Université Catholique de Louvain

Master en Sciences de l'informatique

Intelligence Artificielle

LINGI2263: Computational Linguistics

Author:
Denis Genon

Contents

1 Introduction

1.1 What is NLP ?

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages. There is differents applications:

- Communicating agent;
- Translator;
- Information extraction;
- Inference of knowlegde;
- ...

1.2 Three approches to NLP

1. **Symbolic**: Use of language ressources (dictionnaries, grammars, thesaurus, ...);
2. **Statistical**: Use of empirical techniques to learn from text and construct language models;
3. **Hybrid**: Combining language ressources and statistical algorithms.

1.3 Basic linguistics concepts

1.3.1 Language vs. languages

- **A language**: A system of spoken/written communication used by a particular country, people, ... typically consisting of words used within a regular grammatical and syntactic structure;
- **The language**: Power or faculty of speech.

1.3.2 Competence vs. performance

- **Competence**: knowledge of a language;
- **Performance**: Use of language in concrete situation.

1.3.3 Nature of the linguistic sign

Words are *linguistic signs*. They convey some idea.

- **Signifier**: The psychic imprint of the sound (not the sound);
- **Signified**: The concept associated to the signifier;
- **Referent**: Actual *object* in the real world.

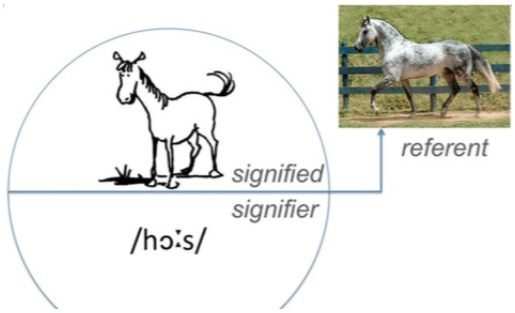


Figure 1: Nature of linguistic sign.

The linguistic sign has the three following characteritics:

- **Arbitrary**: The relationship between the signifier and the signified is not motivated;
- **Conventional**: The link is based on social conventions;
- **Differential**: Signs are part of a system.

1.3.4 Double articulation of the language

A double articulation caracterize human languages.

- At first level, **morpheme** (or moneme) are the smallest meaningful units. A **morph** is a concrete realisation of a morpheme. For example, the morphene *take* has two realizations: *take, took*. The variants of a same morpheme are called **allomorphs**;
- At a second level, **phonemes** are the smallest distinctive units. A **phone** is a realisation of a phoneme. Variants of a given phoneme are called **allophone** (livre and ville).

1.3.5 Two axes: syntagmatic vs. paradigmatic

Morphemes can be replaced by others on the **paradigmatic** axis (vertical) or they can appear in a different environement, combined with other morphemes on the **syntagmatic** axis (horizontal).

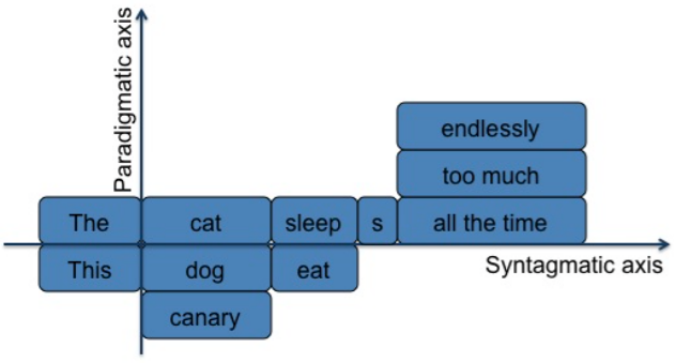


Figure 2: The two axes.

1.3.6 Other general linguistic considerations

- **Creativity**: Finite number of signs permit to elaborate an infinite number of utterences;
- **Evolutivity**: Languages change over the time;
- **Complexity**: Languages are of equal complexity.

1.4 Levels of a linguistic analysis

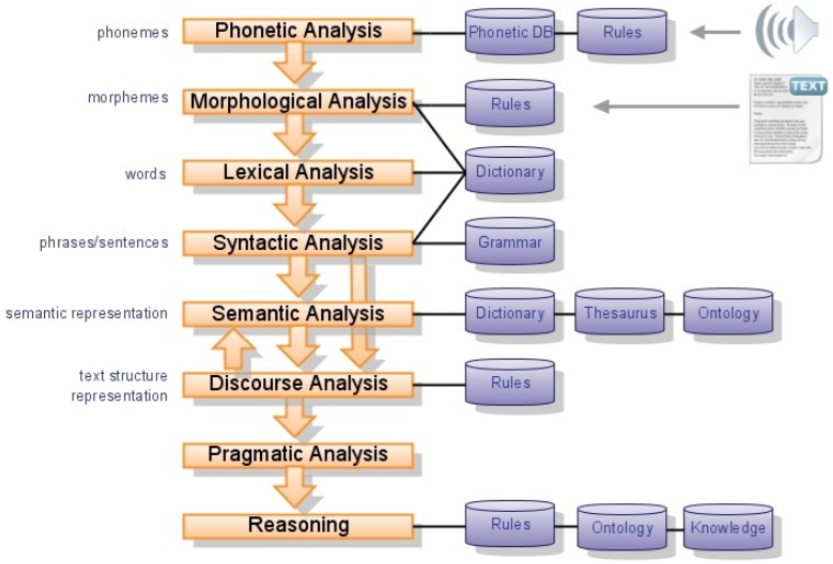


Figure 3: Different levels of linguistic analysis.

1.4.1 Phonology

It is the systematic study of the sounds used in language, and their composition into syllables, words, phrases. Computational phonology is the application of formal and computational techniques to the representation and processing of phonological information.

There is several applications:

- Speech recognition;
- Speech synthesis;
- Phonetic algorithms;
- ...

1.4.2 Morphology

The notion of word is ambiguous:

- **Tokens**: Number of words; **Types**: Number of unique words.
- **Phonemic** words: *[hai]*; **Graphemic** words: *high*.
- **Mono-lexical** unit: *feuille*; **Polylexical** unit: *arc-en-ciel*.
- **Function** word: *to, from*; **Full** word: *home, sea*.

There is too abstract notions behind words:

- **Lemma**: Is the cannonical form of a word (entry of a dictionary, infinitive form or masculine form);
- **Part of Speech**: Are grammatical categories indentifying the nature and/or syntactic function of a word.

Morphene are abstract entities expressing semantic concepts or grammatical features.

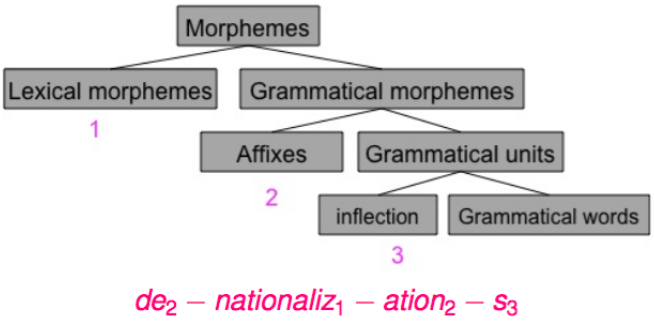


Figure 4: Grammatical and Lexical morphology.

There is several ways to create new words:

- **Inflection**: Grammatical adaptation of a word in a particular syntactic contexts (grammatical morphology);
- **Derivation**: Creation of a new word by adding a bound morpheme (affix) to a derivational base (lexical morphology);
- **Composition**: joining of two or more base forms: hot pepper, cool-headed, online, ...

There are three main aspects linked to morphological analysis:

- **Lemmatization & Stemming**: The goal is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.
  - **Lemmatization**: Based on morphological dictionnaries and analyzers.

Morphological analyser are able to analyze new words using a base dictionary (words and affixes) and morphological rules (add/remove affixes). **Morphological dictionaries** contain forms, lemma, morphological information, ...

– **Stemming**: Based on predefined rules.
Algorithm of Porter:

1. Check if word respect some form;

Porter Algorithm: Definitions

- v (vowel) $\in$ {A, E, I, O, U, Y iff C+Y} (Y follow conson)
- c (consonant) $\in$ {alphabet - v}
- V a sequence of *n* v where $n > 0$
- C a sequence of *n* c where $n > 0$
- any word can be represented by [C] (VC){m} [V]
  where [C] and [V] ≥ 0
  and (VC) can be repeated *m* times

2. If word respect some form, apply 5 runs of somes rules.

Porter Algorithm: More Examples

- (m > 0) EED $\rightarrow$ EE (try: feed $\rightarrow$? ; agreed $\rightarrow$?)
- (*v*) ED $\rightarrow \varnothing$ (try: plastered $\rightarrow$? ; bled $\rightarrow$?)
- (*v*) ING $\rightarrow \varnothing$ (try: motoring $\rightarrow$? ; sing $\rightarrow$?)

