NLP

0.1 - Introduction

`Motivation for NLP`:

- Voice assistants (Siri, Alexa)
- Automatic translation
- Chatbots
- Analysis of user feedback and classification

`Computational Linguistics` - application of theories and models about language (grammars, knowledge, models) to computer systems that use them to understand human language. It studies language using algorithms that can run on a computer

`Language description`:

- phonetics - physical properties of speech sounds
-
    - Which sounds (phonemes) can be represented in writing, how can they be recorded. Phonetics studies the factors and componenets of linguistic sounds.
- syntax - structural properties of sentences
-
    - What structures do we have? Can we specify grammars that generate language? (Chomsky CFG). Can we specify grammars to recongnize language?
- semanctics - meaning of sentences
-
    - What meaning can we assign to language?. Meaning theory, is called the theory of the meaning of signs. Signs are words, phrases, or symbols. Deals with the relationships between signs and meanings of those signs.
- pragmatics - how sentences are interpreted in context, how the language is used, intentions of utterance
-
    - use of language, as opposed to semantics, which focuses on the context-independent meaning of the words and the truth conditions of the words.

`Language forms`:

- spoken (speech), application: speech recognition, synthesis
- written (text)
-
    - Analysis results: indexes, dictionaries, text and language statistics
-
    - Application: translation, expert system interfaces, information extraction, model building, grammar theories, semantic capture.

`Theories and models`:

- Words:
-
    - structure of human dictionary
-
    - structure of the language memory
-
    - formation of words and their inflection (morphology)
-
    - meaning of words

- Sentences:
-
  - structure of sentences
-
  - relationship between the sentence parts
-
  - function of the parts of sentences
-
  - relationship of the sentences to each other

`Methods and tools`:

- Corpora:
-
  - Language corpora (collections of recorded utterances)
-
  - Text and discourse corpora (collections of written texts)

Corpora contain explicit inguistic information, such as:

- Content categorization (genres)
- Labelling the linguistic phenomena contained (for example, parts-of-speech or word types)

`Knowledge representation on the computer`:

- dictionaries, grammars, probabilistic models, knowledge bases, world knowledge
- programs for syntatic, semantic, pragmatic analysis (parses)
- synthesis programs (generators)

`Application-oriented research`:

- Error correction (spelling and grammar)
- Machine speech recognition and control (ViaVoice, dialog systems, ...)
- Machine speech synthesis (car navigation, e-mail reading, ...)
- Machine translation
- Handwriting recognition
- Fact search in documents
- Text evaluation
- Support for physicians during research
- Marketing analysis (analysis of user opinions and preferences) in blogs,
- forums, news/user groups
- "Voice understanding", voice commands (Siri, Alexa, IBM Watson, ...).

0.2 - History

`Formal language theory history`:

- Turing (automata theory)
- Kleene (regex)
- Chomsky (finite automata)
- Backus and Naur (CFG for algol)

`Speech recognition history`:

- Shannon (noisy channel, entropy)
- Bell labs (for digits, single speaker)

`Symbolic paradigm`:

- based on formal languages, generative syntax, parsers
- includes AI with logic and deduction
- search pattern based with keyword searches

`Stochastic paradigm`:

- calculating letter probabilities
- bayesian methods

`NLP encompasses different fields`:

- Computational Linguistics in Linguistics
- Natrual Language Processing in Computer Science
- Speech Recognition in Electrical Engineering
- Computational Psycholinguistics in Psychology

`Foundational Insights: 1940s-50s`:

`Two foundational paradigms`:

- Automaton
- Probabilistic or Information-theoretic models

`McCulloch-Pitts Neuron` - Simplified model of the neuron as a kind of element that could be described as in terms of propositional logic and then to work of Kleene and on finite automata and regexes.

`Formal Language Theory`:

- Chomsky context-free grammars

`Probabilistic Algorithms for NLP`:

- Noisy channel model
- Decoding for the transmission of information

`Entropy` - can be used to measure the information capacity of a channel

`The Two Camps: 1957-70`:

`Two paradigms of speech recognition`:

- Symbolic
- Stochoastic

`Symbolic language 2 lines of research`:

- Chomsky on formal language theory and generative syntax
-
    - `TDAP` - Earliest complete parsing system
- Artifical Intelligence

- Limited to narrow domains, more search pattern based with keyword searches and simple heuristics for deductions and answering questions.

`Stochastic mostly took departments of`:

- Statistics
- Electrical Engineering
- Calculates with letter/word probabilities

`Bayesian system` - used for text recognition that was used a large dictionary to compute the likelihood of each observed letter by multiplying the likelihood of each letter in the word.

`Four Paradigms: 1970 - 1983`:

`Stochastic`:

- Hidden Markov Models
- Metaphors of noisy channel and decoding

`Logic-based`:

- Q-Systems and metamorphosis grammars
- Forerunners of Prolog
- Definitne Clause Grammars
- Logicla facts, implications and inference rules lead to solutions

`Natural Language Understanding`:

- SHRDLU - attempt to build extensive grammar for English
- Network-based semantic
- Case roles
- LUNAR question-answering system

`Discourse Modelling`:

- automatic reference resolution

`Four key areas`:

- Study of substructure in discourse

`BDI` - Belief, Desire, Intention

`Empiricism and Finite-State Models Redux: 1983 - 1993`:

`Two classes of model which were brought back after Skinner's Verbal Behaviour was doubted by Chomsky`:

- Finite-State models (received attention after work on finite-state phonology and morphology)
- Return of Empiricism (Rise of probabilistic models in speech and language processing)

Church invented finite automata for modeling syntax

`The Field Comes Together: 1994-1999`

Data-driven approaches are also increasingly found in tagging, parsing, ambiguity resolution and semantics.

