**Natural Language Processing (NLP)**

**Session 5**

1. Why are sequence models not enough for natural language?
   1. Language has hierarchical structure
   2. Formal grammars were designed to capture the structure of NL
   3. Formal grammar is sometimes referred to as the study of syntax (vs semantics or pragmatics).
   4. In a linguistic context, this word is used to refer to “the ways words are arranged together” e.g., in the sentences and other constructions of language.
   5. Principle of compositionality: The meaning of a complex term is a function of meanings of its parts.
   6. Formalisation of communication using language: “A speaker expresses a psychological idea using a physiological act, which produces a signal that is transferred to a hearer. Upon hearing this signal, the hearer uses the physiological impressions received and recovers the idea. For the communication to be a success, both have to have shared associations between form and meaning.” – de Saussure
2. **Generative grammar**:
   1. The main concept underlying generative grammars is constituency.
   2. A significant part of developing a grammar/grammar engineering involves discovering the constituents present in a particular corpus, or in a language more generally.
3. **Formal grammar**:
   1. Grammatical Relations
      1. These are relationships between the constituents.
      2. Examples are Subject and Object.
      3. For example, in the sentence: “She adores the deep blue sky”.  
           
         “she” and “the deep blue sky” are noun phrase constituents that are the subject, and the object of verb “adores”.
   2. Dependency Relations
      1. Special types of relations between words and phrases. e.g. the verb “want” can be followed by an infinitive:
         1. “I want to sleep.”
      2. It can also be followed by a noun phrase: “I want a sleeping bag.”
         1. This is not the case for all verbs, for example the verb “find", cannot be followed by an infinitive. One cannot say: \*“I find to fly to Edinburgh.”
4. **Context free grammars:**
   1. Context free grammars (from the Chomsky hierarchy of formal languages) are the backbone of many formal models of syntax.
   2. Reasons:
      1. Powerful enough to formalise (most) relationships between words in a sentence.
      2. Computationally tractable enough for implementation, so there exists lots of parsing algorithms and tools for them.
   3. Context free grammars or CFG’s are also called Phrase Structure Grammars. Their formalisation is like Backus-Naur Form (BNF).
      1. A CFG has: a set of production rules: how constituents of language are grouped and ordered together a lexicon: determining which constituents words of a language are part of.
      2. Example: production rules: (NP – ProperNoun; NP – Det Nominal) A noun phrase NP can be generated from either a ProperNoun, or a determiner followed by a Nominal.
      3. Example: lexicon: (Det – a; Det – the) Words “a” and “the” are determiners.
      4. We can combine the production rules with the lexicon and other production rules to form phrases and sentences:
         1. “a” can be a determiner (Det – a), “flight” can be a Noun (Noun – flight), “a flight” can be a noun phrase (NP – Det Nominal).
      5. The symbols on the left-hand side of lexical rules are the lexical categories of words (Det – a; Det – the; Det – this; Det – that; Noun – flight; Noun – morning; Noun – star).
      6. Rules with the same left-hand side can also be denoted using the delimiter | to save space. This form is often used for lexical rules. (Det -> a | the | this| that Noun -> flight | morning | star)
      7. A CFG can be thought of as:
         1. Generating sentences of language in the lexicon.
         2. In generation, the rules are treated as rewrite rules.
            1. NP->Det Nom rewrites NP to Det Nom
            2. Nominal -> Noun rewrites Det Nom to Det Noun
            3. Noun -> flight rewrites Det Noun to Det flight
            4. Det -> a rewrites Det flight to a flight
      8. Here, the terminology we use is:
         1. The string “a flight” can be derived from non-terminal NP.
         2. The sequence of rules is called a derivation.
         3. The language of a CFG is the set of strings that are derivable from the designated start symbol S.
         4. The set of strings derived from S is the set of sentences of a simplified version of English.
      9. A set of possible derivations is commonly represented by a parse tree. For example the parse tree of “a flight” is: NP (root), Det – a (left), Nom – Noun – Flight (right)
      10. A few more rules for the grammar of English:
          1. S -> NP VP VP -> Verb NP VP -> Verb NP PP VP -> Verb PP PP -> Preposition NP Pronoun -> me| I| you| it
          2. I prefer a morning flight prefer a morning flight leave London at noon Leave on Sunday
          3. From London on Wednesday on July 16th
      11. PP’s are often used with times, dates, and locations. PP -> Preposition NP; From London on Wednesday On July 16th.
      12. A sample lexicon:

|  |  |
| --- | --- |
| Noun | flights | breeze | trip | morning |
| Verb | is | prefer | like | need | want | fly |
| Adjective | cheapest | non-stop | first | latest | other | direct |
| Pronoun | me | I | you | it |
| Proper-Noun | Alaska | Baltimore | Los Angeles | Chicago | United | American |
| Determiner | the | a | an | this | these | that |
| Preposition | from | to | on | near |
| Conjunction | and | or | but |

* + 1. Some grammatical rules:

|  |  |
| --- | --- |
| Grammar Rules | Examples |
| S -> NP VP | I + want a morning flight |
| NP -> Pronoun | Proper-Noun | Det Nominal | I | Los Angeles | a + flight |
| Nominal -> Nominal Noun | Noun | Morning + flight | flights |
| VP -> Verb | Verb NP | Verb NP PP | Verb PP | Do | want + a flight | leave + boston + in the morning | leaving + on Thursday |
| PP -> Preposition NP | from + Los Angeles |

* + 1. The bracketed notation for the parse tree:  
         