There is two typical problems: **Under-stemming** error (algo failed to group two words sharing conceptual meaning: divide and division does not give the same result); **Over-stemming** error (algo grouped words having no shared conceptual meaning: wand and to wander).

- **POS Tagging**: labeling word with POS (rules vs probabilities);
- **Disambiguation**: Clarify the nature or function of a word in a sentence.

1.4.3 From lexis to syntax

Lexicalization is the process of adding words, set phrases, or word patterns to a language. This step is complicated for **compound words** and **frozen expressions**.

2 Corpus

2.1 What is a corpus ?

A large body of linguistic evidence typically composed of attested language use.

There is several desired characteristics of a corpus:

- Machine readable;
- Sampled (aiming at balance and representativeness);
- Well-organized and formatted;
- Big.

2.1.1 Whats is a corpus for ?

- For linguists:
  - Empirical approach (based on facts)
  - Chomsky does not like it because it captures bits of performance but not competence.
- In NLP: the raw fuel of NLP.

2.1.2 Typology

There is two category of corpus: **monolingual** and **multilingual**. The first one has three types:

- **Reference corpus**: large, balanced and *representative*, it is designed to provide comprehensive information about a language;
- **Specialized corpus**: corpus designed to cover one particular aspect of the language;
- **Static** vs. **Monitor corpus**: Static corpus does not evolve.

The second one (**multilingual**) has two types:

- **Comparable corpora**: several corpora of various languages collected using the same sampling method and similar balance in order to permit comparisons between corpora;
- **Parallel corpora**: gather a corpus in one language and the translation of this corpus in one or more orther languages.

2.1.3 Collection

Oral corpora are collected from recordings of interview, conferences, radio or TV programs, ... **Written corpora** are collected from digitalization of analogical sources, digital sources, web, ...

2.2 Corpus Annotation

The aim of corpus annotation is to make explicit some linguistic infor- mation that is part of the text. This is done by adding metadata to the text. Automatic or manual.

There is 6 levels of annotations:

- morpho-syntactic annotation;
- lemmatization;
- syntactic annotation (treebanks);
- semantic annotation (entities, word meaning, predicates, etc.);
- discourse annotation (ie: anaphora);
- phonetic transcription.

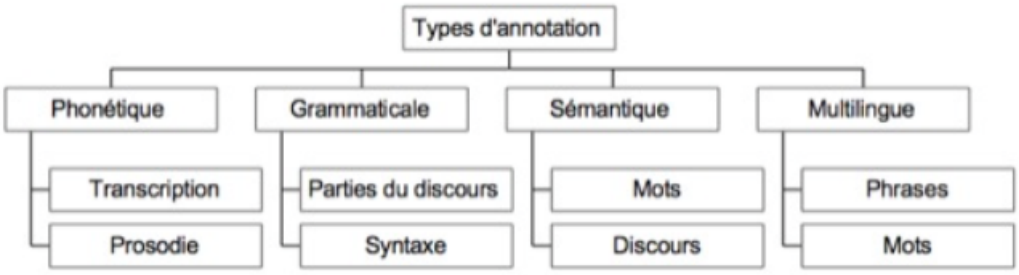


Figure 5: Levels of annotation.

There is some standards for annotation: the goal is to ensure corpora **reusability** and software **interoperability**.

2.3 Corpus Processing

Here is the typical steps of corpus processing:

- **Text formatting and normalization**
- **Tokenization**: Sevral ways to do it. Type/Token Ratio.
- **Lexical analysis**
  - **Morphological analysis**: Based on dictionary and morphological rules. On the fly and can process new words. If only dictionary based, it is quick but limited on the dictionary coverage.
  - **POS tagging**: Remove ambiguity (walk: noun or verb ?)
  - **Lemmatization**
- **Entitites recognition**
- **Syntactic analysis**
- ...

3 Ngrams

3.1 Introduction

In N-Grams, each word occurence is considered as a random event.

3.1.1 Zipf's law

The count histogram follow the Zipf's law: $y = bx^{-a}$.

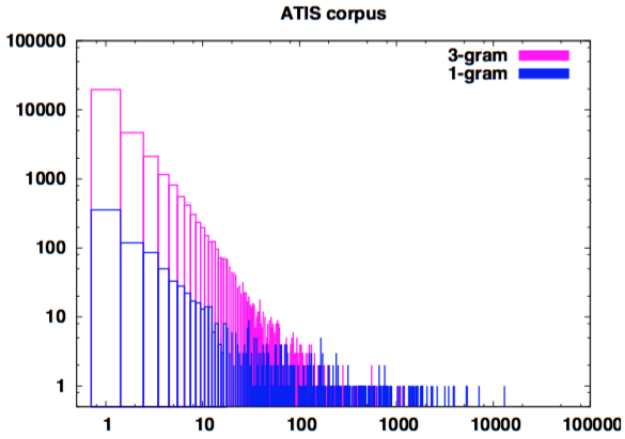


Figure 6: Zipf's law.

3.1.2 N-Gram probability

The conditional probability of observing a word w at a position i given a history $h = \ldots w_{i-2}w_{i-1}$ (or N-gram context) of preceding words

$$P(X_i = w_i \mid \ldots, X_{i-2} = w_{i-2}, X_{i-1} = w_{i-1}) \overset{notation}{=} P(w_i|h)$$

- The random variables X_i, X_{i-1}, X_{i-2}
  - are implicit in the shorter notation $P(w_i|h)$
  - take their value in a fixed vocabulary $w_i, w_{i-1}, w_{i-2}, \cdots \in W$
- The model is assumed stationary
  - it does not depend on the position i in the text: $P(w_i|h) = P(w|h), \forall i$
- The history h for an N-gram is made of the $N-1$ preceding words
- The word w is said to be predicted given the known history h

Figure 7: N-Gram probability.

We can then compute a sentence probability if we use the chain rule and the N-Gram assumption.

$$P(w_1 \dots w_n) = P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\dots P(w_n|w_1^{n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$
$$\approx \prod_{k=1}^{n} P(w_k|w_{k-N+1}^{k-1})$$

Figure 8: Sentence probability.

For example (3-gram): P(The rain in Spain) ≈ P(The) ∗ P(rain | The) ∗ P(in | The rain) ∗ P(Spain | rain in).

3.1.3 Consistency property

Must be verified for all models.

$$\forall h, \sum_{w \in W} P(w|h) = 1$$

Figure 9: Consistency property.

3.2 N-Gram estimation

3.2.1 Maximum Likelihood estimation

$$\hat{P}(w|h) = \begin{cases} \frac{C(h,w)}{C(h)} & \text{if } C(h) > 0 \\ 0 & \text{otherwise} \end{cases}$$

- $C(h, w)$ is the number of times the history h followed by w has been observed in the training set
- $C(h) = \sum_{w \in W} C(h, w)$ is the number of times the history h has been observed in the training set

Figure 10: Maximum likelihood estimation.

For unigram, $\hat{P}(w) = \frac{C(w)}{N}$ where N is the size of the corpus.

3.3 Smoothing

Smoothing is necessary because MLE assign a zero probability to any unseen events in the training set. Many possible events are assigned a zero probability and the probability of observed events is overestimated. Smoothing techniques must respect the consistency property.

3.3.1 Bayesian estimation

The goal of bayesian estimation (or additive smoothing) is to add a pseudo count to avoid having probabilities of 0 for unseen events in the traning set.

Define a priori pseudo-counts $C^*(h, w) > 0$ for any possible events

$$\hat{P}(w|h) = \frac{C(h, w) + C^*(h, w)}{C(h) + \sum_w C^*(h, w)}$$

Figure 11: Bayesian estimation.

The **prior probability** of the event (h, w) is defined as $\frac{C^*(h,w)}{\sum_w C^*(h,w)}$. $\hat{P}(w|h)$ can be interpreted as a **posterior estimate**.

A particular form of the bayesian smoothing is the **Laplace** smoothing (or add one method).

Set $C^*(h, w) = 1$ for all events $\Rightarrow$ uniform priors

$$\hat{P}(w|h) = \frac{C(h, w) + 1}{C(h) + |W|}$$

Figure 12: Add one method.

In general add-one smoothing is a poor method of smoothing because too much probability mass is moved to all the zeros.

3.3.2 Linear estimation

The goal is to build different estimators for several model orders. The differents estimators vary by history and are combined linearly.

$$\hat{P}(w|h) = \lambda_0 \hat{P}_{ML}(w|h) + \lambda_1 \hat{P}_{ML}(w|h_{-1}) + \cdots + \lambda_p \hat{P}_{ML}(w|h_{-p})$$

Figure 13: Linear interpolation. h_{-1}: h without the front word.