`Current research area is formed: 1999 - 2008`:

- Probabilistic and data-driven models are standard
- Parsing, labeling, reference resolution uses probabiilties and empirical evaluation

`The rise of ML: 1994 - 2013`:

- Large amounts of material (both sound and text) were published (corporas, benchmarks and contests for parsing, information extraction, answering questions)
- Machine learning increasingly in focus
-   - Support Vector Machines (SVM)
-   - Maximum Entropy Techniques
- High Performance Computing (HPC) more and more widespread and available, therefore unimagined development thrusts

`Dominance of the Language Models: 2013 - Today`:

- Learn models on large amounts of unlabeled data

-   - The model should derive linguistic relationships by itself
-   - Benefits from deep neural networks

- Actual application benefits from transfer learning

-   - Task-specific fine-tuning of pre-computed models requires less data

1.1 Corpora

`Corpus of a language`: totality of all linguistic phenomena, "the population".

`Selection corpus`: the empirical basis of corpus linguistics

`Representative corpus`: criteria for the representativeness of a selection from the population. It studies:

- The language of an author
- A historical language
- The language or conversational behavior of a social group
- The language at a given time

The probability of occurrence of new word forms decreases with text length

As the text length increases, new words/meanings appear more and more

`Typology of text corpora`:

- Total corpus of a language (Thesaurus Linguae Graecae)
- Author corpus (Shakespeare corpus)

- Selection corpus
- Monitor corpus are used to detect new phenomena and to optimize statistical analysis procedures

`Typology of corpora after editing`:

- Unedited: raw data, pure text form
- Edited: texts with labels
-     - Phonetic, phonological,
-     - Morphological
-     - Word types (parts of speech)
-     - Syntactic
-     - Semantic
-     - Pragmatic

`Use of corpora`:

- Descriptive phonlogy (determination of the phoneme system of dialects)
- Descriptive morphology (determination of the morpheme system of dialects and word formation rules)
- Lexicography (determination and documentation of vocabulary = creation of dictionaries)
- Descriptive syntax (identifying the types of sentence pattersn, phrase patters)
- Descriptive semantics (determination of word sets)
- **Linguistic Technology (development and optimization of stochastic analysis methods)**

`Corpus examples`:

- Brown Corpus: US English, Written, categorized into genres, word types labeled
- Penn Treebank: English, Written, POS labeled, syntax trees, semantic predicate argument structures, transcriptions
- British National Corpus: British English, Written, word types labeled
- Taylor Swift Corpus.

1.2 Background to markup/tagging

`Tagging`: identification of the units of a text by defined labels, which are contained in a `tag set`.

`Unit (token)`: smallest set of characters that belong together

- Letters
- Syllables
- Words
- spaces, line breaks, punctuation marks

`Linguistic marking`: Identifying linguistic units of a text by labels. Labels mark the linguistic quality of the unit in question.

`Feature structure`: A representation of the grammatical features of a word or phrase.

`Linguistic marking touches`:

- Need to explicitly present the results of the linguistic analysis

- Making results usable as a basis for further analyses
- Claims of descriptive linguistics
  - Descriptive syntax
  - Descriptive semantics
  - Statistics
  - Text typology

`Methods of linguistic marking`:

- Manual marking
- Machine marking - automatic syntactic analysis
  - Procedures that use mostly criteria of form
    - Assignment of the possible forms of an isolated word to the class via a set of rules directly or on the basis of an analysis.
    - Marking by means of lexicon
    - Marking by means of morphological analysis
  - Procedures that use mostly criteria of the position of words.
    - Use syntactic structure of higher-level units
    - Often the examination of the immediate surroundings of the word is sufficient
    - Often realized via partial parsers Trying to reduce ambiguities or select a variant via rules.
      - Can use frames
  - Procedures that use mostly probability of occurrence
- Marking with word lists
  - Gazetteer lists are word lists
  - Lists are created specifically for a subject area
    - Lists with first names (e.g. separated into male/female)
    - Lists with surnames
    - Lists with cities
    - Lists with countries
    - Lists with company names
    - Lists with abbreviations

`Basis for probability of occurence`: Hidden Markov Models

`Markov chains`: finite state automata in which each edge is provided with transition probabilities

HMM are generalized Markov chains

- Multiple outgoing edges with the same symbol possible
- Therefore, not possible to judge which nodes are visited if one has a valid sequence of symbols
- Probability of a state thus depends on its predecessor state:
- For word types: P (word|tag) * P(tag|previous n tags)

`Brill tagging`:

- Initial phase: Start with a starting marker
- Training phase: Learn context-based transformation rules
- Iterate until a termination criterion is met

Transformation rules are limited to a few possible templates, such as:

- Replace marker A with marker B if
    - the predecessor (successor) has mark Z
    - A B prevTag Z | A B nextTag Z
- Or such as:
    - the 2nd predecessor (2nd successor) has mark Z
    - A B prev2tag Z | A B next2tag Z

Transformation rules are limited to a few possible templates, such as:

- Replace marker A with marker B if
    - the predecessor has mark Y and the successor has mark Z
    - A B sourroundTag Y Z
- or such as:
    - the predecessor (successor) has marker Y and the pre-predecessor (2nd successor) has marker Z
    - A B prevBigram Y Z | A B nextBigram Y Z

`Evaluation criteria`:

- Accuracy = (true positives + true negatives) / all
    - Determines the percentage of syntactic structures assigned to the sentences that are correct
    - It is possible that no syntactic structure is assigned to a sentence at all
- Coverage
    - Determines the percentage of records that are assigned a structure
    - Correctness of the assigned structure is not considered

Weighting can be selected during development higher coverage usually means lower accuracy

`Formulas`:

- Accuracy = **(true positives + true negatives) / all**
- Precision = **true positives / (true positives + false positives)**
- Recall = **true positives / (true positives + false negatives)**
- F_b = (1 + b^2) * (precision * recall) / (b^2 * precision + recall) (if harmonic, b = 1)

`Evaluation of the error rate for markers`:

- Error rate of 4% for word markings means the following
- If average sentences are 20 words long, we achieve that 56% of all sentences contain a mislabeled word
- If we want to achieve an error rate of 4% for sentences, we need an error rate of 0.2% for words!
- Ripple effect due to error propagation: marking the word types is often the beginning of an NLP processing chain.

1.3 Classification of the app reviws

Feature Identification

`Keywords`:

- Done by string matching

- - Manually created list of keywords per category
- - Search in ratings

Word Environment

Similar/same words often have similar words in their environment.