       [S [NP [Pro I]] [VP [V prefer] [NP [Det a] [Nom [N morning] [Nom [N flight]]]]]]
    2. The long, lonely night is full of stars and moonlight:  
       ((S (NP-SBJ (DT The) (JJ long) ( , , ) (JJ lonely) (NN night)) (VP (VBD is) (ADJP-PRD (JJ full) (PP (IN of) (NP (NN stars) (CC and) (NN moonlight) )))) (. . ) ))
    3. **Formal definition of a CFG:**(N, sigma, R, S)  
         
       where,   
         
       N is a set of non-terminal symbols; sigma is a set of terminal symbols, S is a designated start symbol, R is a set of production rules of the form alpha implies beta (alpha is a non-terminal & beta is a string of symbols from the strings [sigma union N]^\*).
    4. The symbols of a CFG are classified into two groups:
       1. Terminals: These correspond to the words of language. The words are introduced via these rules in the lexicon. e.g. flight, morning, star, a, the, this, that
       2. Non-Terminals: Symbols that express generalisations of these. a.g. S, NP, VP, Noun, Det
       3. and of course, we have the -> (right arrow) symbol
    5. In a CFG:
       1. The items to the right of -> are:
          1. an ordered list of one or more T’s or NT’s
          2. The item to the left of -> is a single NT
          3. This is what makes the language generated by these rules’ context free-there is no context for the application of the rule: on the left-hand side we only have a single NT.
    6. A language is defined through the concept of derivation. A string derives another if it can be rewritten as the second one by a series of rule applications.  
         
       If A -> beta is a production rule and alpha and gamma are two string in [sigma union N]^\*, then we say: (alpha A gamma) directly derives (alpha beta gamma) [obtained by substituting A by beta].  
         
       This is more formally denoted by: (alpha A gamma) only if (alpha beta gamma).
    7. A derivation is a generalisation of a direct derivation. If we have:  
         
       alpha\_{n-1) only if alpha\_{n}  
       then we say  
       alpha\_{1} derives alpha\_{n}

And formally write:  
alpha\_{1} only if alpha\_{n}

* + 1. Parsing is the problem of mapping a string of words to its derivation.
    2. The language generated by a CFG is the set of strings composed of terminals that can be derived from the designated start symbol. A context-free language is therefore:  
         
       L\_{CFG} = { w given w contained with sigma\* and S only if w }
    3. Sentences (strings of words) that can be derived by a grammar are in the formal language defined by that grammar, and are called grammatical sentences.
    4. Sentences that cannot be derived by a given formal grammar are not in the language defined by that grammar and are referred to as ungrammatical.
    5. In linguistics, this is called a generative grammar since the language is defined the set of possible sentences “generated” by the grammar.
    6. In the next lecture we go beyond ‘in’ and ‘out’ binary notions of grammaticality and move towards probabilistic notions of grammaticality.

1. **Logical Grammars**:
   1. The first logical grammar was formalised by Ajdukiewicz in 1935 and only had one rule:
      1. B given A, A => B
      2. This rule says that when an expression of grammatical type A is preceded by an expression of type B given A, then we obtain an expression of type B.
      3. Ajdukiewicz’s rule can be thought of like multiplying a fraction: B given A x A = B
      4. B given A can be thought of as the fraction: B over A
      5. The rule can then be thought of as multiplication: when B over A is multiplied by A, we get B.
      6. Ajdukiewicz called this rule a cancelation scheme. He defined a notion of grammaticality as follows: A string of words has a satisfying syntactic connection if some ordering of its word types reduced to the distinguished type S (of sentence) via successive uses of the cancelation scheme.
      7. Some basic logical types:
         1. NP: noun phrase
         2. S: sentence
      8. Some types assignments:
         1. flight: NP
         2. morning: NP given NP
         3. a: NP given NP
         4. prefer: (S|NP)|NP
   2. In order to make the calculus more refined, Bar-Hillel (1953) introduced directional division types: A\B and B/A and introduced a directional version of the cancelation schema:   
        
      A A\B => B   
      B/A A => B   
      The resulting system is called the AB calculus.
      1. Some basic logical types:
         1. CN: common noun
         2. N: nominal
         3. S: sentence
      2. Some types assignments using directional types:
         1. flight: NP
         2. morning: NP/NP
         3. a: NP/NP
         4. prefer: (NP\S)/NP
      3. An extended set of basic logical types:
         1. NP: noun phrase
         2. S: sentence
         3. PP: prepositional phrase
      4. More advanced type assignment:
         1. we, flight, Geneva, Chamonix: NP
         2. to: PP/NP
         3. and: X\(X/X)
         4. flew,drove: (NP\S)/PP
      5. X can be any type, basic or complex:
         1. X = NP, X = S, X= PP
         2. X=PP/NP, X = NP\S, etc…  
              
            There are some slides on page 64-73 that I do not know how to explain.
   3. In 1958, the logician J. Lambek provided a logical reading of BarHillel’s directional types using the logical law of Modus Ponens: A -> B, A => B
      1. He formalised Bar-Hillel’s rules using proof rules and since then many people (Pollard and Sag 1987, Morrill 1994, Moortgat 1997, Steedman 2000) have used his setting to augment the AB calculus.
      2. In 1992 Pentus showed that the expressive power of Lambek’s calculus is the same as Chomsky’s context free grammars.
      3. He also presented a translation between the formalism of generative grammars and that of logical grammars.
      4. The two grammatical paradigms were finally united!
   4. Summary:
      1. Formal grammars define a mechanism for generating all strings of a formal language which approximate natural languages.
      2. The notion of constituency is key to understanding them.
      3. CFGs are common grammars which define context-free languages, but these don’t cover all of a natural language like English.