We need then to estimate the weights of the model: **Expectation-Maximization estimation**. The model before is a special case of the **mixture model**.

$$\hat{P}(w|h) = \sum_{i=1}^{k} \lambda_i P_i(w|h) \text{ with } \sum_i \lambda_i = 1$$

Figure 14: Mixture model.

We can estimate the λ's with this algorithm:

Use a held-out sample $S = \{w_1, \ldots, w_M\}$ (which can be seen as a single sequence)

- Initialize: $\lambda_i = 1/k$
- E-step: $E(Model_i|S) = \sum_{j=1}^{M} \frac{\lambda_i P_i(w_j|h)}{\sum_{i=1}^{k} \lambda_i P_i(w_j|h)}$
- M-step: $\lambda_i = \frac{E(Model_i|S)}{M}$

Figure 15: Expectation-Maximization algorithm.

3.3.3 Backoff smoothing

Idea: if we have no examples for a particular trigram, we can estimate its probability by using his bigram probability (backoff).

$$\hat{P}(w|h) = \begin{cases} \frac{C(h,w)-d_c}{C(h)} + \gamma(h)\hat{P}_{back}(w|h) & \text{if } C(h,w) > 0 \\ \gamma(h)\hat{P}_{back}(w|h) & \text{if } C(h,w) = 0 \text{ and } C(h) > 0 \\ \hat{P}_{back}(w|h) & \text{if } C(h) = 0 \end{cases}$$

Figure 16: Backoff smoothing. $\hat{P}_{back}(w|h) = \hat{P}(w|h_{-1})$.

d_c are the **discounting coefficients**, and $\gamma(h) = \sum_{w:C(h,w)>0} \frac{d_c}{C(h)}$.

3.4 Performance assessment

To test the smoothing, we can do **data splitting**:

$$d_c \triangleq d, \forall c$$

An upper bound d_* on the optimal discounting coefficient can be estimated from n_c the number of events occurring c times (see count histogram)

$$d \leq d_* = \frac{n_1}{n_1 + 2n_2}$$

In most practical cases, $n_1 > 0$ and $n_2 > 0 \Rightarrow 0 < d_* < 1$

Figure 17: Absolute discounting.

- **Basic protocol**: split data to into training (90%, estimate the model) and test (10%, evalute the quality of the model) files.
- **Refinement 1**:
  - Split training set into 90% of actual training and 10% of held-out set (used to tune *meta-parameters*: pseudo counts or λ's and select an optimal model order)
  - Re-train on the whole training set with any meta-parameter being fixed.
- **Refinement 2**: Repeat the above using 10-fold cross validation.

We can compute the **perplexity** too. The perplexity is a measure of quality of an iterated Shannon game.

$$LL = \frac{1}{M} \log \left(\prod_{i=1}^{M} P(w_i|h) \right) = \frac{1}{M} \sum_{i=1}^{M} \log P(w_i|h)$$

Figure 18: Per-symbol log likelihood.

$$PP = 2^{-LL} = 2^{\left[-\frac{1}{M} \sum_{i=1}^{M} \log_2 P(w_i|h)\right]}$$

Figure 19: Test set perplexity. If random (uniform) over W, $PP = |W|$. Lower the better.

If there is a word we have never see, even with a consistent and smoothed model, $P(w_{new}|h) = 0$, $PP = +\infty$. The astuce is to define a *UNK* word as part of the vocabulary and relies on smoothing.

4 HMMs

A motivating example is the POS tagging. The goal is to assign a POS tag to each word of a sentence. This tagging is ambiguous, so we will choose a tag according to the context of the word. Words are observed and tags are hidden. We will use a finite state model where tags are states and tagging a sentence reduces to find the most likely state sequence.

4.1 Markov Chains

A discrete time finite *Markov Chain (MC)* is a stochastic process $\{X_t | t \in \mathbb{N}\}$ where the random variable X takes its value at any discrete time t in a finite set W and such that:

$$P[X_t = w|X_0, \ldots, X_{t-2}, X_{t-1}] = P[X_t = w|X_{t-p}, \ldots, X_{t-1}] \triangleq P(w|h)$$

Figure 20: A Markov chain of order p is a N-Gram model with $N = p + 1$.

A stationary finite order 1 MC is characterized by a transition matrix

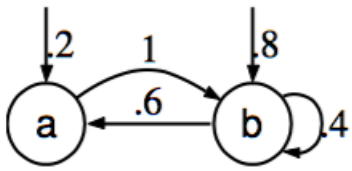
$$\mathbf{P} = [p_{ij}]$$

and an initial probability (row) vector π^0 with

$$\pi_i^0 = P[X_0 = w_i]$$

Figure 21: Transition and initial probabilities.

The maximal number of states $|W|^p$ grows exponentially with p.



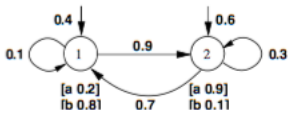
$$W = \{a, b\} \qquad \mathbf{P} = \begin{bmatrix} 0 & 1 \\ .6 & .4 \end{bmatrix} \qquad \pi^0 = [.2 \;\; .8]$$

The probability of a sentence $s = w_0 \dots w_{|s|-1}$, also called the likelihood $P(s|M)$ of s according to a first-order Markov Chain M

$$P(s|M) = \prod_{i=0}^{|s|-1} P(w_i|M) = P[X_0 = w_0] \prod_{i=1}^{|s|-1} P[X_i = w_i|X_{i-1} = w_{i-1}]$$

Figure 22: A sentence probability for a Markov Chain.

4.2 Hidden Markov Models



$W = \{a, b\}$ $Q = \{1, 2\}$

$$\mathbf{A} = \begin{bmatrix} 0.1 & 0.9 \\ 0.7 & 0.3 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0.2 & 0.8 \\ 0.9 & 0.1 \end{bmatrix} \quad \pi = \begin{bmatrix} 0.4 & 0.6 \end{bmatrix}$$

A discrete HMM (with state emission)

- W is a finite vocabulary
- Q is a set of states
- $\mathbf{A}$ a $|Q| \times |Q|$ transition probability matrix ($\sum_{q' \in Q} \mathbf{A}_{qq'} = 1$)
- $\mathbf{B}$ a $|Q| \times |W|$ emission probability matrix ($\sum_{a \in W} \mathbf{B}_{qa} = 1$)
- π an initial probability distribution ($\sum_{q \in Q} \pi_q = 1$)

4.2.1 Path likelihood

The likelihood $P(s, \nu|M)$ of a sequence $s = s_1 \dots s_{|s|}$ along a path or state sequence $\nu = q_1 \dots q_{|s|}$ in a HMM M

$$P(s, \nu|M) = \prod_{i=1}^{|s|} P(s_i, q_i|M) = \pi_{q_1} \mathbf{B}_{q_1 s_1} \prod_{i=2}^{|s|} \mathbf{A}_{q_{i-1} q_i} \mathbf{B}_{q_i s_i}$$

Figure 23: Path likelihood.

4.2.2 Sequence likelihood

Complicated because $O(|Q|^{|S|})$ possible state sequences ($|Q|$ is number of states and $|S|$ is sequence length).

4.3 Most likely state sequence

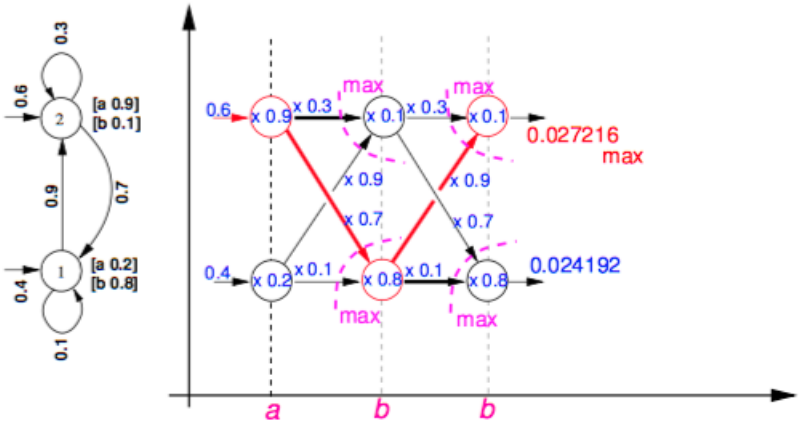


Figure 24: Example.

$$\nu^* = \text{argmax}_\nu \, P(s, \nu | M)$$

Auxiliary quantity: $\gamma(k,t) = P(s_1 \dots s_t, \nu_t^* = k|M)$
The probability of the most likely path ν^* reaching state k at step t

Initialization: $\gamma(k,1) = \pi_k \mathbf{B}_{ks_1}$
Recurrence: $\gamma(k,t) = \max_l[\gamma(l,t-1)\mathbf{A}_{lk}]\mathbf{B}_{ks_t}$
$back(k,t) = \text{argmax}_l[\gamma(l,t-1)\mathbf{A}_{lk}]$
Termination: $P(s,\nu^*|M) = \max_l \gamma(l,|s|)$ $\quad q^*_{|s|} = \text{argmax}_l \gamma(l,|s|)$

Figure 25: Viterbi decoding (recurrence) Algorithm.

