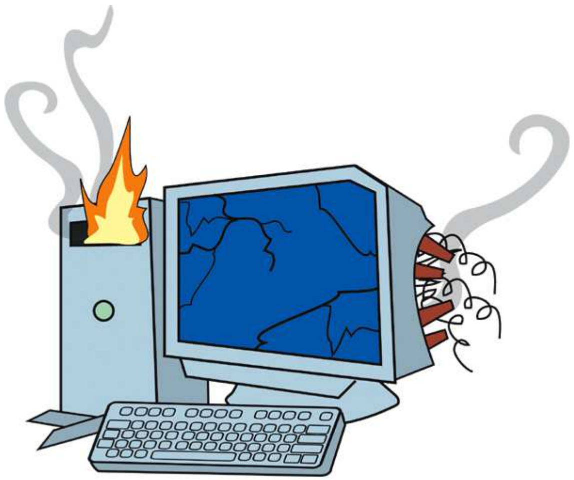
Language is highly complex and ambiguous (cont'd)

Lot of implicit knowledge and common sense in (human) language interpretation (that computers lack):

- "Obvious" interpretation (anaphora/coreference resolution)
  - ▷ She ordered a beer to the waitress. Then she left without paying.
  - ▷ The profressor sent the pupil to the principl becase
    - ◇ he was throwing pellets
    - ◇ he wanted peace
    - ◇ he wanted to see him

(example borrowed from F. Yvon)

- Metaphor, metonymy
  - ▷ you're a black sheep, you have her ear, the room applauded
- Common sense knowledge
  - ▷ I read an article on the accident in the newspaper.
  - ▷ I read an article on the accident in the metro.



Raw text is not what you want to play with

```
I got this as both a book and an audio file.  I had waited to read it and was
surprised by both the enthusiasm of the content and its author, but also by
how he snuck in some odd biblically unsound thoughts (e.g., I gasped when he
suggested Christ went to Hell...what Bible passage evidences this?).

I agree with how he suggests the enemy is out to deceive us and keep us
asleep...but wonder if I go further how much more Eldredge will slip in of
his own peculiar biblical misintrepretations.  Where were his editors when
this was being written?  Why take sensible good sections and mar them with
oddities?

I havent read all the reviews here but as one of those "conservatives"
frequently mentioned in them I have to admit I may not even finish this book
for fear of what else Eldredge has slipped up on.

I did appreciate his story about Daniel and the "delayed" angel...but am left
wondering if I need go deeper into researching that as possible
misintepretation  too.
```

[extract from CLS corpus: review polarity classification task]

Raw text is really not what you want to play with

@fa6ami86 so happy that salman won. btw the 14sec clip is truely a teaser

@phantompoptart ....oops.... I guess I'm kinda out of it.... Blonde moment -blushes- epic fail

@bradleyjp decidedly undecided. Depends on the situation. When I'm out with the people I'll be in Chicago with? Maybe.

Just grabbed some bagels from Panera for everyone at wk. It's Brittany's last day

AHH i'm so HAPPY. I just found my ipod. God is sooo good to me!

Hoping for a better day today. Yesterday was fine until I passed out at my desk, fell off chair, and carpet burned my forehead.

@poepiandzegiant oops just saw you said hello! Hi there

att workkk dyinggg to get the fxckkkkk outt

[extract from carblaca/twitter-sentiment-analysis]

Tokenization: a crucial step at the graphical level

Tokenization is about breaking a text into *sentences* (or sort of sentences) and sentences into *tokens*

- based on punctuation marks and spaces
- plus a huge lot of rules and exceptions
  - j'ai $\rightarrow$ j' + ai *vs.* aujourd'hui $\rightarrow$ aujourd'hui
  - 31/12/2019 $\rightarrow$ 31/12/2019 *vs.* 31 décembre 2019
  - what do we do with quotation marks???? parenthesis????
- might overlap with lexical and morphological anlalysis (but not necesssarily)
- painful and language-dependent but crucial step in NLP pipelines
- a few existing (free) tools available

Stanford	`https://nlp.stanford.edu/software/tokenizer.shtml`
NLTK	`http://www.nltk.org/api/nltk.tokenize.html`
spaCy	`https://spacy.io/api/tokenizer`

Tokenization at work with NLTK and spaCy

```
> from nltk.tokenize import word_tokenize, sent_tokenize
> s='A $2 example\nof a sentence. And a 2nd sentence.'
> word_tokenize(s)
['A', '$', '2', 'example', 'of', 'a', 'sentence', '.',
'And', 'a', '2nd', 'sentence', '.']
>  [word_tokenize(x) for x in sent_tokenize(s)]
[['A', '$', '2', 'example', 'of', 'a', 'sentence', '.'],
['And', 'a', '2nd', 'sentence', '.']]

> import spacy
> nlp = spacy.load('en_core_web_sm')
> doc = nlp(s)
> [token for token in doc]
[A, $, 2, example,
, of, a, sentence, ., And, a, 2nd, sentence, .]
```

Sub-word tokenization

Recent models makes use of sub-word tokens derived from large amount of data based on the frequency of occurrence of substrings, e.g.:

- byte pair encoding (BPE)
  $\rightarrow$ used in GPT-3
- WordPiece
  $\rightarrow$ used in BERT
- Unigram / SentencePiece

```
aaabdaaabac

AabdAabac
A=aa

ABdABac
A=aa B=ab
```

```
> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
> s = "Using a Transformer network is simple"
> tokens = tokenizer.tokenize(s)
> print(tokens)
['Using', 'a', 'transform', '##er', 'network', 'is', 'simple']
```

Morphology and lexicons

Lexicology = inventory and classification of the usages of “words”

- *lexical unit* $\simeq$ entry in a lexicon or dictionary
- existence of many *forms*: aimer, aime, aimé, aimions, aimât, *etc*.
- categories of lexical units: noun, verb, adjective, preposition, *etc*.

