Introduction to session 3, linguistic approaches

- Pierre-Edouard Portier, MCF
- INSA de Lyon, Laboratoire LIRIS, équipe DRIM
imbecile

/ 'imbisisl/ 

noun informal
noun: imbecile; plural noun: imbeciles

1. a stupid person.
synonyms: fool, idiot, cretin, moron, dolt, halfwit, ass, dunce, dullard, simpleton, nincompoop, blockhead, ignoramus, clod; More
informal dope, thickhead, ninny, chump, dimwit, dummy, dum-dum, dumb-bell, jackass, bonehead, fathead, numbskull, dunderhead, airhead, pinhead, lamebrain, pea-brain, birdbrain, dipstick, donkey, noodle;
informal nit, nitwit, twit, numpty, clot, muppet, plonker, berk, prat, pillock, wally, wazzock, divvy;
informal bozo, turkey, goofus;
vulgar slang knobhead;
vulgar slang asshat
"I'd have to be an imbecile to do such a thing"

antonyms: genius

adjective
adjective: imbecile

1. stupid; idiotic.
"try not to make imbecile remarks"

Origin

LATIN

in-

LATIN
imbecillus
without a supporting staff

LATIN
baculum
stick, staff

FRENCH
imbecile
physically weak

mid 16th century (as an adjective in the sense 'physically weak'): via French from Latin
imbecillus, literally 'without a supporting staff', from in- (expressing negation) + baculum 'stick, staff'. The current sense dates from the early 19th century.
References

- Natural Language Processing of Semitic Languages
Computational linguistics for non western European languages

- Phonemes $\rightarrow$ (orthographic) words with problems of euphony, etc.
- Words $\rightarrow$ segmented tagged sentences morphological formation rules
- Sentences $\rightarrow$ phrasal constituents syntactic analysis
- Phrasal constituents $\rightarrow$ meaning anaphora, discourse analysis
Morphological processing

- Necessary for later tasks (information retrieval, question answering, text summarization, machine translation, **text reuse detection**, …)
- Most of the successful POS tagging approaches are based on machine learning (HMM, Average Perceptron, Maximum Entropy, SVM, …)
- However, languages with rich morphological rules (e.g. Semitic languages, Sanskrit, …) make it inefficient to learn the rules from a training corpora.
The additional complexity for the morphological processing of semitic languages comes mainly from the **non-concatenative rules**.

E.g., in Hebrew the **root** `p.t.x` denoting a notion of opening, combines with the **pattern** `maCCeC` denoting tools and instruments to yield “key”.

Moreover, classical bounded morphemes working as affixes still apply (derivational and inflectional morphology)...
The additional complexity for the morphological processing of Sanskrit comes from the specification of relationships by inflectional and derivational morphology (instead of using position as in western European languages), and from rich prosodic changes (e.g., vidyA ("knowledge") + Apyate ("is attained") → vidyApyate).

Positional grammars and constituency parsers become irrelevant, dependency parsers have to be used.
Morphological processing

- The morphological rules for word formation and sentence formation are not one-one, introducing non-determinism (necessary (?) for poetry...).
The common approach is to derive a tagger (i.e., an interpreter for finite state machine) from a structured lexicon (i.e., a repository of grammatical information) by using another finite state automaton (usually called a transducer).

A transducer is a finite state machine with two tapes (input and output) that computes a relation between two formal languages (i.e. sets of strings).

Transducers can be composed:
Words → words split up at morpheme boundaries
Split up words → morphemes description
If I were to work on computational linguistics, I would be tempted to explore the work of Gérard Huet.

Gérard Huet developed the Zen toolkit library in OCaml for computational linguistics introducing a generic data structure (decorated lexical trees) for deriving taggers from lexicon structures.

A very well founded approach that has been applied to Sanskrit (Héritage du Sanskrit, Dictionnaire sanskrit-français).

How would this approach apply to semitic languages?

How to combine it with machine learning approaches?