Applications of Natural Language Processing

1 What are the applications of NLP?

Here is an incomplete list of tasks that involve processing natural languages to some degree or another. In each case, systems have been built that go some way to performing these tasks automatically.

- Spelling and grammar checking;
- machine translation and machine-aided translation;
- document classification (e.g. spam detection; author identification; plagiarism detection; etc.);
- information retrieval;
- question answering;
- summarisation;
- report generation (e.g. software documentation generation);
- natural language interfaces to databases;
- dialogue interfaces to intelligent agents (e.g. interfaces to non-player characters in games; recommender systems; diagnostic systems; etc.)
- …

Consider what may strike you as the simplest of these tasks: spell-checking. At first glance, surely all that is required is for the system to check whether each word in the document is in the system's lexicon or not! Such a simple-minded spell-checker will correctly identify the error in the following sentence:

(1) *"Three steps are neccesary."*

But, in fact, this is just the bare minimum of what is required. If a spell-checker is to realise that this sentence

(2) *"This is unnecessarily complicated."*

has no error, it will probably need to know the *morphological* rules of English. The 'word' *"unnecessarily"* is unlikely to be in the system's lexicon, not because it is incorrectly spelled but because, as we've discussed before, a typical lexicon contains only word roots and the system needs to realise that *"unnecessarily"* is a correct combination of un + necessary + ly.

This next sentence does have a spelling error but a spell-checker would need *syntactic* knowledge to find it:

(3) *"I don't know weather this is correct."*

The sentence requires a conjunction (presumably *"whether"*) where the author has written the correctly-spelled noun *"weather"*.

To realise that *"principle"* and *"principal"* should probably be swapped in sentences (4) and (5) requires an appreciation of their meaning (and, quite possibly, some world knowledge):

(4) *"The principle of the school asked to speak with me."*

(5) *"The principal of the school is 'Carpe Diem' - 'Seize the Day' "*

In the next example, the system would have to inspect other sentences to know whether *"principal"* has been spelled correctly!

(6) *"The principal of the school is simple… "*

Am I arguing that to build spell-checkers we need to solve all the problems of natural language processing? Of course not! *Any* spell-checker is better than none: anything that assists us in our writing, even if it is not perfect, is useful. And this makes an important point. We are a long way from solving all the problems of natural language processing (if we ever will). But useful systems can be built now.

There is arguably a continuum of systems, ranging from those that do 'shallow NLP' to those that (try to) do 'deep NLP'.

2 'Shallow' and 'Deep' NLP

A 'deep' NLP system is one that attempts to build a meaning representation (usually wffs of logic) for the sentences it processes. Hence, at the very least it has a semantics module (which, if you believe that syntax drives implies that it also has a syntax module, i.e. a parser). It may also have modules for pragmatics and for bringing world knowledge to bear.

But there is a continuum of *useful* systems that get by with less than this. These systems are, in a non-pejorative sense, 'shallow' to different degrees. Shallow systems try to carry out their tasks without building full meaning representations. While the difficulties of building successful deep systems have become ever more apparent (e.g. the vast amount of linguistic and non-linguistic knowledge that is needed), there has grown an increasing amount of research and development into shallower systems, with the goal of building useful systems that would work well enough across wider ranges of text.

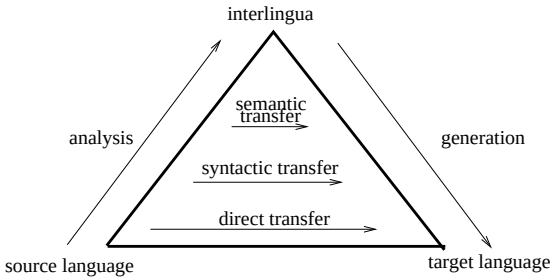
Over the same period, we have seen an increasing use of machine learning to obtain the knowledge that NLP systems, especially shallow NLP systems, use. Learning the knowledge, where possible, is obviously preferable to hand-coding it.

