TAR

2. Basics of Natural Language Processing

TAR-02-NLP.pdf

NLP pipeline

- Sentence segmentation: finding boundaries of sentences in text
    - Often done heuristically, using regular expressions
    - Best performance with supervised machine learning models (prediction of full stop denotion)
- Tokenization: breaking a text up into tokens - words and other meaningful elements
    - tokens are words, punctuation marks, and special characters
    - rule-based (i.e., heuristic) vs. supervised approaches

Morphology

- Morphology: Branch of linguistics concerned with the internal structure of words. Words are made up of **morphemes** (= smallest linguistic pieces with a grammatical function).
    - Inflectional morphology: creating word-forms that express grammatical features
        - fish → fishes
    - Derivational morphology: creating new words from existing ones
        - fish → fishery
    - Compounding: combine two or more existing words
        - sky+scraper

  In IR (Information Retrieval): The query house should match against document talking about houses and maybe housing (but probably not about housewifes)
  In information extraction: If money laundries or money laundry appear in the text, we'd like to extract a keyphrase money laundering
  For syntax and semantics: We need to know the grammatical features of a word: Ana voli Ivana is not the same as Anu voli Ivan
  Many other IR/TM tasks: We simply want to count house, houses and housing as the same thing
- Stemming: reduction of word-forms to stems
        - adjustments → adjust, defensible → defens, revivals → reviv
    - Typically by **suffix stripping** plus some extra steps and checks
    - Pros: simple and efficient
    - Cons:
        - prone to **overstemming** and **understemming** errors
        - difficult to design for morphologically complex languages
        - imprecise (don't differentiate between inflection and derivation)

- Porter stemmer: Popular suffix-stripping stemmer
    - Each word can be represented as $[C](VC)^m[V]$, where C is a sequence of consonants and V is a sequence of vowels
    - Each word has a measure m:
        - m = 0 tr, ee, tree, by
        - m = 1 trouble, oats, trouble, trees, ivy
        - m = 2 troubles, troubles, private
    - Suffix stripping rules: (condition) S1 → S2
        - (m > 1) EMENT →
        - (m>0) ALIZE → AL
        - (m>0) TIONAL → TION
        - (m>1 and (*S or *T)) ION →
    - A cascade of 5 suffix removal steps:
        - Step 1 deals with plurals and past participles
        - Step 2-4: derivation
        - Step 5: tidying up
    - Cons: Porter stemmer occasionally **overstems** (university/universe) and **understems**. Still it works fine in most cases and is very useful in IR applications.

- Lemmatization: Transformation of a word-form into a linguistically valid base form, called the **lemma**
    - nouns → singular nominative form
    - verbs → infinitive form
    - adjectives → singular, nominative, masculine, indefinitive, positive form

  A much more difficult task than stemming, especially for morphologically complex languages, for which you basically need:
    - a **morphological dictionary** that maps word-forms to lemmas
    - a **machine learning model**, trained on a large number of word-lemma pairs

---

Syntax

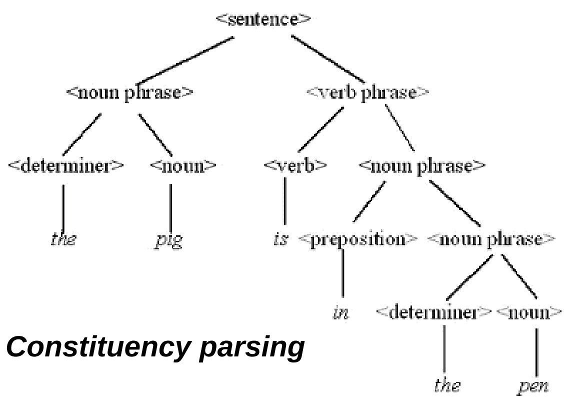
- Language modelling
    - Probabilistic models of text:
        - determine the probability of a word sequence
        - determine a likely candidate for the next word in a sequence
    - We'd like to compute the probability
        - $P(\omega_1,\ \omega_2, \dots ,\ \omega_{n-1},\ \omega_n)$
    - This can be rewritten using the chain rule
        - $P(\omega_1^n) = P(\omega_1)P(\omega_2|\omega_1) \dots\ P(\omega_n|\omega_1^{n-1})$
    - We need a way of estimating these probabilities
    - **Text corpus** (plural: corpora): large and structured set of texts, used for **corpus linguistic analyses** and for the development of **natural language models** (primarily machine learning models)
        - May be manually annotated:
            - e.g., POS-annotated or parsed corpora (tree bank)
            - possibly at different levels (multi-level annotated)

