Computational Linguistics: Introduction

Raffaella Bernardi

1. Admin

Conformemente allart. 13 del Regolamento Generale sulla Protezione dei Dati (UE) 2016/679, si avvisa il pubblico che levento in corso, organizzato dal CIMeC (Universitá di Trento) nell'ambito dei suoi fini istituzionali, sará trasmesso in streaming e audio/videoregistrato.

Chi non volesse apparire in audio o video deve tenere la telecamera ed il microfono del proprio dispositivo in mute. Partecipando all'evento si accetta quanto sopra descritto.

Il titolare del trattamento dei dati personali è lUniversità di Trento, con sede in via Calepina 14, 38122 Trento. I dati di contatto del Responsabile della Protezione Dati sono: rpd@unitn.it, Via Verdi 8, 38122 Trento.

2. Introduction

- **Modality**: Blended. We will have a mixture of in presence classes combined with online classes.
  - in presence blended classes: we will meet in Rovereto with those who can be phisically present and in zoom with those who cannot. The lectures will be registered and uploaded in Moodle afterwords for those who cannot attend the lecture at all.
  - online lectures: short video-lessons (uploaded in Moodle) and alive online meetings via zoom. The online syncronous meetings are scheduled late in the pm and will give the opportunity to meet with all the fellow students.
- **Office hours**: by appointment preferebly before or after classes.
- **Course Materials**: Slides, SLP text book, scientific papers
- **Text Book** D. Jurasfky and J. H. Martin **Speech and Language Processing**.
- **Url**: http://www.disi.unitn.it/~bernardi/Courses/CompLing/20-21.html

3. Rough Schedule

- 5 classes on **Syntax** (Sep-Oct): Formal Grammars of English and Parsing
- 11 classes on **Semantics** (Oct-Nov): Formal Semantics, Distributional Semantics, The Representation of Sentence Meaning, Syntax-Semantics interface
- 1 class on **Evaluation** methods and metrics (Nov)
- 3 classes on **Multimodal Models** (Nov): Language and Vision
- 6 classes on cutting-edge topics related to those discussed through the program: 5 **Reading Groups** and 1 **Guest Lecturer** (Sep-Dec)

4. Coordination wtih Luca Duccheschi's course

Luca Ducceschi and I have coordinated our courses so that with him you have hands-on experience on the concepts you study with me theoretically. To this end, we have organized the courses into blocks.

- 23-30 September: Luca
- 1 and 2 October: me and Luca, resp.
- 5-9 October: me (syntax)
- 12-16 October: Luca
- 19-29 (with the exception of the 21, 26, 28): me (semantics)
- 30 Oct-2 Nov: Luca
- 4 Nov-3 December: me (sytax-semantics, evaluation, language and vision.)

Please, check the web page of the courses for updated information and further details.

5. Goals

1. provide students with an **overview** of the field with focus on the syntax-semantics interface;
2. bring students to be aware on the one hand of several lexicalized **formal grammars**, on the other hand of **computational semantics models** and be able to combine some of them to capture natural language **syntax-semantics interface**;
3. **evaluate** several applications with a special focus to Interactive Question Answering and Language and Vision Models;
4. make students acquainted with **writing scientific reports**.

All these objectives will help students understand how methods from computer science, mathematics and statistics are applied to the modelling of natural language and start being propositive for new ideas.

6. Expected learning outcomes

At the end of the course students will be able to:

1. illustrate the main **challanges addressed in the field**, which are its consolidated results and which are the current research questions;
2. master, at introductory level, the basic rules of some **formal** grammars and of formal and distributional semantics languages and their integration based on the principle of compositionality;
3. compare **approaches** on computational linguistics tasks, in particular within interactive question answering and language and vision integration;
4. apply interdisciplinary approaches to linguistics tasks and write a **scientific report** on their research in `LaTex`.

7. Teaching Methods

We will have

1. frontal classes (online and in presence), pen-and-pencil exercises, discussions on papers lead by students (summary in LaTeX).
2. The exercises will help students better grasp the basic rules of lexicalized formal grammar and formal and distributional semantics and their integration.
3. Students will be individually supervised on a project of their choice, to be selected on the base of their background and interest, in one of the topics discussed during the frontal classes. (suggestion: decide about this in early November.)
4. Students will be supervised on the writing of a scientific report in LaTeX.
5. Students will present their project to their fellow students.
6. Through the courses, you will be given assignments (See the web page.)

