Lecture 22: Introduction to Natural Language Processing (NLP)

- Traditional NLP
- Statistical approaches
- Statistical approaches used for processing Internet documents
- If we have time: hidden variables

Natural language understanding

- Language is very important for communication!
- Two parts: syntax and semantics
- Syntax viewed as important to understand meaning

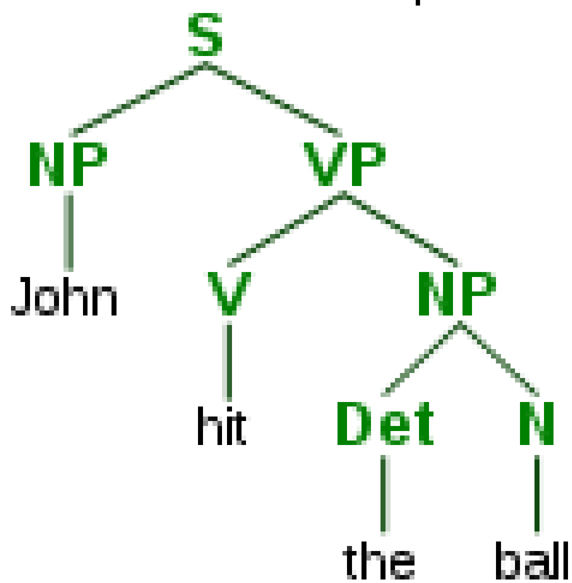
Grammars

Set of re-write rules, e.g.:

$$
\begin{aligned}
S &:= NP\ \ VP \\
NP &:= \text{noun}|\text{pronoun} \\
noun &:= \text{intelligence}|\text{wumpus}|\ldots \\
VP &:= \text{verb}|\text{verb}NP|\ldots \\
&\ \ldots
\end{aligned}
$$

Parse trees

Given a grammar, a sentence can be represented as a parse tree



Problems with using grammars

- Grammars need to be *context-sensitive*
- Anaphora: using pronouns to refer back to entities already introduced in the text

  E.g. After Mary proposed to John, *they* found a preacher and got married. For the honeymoon, *they* went to Hawaii.
- Indexicality: sentences refer to a situation (place, time, S/H, etc.)

  E.g. I am over here
- Metaphor: "Non-literal" usage of words and phrases, often systematic:

  E.g. I've tried killing the process but it won't die. Its parent keeps it alive.

Some good tools exist

- Stanford NLP parser: http://nlp.stanford.edu/software/corenlp.shtml
- Input natural text, output annotated XML, which can be used for further processing:
  - Named entity extraction (proper names, countries, amounts, dates...)
  - Part-of-speech tagging (noun, adverbe, adjective, ...)
  - Parsing
  - Co-reference resolution (finding all words that refer to the same entity) Eg. Albert Einstein invented the theory of relativity. He also played the violin.
- Uses state-of-art NLP methods, and is very easy to use.

Ambiguity

Examples from Stuart Russell:

Squad helps dog bite victim
Helicopter powered by human flies
I ate spaghetti with meatballs
abandon
a fork
a friend

Statistical language models

- Words are treated as *observations*
- We typically have a corpus of data
- The model computes the probability of the input being generated from the same source as the training data
- Naive Bayes and n-gram models are tools of this type

Learning for document classification

- Suppose we want to provide a class label y for documents represented as a set of words $\mathbf{x}$
- We can compute $P(y)$ by counting the number of interesting and uninteresting documents we have
- How do we compute $P(\mathbf{x}|y)$?
- Assuming about 100000 words, and not too many documents, this is hopeless!

  Most possible combinations of words will not appear in the data at all...
- Hence, we need to make some extra assumptions.

Reminder: Naive Bayes assumption

- Suppose the features x_i are discrete
- Assume the x_i *are conditionally independent given* y.
- In other words, assume that:

$$P(x_i|y) = P(x_i|y, x_j), \forall i, j$$

- Then, for any input vector $\mathbf{x}$, we have:

$$\begin{aligned} P(\mathbf{x}|y) = P(x_1, x_2, \ldots x_n|y) &= P(x_1|y)P(x_2|y, x_1) \cdots P(x_n|y, x_1, \ldots x_{n-1}) \\ &= P(x_1|y)P(x_2|y) \ldots P(x_n|y) \end{aligned}$$

- For binary features, instead of $O(2^n)$ numbers to describe a model, we only need $O(n)$!