Morphology = process of the formation of words

- inflection: gender, plural, conjugation, *etc*. (no category change)
  - love $\longrightarrow$ loved, loves, love’.
- derivation: new lexeme derived from existing ones (category change)
  - happy $\longrightarrow$ happiness, happening
- composition: combination/concatenation of forms
  - can opener, neural network, pomme de terre

$\Rightarrow$ *morpheme* = the smallest meaningful morphological unit

- Example: antialcooliques = [anti] [[alcool] [ique]] [s]
- lexical/root morpheme = *lexeme*
- grammatical morphemes, in particular *affixes* such as prefixes, suffixes

word, lexeme, lexie, lemma, form and other fun things

The notion of *word* is fuzzy and ambiguous:

1. Surface: *His answer was only two-words long: certainly not*
2. Sign: *Am, are, is, was ... are some forms of the same word*

We should rather refer to the following concepts

token	a (normalized) graphical string, without meaning
(word)form	linguistic sign with a certain functioning autonomy and a certain internal cohesion
lexeme	lexie clustering wordforms distinguished by inflection (2) ◦ described by its form (signifier) and meaning (signified) ◦ comes with a part-of-speech (PoS) category: N, V, Adj
lemma	(arbitrary) canonical form used to represent a lexie (e.g., aimer)
stem	morphological support of a lexie, bearing the signified (e.g., aim-)

Lemmatization, stemming and POS tagging

- **Stemming**: wordform $\longrightarrow$ stem (usually morphological analysis)

```
> stemmer = nltk.stem.porter.PorterStemmer()
> stemmer.stem('candy')
'candi'
```

- **part-of-speech (POS) tagging**: wordform (token) $\longrightarrow$ POS tag

```
> nltk.tag.pos_tag(word_tokenize('The cat loves milk.'))
[('The', 'DT'), ('cat', 'NN'), ('loves', 'VBZ'), ...]
```

- **lemmatization**: wordform (token) $\longrightarrow$ lemma (requires POS tagging)

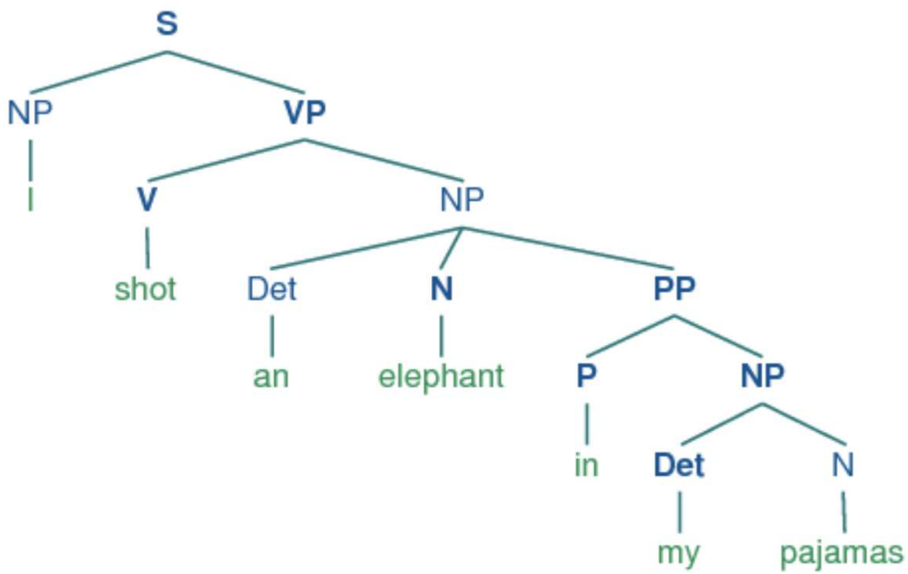
```
> from nltk.stem import WordNetLemmatizer
> lemmatizer = WordNetLemmatizer()
> lemmatizer.lemmatize('candies', pos='n')
> 'candy'
```

POS tagging and lemmatization at work

```
> import nltk
> tok = nltk.word_tokenize(s)
> tok
['The', 'cat', 'sleeps', 'on', 'the', 'mat', '.']
> tag = nltk.pos_tag(tok)
> tag
[('The', 'DT'), ('cat', 'NN'), ('sleeps', 'VBZ'), ('on', 'IN'),
('the', 'DT'), ('mat', 'NN'), ('.', '.')]

> nlp = spacy.load('fr_core_news_md')
> s='Les poules du couvent couvent.'
> doc = nlp(s)
> for token in doc:
...     print(token, token.pos_, token.lemma_)
Les DET le
poules NOUN poule
du DET de
couvent NOUN couvent
couvent ADV couvent
```

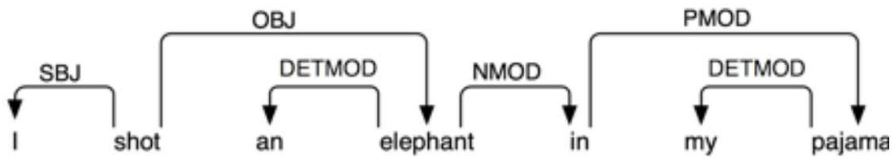
Representing the syntactic level

Parsing: unveil the sentence structure based on the the principles and constraints that govern the combination of words into grammatically correct sentences



constituency parse

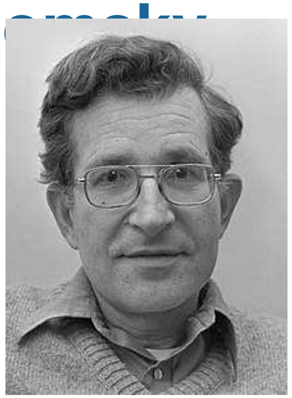
phrase structure grammar



dependency parse

dependency grammar

No syntax model without naming Noam Ch



Formal (generative) grammars are widely used as syntactic models to describe language grammars and perform parsing:

- terminal nodes V (= words)
- non terminal nodes N
- set of derivation rules of the form

$$(N \cup V)^* N (N \cup V)^* \rightarrow (N \cup V)^*$$

```
S -> NP VP
S -> VP NP
VP -> Verb
VP -> Verb NP
NP -> Noun
NP -> Det NP
...
Noun -> time
Noun -> arrow
Verb -> flies
...
```