- $P(s, v^*)$ gives the probability of the optimal path v^*;
- Computation are done with log (because too small values);
- Complexity: $\Theta(m|s|)$;
- Path v^* is an alignement between states and words;
- v^* can be recovered with the backpointers.

4.4 Sequence likelihood

$$P(s|M) = \sum_{\nu} P(s, \nu|M)$$

Auxiliary quantity: $\alpha(k,t) = P(s_1 \dots s_t, \nu_t = k \,|\, M)$
The likelihood of emitting the first t words and reaching state k at time t

Initialization: $\alpha(k,1) = \pi_k \mathbf{B}_{ks_1}$
Recurrence: $\alpha(k,t) = \sum_l [\alpha(l,t-1)\mathbf{A}_{lk}]\mathbf{B}_{ks_t}$
Termination: $P(s|M) = \sum_l \alpha(l,|s|)$

Figure 26: Forward recurrence, same complexity as viterbi recurrence.

4.5 The learning problem

Given an HMM structure and several sentences, you must estimate A, B, π.

4.5.1 Supervised

The learning sentences are annoted with their respective states.

- $\mathbf{B}_{ki} = \frac{C(k,w_i)}{C(k)}$
  - $C(k, w_i)$ = number of times word w_i is observed on state k
  - $C(k)$ = number of times state k is used
- $\mathbf{A}_{kl} = \frac{C(k,l)}{C(k)}$
  - $C(k, l)$ = number of times a transition from state k to state l is used
- $\pi_k = \frac{C_1(k)}{\sum_{j=1}^{|Q|} C_1(j)}$
  - $C_1(k)$ = number of times state k is used as first state

Figure 27: Supervised learning.

4.5.2 Unsupervised

The sentences are not annoted. There is two algorithm. The first one is **Viterbi training**.

1. Fix initial parameter values $\mathbf{A}^0, \mathbf{B}^0, \pi^0$
2. **repeat**
   1. compute a most likely path through a Viterbi alignment for each training sentence given the parameters $\mathbf{A}^i, \mathbf{B}^i, \pi^i$
   2. estimate the emission and transition frequencies from such paths
   3. recompute the parameter values $\mathbf{A}^{i+1}, \mathbf{B}^{i+1}, \pi^{i+1}$ from those frequencies

   till some stopping criterion is met (*e.g.* max number of iterations)

Figure 28: Viterbi learning.

The second is **Forward-Backward** or **Braaum-Welch** algorithm. Viterbi training is an approximation as it considers that each training sentence is generated along a single path. A more accurate estimation is obtained if one considers all possible paths to generate each sentence.

- actual frequencies ($C(k, w_i)$, $C(k)$, $C(k, l)$, ...), are replaced by expected frequencies;
- special case of expectation-maximization (EM) procedure

Viterbi and Baum-Welch training are both sensitive to parameter initialization.

5 POStagging

5.1 Rule-based approach

Use a large dictionary to assign each word a set of possible tags. Apply a large list of disambiguation rules to restrict each set to a single tag for each word. The constraints are used in a negative way, to eliminate the tags that are inconsistent with the context.

Input: **that**
if *(next word is adjective, adverb, or quantifier)*
AND (it is followed by a sentence boundary)
AND (the previous word is not a verb like 'consider')
then eliminate non-ADV tags;
else eliminate ADV tag;

Figure 29: Example of rules.

There are some limitations:

- Specific to a given language;

- Linguistics ressource need to be updated.

5.2 Probabilistic approach

See previous chapter.

5.3 Transformation-based tagging

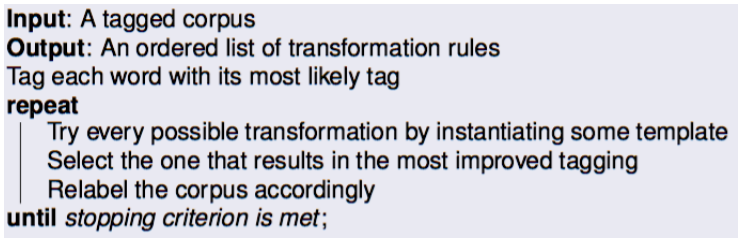


Figure 30: Brill tagger.

The algorithm need transformations templates (Change tag from **a** to **b** when word before is tagged **z**, ...) and a tagged corpus. A stopping criterion could be: insufficient improvement over the previous pass. To know if the transformation give a better tagging, the algorithm need a tagged corpus. The output of the TBL process is an ordered list of transformations; these then constitute a tagging procedure that can be applied to a new corpus.

5.3.1 Pros

- Transformations rules can be interpreted linguistically;
- Learning those rules makes it possible to adapt the tagger to several languages.

5.3.2 Limitations

- A transformation rule can be learned only if it is an instance of an abstract transformation template;
- Supervised only, need a tagged corpus;
- Computational complexity of learning is an issue.

5.4 HMM POS tagging

5.4.1 Supervised learning

- Compute counts from a training set of tagged sentences (sequences of `WORD/TAG` pairs)
- Build the probability estimates

$$\hat{P}(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)} \quad \hat{P}(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

- Smooth the probability estimates
  - Additive smoothing for emission probabilities

$$\hat{P}(w_i|t_i) = \frac{C(t_i, w_i) + \varepsilon}{C(t_i) + \sum_{w \in W} \varepsilon} \quad 10^{-6} \leq \varepsilon \leq 1$$

  - Possibly, any bigram smoothing for transition probabilities

Figure 31: HMM tagging = Viterbi decoding.

Pro We can interpret the states as true POS tag and there is no need to discover the meaning of the states after learning.

Limitations A tagged corpus is required.

5.4.2 Unsupervised learning

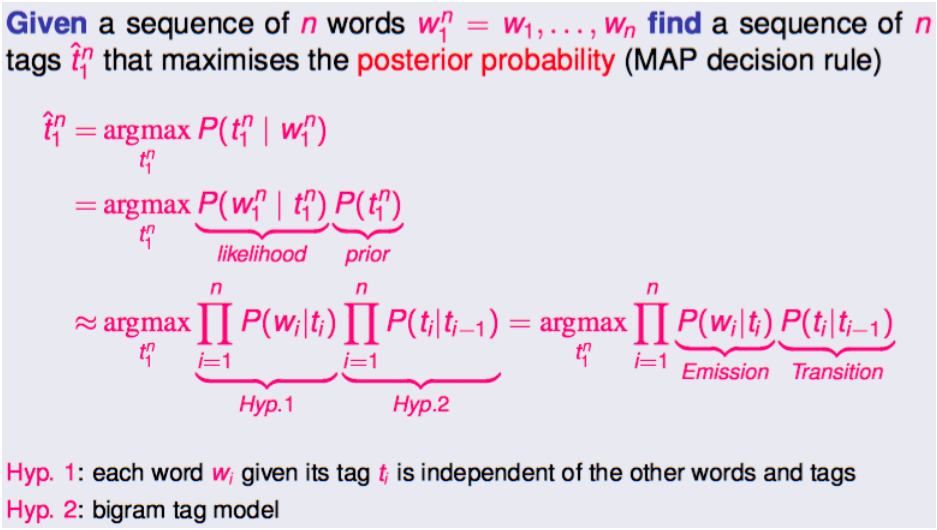


Figure 32: HMM tagging = Viterbi decoding.

Pro Fully automatic. Define a HMM structure with one state per tag in a tagset and learn the HMM parameters $P(w_i|t_i)$ and $P(t_i|t_{i-1})$ using Viterbi training or Baum-Welch on a text corpus (not tagged!).

Limitations Viterbi training and Baum-Welch are sensitive to the initialization. The states are not necessarily associated with relevant tag set, some linguistically informed post-processing needs to be done if one wants to map states to actual tags, whenever possible.

5.4.3 Notes

A **out-of-vocabulary** word in the test set is assigned as a zero probability, there is no Viterbi path. The usual solution is to replace any word occuring only once in the training set with the marker *UNK*. We then need to reduce the observed vocabulary and add *UNK* to it and smooth the emission probability.

It is possible to extend HMM to tag trigrams.

