Advanced Natural Language Processing
Lecture 26
Question Answering and Word Sense Disambiguation

Johanna Moore (some slides by Philipp Koehn)

26 November 2013
Question Answering

- We would like to build
  - a machine that answers questions in natural language
  - may have access to knowledge bases, dictionaries, thesauri
  - may have access to vast quantities of English text

- Basically, a smarter Google. IBM Watson.

- This task is typically called **Question Answering**

- What will we need to be able to do this?
Example Question

• Question
  
  When was Barack Obama born?

• Text available to the machine
  
  Barack Obama was born on August 4, 1961

• This is easy.
  
  – just phrase a Google query properly:
    "Barack Obama was born on *"
  – syntactic rules that convert questions into statements are straight-forward
Example Question (2)

• Question
  What kind of plants grow in Scotland?

• Text available to the machine
  A new chemical plant was opened in Scotland.
  Heather is just one of the many plants that grow in Scotland.

• Why is this hard?
  – words may have different meanings
  – we need to be able to disambiguate them
Example Question (3)

- Question
  
  Do the police use dogs to sniff for drugs?

- Text available to the machine

  The police use canines to sniff for drugs.

- Why is this hard?

  - words may have the “same” meaning (synonyms, hyponyms)
  - we need to be able to match them
Example Question (4)

• Question
  Which animals love to swim?

• Text available to the machine
  Ice bears love to swim in the freezing waters of the Arctic.

• Why is this hard?
  – some words belong to groups that are referred to by other words
  – we need to have database of such A is-a B relationships, such as the WordNet object hierarchy
Example Question (5)

- **Question**
  
  What is the name of George Bush’s poodle?

- **Text available to the machine**
  
  President George Bush has a terrier called Barney.

- **Why is this hard?**
  
  - we need to know that *poodle* and *terrier* are related—they share a common ancestor in a taxonomy such as the WordNet object hierarchy
  - words need to be grouped together into semantically related classes
Example Question (6)

- **Question**
  Did Poland reduce its carbon emissions since 1989?

- **Text available to the machine**
  Due to the collapse of the industrial sector after the end of communism in 1989, all countries in Central Europe saw a fall in carbon emissions. Poland is a country in Central Europe.

- **Why is this hard?**
  - We need to do logical inference to relate the two sentences
Word Sense Disambiguation (WSD)

An important capability for automated question answering is word sense disambiguation, i.e., the ability to select the correct sense for each word in a given context.

What types of plants grow in Scotland?

There are many approaches to this problem:

- Constraint satisfaction approaches
- Dictionary approaches
- Supervised ML
- Unsupervised ML
Constraint Satisfaction

Three cases:

- Disambiguate an argument by using the selectional restrictions from an unambiguous predicate.

- Disambiguate a predicate by using the selectional restrictions from an unambiguous argument.

- Mutual disambiguation of an argument and a predicate.
Constraint Satisfaction Examples

Disambiguating arguments using predicates:

“In our house, everybody has a career and none of them includes washing dishes,” he says.

In her tiny kitchen at home, Ms. Chen works efficiently, stir-frying several simple dishes, including braised pig’s ears and chicken livers with green peppers.

Disambiguate dishes using the selectional restrictions that predicates washing and stir-fry place on their arguments
Constraint Satisfaction Examples

Disambiguating predicates using arguments:

1. Well, there was the time they served green-lipped mussels from New Zealand.

2. Which airlines serve Denver?

3. Which ones serve breakfast?

Sense of serve in 1 requires its THEME to be edible
Sense of serve in 2 requires its THEME to be a geographical entity
Sense of serve in 3 requires its THEME to be a meal designator
Constraint Satisfaction Examples

Mutual disambiguation

I’m looking for a restaurant that serves vegetarian dishes.