And over the same period, statistical and probabilistic approaches have grown in popularity. The probabilities are relatively easily 'learned' from bodies of text. We can thank the Web in part for making this machine-readable text plentifully available.

We will illustrate some of these developments with reference to machine translation systems.

3 Machine Translation

Work on Machine Translation (MT) started in the 1950s, with a focus on translating between Russian and English. The different approaches, from shallowest to deepest, can be illustrated, in a simplistic way, using the Vauquois Triangle:



The deepest approach (which has never been achieved in practice) is to analyse the source document as much as possible (syntax, semantics, pragmatics, etc.), producing a meaning representation in a language referred to as an *interlingua*. This is typically to be thought of as some kind of logic (or, equivalently, knowledge representation language) that is as language-neutral as possible: the language of 'pure thought'! Then, similar processes are applied, in reverse, to produce the corresponding document in the target language.

But there is a range of shallower approaches. These often work adequately well for the purposes to which we put them. For example, a rough translation of a scientific publication from Russian to English by a shallow MT system will probably be sufficient for the English reader to get the 'gist' of the article, to determine whether the article is relevant, to determine its main contribution and to decide whether to commission a high-quality human translation.

The simplest shallow approach (at the bottom of the Triangle) is called direct transfer: take each word in the source, find a corresponding word in a bilingual dictionary, and replace one by the other. Here's how this might work in the case of English to French:

(7) *The man sees the woman.*"

(8) *"La homme voit la femme.*"

In this example, the translation is nearly correct! In fact, often the output will be intelligible. But very frequently the output will be ungrammatical; occasionally, it will be word salad!

Even slightly less shallow approaches can improve the quality of the translation. For example, suppose we do some syntactic analysis. Maybe we don't *parse* the sentences and build phrase-structure trees. Maybe all we do is work out the categories of the words in the sentence. (This is called part-of-speech tagging, see below.) This will help! Suppose the word we are translating is a noun. Suppose too that the word has multiple translations, but only one of these is a noun. Then, this is the one we should probably choose. For example, in the next sentence, knowing that here *"sleep"* is being used as a verb is enough to know to translate it using the verb *"dormir"* and not the noun *"sommeil"*:

(9) *"I sleep well.*" i.e. Pro V Adv

(10) *Je dors bien.*"

(Similarly, the adverb *"well'* gets translated by adverb *"bien"* and not noun *"puits"*.)

We can also equip the system with transformation rules that rely on knowing the categories of the words. For example, if the source is English and contains an adjective followed by a noun, then the output in French can place the adjective after the noun (which will often be correct — but not always).

(11) *"red wine"*

(12) Adj N

(13) *"vin rouge"* i.e. N Adj, not *"rouge vin"* i.e. Adj N

If we could work out the likely meanings of the words (word sense disambiguation, see below), then this might help further. In the following example,

(14) *"I keep my money in the bank"*

by realising that *"bank"* here is most likely a financial institution and not a river bank, we would know to translate it as *"banque"* and not *"rive"*.

The most amazing of these shallow MT systems is Systran, which was developed in the 1960s, has been continually updated, has been used in numerous commercial and government organisations, and is still going strong. It has recently gained a new lease of life since it powers AltaVista's BabelFish for translating web pages. Systran uses a combination of simple syntactic transfer, ad hoc rules and a vast lexicon of pre-translated phrases, resorting to direct transfer when all else fails.

We've mentioned part-of-speech tagging and word sense disambiguation. In the next two sections, we'll discuss (shallow) ways in which these can be done. (Note these are not necessarily techniques used by Systran.)

Class Exercise. Imagine the European Union has an MT system than can translate between every pair of national languages. One day, the Irish successfully argue that Irish Gaelic is their national language and should therefore be a language recognised by the EU and added to the MT system. Which do you think would be the easier kind of MT system to update: one based on shallow processing or one based on an interlingua?