Naive Bayes for binary features

- The parameters of the model are $\theta_{i,1} = P(x_i = 1|y = 1)$, $\theta_{i,0} = P(x_i = 1|y = 0)$, $\theta_1 = P(y = 1)$
- We will find the parameters that *maximize the log likelihood of the training data*!
- The likelihood in this case is:

$$L(\theta_1, \theta_{i,1}, \theta_{i,0}) = \prod_{j=1}^{m} P(\mathbf{x}_j, y_j) = \prod_{j=1}^{m} P(y_j) \prod_{i=1}^{n} P(x_{j,i}|y_j)$$

- First, use the log trick:

$$\log L(\theta_1, \theta_{i,1}, \theta_{i,0}) = \sum_{j=1}^{m} \left(\log P(y_j) + \sum_{i=1}^{n} \log P(x_{j,i}|y_j) \right)$$

- Observe that each term in the sum depends on the values of y_j, $\mathbf{x}_j$ that appear in the jth instance

Maximum likelihood parameter estimation for Naive Bayes

$$\begin{aligned}
\log L(\theta_1, \theta_{i,1}, \theta_{i,0}) &= \sum_{j=1}^{m} [y_j \log \theta_1 + (1 - y_j) \log(1 - \theta_1) \\
&+ \sum_{i=1}^{n} y_j \left(x_{j,i} \log \theta_{i,1} + (1 - x_{j,i}) \log(1 - \theta_{i,1}) \right) \\
&+ \sum_{i=1}^{n} (1 - y_j) \left(x_{j,i} \log \theta_{i,0} + (1 - x_{j,i}) \log(1 - \theta_{i,0}) \right)]
\end{aligned}$$

To estimate θ_1, we take the derivative of $\log L$ wrt θ_1 and set it to 0:

$$\frac{\partial L}{\partial \theta_1} = \sum_{j=1}^{m} \left(\frac{y_j}{\theta_1} + \frac{1 - y_j}{1 - \theta_1}(-1) \right) = 0$$

Maximum likelihood parameters estimation for naive Bayes

By solving for θ_1, we get:

$$\theta_1 = \frac{1}{m}\sum_{j=1}^{m} y_j = \frac{\text{number of examples of class 1}}{\text{total number of examples}}$$

Using a similar derivation, we get:

$$\begin{aligned}
\theta_{i,1} &= \frac{\text{number of instances for which } x_{j,i} = 1 \text{ and } y_j = 1}{\text{number of instances for which } y_j = 1} \\
\theta_{i,0} &= \frac{\text{number of instances for which } x_{j,i} = 1 \text{ and } y_j = 0}{\text{number of instances for which } y_j = 0}
\end{aligned}$$

Text classification revisited

- Consider again the text classification example, where the features x_i correspond to words
- Using the approach above, we can compute probabilities for all the words which appear in the document collection
- But what about words that do not appear?

  They would be assigned zero probability!
- As a result, the probability estimates for documents containing such words would be $0/0$ for both classes, and hence no decision can be made

Laplace smoothing

- Instead of the maximum likelihood estimate:

$$\theta_{i,1} = \frac{\text{number of instances for which } x_{j,i} = 1 \text{ and } y_j = 1}{\text{number of instances for which } y_j = 1}$$

use:

$$\theta_{i,1} = \frac{(\text{number of instances for which } x_{j,i} = 1 \text{ and } y_j = 1) + 1}{(\text{number of instances for which } y_j = 1) + 2}$$

- Hence, if a word does not appear at all in the documents, it will be assigned prior probability 0.5.
- If a word appears in a lot of documents, this estimate is only slightly different from max. likelihood.
- This is an example of *Bayesian prior* for Naive Bayes

Example: 20 newsgroups

Given 1000 training documents from each group, learn to classify new documents according to which newsgroup they came from

comp.graphics	misc.forsale
comp.os.ms-windows.misc	rec.autos
comp.sys.ibm.pc.hardware	rec.motorcycles
comp.sys.mac.hardware	rec.sport.baseball
comp.windows.x	rec.sport.hockey
alt.atheism	sci.space
soc.religion.christian	sci.crypt
talk.religion.misc	sci.electronics
talk.politics.mideast	sci.med
talk.politics.misc	talk.politics.guns

Naive Bayes: 89% classification accuracy - comparable to other state-of-art methods

Computing joint probabilities of word sequences

- Suppose you model a sentence as a sequence of words $w_1, \ldots w_n$
- How do we compute the probability of the sentence, $P(w_1, \ldots w_n)$?

$$P(w_1)P(w_2|w_1)P(w_3|w_2, w_1)\cdots P(w_n|w_{n-1}\cdots w_1)$$

- These have to be estimated from data
- *But data can be sparse!*

n-grams

- We make a *conditional independence assumption*: each words depends only on the n words preceding it, not on anything before
- This is a Markovian assumption!
- 1-st order model: $P(w_i|w_{i-1})$ - bigram model
- 2nd order Markov model: $P(w_i|w_{i-1}, w_{i-2})$ - trigram model
- Now we can get a lot more data!