Comes in different types and flavors, in particular

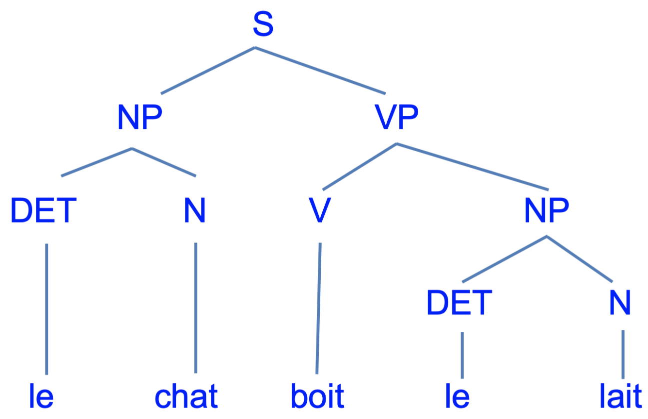
- *context-free grammars* (type 2)
  $\rightarrow$ *left rule limited to a node from* N
- *regular grammars* (type 3)
  $\rightarrow$ *left rule:* N
  $\rightarrow$ *right rule:* $\emptyset$, *V*, *V N*

which can both be efficiently implemented

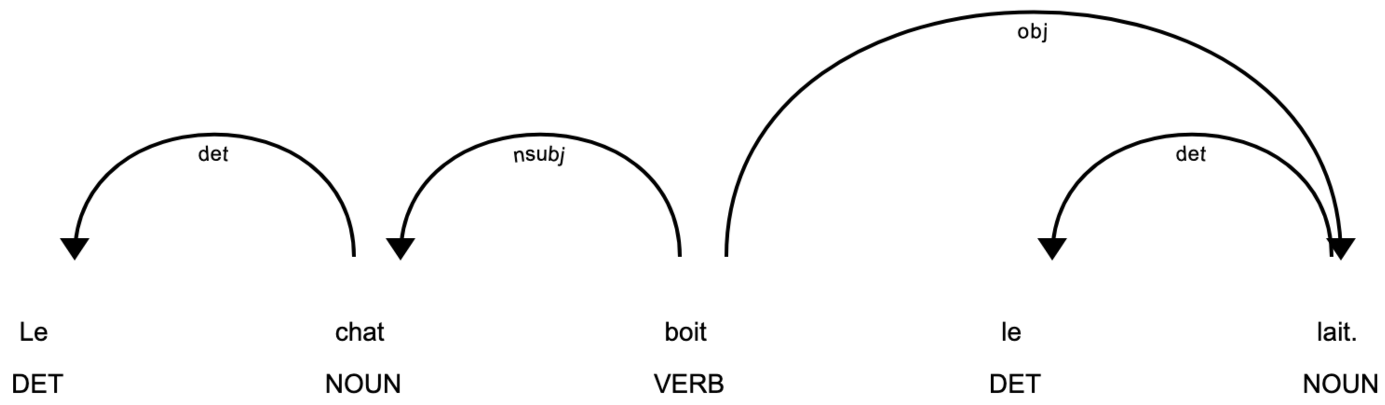
Grammars and parse trees

CFGs lead to constituency parse trees

S → NP VP
NP → Npr
NP → DET ADJ N
VP → V
VP → V NP
DET → la | le
N → chat | pomme | lait
Npr → Jean
V → mange | court | boit



from which one can infer dependency parse trees

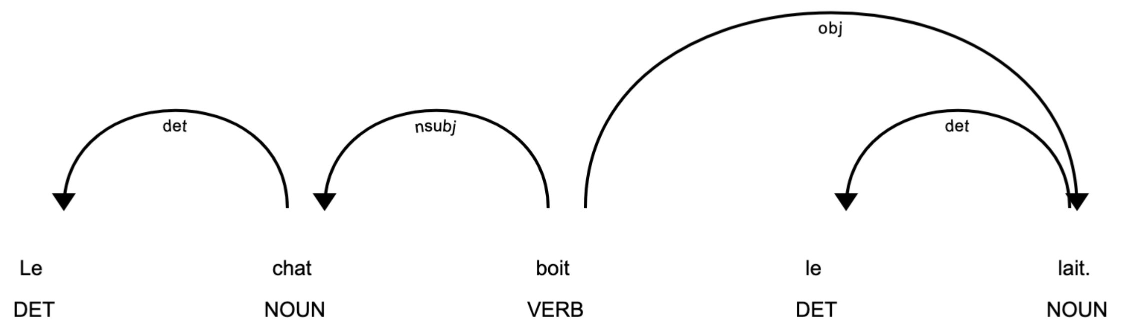


Note: this is not what is done in practice

Grammars and parse trees

Many tools for dependency parsing: MALT, Stanford CoreNLP, spaCy, etc.

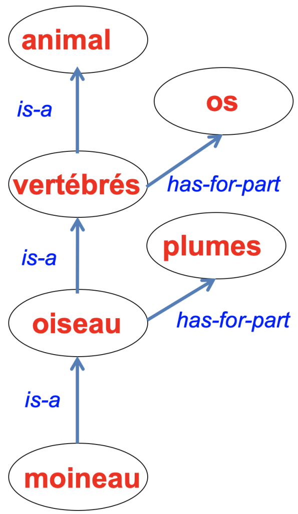
```
> import spacy
> process = spacy.load('fr_core_news_md')
> res = process('Le chat boit le lait.')
> for token in res:
>     print(token.text, token.pos_, token.lemma_, token.dep_)
Le       DET    le     det
chat     NOUN   chat   nsubj
boit     VERB   boire  ROOT
le       DET    le     det
lait     NOUN   lait   obj
.        PUNCT  .      punct
> spacy.displacy.render(xr, style="dep", jupyter=True)
```



About semantics

Representation of semantics in general goes (way) beyond NLP with no commonly adopted framework, among for instance

- precidate and description logics
- abstract meaning representation (AMR)
- semantic networks
- role semantics, frame semantics




```
    (w / want-01
:arg0 (b / boy)
:arg1 (g / go-01 :arg0 b))
```

$$\exists w, b, g : \text{instance}(w, \text{want\_01}) \wedge \text{instance}(g, \text{go\_01}) \\ \wedge \text{instance}(w, \text{boy}) \wedge \text{arg0}(w, b) \\ \wedge \text{arg1}(w, g) \wedge \text{arg0}(g, b)$$

however many ways and resources for lexical semantics (i.e., reprensenting meaning of words)

- coocurrence and compositionality
- word nets

ENSAI École nationale de la statistique et de l'analyse de l'information

Newpaper quizz

Read in the newspaper Le Monde, Nov. 16 2023, article entitled “How to know if a text has been used by an AI” by David Larousserie:

These capacities are obtained by a rather brute force training procedure that consists in guessing the next word in a sentence taken from a very large corpus of textes, reaching thousands of billions of “tokens” (or semantic sub-units, such as sullables, prefixes, suffixes, etc.).

