Computational Semantics and Pragmatics

Autumn 2011

Raquel Fernández

Institute for Logic, Language & Computation
University of Amsterdam

Plan for Today

We'll discuss the main features and differences between supervised and unsupervised learning methods

As a case study we'll consider word sense disambiguation (WSD): the task of determining which sense of a word is being used in a particular context.

ML in Computational Linguistics

- In computational linguistics we use machine learning (ML) techniques to model the ability to classify linguistic objects (in a very broad sense) into classes or categories – the ability to categorise.
- Of course, often ML is used with a practical motivation, to get a particular NLP task done in an effective way.
- But ML techniques can also be a powerful tool for analysing natural language data from a more theoretical point of view:

  $\Rightarrow$ they can help to clarify the patterns in a complex set of observations and at the same time shed light on the underlying processes that lead to those patterns.

Supervised vs. Unsupervised Learning

ML, in all its modalities, always involves a training phase where a model is learned from exposure to data, and a testing phase where new, previously un-seen data are classified according to the model.

- In supervised learning, the learning algorithms are trained on data annotated with the classes we want to be able to predict.
  - ∗ in supervised WSD, the data would be a corpus where uses of the target words have been hand-labelled with their senses.
- In unsupervised learning, the algorithms are trained on data that is not annotated with specific classes; they must learn the classes by identifying patterns in un-annotated data.
  - ∗ in unsupervised WSD, words are not labelled and we don't know a priori which senses a word may have.
- There are also semi-supervised forms of learning that require only small amounts of annotated training data.

Possible Classification Tasks

A few semantic/pragmatic tasks that can be approached as classification learning tasks:

- textual entailment (binary: TRUE / FALSE)
- word sense disambiguation (multi-class)
- semantic relations (multi-class)
- correference resolution (can be conceptualised as a binary task)
- dialogue act tagging (multi-class)
- polarity of indirect answers (binary: POS / NEG)
- generation (e.g. article or pronoun generation – multi-class)
- . . .

Data for Supervised Learning: Annotation

Supervised learning requires humans annotating corpora by hand. This is not only costly and time-consuming... Can we rely on the judgements of one single individual?

- an annotation is considered reliable if several annotators agree sufficiently – they consistently make the same decisions.

Several measures of inter-annotator agreement have been proposed. One of the most commonly used is Cohen's *kappa* (κ). κ measures how much coders agree correcting for chance agreement

$$\kappa = \frac{A_o - A_e}{1 - A_e}$$

A_o: observed agreement
A_e: expected agreement by chance

$\kappa = 1$: perfect agreement
$\kappa = 0$: no agreement beyond chance

There are several ways to compute A_e. For further details, see:

Arstein & Poesio (2008) Survey Article: Inter-Coder Agreement for Computational Linguistics, *Computational Linguistics*, 34(4):555–596.

Data for Supervised Learning: Annotation

- We use whatever portion of the corpus has been annotated by multiple annotators to compute a κ score that measures the reliability of the annotation.
- To train (and later test) an automatic classifier, we only use the classification done by **one** of the annotators (possibly an expert on the topic) – the particular version of the annotation used is considered the gold standard.

Supervised WSD

The SENSEVAL project http://www.senseval.org/ has produced a number of freely available hand-labelled datasets where words are labelled with their "correct" senses.

These datasets can be used to develop supervised classifiers that can automatically predict the sense of a word in context. This involves:

- extracting features that we hypothesise are helpful for predicting senses — each word token (i.e. each use in a particular context) is represented by a feature vector;
- training a classification algorithm on the feature vectors annotated with the hand-labelled senses;
- testing the performance of the algorithm by using it to predict the right sense of un-seen word tokens (feature vectors) whose hand-labelled sense is not made available to the algorithm.

Features for Supervised WSD

From Weaver (1955) in the context of machine translation:

> *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?"*

This contextual information can be encoded as features (numeric or nominal) within a feature vector associated with each target word.

- Collocational features: information about words in specific positions with respect to the target word
- Co-occurrence features: information about the frequency of co-occurrence of the target word with other pre-selected words within a context window ignoring position ($\sim$ *similar to DSMs*)

Features for Supervised WSD: Example

- For instance, consider the following example sentence with target word w_i = **bass**:

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.

- Example of possible collocational features:

w_{i-2},	POSw_{i-2},	w_{i-1},	POSw_{i-1},	w_{i+1},	POSw_{i+1},	w_{i+2},	POSw_{i+2}
⟨ guitar,	N,	and,	C,	player,	N,	stand,	V ⟩

- Example of possible co-occurrence features:

fishing,	big,	sound,	player,	fly,	rod,	pound,	double,	guitar,	band
⟨ 0,	0,	0,	1,	0,	0,	0,	0,	1,	0 ⟩

- These two types of feature vectors can be joined into one long vector.