Assuming 3 senses of serve and 2 of dishes gives 6 possible sense combinations, but only 1 satisfies all selectional restrictions.
Problems with Constraint Satisfaction Approach

- Need to parse to get the verb-argument information needed to make it work
- Scaling up to large numbers of words (WordNet helps with this)
- Getting details of all selectional restrictions correct
- Wider context can sanction violation of selectional restriction

  But it fell apart in 1931, perhaps because people realized you can’t eat gold for lunch.

- Metaphorical uses may violate the constraints

  If you want to kill the Soviet Union, get it to try to eat Afghanistan.
WSD as a Classification Problem

Take a vector representing a word use in context and assign it to the class representing the right sense of that word.

Assume corpus of texts with words labeled with their senses
- She pays 3% interest/INTEREST-MONEY on the loan.
- He showed a lot of interest/INTEREST-CURIOSITY in the painting.

Similar to POS tagging
- given a corpus tagged with senses
- identify features that indicate one sense over another
- learn a model that predicts the correct sense given the features

We can apply similar supervised learning methods
- Naive Bayes
- Decision lists/trees, etc.
What features are useful for WSD?

“If one examines the words in a book, one at a time as through an opaque mask with a hole in it one word wide, then it is obviously impossible to determine, one at a time, the meaning of the words. . . But if one lengthens the slit in the opaque mask, until one can see not only the central word in question but also say N words on either side, then if N is large enough one can unambiguously decide the meaning of the central word. . . The practical question is: "What minimum value of N will, at least in a tolerable fraction of cases, lead to the correct choice of meaning for the central word?"

Warren Weaver, A Synopsis of Linguistic Theory, 1955
Feature Extraction: Collocational Features

Collocational Features: information about words in *specific* positions to the left or right of the target word

- **plant** life
- **plant** closure
- manufacturing **plant**
- assembly **plant**

Features extracted for context words:

- word itself
- POS
Example

An electric guitar and **bass** player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps.

Collocational feature vector extracted from window of 2 words (+ POS tags) to right and left of target word.

[guitar, NN1, and, CJC, player, NN1, stand, VVB]
Feature Extraction: Bag of Words Features

Bag of Words Features: all content words in an N-word window

E.g., vector of binary features indicating whether word $w$, from vocabulary, $V$, occurs in context window

An electric guitar and bass player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps.

$V = [\text{fishing, big, sound, player, fly, rod, pound, double, runs, playing, guitar, band}]$

Window = 10

Bag of Words Feature Vector:

$[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0]$
More about Features

Most approaches use a combination of collocational and bag-of-word features.

- collocation encodes local lexical and grammatical information
- bag of word features give the general topic of the discourse

Of course, many other features may be included:

- Syntactically related words
- Syntactic role in sense
Supervised Learning Approaches to WSD

Learn a WSD model from a representative set of labeled instances from the same distribution as the test set

• input is a training set consisting of feature-encoded inputs labeled with the appropriate sense

• output is a classifier that assigns labels to new, unseen feature-encoded inputs
Naive Bayes Classifiers

Choose the most probable sense, $\hat{s}$, from the possible senses, $S$, for a given feature vector, $V = v_1, v_2, \ldots v_n$

$$\hat{s} = \arg\max_{s \in S} P(s|V)$$

rewriting and assuming independent features yields:

$$\hat{s} = \arg\max_{s \in S} P(s) \prod_{j=1}^{n} P(v_j|s)$$

I.e., we can estimate the probability of an entire vector given a sense by the product of the probabilities of its individual features given that sense.
Naive Bayes Classifiers

Where do the numbers come from?

From a tagged corpus.

- For example, the probability of *guitar* occurring one position to the right of each sense of the word *bass* is computed from the corpus.