8. Grading NB. I will revise this by next class

The final grade will be computed by the two grades below

1. **Assignments (xx %)**: quizz, comments about papers.
2. **Written Exam (xx%)**: exercises on Syntax, Semantics and their interface.
3. **Term paper (xx%)**: You are to complete a project on topics of your choice upon agreement with me. The term paper has to be sent by mail to me by the day of the written exam. During the last classes, we will have project proposals presentations.
4. **Term paper expectation** In the project report, students will show they are able to compare approaches to computational linguistics tasks. The term paper must present an open problem in the CL field, review the relevant SoA and describe a proposal to address the problem or report about a project on it. It is meant to verify that students are able to read and understand technical works in computational linguistics, and to apply the relevant knowledge in a critical manner, showing they have learned how to reason in an interdisciplinary setting and write a scientific report in LaTeX.

9. Your background

Your background:

- BSc in Comp. Science vs. Humanities?
- Logic (PL?, FoL?)?
- Formal Semantics, Distributional Semantics (Vector Space Semantics)?
- Programming skills? Python?
- LaTeX?

Pool: http://etc.ch/WssB

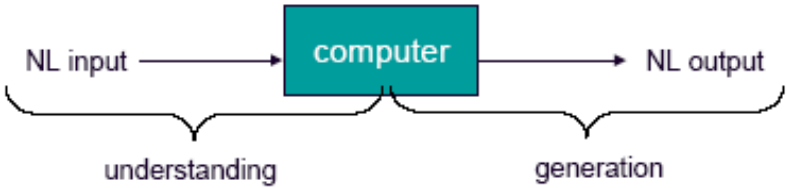
10. What do you know/think of Computational Linguistics

- What do you think is Computational Linguistics?
- Which disciplines are involved?
- Why do you think people are interested in CL?
- Why are you interested in CL?

https://answergarden.ch/share/1404021

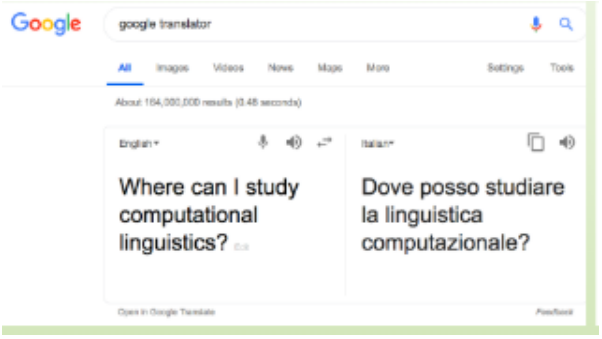
11. Goals of Computational Linguistics

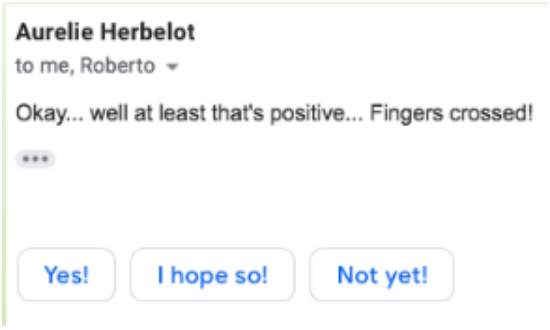
- **Ultimate goal**: To build computer systems that perform as well at using natural language as humans do.
- **Immediate goal** To build computer systems that can process text and speech more intelligently.

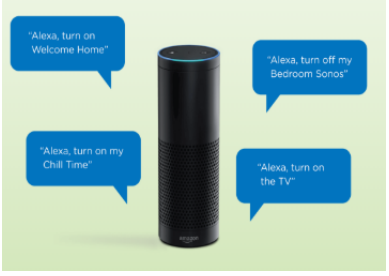


where, NL (**Natural Language**) is the language that people use to communicate with one another and **process** means to analyze.

12. Quite a lot has been reached








AT&T
10:35 AM
“ Call me an ambulance ”
From now on, I’ll call you ‘An Ambulance’. OK?
Cancel
Yes

13. Why computational models of NL

There are two motivations for developing computational models:

- **Scientific**: To obtain a **better understanding** of how language works. Computational models may provide very specific predictions about human behavior that can then be explored by the psycholinguist.
- **Technological**: natural language processing capabilities would revolutionize the way computers are used. Computers that could understand natural language could access to all human knowledge. Moreover, **natural language interfaces** to computers would allow complex systems to be accessible to everyone. In this case, it does not matter if the model used reflects the way humans process language. It only matters that it works.