    - **Zipf's law** states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table
        - **Happax legomena** (wiki: A word that occurs only once within a context, either in the written record of an entire language, in the works of an author, or in a single text) account for ~50% of the words in corpus
    - Even short sequences of 5-6 words would barely ever appear in a large corpus
    - Solution: approximate each full conditional
        - e.g. approximate $\mathrm{P(rabbit \mid The\ other\ day\ I\ saw\ a\ white\ fluffy)}$ by computing $\mathrm{P(rabbit \mid white\ fluffy)}$ instead
        - Now we can use counting to compute the required probability
            - in the above example we would divide the frequency of **white fluffy rabbit** by the frequency of **white fluffy** in a large corpus.
- Parts-of-speech (POS) explains not what the word is, but how it is used. Universal parts across languages:
    - **Verbs** assert something about the subject of the sentence and express actions, events, or states of being
    - **Nouns** are words that we used to name a person, an animal, a place, a thing, or an abstract idea
    - **Adjectives** modify nouns and pronouns by describing, identifying, or quantifying them.
    - **Pronouns** replace nouns or another pronouns and are essentially used to make sentences less cumbersome and less repetitive
    - **Adverbs** modify a verb, an adjective, another adverb, a phrase, or a clause. An adverb indicates manner, time, place, cause, . . .
    - **Prepositions**, conjunctions, . . .

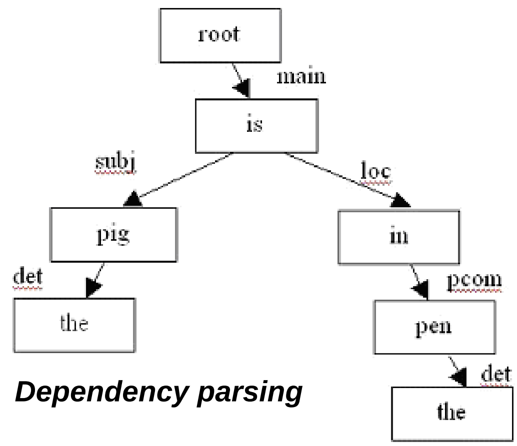
    - POS tagging (grammatical tagging, word-category disambiguation) is the process of marking up a word in a text as corresponding to a particular part of speech

        - POS taggers assign tags from a finite predefined **tagset**
        - State-of-the-art POS taggers are supervised machine learning models

- Parsing is the task of analyzing the grammatical structure of natural language sentences
    - Given a sequence of words, a parser forms units like subject, verb, object and determines the relations between them according to some **grammar formalism**
    - Two types of parsers
        - **Constituency parsers/phrase structure tree (PST) parsers** - based on constituency/PS grammars
        - **Dependency parsers** - based on dependency grammars
    - **Constituency parser** produces a tree that represents the syntactic structure of a sentence (i.e., a break down of the sentence)
        - Words appear only as leaves of the tree:
    - **Dependency parsing** represents the structure of the sentence as the tree of syntactic dependencies between pairs of words
        - Each dependency relation has a governing word and a dependent word
        - Verb is the syntactic center of the clause, all other words directly or indirectly dependent on the verb



