**Natural Language Processing (NLP)**

**Session 8**

1. Semantics
   1. With syntax we looked at the form of sentences of natural language, now we need to look at the meaning of sentences- that is the study of semantics.
   2. For our applications to become useful, we need our systems to interpret or understand the user’s language correctly.
   3. Semantic information or semantic representations bridge the gap between syntax and our knowledge about the world.
   4. We need a semantics whenever the syntactic information is not enough to do the processing needed for the task at hand.
      1. This is the case for most real-world applications, beyond parsing or language modelling. Producing the syntactic tree for ‘put the remote control on the table’ won’t put it there without extra reasoning!
      2. We want our systems to do something for us by understanding our written and spoken behavioural input.
   5. Most human written/spoken communication needs semantic processing, that is, determining what sentences and words mean in the real world):
      1. Following a recipe: you are given a list of linguistic expressions and are supposed to follow them with the goal of preparing some food.
      2. Answering exam and quiz questions.
      3. Understanding jokes and insults.
      4. Learning a programming language using a book.
   6. Applications are the same, needing natural language understanding (NLU) capabilities, but state-of-the-art systems are more constrained:
      1. Ordering Pizzas.
      2. Booking reservations/flights.
      3. Asking queries of a database in a natural language.
      4. Return specific internet search results.
      5. Robot picking objects from commands.
   7. Example: What is the largest city in Europe by population?
      1. Syntactic parse tree from words (e.g., CFG/CCG)
      2. Maximise argument of (cities intersect containedBy(Europe), Population): semantic formula compilation from tree
      3. Istanbul: Evaluation of semantic formula from a model (ending in action)
   8. There are different kinds of semantic representations which can apply to a number of target domains/situations and reasoning can be done with them for achieving different tasks:
      1. First Order Logic
      2. Abstract Meaning Representations
      3. Frames
   9. There are different kinds of semantic representations around:
   10. First Order Logic semantic representation for the sentence “I have a car
   11. There are different kinds of semantic representations around:
   12. Abstract Meaning Representation for the sentence “I have a car”
       1. Having:
          1. Haver: Speaker
          2. HadThing: Car
   13. We are moving from a natural language:
       1. the language of English
   14. to another formal language:
       1. a semantic representation language
   15. The semantic representation language cannot be just any set of symbols. It has to satisfy a set of properties:
       1. Verifiability
       2. Unambiguous Representations
       3. Canonical Forms
       4. Expressiveness
2. Verifiability:
   1. We should be able to use the representation to be able to verify/ check whether what the syntax is expressing holds (is true) in the world, as we know it.
   2. syntax: expression (a) ``Kipling serves vegetarian food”
   3. semantics: expression (b) Serves(Kipling, Vegetarian(food))
   4. Can we use (b) to tell if (a) is true? Can we use (b) to verify (a)?
   5. If we had a Knowledge base, with properties “food”, “Vegetarian”, and “serves” we could indeed do so.
   6. We could go and query the knowledge base using (b) and see whether it holds or not.
   7. We are indeed assuming that the knowledge base is representing the information of the world and correctly so.
3. Unambiguous Representations:
   1. I want to eat somewhere close by.
   2. This can mean two things:
      1. I want to eat something at a place that is close to me.
      2. I want to eat a place that is close to me.
      3. A good semantic representation should be able to represent both.
      4. While the Natural Language query can be ambiguous, our meaning representation itself cannot be ambiguous.
   3. I want to eat delicious food.
      1. Here, it is not very clear cut what delicious food means.
      2. It is a vague concept that varies from person to person and culture to culture.
      3. Other similar concepts are adjectives such as ``tall, big, small”.
      4. These kinds of considerations fell under semantic vagueness. Our semantic representations should be able to deal with vague concepts.
4. Canonical Form:
   1. Kiplings serves vegetarian food.
   2. Vegetarian food is served at Kiplings.
   3. Kiplings has vegetarian food.
   4. Kiplings serves vegetarian meals.
   5. Kiplings serves vegetarian dishes.
   6. All of these syntactic forms mean the same things. Their semantic representations should also be the same.
   7. In other words, if one of them is true in the world, so should be others.
   8. Or, when two distinct inputs that mean the same thing should have the same meaning representation
5. Expressiveness
   1. Finally, a meaning representation scheme must be expressive enough to handle a wide range of subject matter, ideally any sensible natural language utterance.
   2. Although this is probably too much to expect from any single representational system, First-Order Logic is expressive enough to handle quite a lot of what needs to be represented for NLP.
6. First Order Logic Semantics
   1. We should be able to use the representation to be able to tell whether what the syntax is expressing holds (is true) in the world, as we know it.
   2. syntax: expression (a) ``Kipling serves vegetarian food”
   3. semantics: expression (b) Serves(Kipling, Vegetarian(food))
   4. Can we use (b) to tell if (a) is true? Can we use (b) to verify (a)?
7. FOL Semantic Representation Language:
   1. We are defining a representation language.
   2. This constitutes of:
      1. Two sets of symbols that make the vocabulary of this language:
         1. Logical symbols: /\, \/, =>, quantifiers
         2. Non-Logical symbols: John, Mary, eat, vegetarian, food
      2. A set of rules that tells us how to mix (1) and (2) to get sentences.
      3. Each of (2) has a denotation (meaning) in a model.
      4. An interpretation map between 2 and (III): between every nonlogical symbol and its denotation. This map is compositionally extendible to sentences using (II).
   3. A model consists of the following elements:
      1. Domain: a set of individuals/symbols
      2. Properties: sets of individuals
      3. Relations: sets of tuples of individuals
   4. Representation language: Logical and Nonlogical symbols Sentences built from them
   5. Interpretation Map
   6. Model: Domain Properties Relations
   7. Example:
      1. Representation Language: John likes vegetarian food and Mary likes steak.
      2. and is our only logical symbol. All the other symbols, i,e. John, likes, vegetarian food, Mary, steak are non-logical.
      3. Interpretation Map:
         1. John => J
         2. Mary => M
         3. Steak => S
         4. Vegetarian food => Veg
         5. likes => Likes
      4. Domain:
         1. J, M, P, T, S
      5. Properties:
         1. Veg:{T}
      6. Relations:
         1. Likes: {(J, T), (M, S)}
   8. Is our meaning representation verifiable? If so, how?
      1. According to the interpretation map, verifying whether “John likes vegetarian food and Mary likes Steak” is true, we have to check whether the following expression holds in our model:
         1. Veg(T) /\ Likes(J, T) /\ Likes (M, S)
      2. That is, we have to check whether the following hold:
         1. T has/belongs to the property Veg
         2. J is related to T via the relation likes
         3. M is related to S via the relation likes
      3. These are all true, so our sentence is true.
   9. We can do more!
      1. We can also verify that the following sentences are true in our model:
      2. “Mary does not like tofu.”
      3. “John does not like steak.”
      4. “Not every one likes tofu”
      5. “Not every one likes tofu or steak”
      6. “Not every one likes vegetarian food”
      7. “Either John or Mary does not like vegetarian food”
8. Truth Theoretic Semantics:
   1. How do we do that? Using the truth theoretic semantics of FOL. This is the language of FOL:
      1. Formula 🡪 AtomicFormula
         1. | Formula Connective Formula
         2. | Quantifier Variable,... Formula
         3. | ¬ Formula | (Formula)
      2. AtomicFormula 🡪 Predicate(Term,...)
      3. Term 🡪 Function(Term,...)
         1. | Constant
         2. | Variable
      4. Connective 🡪 /\ | \/ | =>
         1. Quantifier 🡪 all | E
         2. Constant 🡪 A | VegetarianFood | Maharani···
         3. Variable 🡪 x | y | ···
         4. Predicate 🡪 Serves | Near | ···
         5. Function 🡪 LocationOf | CuisineOf | ···
      5. This is the its part of its truth theoretic semantics:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | Q | ¬P (not/negation of P) | P /\ Q | P \/ Q | P 🡺 Q |
| False | False | True | False | False | True |
| False | True | True | False | True | True |
| True | False | False | False | True | False |
| True | True | False | True | True | True |