`Features of the environment`:

- words at index -3, -2, -1, +1, +2, +3 seen from word
- Properties typically include the word, part-of-speech tag, etc.

`Bag-of-Words: General neighborhood`

- Words in a (symmetrical) window around the word
- Ignores position and order of words
- Often only binary (i.e. rather set-of-words) and for selected words

`Stop words`: Frequently occurring words (which carry little information)

Removal of so-called "stop words" from the text to be processed. Reduces noise (important words get more weight), but can exclude words important for the domain.

`N-grams`: Sequences of n

`Basic word forms`:

- Stemming: Removing the word ending using simple heuristics
- Lemmatization: forming the basic form with the help of the word environment (statistical models) and dictionaries

Classification

`Learning method`:

- Supervised learning
- - Naive Bayes
- - Decision trees
- - Multinomal logistic regression (maximum entropy)

`Types of classification`:

- Binary classification (Multiple binary calssifications, allows membership in multipe classes, high training effort, potentially more accurate)
- Multiple classification (single classifier returns probability for each class, somewhat less training effort)

`Bayes rule`:

- P(B|A) = P(A|B) * P(B) / P(A)
- P(A|B) = P(A ∩ B) / P(B)
- Starting point: p(c_i | f_1 ... f_n) = p(f_1 ... f_n | c_i) * p(c_i) / p(f_1 ... f_n)
- End evaluation: P(c_i | f_1 ... f_n) = 1 / Z * argmax p(c_i) * prod from j = 1 to n of p(f_j | c_i)

Evaluation Improvement

`Classifications can be imporved by`:

- Username
- Rating reviews
- Publication date
- Training model
  - for specific application (especially accurate - data must be available!)
  - for certain categories (games are rated differently than news apps)

`Threads to validity`:

- Internal validity
  - Incorrect manual markings
  - Poor choice of classes
  - Keywords overlooked
  - Not chosen the best algorithms, features, etc.
  - Feature selection (not machine learned)
- External validity
  - not valid for other platforms (Amazon etc.)
  - not transferable to other languages
  - other rating types have different characteristics (hotel ratings etc.)

`Countermeasures`:

- Incorrect manual markings
  - Design of a marking guideline
  - Two people mark independently
  - Third person mediates in case of disagreement: Only markers with at least two votes selected
  - Markers were engaged
- Poor choice of classes
  - Pre-study: these classes were the relevant ones for SW development
- Keywords overlooked
  - Keywords have little influence -> Error negligible
- Not chosen the best algorithms, features, etc. Should be investigated further But: investigated algorithms and characteristics/combinations of characteristics in detail (t- test, multiple runs, etc.) Feature selection (not machine learned) Should be the subject of future research: Influence unknown

1.4 Parsing

`Pipeline goes as so`: Reader -> Tokenizer -> POS Tagger -> Stemmer -> Parser -> Semantic Analyzer -> Discourse Interpreter

`Classic parsing`:

- Produce a grammar and an associated lexicon
- However, this scales quite poorly and results in only moderate coverage

-
    - A minimal (and reasonable) grammar already generates 36 alternatives for the input example above
-
    - A grammar with 10 rules already 592
-
    - A realistic grammar generates very many alternatives

`Problem with classic parsing`:

- Strongly constrained grammars try to avoid improbable or weird results
-
    - This causes the grammar to be no longer robust
-
    - Many sentences cannot be parsed
- Less severely constrained grammars parse more sentences
-
    - "Simple" sentences, however, get many possible parsings

`Solution for classic parsing`:

- We need a mechanism to determine the probability of a parse
- Statistical parsing allows loose grammars, just pick the best result

`Annotated databases`: Building a corpus with treebank seems slower and less useful than building a grammar.

`Pros`:

- Basis for evaluations
- Reusability of the parses
- Wide coverage
- One can determine frequencies and distribution information

Syntax and dependency trees

`Syntax structure` divides sentences into nested constituents

`Dependency structure` shows which words depend on which other words, closer to semantics

`Syntax tree vs dependency tree` = Nested constituents from multiple words vs Grammatical relations between words

Two paraphrased sentences usually have very different syntax trees but rather similar dependency trees

Context free grammars

CFGs capture structure and sequence of constituents, order of the words and studies how are groups of wrods formed and how do they behave?

`Constiuents are`:

- Nominal phrases, come before verbs
- Subordinate clauses

`CFG consists of`:

- A set of non-terminal symbols
- A set of terminal symbols
- A set of production rules
- A start symbol

`Derivation`: Sequence of rule applications that leads to an input. Covers all elements of the input, and only elements that actually occur in the input

`Problems in CFG`:

- Dependencies over longer distances
- Non-lexical dependencies that are important for semantics
- All this can be captured, but not elegantly

Parsing

`Parsing`: Assign correct syntax trees to input strings

`Correctness`:

- All - and only those - elements of the input are present as leaves
- The start symbol is the root of the tree

`Steps for parsing`:

- Identify all possible structures for a sentence
- - Bottom-upp: Starting from the word sequence, an attempt is made to reduce to the starting symbol
- - Top-down: expansion of the start symbol with the goal of expanding to the entire word sequence

`Top-down vs bottom-up`:

- Top-down has a problem when there are left-recursive rules of the form. A -> A B (can lead to infinite loops)
- Top-down has less problems with right-recursive rules of the form A -> B A
- Bottom-up has a problem when a non-terminal is reduced to an empty terminal
- Bottom-up has fewer problems with left-recursive rules

`Left-corner parsing`: combine top-down "predictions" with the bottom-up source of information: the words of the input

`Left corner`: first (leftmost) symbol on the right side of the rule

`Steps`:

- Start with the start symbol
- Apply only a rule for S whose left corner matches the first input symbol
- continue with the next, leftmost nonterminal and expand it in a similar manner matching the next input symbol
- do it untill all is matched.