Correct or not?

Words = tokens = strings of characters

A natural solution is to represent the wordform as the corresponding string of characters

- very easy to implement
- rather efficient comparison with hash lookup tables (fixed vocabulary)
- symbolic representations are pratical for statistical models and meaning

but poor semantic information in the wordform representation

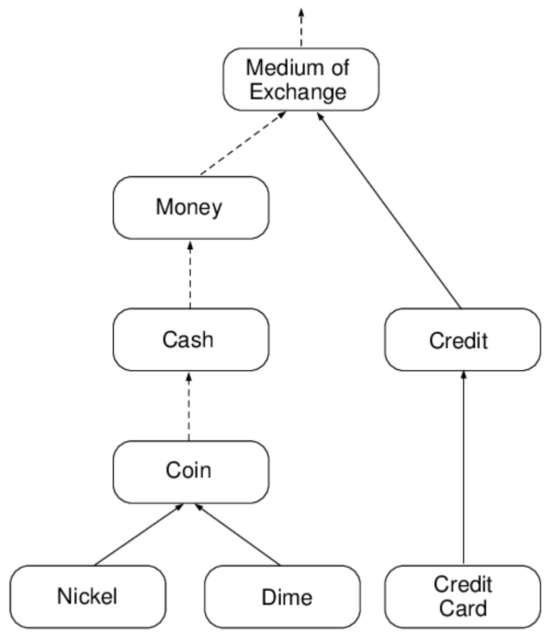
- hard to encode semantic relations with binary comparison
  - vélo $\neq$ bicyclette in the wordform domain
  - can maintain a matrix of semantic relations but costly
- hard to relate inflected forms
  - manger $\neq$ mangera $\neq$ mangerait in the wordform domain
  - can work with lemma but would loose semantic

plus not too convenient for neural networks!

Lexicons and semantic networks

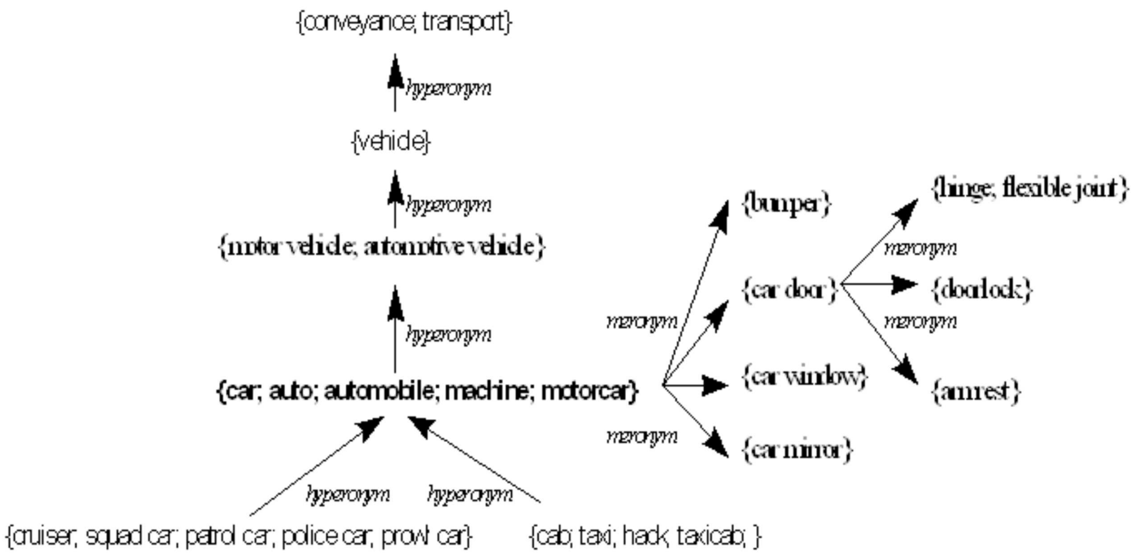
Existence of rich lexicons organized as frames or semantic network, e.g.,

- *FrameNet* – `https://framenet.icsi.berkeley.edu/fndrupal`
  - 13k word senses, 1.2k frames
  - mostly verbs with semantic roles
  - small French version available (ASFALDA)
- *WordNet* – `https://wordnet.princeton.edu`
  - 100k+ nouns, 20k+ adjectives, 10k+ verbs
  - grouped into sets of cognitive synonyms (synsets)
  - synsets interlinked with semantic and lexical relations
  - Wordnet Libre du Français (WOLF)