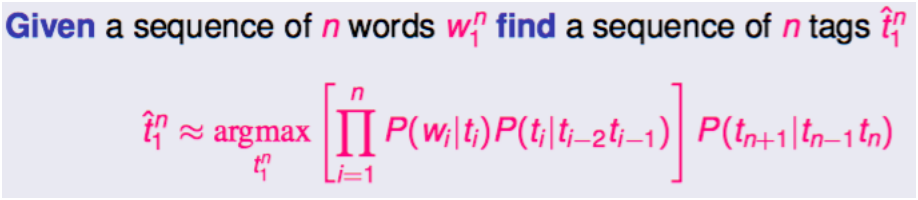


Figure 33: Need to apply n-gram smoothing techniques.

5.5 Evaluation

	IN	JJ	NN	NNP	RB	VBD	VBN
IN	—	.2			.7		
JJ	.2	—	3.3	2.1	1.7	.2	2.7
NN		8.7	—				.2
NNP	.2	3.3	4.1	—	.2		
RB	2.2	2.0	.5		—		
VBD		.3	.5			—	4.4
VBN		2.8				2.6	—

TAG Confusion Matrix

- Row is an actual tag;
- Column is predicted tag;
- Entry is error percentage.

There is the formula to assess the perfomance of the tagging.

- **Average error rate per TAG** $= \frac{1}{\#ofrows} * \sum_i$(total of row i);
- **Tagging error rate** $= \frac{\sum_i f(i) * \text{(total of row i)}}{\sum_i f(i)}$;
- **Tagging accuracy** $= 100\% -$ Tagging error rate.

6 Formal Grammar

The **syntax** correspond to an arrangement to form correct sentences. A **formal grammar** is the set of rules which define the syntax. The **syntactic analysis** is the application of formal rules to filter grammatical and ungrammatical sentences. There is three fundamental concepts in formal grammar.

- **Constituency**: a group of words may behave as a single unit. For example, nominal sentences. There is two clues of words that are grouped togheter: the first one is the context (before a verb for example) and the second is that we can move the group of word in different places of the sentence (preposed/postposed constructions);
- **Grammatical relations**: formalization of ideas from traditional grammar (Subject/Object);
- **Subcategorization and dependency relations**: Some types of relations between word and phrases (for example transitive verbs like *find*).

6.1 Context Free Grammar

A CFG consists of a set of rules and a lexicon of words and symbols.

- N a set of non-termina symbols;
- $\sum$ a set of terminal symbols;
- R a set of rules of the form $A \rightarrow \beta$;
  - A is non-terminal;
  - β is a string of symbols: $(N \cup \sum)^*$.
- S a start symbol.

A CFG define a formal language. The term generative grammar underlines the fact that in this approach a language is defined by a set of possible sentences generated by the grammar.

6.2 Some grammar rules and syntactic structures

6.2.1 Agreement problem

To handle the particular cases (for example a verb which take *s* at the third singular form), we need to add rules on the grammar. The consequence is a increase of the size of the grammar. A solution is to use a constraint-based formalism to take into accont fine-grained information about number and person agreement, subcategorization and even semantic categories.

Figure 34: Feature structure.

6.2.2 Subcategorization

All verbs are not compatible with all kind of constituents. The solution is to have different catagories of words (for example, transitive verb and instransitive verb).

6.2.3 Treebank

6.3 Dependency Grammars

In dependency grammars, constituents and phrase structure rules do not play any fundamental role. Instead, the syntactic structure of a sentence is described purely in terms of words and binary semantic or syntactic relations between these words (called lexical dependencies). Dependency grammars use dependency relationships

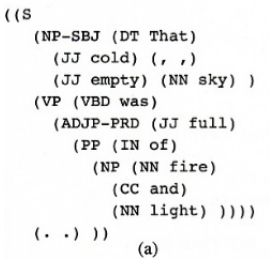


Figure 35: Brown treebank.

between lexicon elements. The nature of the relationship can be morphologic, syntaxic or semantic. The main advantage is to handle languages with free word order (Czech, Latin).

7 Syntactic Parsing

7.1 Introduction

Problem: Given a formal grammar G (CFG) and a sentence s, build a parse tree of s according to G. There may be no solution or many solutions (ambiguity). Context-free means that any rules used to replace the left hand side can be used anywhere in a parse tree.

Parsing is the search among all possible parse trees that can be generated from S for the one generating the observed sentence s. The search space can be explored **bottom-up** or **top-down**.

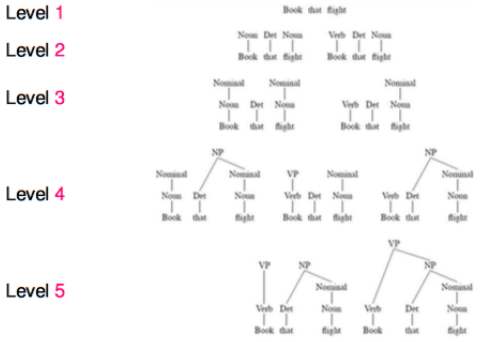


Figure 36: Bottom-up. Never wastes time exploring tree that cannot result is an S.

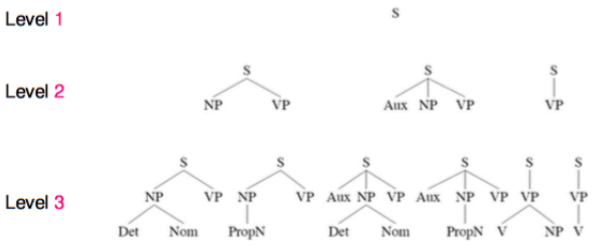


Figure 37: Top-down. Never wastes time on trees that are not consistent with the input.

Due to ambiguity, in the worst case, the number of parse trees is exponential to the sentence length.

7.2 CYK Algorithm

Before using the algorithm, we need to change the grammar in Chomsky Normal Form.

A CFG is in Chomsky Normal Form
if it contains only two kinds of rules:

$A \to B\ C$
$A \to w$

with A, B, C non-terminals and w a terminal

Figure 38: CNF.

1. Copy all conforming rules from G to the new grammar G'
2. Convert terminal within rules to dummy non-terminals in G'
   $INF\text{-}VP \to to\ VP$
   becomes
   $INF\text{-}VP \to TO\ VP$
   $TO \to to$
3. Convert unit-productions
   If $A \overset{*}{\Rightarrow} B$ by a chain of unit-productions and $B \to \gamma$ is a non-unit production
   Then add $A \to \gamma$ to the new grammar G'
4. Make all rules binary and add them to G'
   1. **Replace** any rule $A \to B\ C\ \gamma$ **by** $A \to X_1\ \gamma$ and $X_1 \to B\ C$
   2. Iterate 1 until all non-terminal rules are binary

Figure 39: Conversion algorithm.

Input: *words*
Input: *grammar*
Output: *table*
for $j \leftarrow 1$ **to** LENGTH(*words*) **do**
  $table[j-1, j] \leftarrow \{A \mid A \to words[j] \in grammar\}$
  for $i \leftarrow j-2$ **down to** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      $table[i,j] \leftarrow table[i,j] \cup \{A \mid A \to B\ C \in grammar, B \in table[i,k], C \in table[k,j]\}$
return *table*

Figure 40: CYK algorithm. The sentence is accepted if S in $table[0, N]$. Back-pointers. Bottum-up. $O(n^3)$.

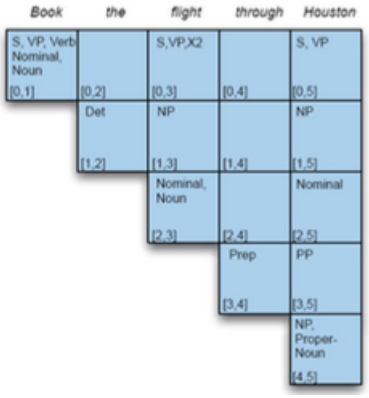


Figure 41: Example.

Figure 42: Order.

7.3 Probabilistic CFG and probabilistic parsing

Problem: Given a PCFG G and a sentence S, compute a most likely parse tree.

$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T \mid s, G) \\ &= \operatorname{argmax}_T \frac{P(s, T \mid G)}{P(s \mid G)} \\ &= \operatorname{argmax}_T P(s, T \mid G)\end{aligned}$$

Figure 43: Disambiguation rule.

Solve the exponential blow-up issue by computing most likely parse tree.

```
table[i, j, A]: probability of spanning the input from i to j with non-terminal A
Input: words
Input: a PCFG in CNF
Output: a most likely parse and its probability
for j ← 1 to LENGTH(words) do
    table[j − 1, j, A] ← P(A → words[j])        // Init with terminal rules
    for i ← j − 2 down to 0 do
        for k ← i + 1 to j − 1 do
            foreach {A → B C ∈ grammar and
                     table[i, k, B] > 0 and table[k, j, C] > 0} do
                if table[i, j, A] < P(A → B C) × table[i, k, B] × table[k, j, C]
                then
                    table[i, j, A] ← P(A → B C) × table[i, k, B] × table[k, j, C]
                    back[i, j, A] ← {k, B, C}    // Backpointer
return BUILDTREE(back[0, LENGTH(words), S]), table[0, LENGTH(words), S]
       Most likely parse                      probability
```

Figure 44: Probabilistic CYK.