1. The Lambda Notation:
   1. The Lambda notation was introduced by Church in 1940. He extended the language of FOL with lambdas in oder to make reasoning about semantics easier. Introduces the idea of a variable and functional application.
   2. A lambda term consists of the Greek letter Lambda, followed by variables and predicates.
   3. Examples:
      1. Lambda x.P(x)
      2. Lambda x. VEG(x)
      3. Lambda x. Lambda y. Likes (x,y)
   4. We can apply lambda terms to each other and the result will be another lambda term.
      1. Lambda x.P(x)(A)
      2. P(A)
   5. Examples:
      1. Lambda x. VEG(x) (J) = VEG (J)
      2. ambda x. Lambda y. LIKES (x, y)(J) (T) = LIKES (J, T)
2. Events, Time, Aspects:
   1. Usually the natural language semantics can express more than just properties and relations.
   2. Specific properties and relations are formalised and used to reason about specific information, such as:
      1. Events: certain verbs express events, e.g.. eating
      2. Time: events happen in time, eating after or before
      3. Aspect: activity (eating), achievement (found), accomplishment (booked)
      4. (read the J & M book for more)
3. Mapping from Syntax to Semantics
   1. We have shown how to verify semantic representations made in FOL. But how do we get to these semantic representations from syntax?   
        