We are interested in **linguistically motivated computational models** of language understanding and production that can be shown to perform well in specific example domains.

14.1. Ambiguity: Phonology

Phonology: It concerns how words are related to the sounds that realize them. It's important for speech-based systems.

1. "I scream"
2. "ice cream"

14.2. Ambiguity: Morphology

Morphology: It's about the inner structure of words. It concerns how words are built up from smaller meaning-bearing units.

1. Unionized (characterized by the presence of labor unions)
2. un-ionized in chemistry

14.3. Ambiguity: Syntax

Syntax: It concerns sentence structure. Different syntactic structure implies different interpretation.

1. I saw the man with the telescope
   - [I[[saw]$_v$[the man]$_{np}$[with the telescope]$_{pp}$]$_{vp}$]$_s$ [(I have the telescope)]
   - [I[[saw]$_v$[[the man]$_{np}$[with the telescope]$_{pp}$]$_{np}$]$_{vp}$]$_s$ [(the man has the telescope)]
2. Visiting relatives can be tiring.

14.4. Ambiguity: Semantics

Semantics: It concerns what words mean and how these meanings combine to form sentence meanings.

1. Visiting relatives can be tiring.
2. Visiting museums can be tiring.

Same set of possible syntactic structures for this sentence. But the meaning of museums makes only one of them plausible.

14.5. Ambiguity: Discourse

Discourse: It concerns how the immediately preceding sentences affect the interpretation of the next sentence

1. **Merck & Co.** formed **a joint venture** with **Ache Group**, of Brazil. **It** will . . .?
2. Merck & Co. formed a joint venture with Ache Group, of Brazil. **It**$_i$ will be called Prodome Ltd.

   (**a joint venture!**$_i$)
3. Merck & Co. formed a joint venture with Ache Group, of Brazil. **It**$_i$ will own 50% of the new company to be called Prodome Ltd.

   (**Merck & Co.**$_i$!)
4. Merck & Co. formed a joint venture with Ache Group, of Brazil. **It**$_i$ had previously teamed up with Merck in two unsuccessful pharmaceutical ventures.

   (**Ache Group**$_i$!)

The difficulty

can't, $22.50, New York, so-called, a priori
Buy, bought, bought, cat, cats

The old train...
(“(The old train) left (the station)") vs (“(The old)train (the young)").

I know more beautiful women than Julia Roberts.
"I know women more beautiful than Julia Roberts" or "I know more beautiful women than Julia Roberts does".

The boy sees the girl on the hill with the telescope

How to split sentences into words?

How to link words together?

Who knows what?

How to make sense of a phrase/how to group words together?

Mary takes a glass of wine

Mary takes a gym course

Mary takes a husband

actor

experiencer

?

16. NLP Systems

1. Tokenization [Luca]
2. PoS tagging [Luca]
3. Morphological analysis
4. Shallow parsing
5. Deep parsing
6. Semantic representation (of sentences)
7. Discourse representation

Tokenization It consists in dividing the sequence of symbols in minimum units called tokens (words, date, numbers, punctuation etc..). Many difficulties: e.g.

Sig. Rossi vs. 05.10.05 vs. www.unitn.it;

given up (multi words 1 token).

17. Words: Classes

Traditionally, linguists classify words into different categories:

- **Categories**: words are said to belong to *classes*/categories. The main categories are nouns (n), verbs (v), adjectives (adj), articles (art) and adverbs (adv).

The class of words can be divided into two broad supercategories:

1. **Closed Class**: Those that have relatively fixed membership. E.g. prepositions, pronouns, particles, quantifiers, coordination, articles.
2. **Open Class**: nouns, verbs, adjectives, adverbs.

18. Words: Classes (Cont'd)

A word in any of the four open classes can be used to form the basis for a phrase. This word is called the **head** of the phrase and indicates the type of thing, activity, or quality that the phrase describes. E.g. "dog" is the head in: "The dog", "the small dog", "the small dog that I saw".

- **Constituents**: Groups of categories may form a single *unit or phrase* called constituents. The main phrases are noun phrases (*np*), verb phrases (*vp*), prepositional phrases (*pp*). Noun phrases for instance are: "she"; "Michael"; "Rajeev Goré"; "the house"; "a young two-year child".

Tests like substitution help decide whether words form constituents.

Can you think of another test?

18.1. Applications of PoS tagging

More recently, linguists have defined classes of words, called **Part-of-Speech** (PoS) tagsets with much larger numbers of word classes. PoS are used to label words in a given collection of written texts (Corpus). These labels turn out to be useful in several language processing applications.