4 Part-of-Speech Tagging

Part-of-speech tagging (POS tagging) is not the same as parsing. In tagging, the goal is not to build a phrase-structure tree. It is simply to work out the most likely syntactic categories of the words in the sentence, e.g.:

(15) *"Ann saw the man with the telescope."* i.e. Name V Det N P Det N

Note that, while tagging involves much less work than parsing, it is not a simple look-up process. If we look up *"saw"* in a lexicon, we find that it is both a noun and a verb. A good POS tagger must realise that, in the sentence above, *"saw"* is most likely a verb (V).

An immediate observation is that this is a classic *classification* problem: given information about an object, predict which of a small, finite set of predefined classes it belongs to. Here, we are given a word and we want to predict its syntactic category (its part-of-speech), and there is a relatively small, finite number of these.

In the earlier part of this module, we saw several ways of building classifiers. For POS tagging, probabilistic methods are the most common. The currently most effective methods work on the basis of n-grams, which just means that the $n-1$ previous words are used to guide the prediction. To keep the notation simple, we'll look at a *bigram* approach, in which just the previous word is used. (And we won't get bogged down in what happens with the first word of a document, which has no previous word!)

So, given a word w_i, we want to find out how probable it is that this word is, for example, a noun (N), given the previous word w_{i-1},

$$P(\textit{class} = N | \textit{current} = w_i, \textit{previous} = w_{i-1})$$

We want to do the same for all the other possible categories that w_i can have (e.g. V, Det, Pro, etc.), and then choose the one that has the highest probability. If we let L be all of w_i's possible categories, then we want to compute the above probability for each $cl \in L$ and choose the cl with the highest probability.

(The rest of this section is revision! It simply explains the naïve Bayes classifier again, applied to POS tagging.)

Since it's unlikely that we have the joint probability distribution, we reformulate using Bayes' rule:

$$P(\textit{class} = cl|\textit{current} = w_i, \textit{previous} = w_{i-1}) = \frac{P(\textit{current} = w_i, \textit{previous} = w_{i-1}|\textit{class} = cl) \times P(\textit{class} = cl)}{P(\textit{current} = w_i, \textit{previous} = w_{i-1})}$$

It's unlikely we have these probabilities either! But, we make the simplifying assumption that the value of *current* and *previous* are conditionally independent given the class. In other words, we use the naïve Bayes classifier. What we need to compute is:

$$= \frac{P(\textit{current} = w_i|\textit{class} = cl) \times P(\textit{previous} = w_{i-1}|\textit{class} = cl) \times P(\textit{class} = cl)}{P(\textit{current} = w_i, \textit{previous} = w_{i-1})}$$

for each $cl \in L$. Remember we don't need the divisor because it will be the same for each category $cl \in L$.

Where will we get these probabilities? We must take a large body of text and ask a human to tag it manually. Then we can obtain the probabilities by computing relative frequencies. Note that many of the probabilities will be zero, and this is a problem for this classifier. However, in our earlier lecture on this topic we briefly discussed the remedies for this.

In practice, POS taggers often use trigrams, not bigrams, i.e. they base the classification on the previous *two* words. These taggers typically have about a 95% success rate. This sounds great until you learn that a unigram approach (i.e. taking no previous words into account, just basing the prediction on the current word's most frequent tag) scores about 90%. So, the extra effort isn't buying a great deal of extra accuracy.

5 Word Sense Disambiguation

Word sense disambiguation (WSD) is also a classification problem (assuming that we believe that each word has only a finite number of distinct meanings).

5.1 Naïve Bayes

We can, of course, use the naïve Bayes classifier again. This time, L will be all of word w_i's possible meanings. The formulae are all otherwise the same. However, if we are to obtain the probabilities by computing relative frequencies, we require humans to manually disambiguate large quantities of text. This is even more time-consuming than manual POS tagging.