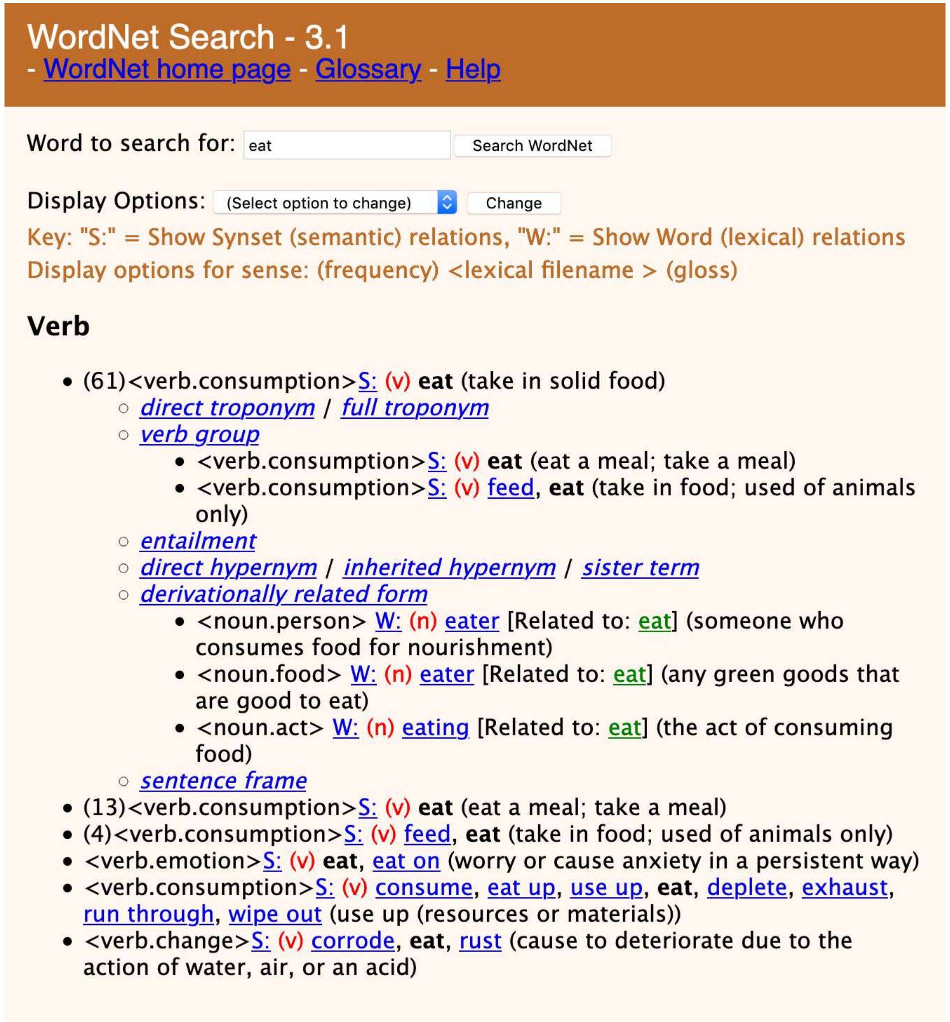


- *SentiWordNet* – `https://github.com/aesuli/sentiwordnet`
  - annotation of WordNet synsets with sentiment cues
  - positivity, negativity, objectivity

Lexicons and semantic networks: a quick zoom on WordNet



(a) WordNet structure



(a) WordNet browser

WordNet similarity

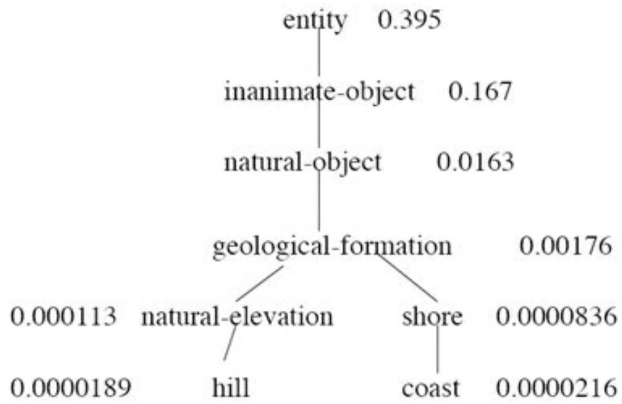
Exploit hierarchical structure of WordNet, with *least common subsumer* (LCS) of the two synsets to determine semantic proximity between two synsets.

$$\text{simpath}(c_1, c_2) = \frac{1}{1 + \text{shortest\_path\_length}(c_1, c_2)}$$

Generalize to words

$$\text{simword}(w_1, w_2) = \max_{c_1 \in S_1, c_2 \in S_2} \text{simpath}(c_1, c_2)$$

with $S_i = \text{senses}(w_i)$.



Many many many variants, e.g., with probabilities associated to the specificity of the concepts (P(entity) = 1) or ratio between LCS depth and synset depth

$$\text{Wu Palmer}(c_1, c_2) = 2\frac{\text{depth}(\text{lcs}(c_1, c_2))}{\text{depth}(c_1) + \text{depth}(c_2)}$$

Pedersen *et al.*, 2004. WordNet:: Similarity-Measuring the Relatedness of Concepts.

ENSAI École nationale de la statistique et de l'analyse de l'information

Playing with WordNet in practice

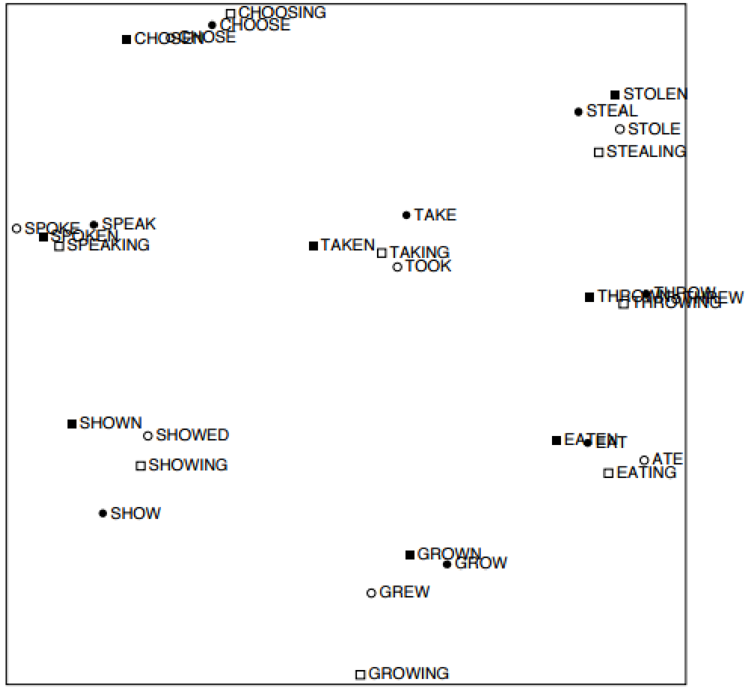
```
> from nltk.corpus import wordnet as wn

> wn.synsets('rat')
[Synset('rat.n.01'), Synset('scab.n.01'), Synset('rotter.n.01'),
Synset('informer.n.01'), Synset('rat.n.05'), Synset('rat.v.01'),
Synset('rat.v.02'), Synset('fink.v.01'), Synset('rat.v.04'),
Synset('rat.v.05'), Synset('denounce.v.04')]
> rat = wn.synset('rat.n.01')
> rat.hypernyms()
[Synset('rodent.n.01')]
> rat.hyponyms()
[Synset('bandicoot_rat.n.01'), Synset('black_rat.n.01'),
Synset('brown_rat.n.01'), Synset('jerboa_rat.n.01'),
Synset('pocket_rat.n.01'), Synset('rice_rat.n.01')]
> rat.lowest_common_hypernyms(wn.synset('man.n.01'))
[Synset('organism.n.01')]
> rat.wup_similarity(wn.synset('man.n.01')))
0.54
```

https://www.nltk.org/howto/wordnet.html

Distributed or embedded representations: What the heck?