7.3.1 Learning rule probabilities from a Treebank

$$\hat{P}(A \to \beta) = \hat{P}(A \to \beta \mid A) = \frac{C(A \to \beta)}{\sum_\gamma C(A \to \gamma)} = \frac{C(A \to \beta)}{C(A)}$$

$C(A \to \beta)$ = number of times the rule $A \to \beta$ is used
$C(A)$ = number of times A is used as left-hand-side of some rule

Figure 45: Rule probability estimate. Smoothing may be used. Unsupervised learning possible: Inside-Outside algorithm.

7.4 Earley Algorithm

```
Input: words
Input: grammar
Output: chart
ENQUEUE((γ → •S, [0,0]), chart[0])        // Initialize chart
for i ← 0 to LENGTH(words) do
    foreach state ∈ chart[i] do
        if INCOMPLETE(state) then
            if NEXT-CAT(state) is a POS then SCANNER(state) ;
            else PREDICTOR(state) ;
        else
            COMPLETER(state)
return chart
```

Figure 46: Earley algorithm. Top-down. No need for CNF. A state $S \to \alpha\bullet, [0, N]$ in $chart[N]$ denotes a successful parse. $O(n^3)$.

7.5 Partial parsing

A style of partial parsing is **chunking**: a simple bracketing without hierarchical structure.

7.5.1 Algorithms

- HMMs;
- Local classifier from selected features;
- Iterated chunking based on a hierarchy of chunk tag offers an approximation to full parsing.

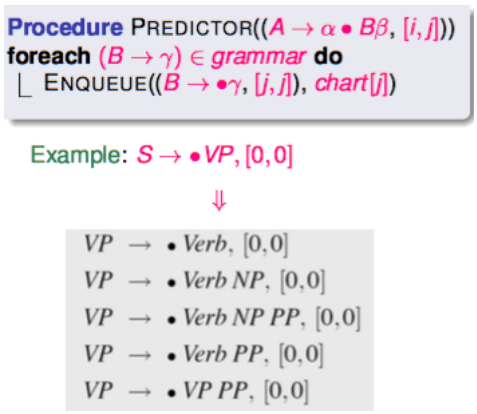


Figure 47: Predictor.

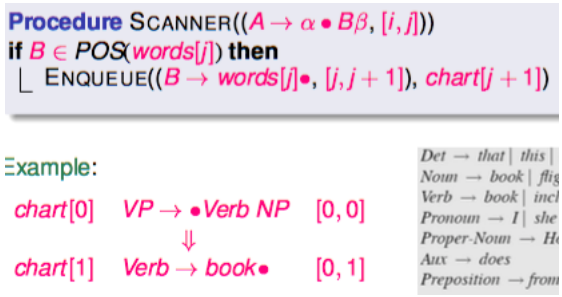


Figure 48: Scanner.

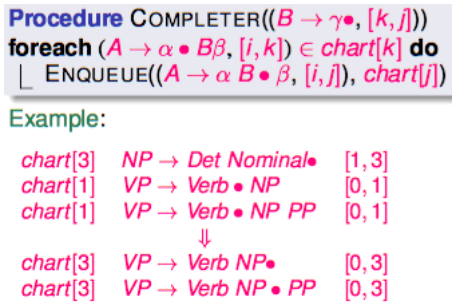


Figure 49: Completer.

7.5.2 Chunking system evaluation

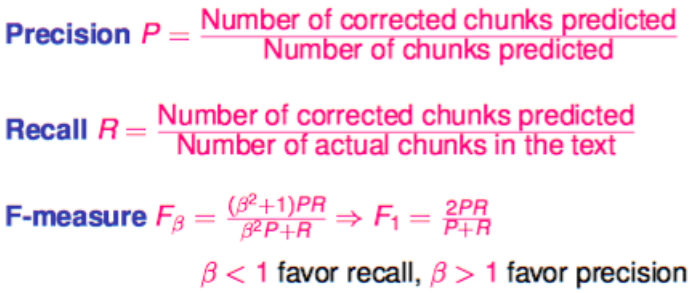


Figure 50: Evaluation. State of art is 92% F_1.

8 Machine Translation

Problem: Machine translation concenrs the use of computers to automate translation from a source language to a target language.

8.0.1 Why is MT difficult ?

- **Strucutral divergences**;
  - **Isolating** (one morphene per word) vs **polysynthetic** (one word is a sentence);
  - **Word orders**;
  - **Argument structure**: verb-framed, satellite-framed;
  - **Pronoun dropping**;
  - **Specific divergences**: adjective precede or not nouns.
- **Lexical divergences**.
  - **Homonymous words**: ex. *bass*;
  - **Polysemous words**: ex. *know*;
  - **Grammatical lexical divergences**: ex. gender on adjectives;
  - **Lexical gaps**: no word in one language.

8.0.2 Various translation tasks

- **Rough translation**;
- **Draft translation with human post-editing**;
- **Fully automatic translation**.

8.1 Approaches

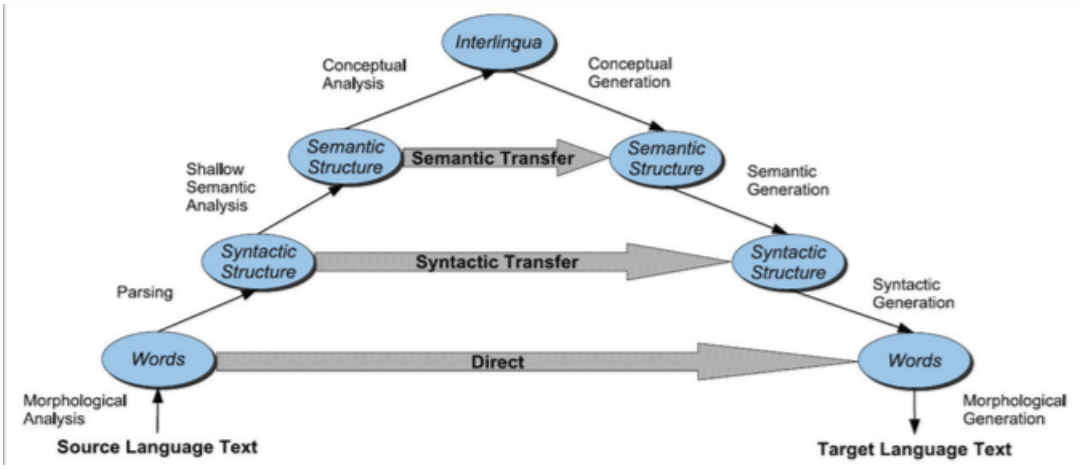


Figure 51: Levels of transfer.

8.1.1 Direct transfer

Word-by-word through the source language text, incrementally transforming the source language text into a target language text (bilingual dictionary for lexical transfer).

Limitations

- Complex and numerous transformation rules;
- No parsing component (cannot handle reliably long distance reordering).

8.1.2 Syntactic and semantic transfer

Parse, then transfer the syntactic structure and finally generate the target text.

Limitations

- Translating from SVO languages to SOV languages require complex syntactic transformations;
- Lexical transfer is also required with a bilingual dictionary but unsatisfactory for ambiguous words;
- Additional semantic analysis is required but it is even more complex to define reliable semantic transformations.

8.1.3 Interlingua

Treat translation as a process of extracting the full meaning of the source and expressing it in the target language.

Limitations

- Deep conceptual analysis is even more difficult than shallow semantic analysis;
- The generation steps are far from trivial;
- Some concepts simply do not exist in common between languages.

8.2 Statistical machine translation

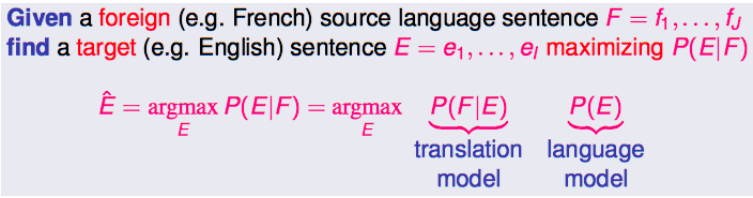


Figure 52: Problem definition. Require: a **language model** to compute P(E); a **translation model** to compte P(F|E); a **decoder** to compute $\hat{E} = \text{argmax}_E P(F|E)P(E)$.