I've no straightforward figures for how accurate naïve Bayes classifiers are for WSD. One much-quoted figure is that the unigram approach (as above, i.e. taking no previous words into account, just basing the prediction on the current word's most frequent meaning) scores about 75%.

5.2 Lesk's approach

Michael Lesk proposed a shallow, simple but quite effective WSD algorithm. Again to choose the meaning of word w_i it looks at surrounding words and, for simplicity, we will assume it just looks at the preceding word, w_{i-1}.

The algorithm requires access to a machine-readable dictionary. The different meanings of words w_i and w_{i-1} are retrieved from this dictionary. For each of w_i's meanings and for each of w_{i-1}'s meanings, we count the number of

words in common between the dictionary definitions. The pair of meanings that have the most in common are chosen to be the meanings of the two words.

Here's Lesk's example. Suppose the phrase *"pine cone"* appears in your document. *"pine"* and *"cone"* are both ambiguous. We look up each one in the dictionary and we find the following:

pine	1.	kind of evergreen tree with needle-shaped leaves…
	2.	waste away through sorrow or illness
cone	1.	solid body which narrows to a point…
	2.	something of this shape whether solid or hollow…
	3.	fruit of certain evergreen tree…

Now compute the size of the intersections between the pairs of meanings:

pine-1 ∩ cone-1	= 0		pine-2 ∩ cone-1	= 0
pine-1 ∩ cone-2	= 1		pine-2 ∩ cone-2	= 1
pine-1 ∩ cone-3	= 3		pine-2 ∩ cone-3	= 0

So the meanings are pine-1 and cone-3! (In the example, word such as *"of"* were counted. In practise, they wouldn't be.)

5.3 Sortal Restrictions

The final approach we will mention requires a somewhat deeper analysis.

We can group individuals into *classes*. For example, there are individuals that are physical objects and those that are abstract objects. And we can devise a *class hierarchy*. For example, the physical objects subdivide into the animate ones and inanimate ones; the abstract objects subdivide into the events, processes and states. These subclasses might further subdivide into their own subclasses.

It is clear that the arguments of a relation or a function are often restricted to individuals of certain classes. For example, the two-place 'drink' relation requires the drinker to be animate and the object drunk to be a fluid physical object; the one-place 'crawl' relation requires the crawler to be an animal. These restrictions on the sorts of the arguments of relations are referred to as *sortal restrictions* or *selection restrictions*.

If a sentence is ambiguous and one of its meanings violates sortal restrictions and the other does not, then we can choose the meaning that does not violate the sortal restrictions. For example,

(16) *"A bug crawled into the sugar bowl."*

contains an ambiguity: the word *"bug"* may denote insects, microphones or software errors. However, the sortal restriction on the argument of *"crawled"* dictates that only the first meaning gives rise to a meaning for the sentence as a whole.

Interestingly, not only does this help in WSD, it may help with other forms of ambiguity too. For example, it can help with resolving referents (which we discussed in the previous lecture):

(17) *Ann took her wine to the table and drank it.*"

We take *"it"* to refer to the wine and not the table because of the sortal restrictions on 'drink'. And it can help with structural ambiguity (where there is more than one phrase-structure tree). For example,

(18) *The man walked past the shop smoking a cigar.*"

has two phrase-structure trees: one in which "*smoking a cigar*" modifies "*walked past the shop*" and therefore tells us more about the way the man was acting, and another in which "*smoking a cigar*" modifies "*the shop*", but which violates sortal restrictions.

Intriguingly, some people claim that violated sortal restrictions help reveal non-literal uses of language. In sentence (19), for example,

(19) "*My car drinks petrol.*"

the literal reading would violate a sortal restriction, but a non-literal reading might be possible.