Application: Speech recognition

- Input: wave sound file
- Output: typed text representing the words
- To disambiguate the next word, one can use n-gram models to predict the most likely next word, based on the past words
- n-gram model is typically learned from past data
- This idea is at the core of many speech recognizers

NLP tasks related to the Internet

- *Information retrieval (IR):* give a word query, retrieve documents that are *relevant* to the query

  Most well understood and studied task

- *Information filtering (text categorization):* group documents based on topics/categories
  - E.g. Yahoo categories for browsing
  - E.g. E-mail filters
  - News services

- *Information extraction:* given a text, get relevant information in a template. Closest to language understanding

  E.g. House advertisements (get location, price, features)

  E.g. Contact information for companies

How can we do information retrieval?

- Two basic approaches
  - Exact matching (logical approach)
  - Approximate (inexact) matching
- The exact match approaches do not work well at all!
  - Most often, no documents are retrieved, because the query is too restrictive.
  - Hard to tell for the user which terms to drop in order to get results.

Basic idea of inexact matching systems

- We are given a collection of documents
- Each document is a collection of words
- The query is also a collection of words
- We want to retrieve the documents which are "closest" to the query
- The trick is how to get a good distance metric!

Key assumption: If a word occurs very frequently in a document compared to its frequency in the entire collection of documents, then the document is "about" that word.

Processing documents for IR

1. Assign every new document an ID
2. Break the document into words
3. Eliminate stopwords and do stemming
4. Do term weighting

Details of document processing

- Stopwords very frequently occurring words that do not have a lot of meaning

  E.g. Articles: the, a, these... and Prepositions: on, in, ...

- Stemming (also known as suffix removal) is designed to take care of different conjugations and declinations. E.g. eliminating 's' for the plural, -ing and -ed terminations, etc.

  Example: after stemming, win, wins, won and winning will all become WIN

How should we weight the words in a document???

Term weighting

Key assumption: If a word occurs very frequently in a document compared to its frequency in the entire collection of documents, then the document is "about" that word.

- Term frequency:

$$\frac{\text{Number of times term occurs in the document}}{\text{Total number of terms in the document}}, \text{ or}$$

$$\frac{\log(\text{Number of times term occurs in the document}+1)}{\log(\text{Total number of terms in the document})}$$

  This tells us if terms occur frequently, but does not tell us if the occur "unusually" frequently.

- Inverse document frequency:

$$\log \frac{\text{Number of documents in collection}}{\text{Number of documents in which the term occurs at least once}}$$

Processing queries for IR

We have to do the same things to the queries as we do to the documents!

1. Break into words
2. Stopword elimination and stemming
3. Retrieve all documents containing any of the query words
4. *Rank the documents*

   To rank the documents, for a simple query, we compute:

   Term frequency * Inverse document frequency

   for each term. Then we sum them up!

   More complicated formulas if the query contains '+' '-', phrases etc.

Example

Query: "The destruction of the Amazonian rain forests"

1. Case normalization: "the destruction of the Amazonian rain forests"
2. Stopword removal: "destruction Amazonian rain forests"
3. Stemming: "destruction amazon rain forest"
4. Then we apply our formula!

Note: Certain terms in the query will inherently be more important than others

E.g. amazon vs. rain

Evaluating IR Systems

- Two measures:
  - *Precision*: ratio of the number of relevant document retrieved over the total number of documents retrieved
  - *Recall*: ratio of relevant documents retrieved for a given query over the number of relevant documents for that query in the database.
- Both precision and recall are between 0 and 1 (close to 1 is better).
- People are used to judge the ‘correct” label of a document, but they are subjective and may disagree
- Bad news: usually high precision means low recall and vice versa

Why is statistical NLP good?

- Universal! Can be applied to any collection of documents, in any language, and no matter how it is structured
- In contrast, knowledge-based NLP systems work ONLY for specialized collections
- Very robust to language mistakes (e.g. bad syntax)
- Most of the time, you get at least some relevant documents

Why do we still have research in NLP?

- Statistical NLP is *not really language understanding*! Are word counts all that language is about?
- Syntax knowledge could be very helpful sometimes

  There are some attempts now to incorporate knowledge in statistical NLP
- Eliminating prepositions means that we cannot really understand the meaning anymore
- One can trick the system by overloading the document with certain terms, although they do not get displayed on the screen.
- If a word has more than one meaning, you get a very varied collection of documents...

AI techniques directly applicable to web text processing

- Learning:
  - Clustering: group documents, detect outliers
  - Naive Bayes: classify a document
  - Neural nets
- Probabilistic reasoning: each word can be considered as "evidence", try to infer what the text is about