The idea is to represent words (better said wordforms or tokens) in a Euclidean space with nice (semantic) properties

This is also known as *embedding*, which can be seen as the first layer of a neural network



wordform		*token*		*1-hot*	*layer 1*		*embedding*
aimer	→	aimer i = 101	→	0 0 0 … 0 1 0 … 0 0 0	W_{ij} (nxd)	→	$w_i \in \mathbb{R}^d$

- What properties for w_i?
- How to obtain w_i?
- How to evaluate w_i?

ENSAI de la statistique et de l'analyse de l'information

The seminal idea of distributional semantics

[...] the parts of a language do not occur arbitrarily to each other: each element occurs in certain positions relative to certain other elements. – Zellig S. Harris, Distributional structure, in Word, 10(23):146–162, 1954

You should know a word by the company it keeps – J. R. Firth, A synopsis of linguistic theory 1930-1955 in Studies, in Linguistic Analysis, 1–32, 1957

In practice, words are represented by the context (typically nouns and verbs, possibly lemmatized) they appear in, e.g.,

The dog barked in the park.

The owner of the dog put him on the leash since he barked.

...

	leash	walk	run	owner	leg	bark
dog	3	5	1	5	4	2
cat	0	3	3	1	5	0
lion	0	3	2	0	1	0
light	0	0	0	0	0	0
bark	1	0	0	2	1	0
car	0	0	4	3	0	0

A direct approach: the co-occurrence matrix

count co-occurrences of words within a context window

reduce dimension somehow (and avoid sparsity)

Corpus *I like deep learning. I like NLP. I enjoy flying.*

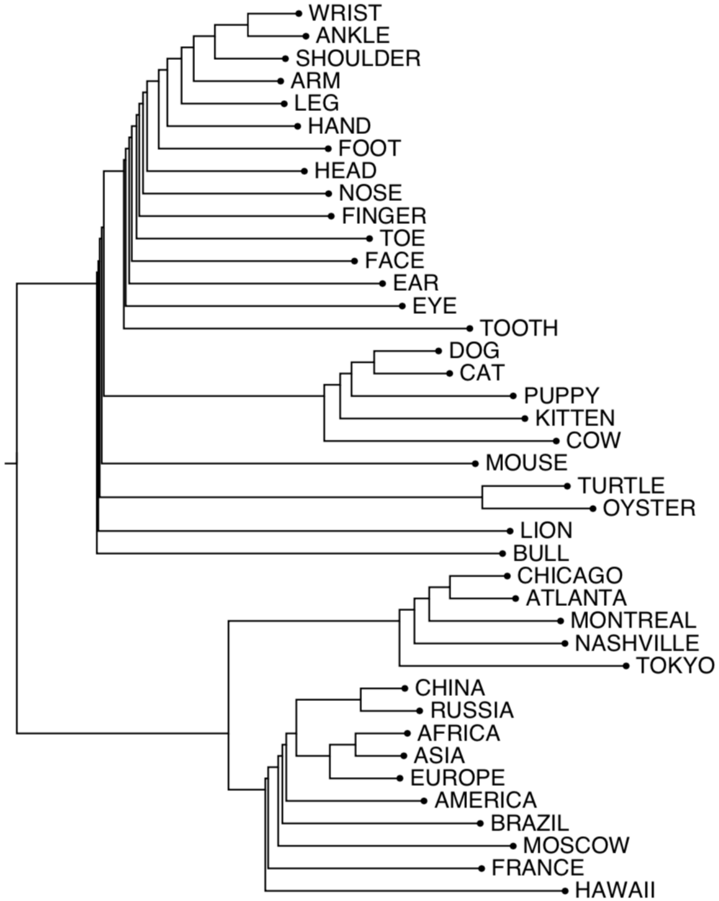
Window size $\pm$ 1 token

$$X = \begin{array}{cc} & \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{array}{c} I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} & \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

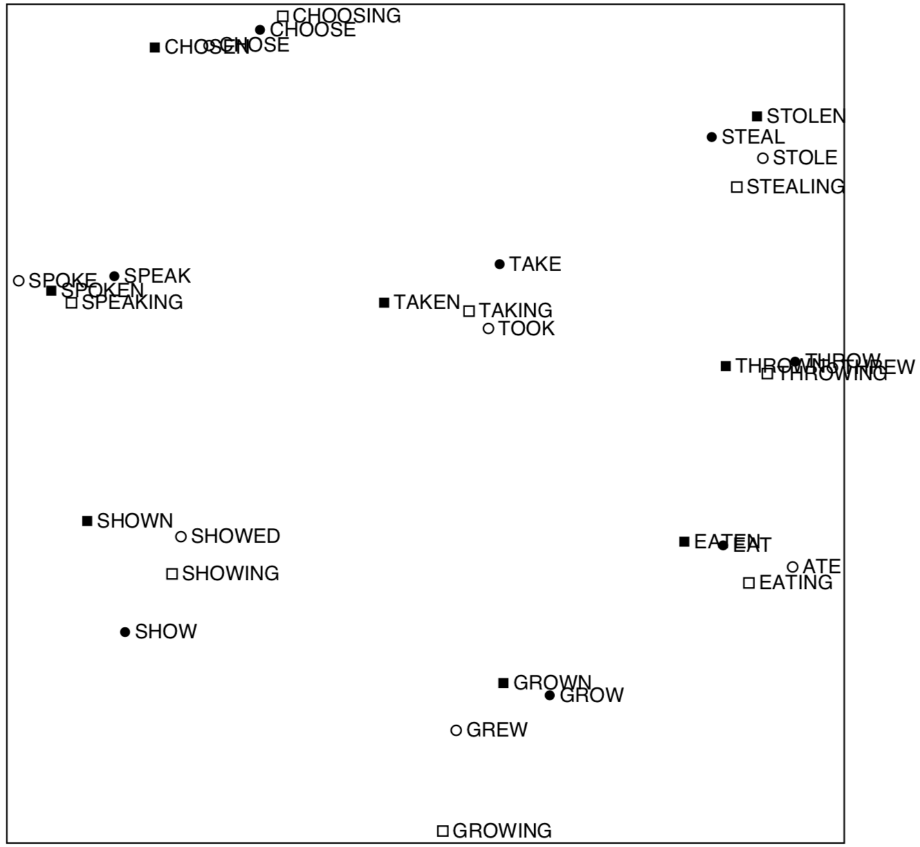
In practice, use singular value decomposition for dimenstionality reduction and limit sparcity.