      Input 🡪 Parser 🡪Tree structure 🡪 Semantic Analyzer 🡪 Output Semantic Representations
   2. We have learnt about different grammatical formalisms. Each gives us a different parse tree or structure. For now, let’s consider the generative grammar model.
   3. The semantic representation of each generative parse tree is obtained by following two main steps:
      1. To each rule of our grammar, we assign a semantic representation in the Lambda notation.
      2. As rules are combined with each other, e.g. by working with them as rewrite rules or by going up the parse tree, the corresponding Lambda terms are “applied” to each other.
   4. Examples:
      1. Syntactic CFG Rule – Semantics
      2. S 🡪 NP VP { VP.sem(NP.sem) }
      3. This means in order to get a semantics for the sentences, apply the semantics of VP (VP.sem) to that of NP (NP.sem)
      4. Similarly for the rest of the rules below:
         1. VP 🡪 Verb NP { Verb.sem(NP.sem) }
         2. Verb 🡪 serves
         3. NP 🡪 ProperNoun { ProperNoun.sem }
      5. A tree with left node as NP and right node as VP. NP has one node connected in the next level Proper Noun which is connected to John which is connected to J. VP has Verb and NP as left and right nodes. Verb has one node “eats” which is connected to “lambda y. lambda x. eat (x,y)” and NP has one node “Tofu” which is connected to node labelled “T”.
      6. The Semantic analyzer labels the root node S as “lambda x. eat(x, T)(J) = eat (J, T)“. The right node of the root node “VP” is represented as “lambda y. lambda x. eat(x, y)(J) = lambda x. eat (x, T)”
      7. In reality, things are more complicated.
         1. The semantic analyser assigns much more complicated semantics to verbs, in order to cover as much as possible of their meaning.
         2. Thus the semantics of the sentence also becomes more complicated. 552 But the computational principles are the same.
4. Distributional Hypothesis
   1. Words that occur in similar contexts tend to have similar meanings. This insight was first formulated by Harris (1954) who said:
      1. “oculist and eye-doctor . . . occur in almost the same environments”
   2. and more generally that
      1. If A and B have almost identical environments. . . we say that they are synonyms.”
   3. The most famous statement of the principle comes a few years later from the linguist Firth (1957), who phrased it as
      1. “You shall know a word by the company it keeps!”
   4. The meaning of a word is thus related to the distribution of the words around it.
5. Automation:
   1. Imagine you had never seen the word tesguino, but given the following text:
   2. A bottle of tesguino is on the table. Everybody likes tesguino . Tesguino makes you drunk. We make tesguino out of corn.
   3. Can you guess what it means?:
      1. a fermented drink similar to beer made of corn
   4. This intuition can be automatised by counting words in the context of tesguino. These will tend to be words like:
      1. bottle and drunk, which are the same as the words around:
      2. beer or liquor or tequila.
   5. Thus, same contexts around words, help us discover the degree of similarity between words.
      1. e.g. between beer and liquor and tesguino.
   6. A more advanced method takes into account more sophisticated features of the context, e.g. syntactic features , e.g. ‘occurs after drunk’, ’occurs before bottle’, ‘is the direct object of ‘drink’. etc.
   7. This intuition can be automatised by counting words in the context of tesguino. These will tend to be words like:
      1. bottle and drunk, which are the same as the words around:
      2. beer or liquor or tequila.
   8. Thus, same contexts around words, help us discover the degree of similarity between words.
      1. e.g. between beer and liquor and tesguino.
   9. A more advanced method takes into account more sophisticated features of the context, e.g. syntactic features , e.g. ‘occurs after drunk’, ’occurs before bottle’, ‘is the direct object of ‘drink’. etc.
6. Distributional/Vector Semantics
   1. The meaning of a word is computed from the distribution of words around it, represented as a vector of numbers, related to frequency counts of the word in context.
   2. Main application is to compute semantic similarity between words, sentences, documents.
   3. A vast domain of NLP tasks:
      1. named entity extraction, parsing, semantic role labelling, relation extraction.
   4. An important tool in applications like paraphrasing, question answering, summarization, automatic essay grading, etc.
7. Basic Linear Algebra
   1. A vector, e.g. v, w is a list/array of numbers,   
        