Constituency parsing

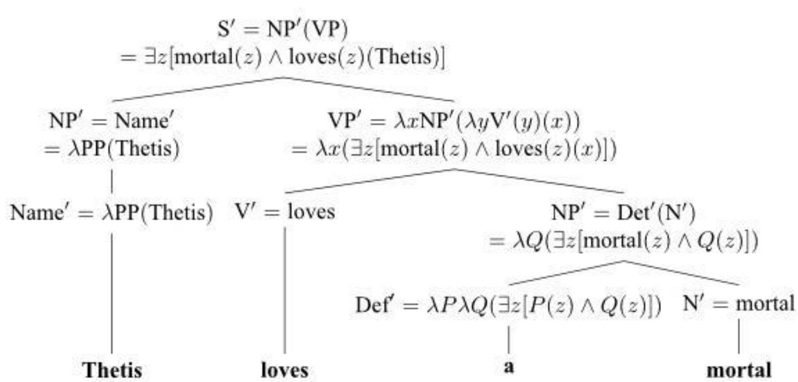


Dependency parsing

- <u>Shallow parsing</u> (chunking, “light parsing”) only identifies the constituents (noun phrases, verbs phrases, prepositional phrases, etc.) but does not specify their internal structure nor their role in the sentence
  - <u>Example:</u> [NP Jack and Jill] [VP went] [ADVP up] [NP the hill] [VP to fetch] [NP a pail] [PP of] [NP water].

Semantics

- <u>Semantics</u>:
  - **Sentence-level semantics**: representing the meaning of a sentence comprised of the meaning of its parts
  - **Discourse-level semantics**: meaning of text that goes beyond a single sentence (anaphors, discourse structures)
  - **Word-level (lexical) semantics**: meaning of words and how they relate to each other, verb and event semantics, distributional semantics…
  - **Computational semantics** is the study of how to automate the process of constructing and reasoning with meaning representations of natural language expressions.
  - **Formal semantics**: Traditional approach to natural language semantics, focused at sentence-level and discourse-level semantics. **Cons:** Falls short of representing the meaning of the individual words
  - **Montague grammar**: based on predicate logic and lambda calculus, constructs predicate formulas based on parse trees:



    - **Distributional semantics:** Representation of word meaning based on **distributional hypothesis**
        - correlation between similarity of words' contexts and words' semantic similarity
        - Words represented as vectors of context features obtained from corpus
        - Semantic similarity predicted via vector similarity
        - **Distributional semantic models** are most useful for finding semantically similar and related words

- WordNet: Manually constructed lexical database, contains nouns, verbs, adjectives and adverbs. Words are organized into synsets - sets of words with the same sense. For each synset WordNet provides:
    - a list of words that can be used in that sense
    - a gloss - a short description of the sense
    - semantic relations to other synsets (hyponymy, meronymy, …)
      Nouns:
    - **Hyperonymy/hyponymy** - IS-A relation (chair - furniture)
    - **Meronymy** - a part of whole relation (finger - hand)
    - **Antonymy** - opposite meaning (wet - dry)
    - **Similarity** - similar (but not identical) meaning (warm - hot)
      Verbs:
    - Troponymy - increasingly specific manner of an event (communicate - talk - whisper, move - jog - run)
    - Entailment - one word entails the other (succeed - try, buy - pay)

    - Similarity (S) = pragmatic relation, Relatedness (R) = syntagmatic relation:
        - airplane - machine (S), airplane - engine (S), pilot - airplane (R), magic - disappear (R), rich - caviar (R)
- Word sense disambiguation (WSD): the task of identifying which sense (meaning) of a word is used in a sentence, when the word has multiple meanings:
    - The newspaper fired the editor. (The company/organization)
    - John spilled coffee on the newspaper. (The physical newspapers)

    - WSD addresses both polysemy and homonymy
        - Polysemy - a word has multiple, related meanings, e.g. ring (wedding ring vs. boxing ring)
        - Homonymy - two, unrelated words, have the same form, e.g., saw (past tense of see vs. a tool)
        - The more fine-grained the senses, the more difficult the disambiguation task