Figure 53: Best translation. Trade-off.

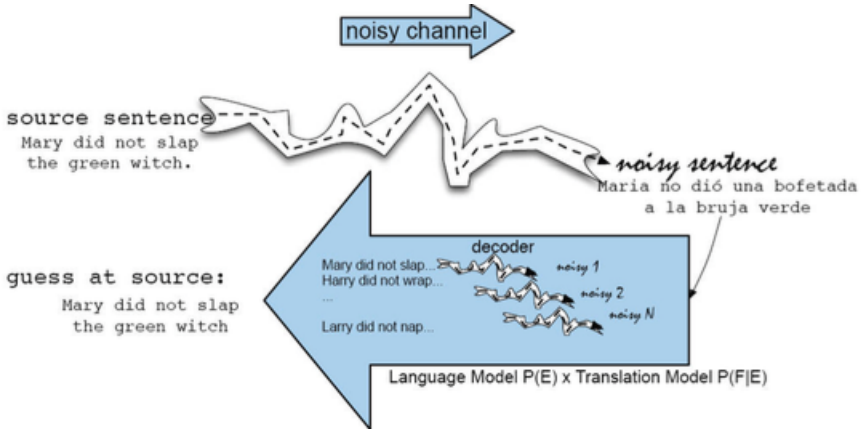


Figure 54: The speaker think in English and produce a noisy version in aother language. The translation aims at decoding the noisy version.

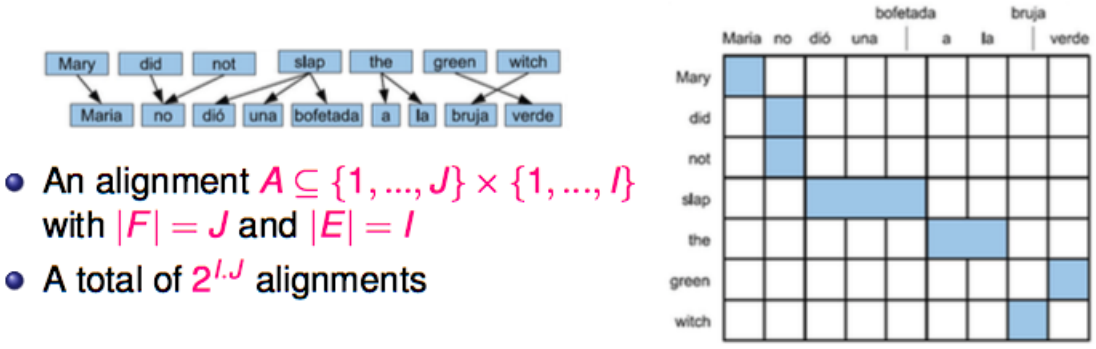
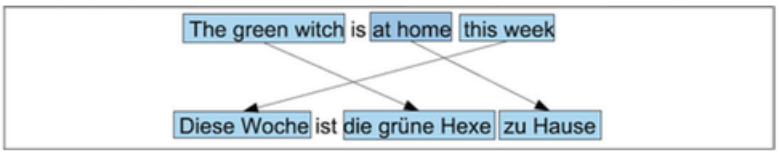


Figure 55: Alignments.

8.2.1 Phrase-based models

Phrases are units of translation, each phrase has exactly one translation, then alignments are permutations. The model includes **translation probabilities** of phrases and **distortion probabilities** to model the permutations.

- **Phrase Translation probability**: $P(\bar{f}_i|\bar{e}_i)$
- **Distortion probability**: $d(a_i, b_{i-1}) = \alpha^{|a_i - b_{i-1} - 1|}$
  - a_i is the start word position of the source phrase generated by the target phrase $\bar{e}_i$
  - b_{i-1} is the end word position of the source phrase generated by the target phrase $\bar{e}_{i-1}$ (with $b_0 = 0$)
  - α is a small positive constant
- **Translation Model** $P(F|E) = \prod_{i=1}^{I} P(\bar{f}_i|\bar{e}_i) d(a_i, b_{i-1})$



$P(\bar{f}_1|\bar{e}_1) = P(\text{die grüne Hexe}|\text{The green witch})$; $P(\bar{f}_2|\bar{e}_2) = P(\text{ist}|\text{is})$; ...
$a_1 = 4, b_0 = 0 \Rightarrow d(a_1, b_0) = \alpha^3$; $a_2 = 3, b_1 = 6 \Rightarrow d(a_2, b_1) = \alpha^4$; ...

Figure 56: Phrase-based translation model. Parameters are $P(\bar{f}|\bar{e})$ and α.

A such model is essentially a large bilingual probabilistic dictionary of phrases which can be estimated from a large corpus of aligned phrases and their respective counts C(.,.), but aligned phrases are rarely available as such in parallel corpora and then word alignments are used as seeds for phrase alignments.

$$\hat{P}(\bar{f}|\bar{e}) = \frac{C(\bar{f}, \bar{e})}{\sum_{\bar{f}'} C(\bar{f}', \bar{e})}$$

Figure 57: Large bilingual probabilistic dictionary.

1. Symmetrizing: produce word alignments $F \rightarrow E$ and $E \rightarrow F$
2. Intersect both alignments
3. Build a classifier to select additional connections from the union of both alignments
4. Extract consistent phrases

Figure 58: Words to phrase alignments algorithm.

8.2.2 Multi-stack decoding with phrase-based models

Decoding problem

$\hat{E} = \text{argmax}_E P(F|E)P(E) = \text{argmin}_E Cost(E, F)$

- Best-first search (A*) to find a minimal cost solution
- The cost combines
  - the **current cost** of currently translated phrases $S = (F, E)$
    $Cost(E, F) = -\log\left(\prod_{i \in S} P(\bar{f}_i|\bar{e}_i) d(a_i, b_{i-1}) P(E)\right)$
  - the **future cost** to translate the remaining words is estimated by ignoring distortion costs
- The algorithm is referred to as **multi-stack decoding** but it actually uses priority queues. One cannot easily compare the cost of partial translations that translate different number of foreign words $\Rightarrow$ there are m "stacks" where "stack" s_m stores all current hypotheses covering m foreign words
- A beam-search pruning decoder ($\neq$ A*) develops all promising hypotheses in parallel and prune the others

Figure 59: Multi-stack decoding with phrase-based models.

8.3 MT evaluation

8.3.1 Criteria

- **Human raters**:
  - **Fluency**: clarity, naturalness;
  - **Faithfulness**: adequacy;
  - **Informativeness**: enough information to accomplish a task;
  - **Edit-cost**: minimizing post editing.
- **Automatic evaluation**:
  - Heuristics to assess translation systems automatically with respect to reference translations provided by humans;
  - Correlated with human judgments.

8.3.2 BLEU evaluation metric

- BLEU metric over a whole test corpus with one candidate and several references per translation
- Modified precision p_n for N-grams of order n $(= 1, 2, 3, \ldots, N)$

$$p_n = \frac{\sum_{C \in Candidates} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C \in Candidates} \sum_{n-gram \in C} Count(n-gram)}$$

- Brevity penalty
  - $|C|$ sum length of candidates
  - $|R|$ sum length of best match reference for each candidate

$$BP = \begin{cases} 1 & \text{if } |C| > |R| \\ e^{\{1-\frac{|R|}{|C|}\}} & \text{if } |C| \leq |R| \end{cases}$$

$$\boxed{BLEU = BP \times \exp\left(\frac{1}{N} \sum_{n=1}^{N} \log p_n\right)}$$

Figure 60: BLEU.

9 Lexical Semantics

9.1 Introduction

Lexical semantics focus on the meaning of words. We call **lexeme** a pair of *form* with its *meaning*. A lexeme is represented by a **lemma**. The meaning of a lema can vary given the context.
A **word sense** is a discrete representatin of one aspect of the meaning of the word. If two senses from a same word have no semantic relation, they are **homonyms**. Else its **polysemy** (ex. *mouse*). If two senses are related, they are viewed as two senses of a **polysemous lexeme**. A kind of polysemy is **metonomy**, where we replace an aspect of a concept by the concept (ex. *boire un verre*).

9.2 Relation between senses

9.2.1 Synonymy and antonymy

Two words are **synonyms** if they are substituable one for the other in any sentence without changing the truth condition of the sentence. **Antonymy** is more complex to define because two senses can be antonyms in many ways:

- **Binary opposition**: alive/dead; true/false;
- **Opposite ends of some scale**: cold/warm; rapid/slow;
- **Reversive**: rize/fall; up/down; buy/sell.

Better to describe synonymy as a relation between senses (ex *hot* - *spicy*). Hard to automatize a task to distinguish synonyms to antonyms.

9.2.2 Hierarchical relations

One sense is a **hyponym** of another sense if the first sense is more specific, denoting a subclass. *Cat* is hyponym of *animal*. A **hypernym** is the inverse.