`Cocke-Younger-Kasami algorithm (CYK)`:

- Requires grammar in Chomsky Normal Form (CNF).
- all rules must have the following form
-
  - A -> B C or
-
  - A -> w or
-
  - S -> epsilon
- The start symbol must not be on any right side

`Covert CFG to CNF`:

- Clean up S ->
-
  - Introducing S' -> S and S' -> epsilon
-
  - Make S' the new starting symbol
- Generate weak CNF (for terminals that occur in right-hand sides with non-terminals)
-
  - For each terminal w, introduce a rule X_W -> w, and use these instead of terminals
- Clean up right sides with more than two non-terminals
-
  - For A -> BCD replace BC by X_BC and insert rule X_BC -> BC
- Removal of epsilon productions
-
  - Delete all rules with A -> epsilon (Execpt for S' -> epsilon)
- Remove chain rules
-
  - Given A -> B, for each B -> w, introduce A -> w

`Steps to determine if a string can be generated by a grammar (nonprobabilistic)`:

- Given a grammar in CNF and a word w = w1...wn, for each partial word wi...wj (with 1 ≤ i ≤ j ≤ n), determine which non-terminals generate it.
- Since CNF is present, the first step is trivial.
- Since the CNF does not contain ε-productions, words with multiple letters can only be generated from rules of the form A → BC, where B generates the left part and C generates the right part of the word.
- For each partial word wi...wj, initialize a matrix Vij with the non-terminals that can generate it.
- Non-terminals covering the entire input are located in V1n.
- For a rule A → BC, we need to find B in Vik and C in Vkj for some k such that i ≤ k < j.
- If we believe that A can generate wi...wj and A → BC is a rule of the grammar, then there must be a B in Vik and a C in Vkj for some k such that i ≤ k < j.
- We can find this by searching over all possible values of k for each i and j, and updating the matrix accordingly.
- If the start symbol S can generate w1...wn (i.e., S is in V1n), then the word can be generated by the grammar; otherwise, it cannot.

Probabilistic grammars

`Probabilistic CFGs and statistical parsing`:

`Probabilistic model`: Each rule and syntax tree is given a probability

- Sum of probabilities is 1 for all rules with the same non-terminal on the left side

- **The probability of a derivation is the `product` of the `probabilities of the derivation rules` used**

PCFGs can also be used for disambiguation

2.1 Lexical Relations and WordNet

`Lexical semantics`: The meaning of individual words

`Formal semantics, compositional semantics, sentential semantics`: How can word meanings be chosen to elicit meaning from a sentence or utterance

`Senses`: A word may have several meanings

`Lexeme`: pair (meaning, form)

`Lemma`: grammatical form used to represent a lexeme; leading form, i.e., what to look for in a dictionary to find a word

`Word forms`: Special surface structures, for inflectable words they are inflectional forms

`Homonyms`: Lexemes that share the same form but not the meaning

`Polysemes`: Lexemes that share the same form but the meaning is similar, **Polysemy can be a `systematic relationship` between meanings**

`Zeugma`: Rhetorical figure in which the common verb is placed only once in sentence compounds

`Zeugma Test`: If the sentence sounds weird, we have two different meaning of the verb

Two lexemes are `perfect synonyms` if they can be successfully/meaningfully **interchanged anywhere**

`For nouns`:

- `Hyponyms`: One meaning is a hyponym of another if it is more specific and denotes a subclass of the other
- `Hyperonym`: One meaning is a hyperonym of another if it is more general and denotes a superclass of the other
- `Meronym`: word that denotes a constituent part or a member of something. (apple is a meronym of apple tree)
- `Holonym`: the name of the whole of which the meronym is a part. (apple tree is a holonym of apple)

`For verbs`:

- `Hypernym`: From events to superordinate events (fly → travel)
- `Troponym`: From a verb (event) to a specific manner elaboration of that verb (walk → stroll)
- `Entails`: From verbs (events) to the verbs (events) they entail (snore → sleep)

`phraseologism/superlemma/idioms`: sequences of word forms with specific meaning not deriveable from a literal reading

`Encyclopedia`: A list of words (vocabulary) of a language

`Lexicon`: generally a reference work, the set of lexemes of a language or an individual speaker

`semantic networks`: graphs whose nodes are words or concepts/meanings, groups of words, frames, or the like, Edges in the graph represent relations

WordNet

`WordNet`: hierarchically organized lexical database, online thesaurus, has aspects of dictionary, ontology for english langauge

`synset`: basic building block of wordnet, set of (near-) synonyms of a single meaning

FrameNet

`FrameNet`: semantic frames, english dictionary, The meaning of a linguistic expression (e.g. a word) is always grasped in the context of world knowledge

Lexical unit is associated with a semantic frame, frames are linked to each other via relations

2.2 Ontologies

`Ontology`: explicit specification of a shared conceptualization, **explicit**: Everything that a system is supposed to know must be explicitly represented, **shared conceptualization**: In particular, what we (as humans) do not verbalize in communication, because it is self-evident for us (the communication partners), has to be made explicit. in CS, **representation of knowledge about a domain or subject area**, possible to talk about ontologies (plural), in which case two ontologies are distinguished in the type of represented knowledge or in the subject area

With the help of ontologies, **tacit knowledge** (that people have) is to be made usable for systems

`Lexicons vs. ontologies`:

- Ontologies are usually domain-specific
- Lexicons contain information about words, ontologies about **concepts**

`Words vs. concepts`:

- Overlaps, polysemy and synonymy
- cross-linguistic synonymy
- lexical gaps (missing term for a concept in a language)
- Idioms
- Word idiosyncrasies (a term is used by a group with a special meaning).

`Declarative knowledge`:

- Factual knowledge (propositional)
- Factual knowledge (as "picture", as "film", ...)
- Knowledge about the situational context
- Rules knowledge

`Procedural knowledge`:

- Knowledge about rules and their application
- Knowledge about performing actions

`Explicit vs implicit knowledge`:

- Explicit
-
    - formulated
-
    - Usable by the computer
- Implicit
-
    - Not formulated
-
    - Possibly even not formulable; then
-
    -
        - Not representable in an ontology
-
    -
        - And therefore, not usable by computer

`Exhaustive partitioning`: All instances belonging to the parent class necessarily belong to one of the partitions.

Conversely, the **union of partitions** includes **all instances of the parent class** `(completeness)`.