      e.g. [1, 3, 37, 5] [1, 2, 58, 117] [8,12, 1, 0] [15,0,0,0]
   2. A vector space, e.g. V, W is a collection of vectors satisfying certain properties, e.g. being closed under addition and scalar multiplication.
   3. The entries of vectors are not arbitrary. They correspond to a quantity with respect to a basis, e.g. 1, 1, 8, 15 are the quantities of the above vectors with respect to basis 1 of the vector space.
   4. The number of basis of a vector space is called its dimension, it is denoted by |V|, |W|, etc, e.g. the above vector space has dimension 4.
8. Term-Document Matrix
   1. Vector or distributional models of meaning are generally based on a co-occurrence matrix, a way of representing how often words cooccur. We’ll look at two popular matrices: the term-document matrix and the term-term (or word-context) matrix.
   2. The first vector space model developed for information retrieval by Salton in 1971, inventing the term-document matrix.
   3. Documents are represented by vectors in a vector space whose basis are words.
   4. Each cell contains the number of times the (row) word occurs in the (column) document.
   5. A vector v in the vector space V can be thought of as a point in the |V| dimensional space.
   6. A visualisation/depiction of these vectors, with only two of the dimensions (battle and fool) is as follows:  
        
      Standard vectors on an x,y graph with [x, y] points representing their magnitude and direction.
   7. What is it good for?
      1. For the task of finding similar documents: Two documents are similar iff they contain similar words.
      2. The geometric or angular distance between vectors is used to represent the similarity between them. The geometric distance between document vectors is used to compute the similarities between documents through the cosine of the angle:  
           
         cosine(v, w) = (V dot W) / (|V| |W|) = sum of (v\_i \*w\_i) / (sum of(v\_i^2) \* sum of (w\_i^2))
      3. In information retrieval (IR), the distance between the vector of a query and the vector of the documents is computers and similar documents to query are retrieved as the result of search.
9. Word-Context Matrix
   1. A different approach to distributional meaning, focussing on word meaning where words which occur in similar contexts are deemed similar.
   2. Columns and rows are both labelled by words but words along the side are the target words we are interested in analysing, and the words along the top are context words which give the target words their distributional meaning.
   3. When using raw counts, each cell represents the number of times a target word (label of the row) occurs in the context/neighbourhood of a context word (label of column).
   4. Being in a context means being in a window of k-words around it.
   5. For example, here are 7-word windows around four words of the Brown corpus
10. Some Parameters of the Model
    1. The dimension of the vector space is the size of the vocabulary of the corpus, e.g. 10,000 or 50,000.
    2. This means that, lots of cells of the matrix will be 0 leading to sparse vectors, so one can use efficient sparse matrix computation algorithms to compute with these vectors.
    3. Size of the context window depends on the task, it is some where between 1 and 8.
    4. The smaller the window, the more syntactic information the cooccurrence counts represent, the longer the window, the more semantic information.
    5. The raw frequency that the cells of a word-context matrix record, is not the best measure of association between the words.
       1. We want a measure that can distinguish between very common words such as `it, the, a, they, …’.
       2. We want a measure by which informative context words are weighed higher than non-informative ones.
    6. Examples of such measures are:
       1. Pointwise Mutual Information: log\_2 (P(x,y) / P(x)\*P(y))
       2. Mutual information: sum of (P(x,y))\*log\_2 (P(x,y) / P(x)\*P(y))
11. Pointwise Mutual Information:
    1. Pointwise mutual information between two events x, y tells us how often they occurred together in comparison to our expectation if they are independent.
    2. For word-context matrices, PMI is defined with regards to the occurrence between a word and a context word:  
         
       PMI(w,c) = log\_2 (P(w,c) /P(w)\*P(c) )  
         
       where,   
         
       P(w,c) = how often w and c are observed together  
       P(w)\*P(c) = how often we expect w and c to occur independently
    3. Positive Pointwise Mutual Information
       1. A logarithm can return negative values.
       2. In this case, this happens when PMI(w,c) < 0
       3. That is, when w occurred with c, less frequent than chance.
       4. Which is unrealistic and returns unreliable quantities for word vectors.
       5. So people consider Positive PMI or PPMI.  
            
          PPMI(w,c) = maximise (log\_2 (P(w,c) /P(w)\*P(c) , 0)
    4. Computing PPMI:
       1. Given a word-context matrix F and W rows in C columns.
       2. Take F\_ij to be the number of the word w\_i occurred in the context of word c\_j.  
            