One major criticism of the use of sortal restrictions is that their enforcement is inflexible. If we make the constraints very specific (as may be needed for high discriminating power), they may reject perfectly acceptable utterances. But as we 'loosen' them to accommodate a wider range of uses of a verb, we may end up with vacuous constraints that are rarely violated. For example, what is the sortal restriction on the killer in the 'kill' relation? Human murderers, wild animals, missiles, diseases, and bad news: all can be killers. Similarly, what is the sortal restriction on the killed object in the 'kill' relation? Living things, time, proposals and interest can all be killed. The sortal restrictions would appear to be that the killer must be a thing, and the killed object must be a thing. This is no constraint at all! This has led to the proposal that these sortal restrictions be treated as *preferences* rather than as absolute restrictions. Violation of a sortal preference would not cause a sentence reading to be rejected. Instead, by keeping some measure of the number and severity of the violation in each reading, readings can be ranked.

6 Conclusions

One theme of this lecture has been how the 'Holy Grail' of 'deep' NLP is not necessary in the short-term for building *useful* systems. The other theme, however, has been that, if we can make the systems even a little deeper, then they will generally (but not always) perform better. We've illustrated this with reference to spell-checking and MT. Here's another illustration.

Recently, there has been a lot of interest in building systems for *Question Answering (QA)*. Everyone agrees that QA answering is different from Information Retrieval (IR), although there is not a sharp separation between the two. In QA, the user poses a question and s/he does not want a long list of possible relevant documents, through which s/he must search for the answer. Rather, it is the job of the system to extract the relevant information and present an answer to the question in a personalised presentation.

Without going into details, it is interesting to note the performances of various systems entered into a QA competition held in 2000. I will show you a graph in the lecture that demonstrated that, on the whole, the systems that used knowledge of language, even quite shallow knowledge, performed better at this task than the more traditional IR systems.

Exercise

1. What is meant by *compositional semantics*?
2. What is meant in compositional semantics by *the rule-to-rule hypothesis*?
3. Here is a Context-Free Phrase-Structure Grammar and lexicon for a fragment of the English language in which grammar rules and lexical entries are paired with semantic rules (using the notation from the lectures):

Grammar rule	Semantic rule	Word : Category : Semantics
$S \rightarrow NP\ VP$	$S' = VP'(NP')$	*Becks* : Name : $\lambda P[P(becks)]$
$NP \rightarrow Name$	$NP' = Name'$	*Posh* : Name : $\lambda P[P(posh)]$
$NP \rightarrow Det\ N$	$NP' = Det'(N')$	*every* : Det : $\lambda Q[\lambda P[\forall x(Q(x) \Rightarrow P(x))]]$
$VP \rightarrow Vi$	$VP' = Vi'$	*player* : N : $\lambda x[\mathit{player}(x)]$
$VP \rightarrow Vt\ NP$	$VP' = Vt'(NP')$	*scored* : Vi : $\lambda P[P(\lambda x[\mathit{scored}(x)])]$
		hates : Vt : $\lambda P[\lambda Q[Q(\lambda x[P(\lambda y[\mathit{hates}(x,y)])])]]$

Draw a parse tree for each of the following sentences. Annotate each node of your trees with its semantics. Use lambda-reduction to simplify as much as possible. Show your working.

(a) *Becks scored*

(b) *every player hates Posh*

4. Each of the following sentences is ambiguous in multiple ways. For each sentence, list the ambiguities and the type of each ambiguity (structural, lexical, etc.)

(a) *Becks reached the ball.*

(b) *Becks kissed a former lady friend.*

(c) *Posh hit the photographer with his tripod.*

5. Identify the knowledge that would enable an agent to resolve the referents of the underlined pronouns (i.e. to whom they refer):

(a) *Posh has a Yorkshire terrier, Lucy. She sings Spice Girls hits when they go for a walk.*

(b) *Becks always takes baby Romeo to the match. He uses the opportunity to teach him to speak.*

6. For each of the following sentences, identify the problem that the sentence causes for compositional semantics.

(a) *Every loving son adores his mother.*

(b) *Someone is loved by everyone.*

(c) *Fergie gave him the sack.*