2.3.1 Representation of Ontologies

`Procedure`:

- Feasibility study
- Start/Preparation
- Structure/Refinement
- Evaluation
- Applications and evolution

`Phase 1: Feasibility study`:

- What is the problem to be solved?
-
    - Subject area
-
    - Solvable, economical with the help of an ontology?
- What resources are available?
-
    - Hardware
-
    - Software
-
    - Staff
-
    - Time

`Phase 2: Start/Preparation`:

- What is needed for solving the problem?
- What problem-relevant elements does the subject matter include?
- Which of these are concepts and which are instances?
- Which attributes of the concepts are important for problem solving?
- Which relations between the concepts are important?

- This step can be done with paper and pencil!

`Phase 3: Construction / Refinement`:

- Top-Down
- - Structure of the class hierarchy (taxonomy) "from general to specific". Can we possibly use an „upper ontology"? (with useful super concepts)
- Bottom-Up
- - Extract (semi-automatically?) concepts, attributes, etc. from existing documents and construct a class hierarchy from them.

`Phase 4: Evaluation`:

- Checking the ontology for deficiencies

- If found, revise

- Evaluation in terms of technology uniform syntax

- - Consistency; absence of contradictions (verification of semantics)
- - Scalability (Can the ontology be extended easily? Will the use of the ontology be still fast enough?)
- - Interoperability (Can the ontology be technically integrated; e.g., in the context of a service-oriented architecture?)

- Evaluation in relation to the users

- - Does the ontology fulfill the task?
- - Are the semantics understandable for the user?

- Formal evaluation

- - Formal checks can possibly be performed with tools

`Evaluation faults`:

- Inconsistencies
- - Circular errors
- - Partitioning errors
- - Semantic errors
- Incompletions
- - Incomplete concept classification
- - Omissions
- Redundancies
- - Grammatical
- - Identical definition of different classes or instances

`Phase 5: Application/Evolution`:

- In which other applications / systems can the developed ontology be used?
- Which applications can be used to access and use the knowledge represented in the ontology?

2.3.2 Relation extraction and automatic ontology construction

`Information extraction`: processing of unstructured information. Also known as `text analytics`.

`Name recognition` has:

- Named Entity Recognition (NER)
- Named Entity Extraction

`Naming markers` are:

- Person; e.g. Mike Tyson
- Location; e.g. Paris
- Organization; e.g. Microsoft

Types of `Relationships between words`:

- Hierarchical (ontological) relations: Karlsruhe **is a** city
- Other relationships between words: alternator **is a part of** car

`5 methods for relation extraction`:

- **`MANUALLY CREATED SEARCH PATTERNS`**
- - `Hearst Patterns`: used to identify **hyponymy** relationships, specifically for **noun phrases**. used in WordNet. 66% precision
- - - `NP such as NP`: "Fruits such as apples and oranges are healthy."
- - - `NP and other NP`: "Fruits like apples and other fruits like oranges are healthy."
- - - `NP or other NP`: "Fruits like apples or other fruits like oranges are healthy."
- - `Part-whole relationship (Berland and Charniak)`: used to identify **meronymy** relationships, by using **holonyms** and patterns. 55% precision
- - - `NP's NP`: "car's engine"
- - - `NP of NP`: "engine of car"
- - - `NP in NP`: "engine in car"
- - `PROBLEMS`:
- - - Rules are difficult to write
- - - Rules are difficult to maintain
- - - There are many rules
- - - Rules are domain specific
- - - Accuracy is not good enough
- **`SUPERVISED METHODS`**
- - Supervised relation extraction: ID all named entities, decide if they have a relationship, if yes, classify. **Doesn't need hand annotated data**.
- - `PROBLEMS`:
- - - Labeled training data is expensive to produce and thus limited in quantity.
- - - Relations are labeled on a particular corpus, the resulting classifiers tend to be biased toward some domain.
- **`BOOTSTRAPPING`**

- - `DIPRE`: extracts relationships from the internet. Learns (URL-specific) search patterns.
    - Start with a **small set of tuples**. i.e: (author, book)
    - Find nearby occurrences of the author, title of a book in WWW. Along with the tuple found, keep the context of every occurrence (url and surrounding text). i.e: The robots of dawn by Isaac Asimov
    - Generate patterns on the set of occurences.
    - Search database for tuples with that pattern and extract them.
    - if the set of tuples that match that tuple is large enough, return, else find occurences with tuples.
  - `PROBLEMS`:
    - We need corresponding initial values for each relation
    - Start values influence the quality of the procedure
    - Low precision and semantic drift.
- **`UNSUPERVISED METHODS`**
  - `KnowItAll`: Extracts instances of given classes, given relations, subclasses, corpus: entire internet. domain-independent. Depends on Generic Extraction Patterns(GEPs). If the pattern has `<relation>` in it, then that is for relation extraction, otherwise, it's instance extraction. Patterns are motivated by Hearts Patterns. It uses PMI to analyze the candidates to manage the tradeoff between precision and recall. Uses Bayesian classifier and Brille tagging.
  - Example patterns:
    - `NP and other <class1>`
    - `NP or other <class1>`
    - `<class1> such as NPList`
    - `<class1> is the <relation> <class2>`
  - It has 3 elements:
    - Pattern Learning to learn domain-specific extraction rules.
    - Subclass Extraction to boost recall.
    - List Extraction to extract elements after learning a wrapper for the list.
  - Steps:
    - Bootstrap with rules, queries and discriminators.
    - For every query, extractor searches with engines and returns a set of Web pages.
    - From web pages, extract facts using rule associated with the query.
    - Write it to extractions list and run assessor.
    - Assessor assigns a probabilty to every extraction using Bayesian classifier on some discriminator.
    - Adds the extraction with probability to knowledge base.
  - `TextRunner`: self-supervised learner (annotates pos/neg examples and learns an extractor), single-pass extractor(automatically discovering possible relations of interest with 1 pass), Redundancy-based evaluation (probabilities to relations). Uses Maximum Entropy for POS tagging.
  - Steps (Learner):
    - labels training data as pos/neg.
    - trains a Naive Bayes classifier, used by the extractor.
    - using a parser, ID trustworthy extractions.
  - Steps (Extractor):
    - with a single pass, tags all words with most probable POS tag.