- **Speech synthesis**: A word's PoS can tell us something about how the word is pronounced. E.g. "content" can be a noun or an adjective, and it's pronounced differently: CONtent (noun) vs. conTENT (adjective).
- **Information Retrieval**: A word's PoS can tell us which morphological affixes it can take, or it can help selecting out nouns or other important words from a document.
- **Theoretical Linguistics**: Words' PoS can help finding instances or frequencies of particular constructions in large corpora.

19. Morphology

Morphology is the study of how words are built up from smaller meaning-bearing units, morphemes. It concerns the inner structure of words.

For instance,

- **fog**: it's one morphem
- **cats**: it consists of two morphemes: cat + -s.

19.1. Morphemes

Morphemes are divided into:

1. **stems**: they are the main morpheme of the word, supplying the main meaning.
2. **affixes**: they add additional meanings of various kinds. They are further divided into:
   - **prefixes**: precede the stem (English: unknown= un + known)
   - **suffixes**: follow the stem (English: eats= eat + -s)
   - **circumfixes**: do both (German: gesagt (said)= ge + sag + t)
   - **infixes**: are inserted inside the stem (Bontoc -Philippines - fikas (strong), fumikas (to be strong))

A word can have more than one affixes (e.g. re+write+s, unbelievably= believe (stem), un-, -able, -ly).

19.2. Ways of forming new words

There are two basic ways used to form new words:

1. **Inflectional forms**: It is the combination of a word stem with a grammatical morpheme, usually resulting in a word of the **same class as the original stem**, and usually filling some syntactic function like agreement. E.g. in English,

   past tense on verbs is marked by the suffix "-ed", form by "-s", and participle by "-ing".

2. **Derivational forms**: It is the combination of a word stem with a grammatical morpheme, usually resulting in a word of a **different class**, often with a meaning hard to predict exactly. E.g.

   Adverbs from noun: friendly from friend.

   Noun from verbs: killer from kill.

   Adjectives from nouns: "computational" from "computation", "unreal" from "real".

20. Computational Morphology

We want to build a system able to provide the stem and the affixes given a word as input (e.g. cats $\rightarrow \{\text{cat} + N + PL\}$), or able to generate all the possible words made of a given stem (e.g. cat $\rightarrow \{\text{cats}, \text{cat}\}$). To this end, we first of all need to have a way to formally represent Morphology Theory studied by Linguists.

20.1. Modules

To build a morphological recognizer/generator, we'll need at least the following:

lexicon: the list of stems and affixes, together with basic information about them (e.g. Noun stem or Verb stem).

morphotactics: the model of the morpheme ordering, e.g. English plural morpheme follows the noun rather than preceding it.

orthographic rules: spelling rules used to model the changes that occur in a word, e.g. city becomes cities, i.e. "y" $\rightsquigarrow$ "ie".

20.2. The Lexicon and Morphotactics

Lexicon: It's a repository of words. Having an explicit list of every word is impossible, hence the lexicon is structured with a list of each of the stems and affixes of the language.

Morphotactics: One of the most common way to model morphotactics is by means of **Finite State Automata** (FSA).

21. FSA for Morphology Recognition/Generation

We have said that a language is a set of strings. An important operation on strings is concatenation.

- At syntactic level, strings are words that are concatenated together to form phrases.
- At morphological level, strings are morphemes that are concatenated to form words. E.g.

$$\begin{array}{ll} \text{Stem Language:} & \{work, talk, walk\}. \\ \text{Suffix Language:} & \{\epsilon, -ed, -ing, -s\}. \end{array}$$

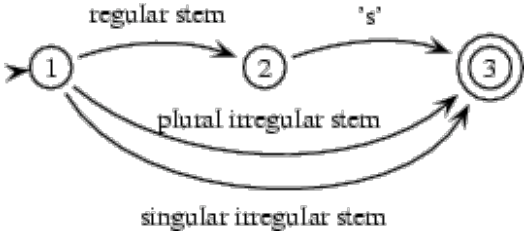
The concatenation of the Suffix language after the Stem language, gives:

$\{work, worked, working, works, talk, talked, talking, talks,$
$walk, walked, walking, walks\}$

21.1. FSA for English Inflectional Morphology

Let's build an FSA that recognizes English nominal inflection. Our lexicon is:

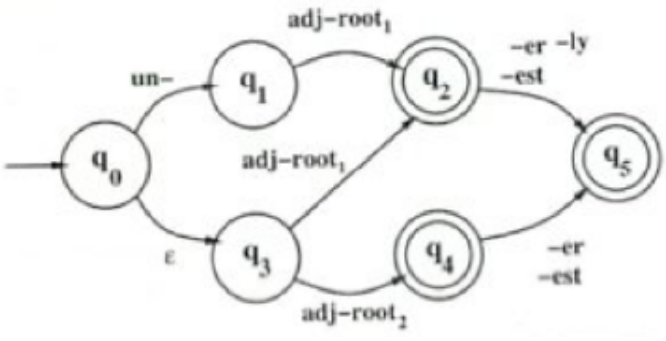
reg-stem	**plural**	**pl-irreg-stem**	**sing-irreg-stem**
fox	-s	geese	goose
cat		sheep	sheep
dog		mice	mouse



21.2. FSA for English Derivational Morphology

Let's build an FSA that recognizes English adjectives. Our lexicon is:

adj-root1	**adj-root2**	**Suffix-1-2**	**Suffix-1**	**Affix-1**
clear	big	-er	-ly	un-
happy	cool	-est		
real				



22. Background notions: Formal Language & FSA

- A formal language is a set of strings. E.g. $\{a, b, c\}$, $\{the, student, \}$ or $\{student, -s\}$.
- Strings are by definition finite in length.
- The language accepted (or recognized) by an FSA is the set of all strings it recognizes when used in recognition mode.
- The language generated by an FSA is the set of all strings it can generate when used in generation mode.
- The language accepted and the language generated by an FSA are exactly the same.
- FSA recognize/generate Regular Languages [you will use RL with Luca].

22.1. Finite State Automata

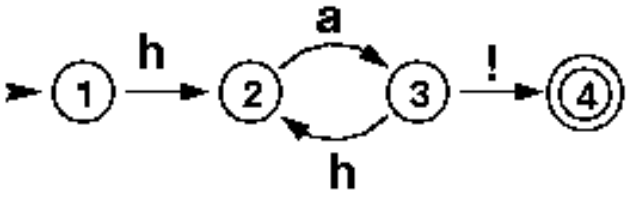
A finite state **generator** is a simple computing machine that **outputs** a sequence of symbols.

It starts in some **initial state** and then tries to reach a **final state** by making transitions from one state to another.

Every time it makes such a transition it emits (or writes or generates) a symbol.

It has to keep doing this until it reaches a **final state**; before that it cannot stop.

22.1.1. FSA as directed graph Finite state generators can be thought of as **directed graphs**. And in fact finite state generators are usually drawn as directed graphs. Here is our laughing machine as we will from now on draw finite state generators:



The nodes of the graph are the states of the generator. We have numbered them, so that it is easier to talk about them. The **arcs** of the graph are the transitions, and the **labels** of the arcs are the symbols that the machine emits. A double circle indicates that this state is a final state and the one with the black triangle is the start.

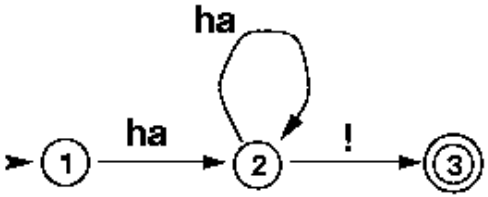
22.1.2. Finite State Recognizer Finite state recognizers are simple computing machines that **read** (or at least try to read) **a sequence of symbols** from an input tape. That seems to be only a small difference, and in fact, finite state generators and finite state recognizers are exactly the same kind of machine. Just that we are using them to output symbols in one case and to read symbols in the other case.

An FSA recognizes (or accepts) a string of symbols if starting in an intial state it can read in the symbols one after the other while making transitions from one state to another such that the transition reading in the last symbol takes the machine into a final state.

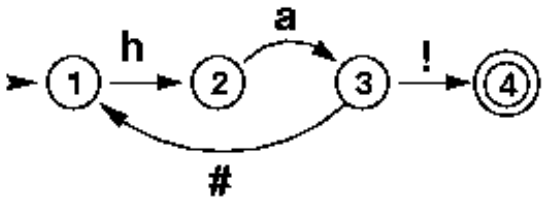
That means an FSA **fails** to recognize a string if:

- it cannot reach a final state; or
- it can reach a final state, but when it does there are still unread symbols left over.