Types of Algorithms

- Many types of algorithms can be used for classification-based supervised learning: Maximum Entropy, Decision Trees, Memory-based learning, Support Vector Machines, etc.
- Essentially, they all estimate the likelyhood that a particular instance belongs to class `C` given a set of observations encoded in the feature vector characterising that instance.

instances in training dataset:		un-seen instances in testing dataset:	
< feature vector >	C_1	< feature vector >	?
< feature vector >	C_2	< feature vector >	?
< feature vector >	C_1	< feature vector >	?
< feature vector >	C_3	< feature vector >	?

- If you are interested in the inner workings of particular algorithms, two rather accessible sources of information are:

Manning & Schütze (1999) *Foundations of Statistical Natural Language Processing*, MIT Press.
Witten, Frank & Hall (2011) *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann.

Evaluation: Partitioning the Data

The development and evaluation of an automated learning system involves partitioning the data into the following **disjoint** subsets:

- Training data: data used for developing the system's capabilities
- Development data: possibly some data is held out for use in formative evaluation for developing and improving the system
- Test data: data used to evaluate the system's performance after development (what you report on your paper).

This split could correspond to 70, 20, and 10 percent of the overall data, for training, development, and testing, respectively.

Evaluation: Cross-Validation

If only a small quantity of annotated data is available, it is common to use cross-validation for training and evaluation.

- the data is partitioned into k sets or *folds* (ofetn $k = 10$)
- training and testing are done k times, each time using a different fold for evaluation and the remaining $k - 1$ folds for training
- the mean of the k tests is taken as final results

To use the data even more efficiently, we can set k to the total number N of items in the data set so that each fold involves $N - 1$ items for training and 1 for testing.

- this form of cross-validation is known as leave-one-out.

In cross-validation, every items gets used for both training and testing. This avoids arbitrary splits that by chance may lead to biased results.

Evaluation Measures

Measures for reporting the system's performance on the test data:

- Accuracy: percentage of instance where the class hypothesised by the system matches the gold standard label.
- Error rate: the inverse of accuracy $1 - A$

Measures computed per class:

- Precision: proportion of correct system hypotheses for class C given the total of sytem hypotheses for that class.
- Recall: proportion of correct system hypotheses for class C given the total number of items labelled with class C in the gold standard.
- F-measure: combination of precision and recall

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

Typically $\beta = 1$
$\beta > 1$ favours recall, while $\beta < 1$ favours precision.

Lower and Upper Bounds

The system's performance needs to be compared to some baseline or lower bound. The results of your system will be more convincing the more it improves over a more challenging baseline.

A baseline can be the accuracy or F-measure achieved by e.g.:

- a random classifier
- a majority class classifier: always choose the most frequent class
- a basic algorithm

Human inter-annotator agreement (the κ score of a multi-coder annoation) can be taken to define an upper bound for the performance of an automatic system:

- we can expect that an automatic system will agree with the gold standard only as much as other humans are able to agree with it.
- when results on inter-annotator agreement are not available, there is typically no upper bound – the upper bound is then to achieve 100% accuracy.

Feature Analysis

A very important part of developing automatic classifiers is the selection of a predictive set of features — theoretical and linguistic insights can help us to come up with interesting features.

Once we have our set of features, we want to investigate which features have the most predictive power. Two possible methods:

- Feature ablation: remove one single feature at a time and re-train and re-test the classifier to compare results with and without that feature.
- Information gain: if we know the value of feature F, how much does that reduce our uncertainty regarding the correct class X?
    - $*$ the difference between the prior probability of X (it's frequency) and the conditional probability $p(X \mid F)$ of X given F gives us the info gain of F for X
    - $*$ also called *Kullback-Leibler divergence* or *relative entropy*

$\Rightarrow$ coming up with well-motivated features and analysing their relative predictive power in a categorisation task is what makes supervised machine learning approaches interesting theoretically.

Unsupervised Learning

- In unsupervised learning the training data is not annotated with the properties that the algorithm is intended to produce as output.
- The algorithm is provided with the data alone and must learn some interesting structure through identifying patterns
- Choice of supervised vs. unsupervised learning:
  - ∗ From a practical or engineering perspective, we are interested in balancing the degree of accuracy achieved in proportion to the cost of resources it requires.
  - ∗ From a theoretical perspective, we may consider these issues:
    - ▸ do these methods tell us anything about the learning mechanisms humans employ in acquiring knowledge of their language?
    - ▸ can they be models of human language acquisition?

Unsupervised WSD

Why use unsupervised learning for WSD?

- It is expensive and difficult to build hand-labelled corpora.
- Hand-labelled senses may not be theoretically sound. Recall Kilgarriff's arguments:
  - ∗ defining a fix set of word senses may be impossible, and would at any rate be a domain-dependent task.
  - ∗ word senses should be reduced to abstractions over clusters of word usages.