- - - relations are ID'd by using heuristics to eliminate useless phrases.
- - - each phrase gets probability if they are part of the entity.
- - - a good probability entitiy, prob tuble `t` is given to classifier.
- - - if the classifier returns trustworthy, store it.
- - Steps (Assessor):
- - - counts the number of distinct sentences from which each extraction was found.
- - - uses these counts to assign a probability to each tuple using the probabilistic model previously applied to unsupervised IE in the KNOWITALL system/
- - `TextRunner` vs `KnowItAll`:
- - - KnowItAll requires lots of downloads of pages, so it takes a lot of time to complete.
- - - KnowItAll takes in relation names as input, so each time a relation is ID'd, it has to be rerun.
- - - TextRunner is scalable.
- **`DISTANT SUPERVISION`**
- - use a database of relationships to get many training examples. uses `is-a` relationships from WordNet. Uses weakly labeled data. uses freebase and wikipedia as corpus.
- - Steps:
- - - For a pair of entities, Search sentences with both entities in a corpus if they have `is-a` relation, parse it.
- - - Extract search pattern.
- - - Train a classifier with patterns.
- - `ADVANTAGES`:
- - - uses corpora and syntax.
- - - relations have a canonical name.
- - - can handle large amount of weak properties
- - - domain independent (insensitive to corpora used)
- - `PROBLEMS`:
- - - Works especially well when text corpus is closely aligned with database
- - - Noise or general sentences/texts break the heuristics

2.4 Topic modelling and labeling

`TOPIC MODELING`:

`Term frequency - inverse document frequency (tf-idf)` is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. Formula is: `tf-idf = tf * idf`, where tf is the term frequency and idf is the inverse document frequency. tf is calculated as `tf(t, d) = count(t, d) / count(d)`, where t is the term and d is the document. idf is calculated as: `idf = log(N / df)`, where N is the number of documents in the corpus and df is the number of documents in the corpus that the term appears in.

`Latent Semantic Indexing/Analysis (LSI/LSA)`: `LSI` uses singular value decomposition of the document-term matrix to represent by r orthogonal factors, which goes as: A^(nxm) = U^(nxr) * S^(rxr) * V^(rxm). n - number of documents, m - number of terms.

`Benefits of LSI`:

- overcomes synonym problem of keyword queries and vector space models. If the documents use synonyms, the topic can still be discovered via context.
- Documents clustered on the basis of their conceptual similarity.

`Limits of LSI`:

- info loss, neglects word order, needed for meaning.
- doesn't account for homonymy and polysemy. It cannot distinguish between the different meanings of terms.
- Topics are only word collections without label/title

`Use of topic modeling`:

- Search Engine Optimization
- Optimization of recommender systems
- Improvement of customer service

`Topic Labeling`:

- Simple method:
  - Take top (=most likely) term as label.
- Ontology based (Unsupervised multi-topic labeling to spoken utterances):
  - disambiguate word senses, wikipedia as pre-labeled corpus to train naive bayes classifier.
  - build topic graphs based on DBPedia relations.
  - Create Sense Graphs for each Noun
    - Create a node for the initial concept
    - Create additional nodes for all concepts that can be reached from the initial concept via two hops in the ontology
  - Create the topic graph by merging all sense graphs at shared concepts.
  - Node selection
    - We use PageRank to determine the central nodes in the graph
    - Two methods of node selection:
      - Top Connectivity: Choose nodes that are connected to the most initial concepts. If equal, choose the node with highest PageRank
      - Max Coverage: First select a node according to Top Strategy. Then select further nodes in such a way that as many as possible initial concepts that are not yet represented are in a labeled graph. The evaluation showed that only 33% of labels was a good fit for the topics.
- Selection of central nodes, which then represent the topics.

`Applications of topic labeling`:

- Query knowledge from knowledge base/ontology
- Selection of an appropriate domain ontology
- Improvement of (previous) steps like disambiguation
- Simplify or accelerate (manual) analyses

3.1 Word similarities

`Applications`:

- Question Answering
- Machine translation
- Natural language generation
- Language Modeling
- Plagiarism detection

3.1.1 Word similarities: Thesaurus-based

`Simple approaches for word similarity`:

- Use synonym relation

`Thesaurus-based word similarity`:

- Word similarity vs. word relatedness
-
  - Similar words are (almost) synonyms
-
  - Related can be anything
- Path based similarity
-

`Path-based similarity`:

- View the thesaurus as a graph. Two meanings are related if they are close to each other in the graph.
- `senses(w)` = set of sense nodes of the word w in the thesaurus graph
- `pathlen(c1, c2)` = length of the shortest path between c1 and c2
- `sim_path(c1, c2)` = - log (pathlen(c1, c2) / 2D), where D is the max. of the minimum path lengths between any two concepts in the thesaurus
- `wordsim(w1, w2) = max sim_path(c1, c2)` for c1 in senses(w1), c2 in senses(w2)

`With information content`:

- `P(c)` = sum of count(w) / N, where w is ∊ words(c), where c is a concept.
- `IC(c)` = - log P(c)

`Resnik's method` (Downside: doesn't account differences): `sim_resnik(c1, c2)` = - log (P(lcs(c1, c2))), where lcs(c1, c2) is the least common subsumer of c1 and c2.

`Dekang Lin's method`: Difference: `IC(description(c1,c2)) - IC(common(c1,c2))` Commality: `IC(common(c1,c2))` `sim_lin(a, b) = 2 * log (P(LCS(c1, c2))) / (log (P(c1)) + log (P(c2)))`

`Extended Lesk metric`:

- Lesk metric: number of words in the glosses of the two concepts that are the same
- `sim_eLesk(c1, c2) = sum of overlap(gloss(r(c1)), gloss(q(2)))`, where r and q ∊ R is the set of all wordnet relations under consideration.

`Evaluation of thesaurus-based approaches`:

- Intrinsics evaluation: calculate correlation coefficients between the results of the algorithms and similarity ratings by humans.
- Extrinsics evaluation: Deploy the algorithm in an application and check its result.