The direct approach illustrated

Semantic



Syntax



Rohde et al., 2005. An improved model of semantics similarity based on lexical co-occurrence.

ENSAI École nationale de la statistique et de l'analyse de l'information

The many variants of co-occurrence counting

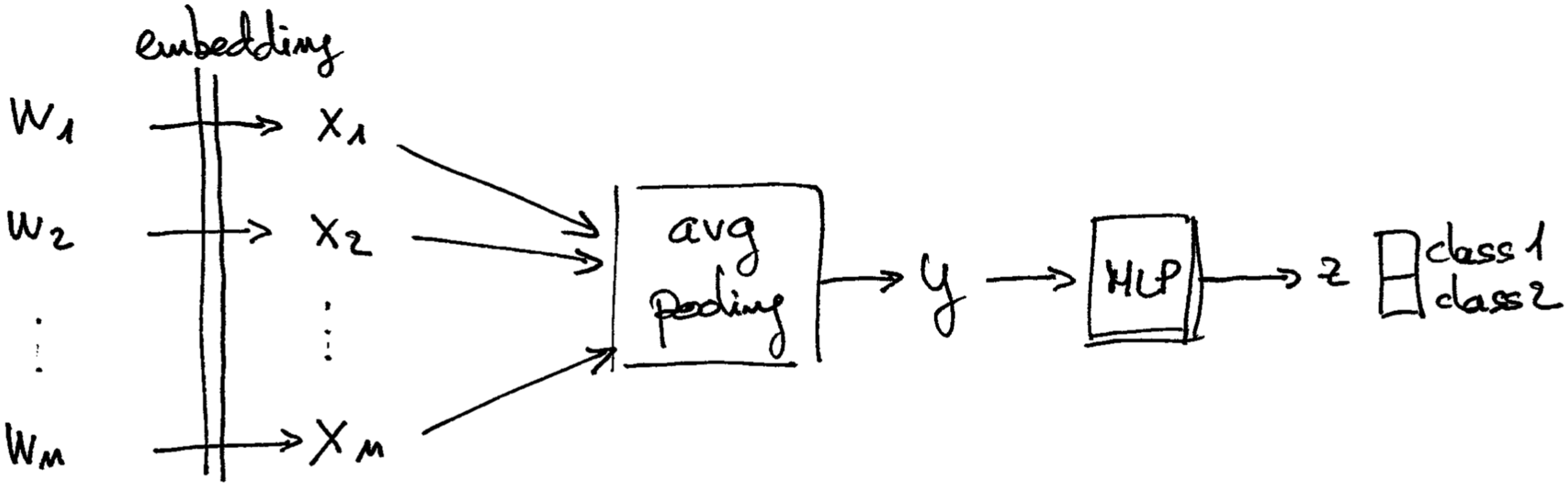
Many variants of the co-occurrence scheme are possible

- weighting w.r.t. the location in the window
- downplay function words
  $\rightarrow$ discard function words, cap counts e.g. at 100, etc.
- use (positive) correlation coefficients instead of counts

Yet, **not too practical** because

- the size of the matrix and the correspinding computational cost
- the cost of adding new words

The notion of word embedding



$x_i = W \cdot 1\mathrm{hot}(w_i) \in \mathbb{R}^d$ $\qquad$ $y = \frac{1}{m}\sum_i x_i$ $\qquad$ $z = \mathrm{softmax}(\tanh(Ay+b)) \in \mathbb{R}^2$

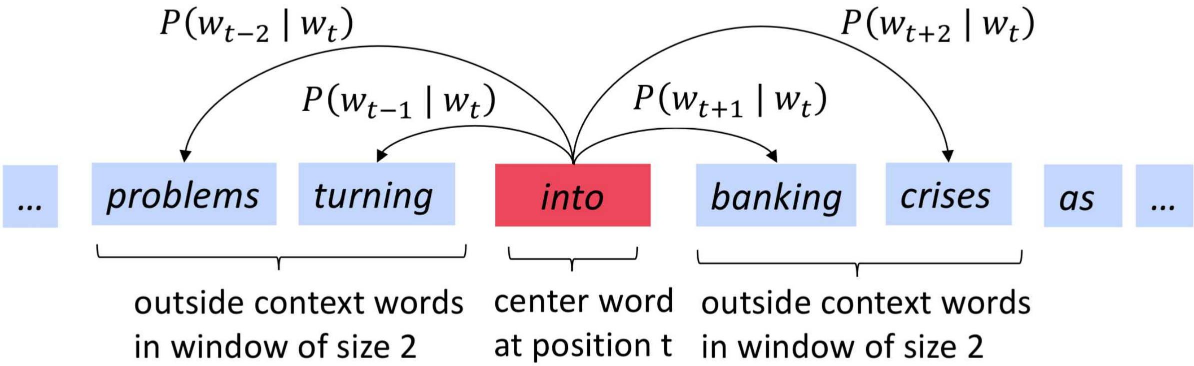
That’s embedding but is it distributional semantics?

The *word2vec* principle

Directly learn word vectors without looking at co-occurrences!

The model comes in two flavors:

1. vectors to “predict” center word from surrounding words (cbow)
2. vectors to “predict” surrounding words from center word (skip-gram)



For the skip-gram model, we thus seek vectors so that

$$J(\theta) = \sum_t \sum_{j \in [-m,m], j \neq 0} \ln p(w_{t+j} | w_t)$$

Towards the skip-gram maths

If we denote

- u_o = vector representing the context/output word o
- v_c = vector representing the center word c

we define the probability of the context knowing the word as a logistic regression (or softmax function), i.e.

$$p(o|c) = \frac{\exp(u_o' v_c)}{\displaystyle\sum_{w \in \mathcal{V}} \exp(u_w' v_c)}$$

and hence

$$J(\theta) = \sum_{t,j} \left(u'_{w_{t+j}} v_{w_t} - \ln \left(\sum_{v} u_v' v_{w_t} \right) \right)$$

This is a mess to optimize!!!!

The skip-gram model approximation

- Approximate the objective function and use negative sampling, i.e.

$$J(o, c, \theta) = \sigma(u'_o v_c) + \sum_{w \sim P[w]} \sigma(-u'_w v_c)$$

  with a sum over a small number of negative samples

- Minimax principle
  - maximize value for co-occurring words $\rightarrow \sigma(u'_o v_c)$
  - minimize value for non co-occuring ones $\rightarrow \sigma(-u'_w v_c)$
  - note that $\sigma(-x) = 1 - \sigma(x)$

In the end (after training properly), we have $u'_a v_b \simeq \ln P[a|b]$ for two arbitrary words a and b.