In unsupervised WSD we do not start with a set of human-defined senses – the "senses" are created automatically from the instances of each word in the training set.

⇒ we can use a version of a DSM where we compute context vectors for each token of interest, i.e. for each usage, instead of computing vectors for *types* of target terms.

Unsupervised WSD

Training: creating “senses” from usages

- For each token t_w of word w in a corpus, compute a context vector $\mathbf{c}_{t_w}$
- Use a clustering algorithm to cluster the vectors into groups or clusters; each cluster defines a sense of w
- Compute the vector centroid (the average or arithmetic mean) of each cluster; each centroid is a vector $\mathbf{s}_{w_i}$ representing that sense of w

Prediction: disambiguating a token t_w of w by assigning it a sense

- Compute a context vector $\mathbf{v}_{t_w}$ for t_w
- Retrieve all sense vectors for w
- Assign to t_w the sense represented by the sense vector $\mathbf{s}_{w_i}$ that is closest to $\mathbf{v}_{t_w}$

This procedures requires a clustering algorithm and a distance metric to compare vectors.

Clustering

Clustering is a general term referring to the task of classifying a set of objects into groups (clusters) so that the objects in the same cluster are more similar to each other than to those in other clusters.

Several clustering algorithms exist. Two common techniques are:

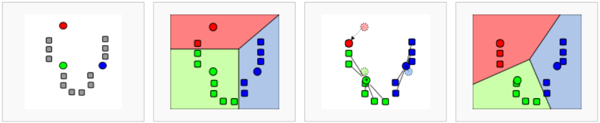
- k-means clustering
- Agglomerative hierarchical clustering

We will briefly review the basic steps involved in these two types of algorithms. For further details, you can consult these reference:

Manning & Schütze (1999) Foundations of Statistical Natural Language Processing, ch. 14: *Clustering*, MIT Press.
Jain, Murty & Flynn (1999) Data Clustering: A Review, *ACM Computing Surveys*, 31:264-323.

k-means Clustering: Basics

1. assume a certain number k of clusters;
2. select k objects that are as distant as possible from each other; these are the starting centroids of the clusters;
3. assign each remaining object to the cluster whose centroid is the closest;
4. when all objects have been assigned, recalculate the positions of the k centroids.
5. Repeat Steps 3 and 4 until the centroids are stable.

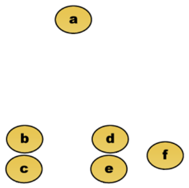


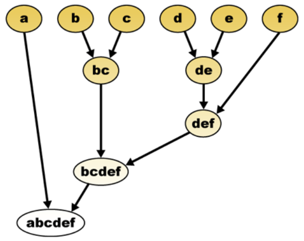
Picture from Wikipedia `http://en.wikipedia.org/wiki/K-means_algorithm`
There seems to be a mistake with red cluster, but good enough for illustration

Agglomerative Clustering: Basics

1. assign each training instance to its own cluster
2. compute the distance between the clusters and merge the most similar pair of clusters
   * similarity between clusters can be computer by taking the shortest, the longest, or the average distance
3. repeat step 2 until either a specified number of clusters is reached or the clusters have some desired property.

By repeating step 2 until all items belong to the same cluster we end up with a tree that can be cut at the desired level of specificity.





Evaluation of Unsupervised Predictions

In unsupervised learning we don't have a gold standard or ground truth against which we can compare the output of our system. Therefore evaluation can be tricky. . .

Some possibilities include:

- extrinsic evaluation: is the system's output positively evaluated by humans judgements?
- end-to-end evaluation: does the output of the system improve the performance of a larger task? (e.g. does unsupervised WSD improve machine translation?)
- if an annotated corpus exists, we can also do an intrinsic evaluation (such as those in supervised learning). For instance, for WSD:
  - map each cluster (induced sense) to the predefined sense that in the training set has most word tokens overlapping with the cluster; or
  - for all pairs of usages of a word in the test set, test whether the system and the hand-labels consider the pairs to have the same sense or not.

What's Next

Change of schedule:

- class on Thursday 27 Oct at the regular time; room TBA.
- break on Thursday 3 November - schedule an appointment with me to discuss your project; more on this next week.

Next week we'll turn to topics more related to pragmatics:

- Gricean pragmatics and conversational implicature
- Required readings (see the overview bibliography for links):
  - ∗ Grice (1975) Logic and Conversation. In *Syntax and Semantics 3: Speech Acts*, 43-58. New York: Academic Press.
  - ∗ Davis (2010) Implicature, The Stanford Encyclopaedia of Philosophy (Winter 2010 Edition), Edward N. Zalta (ed.).