`Problems with thesaurus-based approaches`:

- You need a thesaurus for every language.
- Even if there were a thesaurus, there is no way it could have all words in it.
- Bad for verbs, good for nouns, no info about adjectives and adverbs.

3.1.2 Word similarities: Distribution-based

`Context vector`: vector of word frequencies in a context. w_i -> = (f_1, f_2 ... f_n). Represent the words by their context vectors. Calculate the distance with a vector distance measure. The words are similar if the **distance is small** or the **vectors are similar** according to the measure.

- Defining the vectors:
    - Parse the sentences and extract syntactic dependencies
    - For the word "X" there is now a R-dimensional vector. where R is the number of relations.
- Weighting of the count values
    - Probability of a feature f given w is P(f|w) with the maximum likelihood, `P(f|w) = number(f, w) / number(w)`.
- `assoc_prob(w, f) = P(f|w)`.
- `ML estimator P(f, w) = number(f, w) / sum of number(w')`.
    - Pointwise Mutual Information(PMI): `PMI(x, y) = log_2^( P(x, y) / P(x) * P(y) )`. Mutual information formula: `MI(x, y) = for all x( for all y ( P(x, y) * PMI(x, y) ))`. `assoc_pmi(w, f) = PMI(w, f)`. Lin had a different idea: `assoc_lin(w, f) = log_2^(P(w,f)/ P(w) * P(r|w) * P(w'|w))`, where f = (r, w').
    - `PROBLEMS`:
        - you need a big corpora for this to be reliable. pick pmi or lin > 0.
        - vectors are sparse, for large vectors, they are unwieldy, and for short, they have limited vocabulary.
    - `Word embeddings` is a "relatively **short vectors** with **full semantic representation**":
        - `Word2Vec`: train neural network on pseudo task. uses skip-gram neural network model.
        - Context vector for "x" X output weights for "y" -> softmax `(e^x/sum of e^x)` = Probability that if you randomly pick a word nearby "x", that is "y". In Word2Vec, we throw away output layer, and use hidden layer as model.
    - `Application for word embeddings`:
        - Word/text similarity
        - Automatic summaries
        - Translation
- Define similarity of vectors
    - Normalization: Minkowski distance: (is bad for extreme values)
        - `L_m(a, b) = (sum of |a_i - b_i|^m)^(1/m)`.
    - Metrics for Information retrieval:
        - `sim_cos(a, b)` = a • b / | a | * | b |.
        - `sim_jaccard(v, w) = sum of min(v_i, w_i) / sum of max(v_i, w_i)`.

- - - `sim_dice(v, w) = 2 * sum of min(v_i, w_i) / sum of (v_i + w_i)`. sim_dice and sim_jaccard work particularly well, as does the t-test for weighting.
- - `PROBLEMS`:
- - - out-of-vocabulary words
- - - Homonymous (cannot distinguish between different meanings of a word)
- - - Antonyms (often "close" to each other despite opposite meaning)
- - `FastText embeddings`: Uses sub-word information, word is represented by a set of character-n-grams.
- - `Sense embeddings`:
- - - `ELMo` (Embeddings from Language Models) - learn the context of words, from the current sentece. long short term memory (LSTM) neural networks with two layers and pass through input bidirectionally (BiLSTM), concatenate, weight and sum vectors. Returns a representation of "tezguino" in the context of "...make tezguino out..."
- - - `BERT`: Bidirectional Encoder Representations from Transformers gives Pre-trained language model. Applied on specific domains to improve performance on domain-related NLP tasks. **Has more layers than ELMo**. Uses BooksCorpus and Wikipedia. Result is pre-trained language model.
- - `PROBLEMS`:
- - - generation of embeddings is lossy
- - - BERT is commited to neural networks
- - - Big memory requirement

3.2 Word sense disambiguation

`WSD variants`:

- Lexical sample task - WSD for a small, fixed set of words. Supervised learning works well here.
- All-words task, Lack of training data is a problem.

`Word Sense` **`Discrimination`** uses unsupervised techniques, clusters word usages without predetermined repertoire of senses.

`Approaches of WSD`:

- Knowledge-based approaches used dictionaries, logical inference.
- Corpus-based approaches: Supervised machine learning and contextual features. Needs hand-tagged training data, which is a problem.

`Supervised WSD`:

- Start with a training text where the senses are attached to the words as tags (tagged text)
- Extract properties that describe the context of the words
- Train a classifier
- Use classifier for untagged text
- - `PROBLEMS`:
- - - Training data is expensive
- - - word must be included in the training data

`Naive Bayes for WSD`:

- classifier selects most probable sense, given a feature vector
- apply Bayes' law
- use ML estimator.
- `PROBLEMS`:
-
  - rounding errors
-
  - 0 probabilities for all senses

`Evaluation of WSD`:

- Intrinsic evaluation: Have WSD system tag terms for which there is a tagged test corpus.
- Extrinsic evaluation: Evaluate Machine translation, IR, Question Answering.
-
  - MFS(Most-frequient-sense) baseline: Just use the most common sense for an ambiguous word in the training corpus. 55%
-
  - GCY92 (Gale, Church, Yarowsky): 90%
- `PROBLEMS`:
-
  - Needs a baseline

`MFS baseline`: use most common sense for an ambiguous word in the training corpus, if not known, use first from WordNet

`Dictionary-based approaches`:

- `Lesk`: Get all sense, Compare the definitions of the target word with those in the context, Choose the sense whose description has the greatest overlap.

`Mihalcea 2007 WSD`: Use Wikipedia as a basis for building a sense tagged corpus. Consider (internal) links as sense annotations. Steps:

- Collect markers (piped link [[thing (sense)]] or a plain link)
- Map markers to WordNet senses
- Train a Naive Bayes classifier
- The more data, the better
- `PROBLEMS`:
-
  - Wiki has info mostly for nouns
-
  - some labels are inconsistnent.

3.3 Specification imporvement

`Common errors in specifications`:

- Inherent ambiguities
- Inconsistencies

`Solutions`:

- quality characteristics: indicators for errors that can be found objectively.