73% on the line task, but ensemble of simple NB classifiers gets 89% (Pederson, 2000)

One problem with Naive Bayes is that it’s hard for humans to understand how it makes its decisions.
Decision List Classifiers

**Decision List classifiers** are simple to learn/train and are often effective.

Equivalent to case statements in programming languages, i.e., they consist of an ordered set of conditions with simple conclusions:

- A sequence of tests is applied to each target-word feature vector
- Each test is indicative of a particular sense
- If a test succeeds, that sense is returned, otherwise the next test in the sequence is applied
- At end of sequence, majority class is returned

Easier for people to understand.
## Example Decision List

<table>
<thead>
<tr>
<th>Rule</th>
<th>Sense</th>
</tr>
</thead>
<tbody>
<tr>
<td>fish within window</td>
<td>$\Rightarrow$ bass$^1$</td>
</tr>
<tr>
<td>striped bass</td>
<td>$\Rightarrow$ bass$^1$</td>
</tr>
<tr>
<td>guitar within window</td>
<td>$\Rightarrow$ bass$^2$</td>
</tr>
<tr>
<td>bass player</td>
<td>$\Rightarrow$ bass$^2$</td>
</tr>
<tr>
<td>piano within window</td>
<td>$\Rightarrow$ bass$^2$</td>
</tr>
<tr>
<td>tenor within window</td>
<td>$\Rightarrow$ bass$^2$</td>
</tr>
<tr>
<td>sea bass</td>
<td>$\Rightarrow$ bass$^1$</td>
</tr>
<tr>
<td>play/V bass</td>
<td>$\Rightarrow$ bass$^2$</td>
</tr>
<tr>
<td>river within window</td>
<td>$\Rightarrow$ bass$^1$</td>
</tr>
<tr>
<td>violin within window</td>
<td>$\Rightarrow$ bass$^2$</td>
</tr>
<tr>
<td>salmon within window</td>
<td>$\Rightarrow$ bass$^1$</td>
</tr>
<tr>
<td>on bass</td>
<td>$\Rightarrow$ bass$^2$</td>
</tr>
<tr>
<td>bass are</td>
<td>$\Rightarrow$ bass$^1$</td>
</tr>
</tbody>
</table>
Learning Decision Lists

Generate and order the sequence of tests based on characteristics of the training data

Yarowsky’s (1994) approach:

• Tests of the form: collocation $\rightarrow$ sense
  
  – example: manufacturing \textbf{plant} $\rightarrow$ \texttt{PLANT-FACTORY}

• Rank order tests based on their individual accuracy on the training data, defined as:

\[
\log \left( \frac{p(sense_A | collocation_i)}{p(sense_B | collocation_i)} \right)
\]

• Smoothing is important

95% accuracy on binary decision tasks.
Evaluation

WSD systems are usually developed and evaluated intrinsically, i.e., treated as if they were stand alone systems.

Evaluation metric is accuracy: the percentage of target words that are tagged correctly, i.e., there is an exact match with the hand-labeled tags.

Two major types

- Fine-grained tagging to a dictionary (e.g., WordNet) sense
- Coarse-grained binary tagging, e.g., musical vs. fish sense of bass

And what about partial credit? E.g., confusing a particular musical sense of bass (e.g., instrument) with a fish sense is clearly worse than confusing it with another musical sense (e.g., singer)
Evaluation

How do you choose which you want to do?
You choose based on your application.

Example: A text-to-speech system needs to know if it is a [baes] or a [beys]. It doesn’t need to know if its singer or a bass fiddle, or a fresh or saltwater fish.
Evaluation

Many aspects of WSD evaluation have been standardized by the SENSEVAL and SEMEVAL efforts.

Provide shared task with training and testing materials along with the sense inventories for WSD tasks in a variety of languages.
Evaluation

The bottom line is that classifiers all perform essentially the same on similar tasks in this domain.

naive Bayes, decision lists, decision trees, neural nets.

They differ in their end product (how inspectable is it?), how much data they need, and how long they take to run.

A big drawback is that they all need labelled data.

Now we’ll look at an unsupervised method for WSD.
The Story So Far

• We would like to build a machine that answers natural language questions, given a large collection of text

• We need to address
  – multiple word senses
  – synonyms
  – word classes
  – relationships between concepts
  – inference

• Open question
  – how much of this can be encoded by hand?
  – how much of this can be learned from data?
The meaning of a word is its use.