          P(w\_i, c\_i) = f\_ij / double sum of f\_ij  
          P(w\_i) = sum of (f\_ij) starting at c / double sum of f\_ij  
          P(c\_j) = sum of (f\_ij) starting at w / double sum of f\_ij  
          PPMI(w,c) = maximise (log\_2 (P(w,c) /P(w)\*P(c) , 0)  
            
          P(w=information, c=data) = 6 / 19 = 0.316  
          P(w=information) = 11/19 = 0.579  
          P(c=data) = 7/19 = 0.368  
          (log\_2 (P(w,c) /P(w)\*P(c) , 0) = Log\_2(.316 / .368\*.579) = .568
12. Smoothing
    1. PMI has the problem of being biased toward infrequent events; very rare words tend to have very high PMI values. A solution is obtained by smoothing.
    2. Laplace Smoothing: before computing PMI, add a small constant k (e.g. 1, 2, 3) to all counts. This increases the 0 counts and shrinks the non-zero counts.
    3. Another method (Levy et al 2015) is to raise the probability of context to a power:  
         
       PPMI\_a(w,c) = maximise (log\_2 (P(w,c) /P(w)\*P(c) , 0)  
       P\_a(c) = count(c)^a / sum of (count(c)^a)
    4. Experiments have shown that raising to the power of 0.75 betters the results.
    5. This works because raising the probability to a = 0.75 increases the probability assigned to rare contexts, and hence lower their PMI (Pa(c) > P(c) when c is rare).
13. Other measures
    1. TF-IDF: from Information Retrieval.
       1. TF: frequency of the word in the document (Luhn 1975),
       2. IDF: inverse document frequency (Sparck Jones, 1972).
    2. It is one way of assigning higher weights to the more discriminative words.
    3. IDF is defined using the fraction N/dfi where N is the total number of documents in the collection, and dfi is the number of documents in which term i occurs.
    4. The fewer documents in which a term occurs, the higher this weight.
    5. Because of the large number of documents in many collections, this measure is usually squashed with a log function.  
         
       idf\_i = log(N / df\_i)
14. Measuring Similarity:
    1. The geometric distance between vectors of words.
    2. If vectors have length 1, this is their dot product:
       1. Dot-product(arrow(v), arrow(w)) = arrow(v) dot arrow(w) = sum of (arrow(v)\*arrow(w)) = v\_1w\_1 + v\_2w\_2 + … + v\_N w\_N
       2. If not, it is the cosine between the two vectors: cosine(v, w) = (V dot W) / (|V| |W|) = sum of (v\_i \*w\_i) / (sum of(v\_i^2) \* sum of (w\_i^2))
       3. What is the cosine distance between these vectors?  
          digital [1,1]  
          information [6,4]  
          1. What is the distance between the vectors?
             1. For numerator, get the dot product of the two vectors: 1\*6 + 1\*4 = 10
             2. For denominator, get the product of the magnitudes:  
                  
                sqrt(1^2 + 1^2) = 1.414  
                sqrt(6^2 + 4^2) = 7.211
             3. Put it together:  
                  
                cos a = 10 / 1.414 \* 7.211 = 0.981
15. Syntactic Vector Models
    1. Instead of defining a word’s context by nearby words, we can define it by syntactic relations.
    2. The meaning of entities and the meaning of grammatical relations among them is related to their restrictions to other entities. (Harris 1968)
    3. The similarity between meanings of duty and responsibility is mirrored in their syntactic behaviour.
    4. They both can be modified by adjectives: additional, administrative, collective, congressional;
    5. They both can be the direct objects of verbs: assert, assign, assume, attend to avoid, become, etc.
16. Evaluating Vector Models
    1. WordSim-353 : a set of of ratings for 353 noun pairs.
    2. SimLex-999 : quantifies similarity (cup, mug) rather than relatedness (cup, coffee), and includes adjective, noun and verb pairs.
    3. TOEFL dataset : a set of 80 questions, each consisting of a target word with 4 additional word choices; the task is to
    4. choose which is the correct synonym.
    5. Stanford Contextual Word Similarity (SCWS) : human similarity ratings for 2,003 pairs of words, but in their sentential context.
    6. Word2Vec Dataset: a set of patterns “a is to b as c is to d”. Given a, b, and c, the task is to find d. The words are in certain semantic relationships with each other. For example Athens is to Greece as Oslo is to ? …. Norway.: Mikolov et al. 2013.