`Quality model by Fabbrini`:

- Non-ambiguity (Uniqueness)
    - Ambiguity
    - Subjectivity
    - Optionality
    - Weakness
- Completeness
    - Under-specification
- Consistency
    - Reference error
- Understandability
    - Multiplicities
    - Implicity
    - Unexplanation (Acronyms)

`QUARS (Quality Analyzer for Requirements Specifications)`:

- Lexical Analyzer
- Syntax Analyzer
- Quality Evaluator
- Special purpose grammar
- Dictionaries

`QUARS` steps:

- Get requirement document
- Start with lexical analysis, outputs POS tags
- Do syntatic analysis with grammar, outputs Trees
- Do quality evaluation with domain dictionary and properties-related dictionary
- Get warnings

`RESI (Requirements Engineering Specification Improver)` steps:

- Get requirement document
- Basic object specification
- Preannotation (POS tags, ontology links)
- Preannotated Specification Object
- Apply rules until no more changes
- Improved Specification Object
- Export Graph-System

`RESI linguistic defects`:

- Deletion
    - Incomplete process words: missing actor and action arguments.
    - Incomplete modal verbs: why?
    - Modal verbs requiring further specification in exceptional cases. (what if?)
- Generalization
    - Universal quantifiers should have exceptional cases specified (admin user)
    - Incomplete specified conditions (else missing)

- - Nouns without reference index need a determiner/quantifier.
- Distortion
- - Nominalizations(turning verb into a noun) hide a process behind a simple event
- - Stretched verbs are less precise.
- Cluttering
- - Several words with same meaning
- - Definite and indefinite articles

`RESI` improves Recalls in the specifications.

`DENOM`: Analyze whether a found nominalization is problematic

- Get identified nominalizations
- Check in glossary, if yes, then it's self-descriptive
- Is the nominalization a part of a nominal phrase? if yes, Defined in the sentence wide context.
- Consider the intra-sentence context, if yes, again 2nd.
- If no context at all, Category 3 or 4 (WARNING)
- - **`Improves the precision to 65% from 8%. with the mean being 75% from 15%.`**

4.1 Automatic Modeling with Semantics

The software should work inside with the same terms as its user "outside".

`OO analysis`:

- Identify the reality
- Model the section to be supported by the program
- Make the model executable
- Simulate the selected section of the world

4.1.1 Manual modeling using rule sets

`Linguistics analysis`:

- Naive approach
- - Model predicate as relation sub and obj as classes?
- Superficial analysis
- - Use "simple" NLP techniques to extract candidate classes and relationships from texts, and an analyst makes decision manually.
- `PROBLEMS`:
- - Requires lots of interaction with the analyst
- Deep analysis
- - Determine attributes, relationships, etc. in addition to candidate classes (tailored to one domain)
- `PROBLEMS`:
- - Requires lots of background knowledge
- - Domain-specific
- Domain-independent semantic analysis

- - Use NLP tools that allow syntactic analysis for this purpose, to identify logical sub. and obj-s
- - Determine possessive contructions

4.1.2 Semi-Automatic Modeling via Syntax

`Harmains Class Model Builder` is a CASE tool for analysis and processing of software requirements. Does Domain-independent OO analysis. Takes software requirements document as an input, linguistically analyses this document to build an integrated discourse model and extracts the main object classes and the static relationships among objects of these classes. Analyst selects classes and relationships and a model is generated.(CM 1). Then discourse model is built, interpreted and model is created (CM 2)

- CM-Builder 1: Performs a superficial analysis; makes suggestions
- CM Builder 2: Creates a domain model without asking

`CM steps`:

- Lexical preprocessing
- - Tokenization
- - Sentence splitting
- - POS tagging
- - Morphological analysis
- Parsing and Semantic interpretation
- Discourse interpretation
- Outputs a CDIF Conceptual model

`Evaluation of CM builder`:

- `Recall = n_correct / n_S`, where n_S is elements from sample solution
- `Precision = n_correct / n_correct + n_false`, where n_correct and n_false are if the element is in the sample solution or not.
- `Over-specification = n_extra / n_S`

`Finding matching elements`:

- with Element-specific context comparison
- First perform a mapping of the classes, Maximize the number of hits for classes, Then consider attributes and relationships

4.2 Automatic modeling with Semantics

4.2.1 Labels and more

`Thematic role`: can be described by a predicate and various dependencies, encode this relationship.

Consider not the syntactic roles, but the `semantic or thematic roles`, which are:

- Agens - doer
- Actus - action

- Patiens - target of an action
- Instrumentum - with what
- Transitus - change of state
- Status - state
- Theme - whose state is explained
- Possessor - owner
- Habitum - owned
- Destinatio - to where
- Recipiens - to who
- Beneficiens - for whom
- Origo - from where
- Locus - where
- Creator - creator/destroyer
- Opus - created/destroyed
- Tempus - when
- Omnium - everything/whole
- Pars - part
- Donor - giver
- Recipient - receiver
- Habitum - received

`Application of thematic roles`:

- Question Answering
- simple representation of meaning for
  - Machine translation
  - Document summary
  - Information extraction

`Problems with thematic roles`:

- It is difficult to formally define a role
- Intermediary instruments
- Enabling instruments

4.2.2 From text to UML model via semantics

- Open word classes ≈ Model elements
  - Nouns, verbs, etc. play thematic roles and justify model elements
  - Adjectives justify attributes or parameters
- Closed word classes ≈ context
  - Articles, prepositions, adverbs, etc. encode the relationships

`Model creation from the graph`:

- Simple pattern: create a class for all the words with roles and add attributes to class.
- Copmlex pattern: Donor + Recipiens + Habitum + Actus in one relation: method Actus at the donor with the parameters Habitum and Recipiens

5.1 Language Models

`Neural langauge model`: approach to predicting words/phrases

`Training and fine-tuning of Language Models`:

- Semi-supervised learning: Prediction, sometimes add tasks
- Fine-Tuning: Supervised Learning: Training on a specific related task using a labeled dataset

5.1.1 NorBERT

`NorBERT` has a single-layer feedforward neural network. Good solutions with little training data through transfer learning.