*Philosophical Investigations*

Ludwig Wittgenstein

You shall know a word by the company it keeps.

*A Synopsis of Linguistic Theory, 1957*

J. R. Firth
Unsupervised Word Sense Disambiguation

• The task
  – given a set of occurrences of a word in context
  – group them into classes (different senses)

• Example
  – bat/Sport occurs in context of player, pitcher, catcher, ballpark, win,...
  – bat/Animal occurs in context of cave, flying, night, vampires, blood,...

• Idea
  – contexts of the different occurrences of bat/Sport are similar
  – contexts of an occurrence of bat/Sport and an occurrence of bat/Animal are dissimilar
Unsupervised Word Sense Disambiguation

What we need

- For each token $W_i$ of word $w$ in a corpus, compute a context vector $\vec{c}$

- Use a clustering algorithm to cluster these word-token context vectors, $\vec{c}$ into a predefined number of clusters. Each cluster defines a sense of $w$

- Compute the vector centroid of each cluster. Each vector centroid $\vec{s}_j$ is a sense vector representing that sense of $w$
Vector Representation of Contexts

Contexts may be represented by a vector of word counts.

Example context
Then he grabbed his new mitt and bat, and headed back to the dugout for another turn at bat. Hulet isn’t your average baseball player. ”It might have been doctoring up a bat, grooving a bat with pennies or putting a little pine tar on the baseball. All the players were sitting around the dugout laughing at me.”

The word counts may be normalized, so all the vector components add up to one.
Unsupervised Word Sense Disambiguation

To disambiguate a particular token $t$ of $w$

- Compute a context vector $\vec{c}$ for $t$
- Retrieve all sense vectors $\vec{s}_j$ for $w$
- Assign $t$ to the sense represented by the sense vector $\vec{s}_j$ that is most similar to $t$
Clustering

- Unsupervised learning
  - **given**: a set of contexts around the ambiguous word
  - **wanted**: grouping into appropriate classes

- **Agglomerative clustering**
  - group the two most similar contexts together
  - repeat until
    - we have a pre-defined number of classes
    - the lowest similarity is higher than a pre-defined threshold

- **Complexity**
  - at any point in the Clustering algorithm, we have to compare every context with every other context
  - complexity **quadratic** in the number of contexts $O(n^2)$
Agglomerative Clustering

- Start: 9 contexts, 9 classes

• Contexts $c_3$ and $c_7$ are most similar

• Contexts $c_1$ and $c_5$ are most similar
Agglomerative Clustering (2)

- Contexts $c_6$ and $c_8$ are most similar

- Context $c_4$ and class $\{c_8, c_6\}$ are most similar
Agglomerative Clustering (3)

- Context $c_2$ and class $\{c_6, c_8\}$ are most similar

- Class $c_9$ and class $\{c_3, c_4, c_7\}$ are most similar
Agglomerative Clustering (4)

- Class \(\{c_1, c_5\}\) and class \(\{c_2, c_6, c_8\}\) are most similar

- If we stop now, we have two classes
Clusters of Contexts

[from Schütze, 1998]
A Similarity Metric

How do we know how similar two contexts are?

- One popular similarity metric for vectors is the *cosine*

\[
\text{sim}(\vec{x}, \vec{y}) = \frac{\sum_{i=1}^{m} x_i \times y_i}{\sqrt{\sum_{i=1}^{m} x_i \times \sum_{i=1}^{m} y_i}} = \vec{x} \cdot \vec{y}
\]

- Many other similarity metrics have been defined (see Jurafsky & Martin, Section 20.7)
Summary

- Word Sense Disambiguation is the task of determining the correct sense of a word in context.
- Supervised approaches require sense-tagged training data.
- Classifiers for WSD are trained on collocational and bag-of-words features.
- Unsupervised approaches use clustering and a metric for computing similarity between vectors.
- There are many other clustering methods and other merging strategies, see Jurafsky and Martin Chap 20.