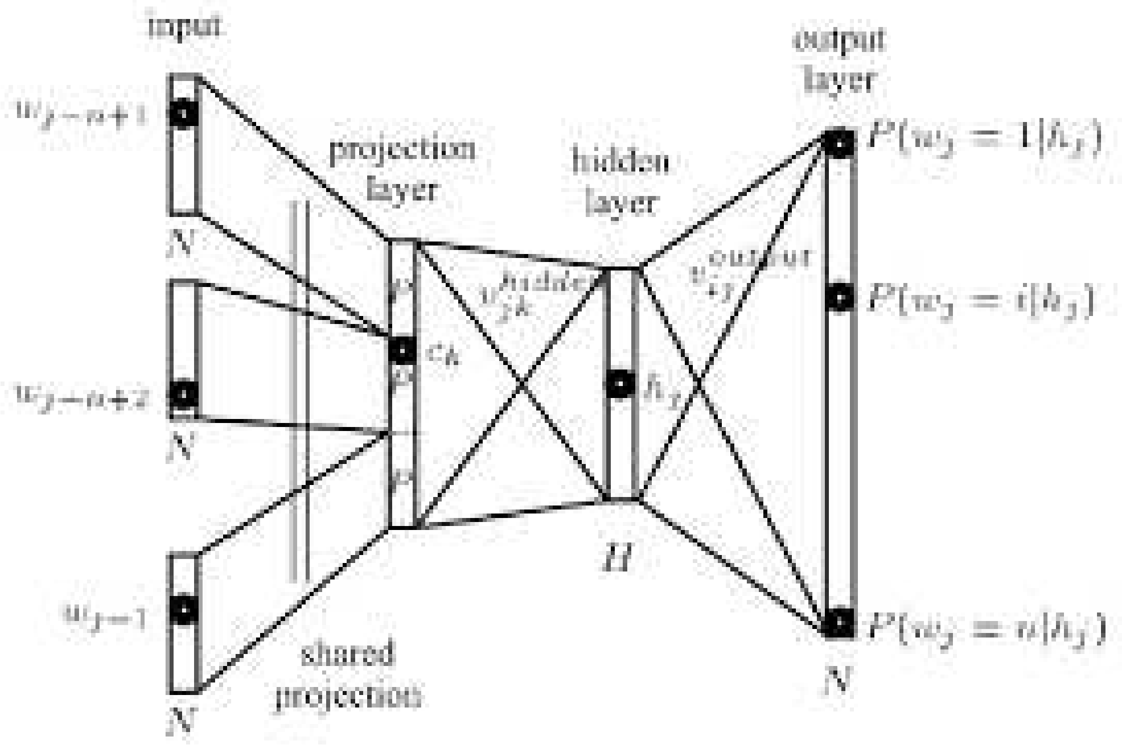
École nationale de la statistique et de l'analyse de l'information

The skip-gram model practical details

- $P[w] = \frac{\text{unigram}(w)^{3/4}}{Z}$ to downplay frequent words
- optimization using
  - stochastic gradient descent: update using gradient for each pair seen
  - minibatch: update using gradient computed over a small batch of pairs
- random initialization of the vectors
- combine u and v to get the output

Mikolov et al., 2013. Distributed representation of words and phrases and their compositionality

What relation with neural networks and embedding layers?



Global vectors

Mikolov's *word2vec* (and other related methods not presented here) are well-behaved

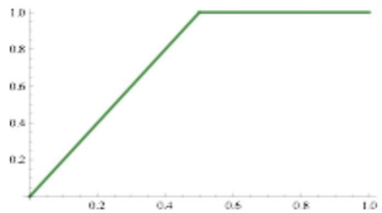
- generalizes pretty well to various tasks
- captures complex linguistic patterns

but also fail in some aspects

- not using global statistics on corpus
- training can be slow and requires very large amounts of data

GloVE combines "the best of both world" with

$$J(\theta) = \frac{1}{2} \sum_{i,j \in \mathcal{V} \times \mathcal{V}} f(P_{ij})(u_i' v_j - \ln P_{ij})^2 \qquad f \sim$$



Pros: fast and efficient, scalable, need for less data

Pennington et al., 2014. Glove: Global vectors for word representation.

Global vectors illustrated

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

litoria

leptodactylidae

rana

eleutherodactylus

Pennington et al., 2014. Glove: Global vectors for word representation.

Similarity evaluation

- Use dot product (cosine similarity) to predict similarity between two words
- Correlates with human judgement

love	sex	6.77
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	keyboard	7.62
computer	internet	7.58
...		

`http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/`

See François Torregrossa *et al*. A survey on training and evaluation of word embeddings. International Journal of Data Science and Analytics, 2021.

fasttext: word embedding through n-gram embedding

$\Rightarrow$ embed subword units rather than words

- decompose word as bag of n-gram with $n \in [3, 6]$
  - where = (<wh + whe + her + ere + re> + <whe + ...)
- word embeding is the sum of subword embeddings
- similarity between w and c is given by

$$s(w, c) = \sum_{g \in \mathcal{G}_w} z_g v_c$$

- optimization of subword embeddings similar to word2vec

Bojanowski et al., 2017. Enriching word vectors with subword information.

ENSAI École nationale de la statistique et de l'analyse de l'information

Model resources on the web

- The word embedding repository `http://vectors.nlpl.eu/repository`
- Fasttext repository `https://fasttext.cc/docs/en/english-vectors.html`
- GloVe models `https://nlp.stanford.edu/projects/glove`
- Pretrained models in NLP pipelines, e.g., gensim model zoo

Practical examples

```
import gensim.downloader
> print(list(gensim.downloader.info()['models'].keys()))
['fasttext-wiki-news-subwords-300',
 'conceptnet-numberbatch-17-06-300',
 'word2vec-ruscorpora-300',
 'word2vec-google-news-300',
...

> glove_vectors = gensim.downloader.load('glove-twitter-25')
> glove_vectors.most_similar('twitter')
[('facebook', 0.948005199432373),
 ('tweet', 0.9403423070907593),
 ('fb', 0.9342358708381653),
...
```