22.1.3. Finite State Automata Try to think of what language is recognized or generated by the FSA below.



22.1.4. Finite State Automata with jumps



It has a strange transition from state 3 to state 1 which is reading/emitting `#`. We will call transitions of this type **jump arcs** (or ϵ transitions). Jump arcs let us jump from one state to another **without emitting or reading a symbol**. So, `#` is really just there to indicate that this is a jump arc and the machine is not reading or writing anything when making this transition.

This FSA accepts/generates the same language as our first laughing machine, namely sequences of `ha` followed by a `!`. Try it yourself.

22.1.5. Important properties of FSA

- All in all, finite state generators can only have a **finite number** of different **states**, that's where the name comes from.
- Another important property of finite state generators is that **they only know the state they are currently in**. That means they cannot look ahead at the states that come and also don't have any memory of the states they have been in before or the symbols that they have emitted.
- An FSA can have several intial and final states (it must have at least one initial and one final state, though).

22.2. Regular Languages

Recall: V^* denotes the set of all strings formed over the alphabet V. A^* denotes the set of all strings obtained by concatenating strings in A in all possible ways.

Given an alphabet V,

1. $\{\}$ is a regular language
2. For any string $x \in V^*$, $\{x\}$ is a regular language.
3. If A and B are regular languages, so is $A \cup B$.
4. If A and B are regular languages, so is AB.
5. If A is a regular language, so is A^*.
6. Nothing else is a regular language.

Examples For example, let $V = \{a, b, c\}$. Then since aab and cc are members of V^* by 2, $\{aab\}$ and $\{cc\}$ are regular languages. By 3, so is their union, $\{aab, cc\}$. By 4, so is their concatenation $\{aabcc\}$. Likewise, by 5 $\{aab\}^*$ $\{cc\}^*$ are regular languages.

22.2.1. Pumping Lemma For instance, a non-regular language is, e.g., $L = \{a^n b^n \mid n \geqslant 0\}$. More generally, FSA cannot generate/recognize balanced open and closed parentheses.

You can prove that L is not a regular language by means of the **Pumping Lemma**.

Roughly note that with FSA you cannot record (**no memory**!) any arbitrary number of a's you have read, hence you cannot control that the number of a's and b's has to be the same. In other words, you cannot account for the fact that there exists a **relation of dependency** between a^n and b^n.

23. Recognizers vs. Parsers

We have seen that we can give a word to a recognizer and the recognizer will say "yes" or "no". But often that's not enough: in addition to knowing that something is accepted by a certain FSA, we would like to have an explanation of why it was accepted. Finite State Parsers give us that kind of explanation by returning the sequence of transitions that was made.

This distinction between recognizers and parsers is a standard one:

- **Recognizers** just say "yes" or "no", while
- **Parsers** also give an analysis of the input (e.g. a parse tree).

This distinction does not only apply to FSA, but also to all kinds of machines that check whether some input belongs to a language and we will make use of it throughout the course.

23.1. Morphological Parsers

The goal of morphological parsing is to find out what morphemes a given word is built from. For example, a morphological parser should be able to tell us that the word "cats" is the plural form of the noun stem "cat", and that the word "mice" is the plural form of the noun stem "mouse".

So, given the string "cats" as input, a morphological parser should produce an output that looks similar to {cat N PL}.

24. What are FSA good for in CL?

Finite-state techniques are widely used today in both research and industry for natural-language processing. The software implementations and documentation are improving steadily, and they are increasingly available. In CL they are mostly "lower-level" natural language processing:

- Tokenization
- Spelling checking/correction
- Phonology
- Morphological Analysis/Generation
- Part-of-Speech Tagging
- "Shallow" Syntactic Parsing

Finite-state techniques cannot do everything; but for tasks where they do apply, they are extremely attractive. In fact, the flip side of their **expressive weakness** being that they usually behave very well computationally. If you can find a solution based on finite state methods, your implementation will probably be **efficient**.

25. Conclusion

- Today we have introduced CL
- Next classes with Luca on tokenization and regular expressions.
- With me: On Thursday 1st of October, 18:00-19:30 online Reading Group;
- On Monday 5th of October we will start the block of classes on Syntax

Next Assignment GP1: in presence-students and GP2: by remote-students.

- Read the paper by Tenney et al ACL 20219 (GP1) and by Jawahar et al ACL 2019 (GP2),
- annotate it with questions and comments -by the 28.09.2020 in https://perusall.com/. I will give you access to it through Moodle.
- Prepare a summary for the other group to be presented in the online-class (1st of October).