A **taxonomy** is a collection of controlled vocabulary terms organized into a hierarchical structure. Each term in a taxonomy is in one or more parent-child relationships to other terms in the taxonomy. A **thesaurus** is a networked collection of controlled vocabulary terms. This means that a thesaurus uses associative relationships in addition to parent-child relationships. An **ontology** is an explicit and formal specification of a conceptualization (Gruber 1993). It uses a controlled vocabulary.

9.3 Three approaches to lexical semantics

9.3.1 Lexical relations

Relations between the senses of words. **WordNet** is a lexical database accessible through word senses. There are three databases: one for nouns, one for verbs and one for adjectives and adverbs. A **synset** is a set of equivalent synonyms. A **glose** describe the concept heind the synset.

9.3.2 Event participation

Thematic roles are an attempt to categorize commonality of different roles (agent, experiencer, force, theme, result, ...). The benefit is to allows simple inferences with a shallow meaning representation. The problem: difficult to standardize and formalize roles.

PropBank Annoted sentences with semantic roles (for verbs). Can use Treebank annotation.

FrameNet Based on **frames** (script-like structure that describes events).

9.3.3 Selectional restrictions

Selectional Restrictions express a kind of semantic constraint that a verb imposes on the kind of concepts that are allowed to fill its argument roles. We can use synset of WordNet to achieve this.

9.4 Computational lexical semantics

9.4.1 Word sense disambiguation

The goal is to select the sense for a given word in a given context. There is two tasks:

- **Lexical sample task**: Only for a limited sample of words. Classifiers can be trained on hand labeled corpora;
- **All-words tasks**: Need other approaches due to sparseness/too much work for labeling by hand.

Supervised WSD

Feature extraction The first step is to extract features that can help to predict word senses. We can use preprocess (POST, syntax, lemmatization, ...). Then two classes of features can be used:

- **Collectional features**: encode informations about specific positions at the left or right of the target word ([wi-2, POS i-2, wi-1, POS i-1, wi+1, POS i+1, wi+2 POS i+2], [guitar, NN, and, CC, player, NN, stand, VB]);
- **Bag od words features**: Unordred set of words in the context of the target word ([fishing,big,sound,player,fly,rod,pound,double,runs,playing,guitar,band], [0,0,0,1,0,0,0,0,0,0,1,0]).

Bayesian approach Choosing the best sense $\hat{s}$ out of a set of possible senses S for a feature vector $\vec{f}$ amounts to choosing the most probable sense given that vector.

$$\hat{s} = argmax_{s \in S} P(s|\vec{f}) = argmax_{s \in S} \frac{P(\vec{f}|s)P(s)}{P(\vec{f})}$$

Problem: not enough data to resolve this problem (20 words = 2^{20} vectors).

Naive Bayes approach With the independence assumption: $P(\vec{f}|s) \approx \prod_{j=1}^{n} P(\vec{f_j}|s)$ we have $\hat{s} = argmax_{s \in S} P(s) \prod_{j=1}^{n} P(f_j|s)$. $P(s_i) = \frac{count(s_i, w_j)}{count(w_j)}$ and $P(f_j|s) = \frac{count(f_j, s)}{count(s)}$. Need smoothing.

Dictionnary and Thesaurus method Hand-labelled corpora is not always available. A dictionary or thesaurus can provide an indirect kind of supervision. We select the sense from the lexical resources whose definition share the most words with the target word's neighborhood.

```
function SIMPLIFIED LESK(word, sentence) returns best sense of word

 best-sense ← most frequent sense for word
 max-overlap ← 0
 context ← set of words in sentence
 for each sense in senses of word do
  signature ← set of words in the gloss and examples of sense
  overlap ← COMPUTEOVERLAP(signature, context)
  if overlap > max-overlap then
      max-overlap ← overlap
      best-sense ← sense
 end
 return(best-sense)
```

Figure 61: Lesk algorithm. Improvements: extend list of words used in classifier and use weighting: inverse document frequency $idf_i = log(\frac{Ndoc}{nd_{w_i}})$. Ndoc is the total number of definitions and nd_{w_i} is the number of definitions which contains the word w_i.

WSD Evaluation Two ways to evaluate a WSD system: **task-based evaluation** (bad because the final application does not only depend of the WSD system) and **intrinsic evaluation** (percentage of correctness). We can use two different metrics: **baseline** (compared with first sense in WordNet) and **ceiling** (human inter annotator agreement). We can used two kinds of data: **hand labelled corpora** (supervised) and **pseudowords** (concatenation of two words, the WSD system must find the correct sense).

9.4.2 Word similarity

Kind of distance between words. There is two classes of algorithms:

- **Thesaurus based method**: $sim_{path}(c_1, c_2) = -log(pathlen(c_1, c_2))$;
- **Distributional approach**: find words with the same distribution in the context.

Note: theoretical distinction between word similaritiy and word relatedness (ex. *car* and *gasoline*).

10 Information Extraction

10.1 Named entities extraction

Named entities are proper names, quantity phrases, events, ... There is two common difficulties when extracting NE: **variants** (USA = United States of America) and **ambiguities** (Washington).
To build a name tagger, there is two approches: **Hand-written** (internal evidence and external evidence) and **Automated training** (relies on tagged corpora).

Here is how automated training works:

- We encode data IOB style (B: word starting entity, I: others words of entity, O: dont belong to entity);
- Select features from those inputs (shape features and predicitve words);
- Encode training set with features;
- Train classifier before labelling new data.

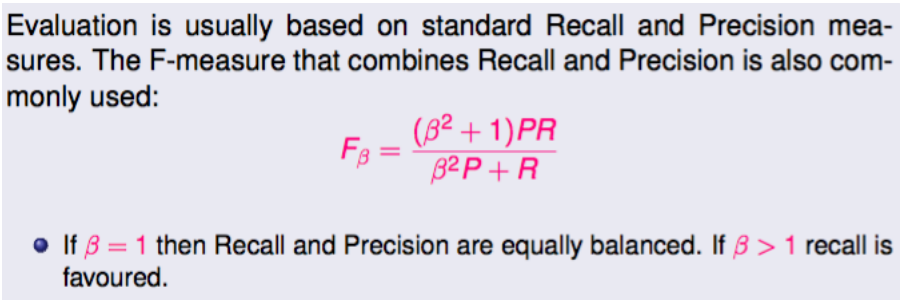


Figure 62: Evaluation of NER system. $R = \frac{TP}{TP+FN}$ and $P = \frac{TP}{TP+FP}$.

10.2 Relation detection

Different kinds of relation.

10.2.1 Hand-written extraction

We can use regular expressions, extraction graphs and language resources.

10.2.2 Statistical approach

Supervised Using hand annoted data. First step is to train a classifier that say if two NE are in relation or not. The second step is to train a classifier that label the relation based on features from the NE, words in the context and syntactic structure.

Semi-supervised Induce new patterns by bootstrapping from the initial search results from a small set of seed patterns.

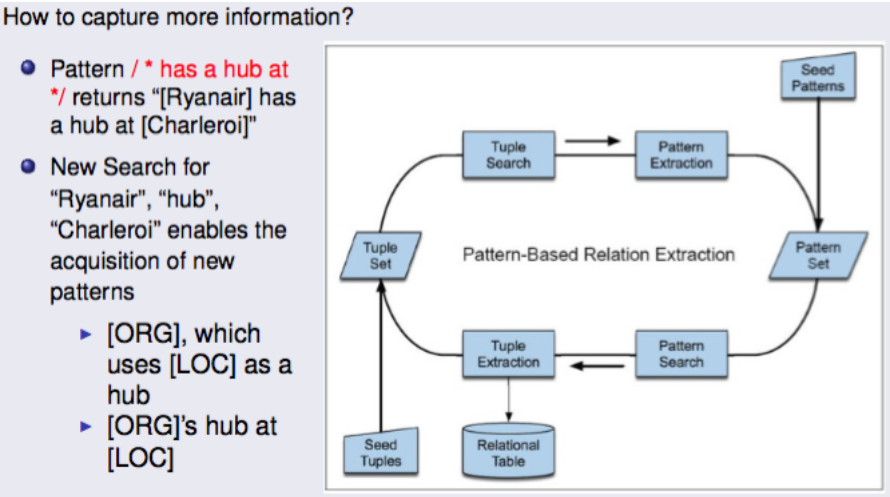


Figure 63: More information with semi-supervised learning.

10.3 Event extraction

Use stereotypical situation in the world. These situations can be characterized as scripts and scripts can be represented as template. The task is to fill the template.

There is somes steps: Named entities, phrase extraction, anaphora resolution (grouping same NE), pattern extraction.