Computational linguistics

Ann Copestake

31.1 Introduction

Computational linguistics concerns the computational modelling of human language. It is a large and complex field, impossible to survey briefly. As an indication of its size, there are over 32,000 papers in the anthology of the Association for Computational Linguistics, which only contains a fraction of the papers published in the discipline. The aim of this chapter, therefore, is to give an idea of the topics covered, and to discuss some of the methodology employed, which differs substantially from the approaches most common in linguistics. In fact, while the term, ‘Computational Linguistics’, suggests a branch of linguistics, it is best to view it as a discipline in its own right, with connections to linguistics which are close in some subareas and hardly exist at all in others. Some authors distinguish between ‘Computational Linguistics’ (henceforth CL) and ‘Natural Language Processing’ (NLP), with the latter being seen as more concerned with practical applications, but other writers use the terms interchangeably. Even more confusingly, speech processing and search and Information Retrieval (IR) are usually seen as separate from CL/NLP, though there are broader terms, such as ‘Human Language Technologies’, which include them all. In this chapter, I mainly concentrate on topics which are core to CL, looking at some applications but concentrating on the computational methodologies involved in modelling language.

Within Computer Science, CL is often seen as closely related to Artificial Intelligence (AI). Until the late 1980s, a mainstream view was that CL concerned language-specific issues while AI would develop a language-independent account of general human reasoning. This approach is generally seen to have failed, but alternative paradigms have emerged. In particular, very large amounts of machine-readable text started to become readily available in the early 1990s and this allowed the successful development of many different types of statistical models. As discussed below, these may either supplement or be alternatives to symbolic models of language, but they also give some of the functionality that AI was previously expected to provide.

For instance, in natural language understanding (NLU), the role of CL was originally seen as to parse sentences according to their morphological and syntactic structure in order to produce meaning representations corresponding to different possible analyses. It was the
role of AI to connect the meaning representations to a model of the world, to reason about the correct analysis in context and to allow the system to act on the utterance appropriately. Given a very limited ‘microworld’, it is indeed possible to do this: i.e. to build a model of the entities and possible events which allow some simple utterances which are directly relevant to the microworld to be interpreted. The most famous example of this (although not the earliest) was Winograd’s SHRDLU system, in the early 1970s, a virtual world in which a number of toy blocks were arranged on a surface and inside a box. The computer’s task was to manipulate the blocks in response to human commands such as *Pick up a big red block!* and to answer questions about the state of the microworld. By the early 1980s, several NLU systems based on microworlds constructed from databases were in commercial use and there was a general perception that computers could understand human language. But the approach was insufficiently robust to work with spoken input and did not extend beyond very small domains, where all the possible types of entities and events could be enumerated.

There are a number of reasons why the limited domain approaches did not scale to general language understanding. On the language processing side, it was necessary to improve parsers so that it was possible to process unrestricted text reasonably quickly. When analysed on the basis of the morphology and syntax alone, without constraints based on the meaning of the sentence, even moderate length sentences have a massive degree of ambiguity. Current parsers rank analyses according to their statistical plausibility rather than trying to disambiguate by reasoning about meaning and are reasonably successful at doing this. Thus the elusiveness of full NLU is not primarily due to limitations in parser performance but the (current?) impossibility of modelling human knowledge and reasoning. The problem is not factual general knowledge of the form ‘the king of Prussia in 1790’, ‘the closest airport to Cambridge’ and so on. There are many databases and gazetteers which contain this type of information and CL systems exist that can extract facts from online text with a high degree of reliability, as was illustrated by the success of the IBM Watson system on the *Jeopardy!* quiz show (for an overview of the system, see Ferrucci et al. 2010). The main difficulties arise with the sort of reasoning that humans would not even regard as ‘intelligent’. Human understanding of space, movement, time and so on is essential to language comprehension, but extremely hard to capture formally. This is difficult to illustrate with a short example, but consider the phrase *the cat in front of the chair.* This has an interpretation where the cat is located according to coordinates which are intrinsic to the chair (as well as one where the location is relative to another entity). However, what determines the notion of front and back of an entity is complex, as consideration of examples such as *chair, stool, television, steps, earthworm, house* and *book* should demonstrate. There is no straightforward algorithm, yet even very young children can interpret these intrinsic spatial expressions correctly. For a human, the deduction ‘if a cat is in front of some steps then it is at the bottom of the steps’ is completely obvious, but for a question answering system this is a more challenging topic than European monarchs of the eighteenth century.

Perhaps the most fundamental issue for NLU is whether there are workable nonlinguistic notions of concepts. The assumption that it is possible to have a language-independent concept such as BIG (distinct from the words *big, groß, grand, gros* and so on) underpinned approaches that separated language-specific processing from general-purpose reasoning. It makes sense to make such an assumption for microworlds, where natural language words are used with very specific referents which can be defined or enumerated. For instance, in SHRDLU, it was possible to list all the blocks which could be described as *big.* There are other cases where language-independent definitions are workable, such as some scientific and mathematic terms. However, the attempt to use symbolic language-independent
concepts as the basis of general language understanding is now regarded as a dead end by most researchers. It is an open question as to whether it will nevertheless be possible eventually to model lexical meaning in a way that will support full NLU, but it is already clear that it is possible to provide useful partial models and that the failure to achieve full natural language understanding has not prevented real applications from using CL techniques. Even if one has the long-term goal of addressing fundamental problems of language and concepts, it makes sense to work on practical techniques for language processing in the shorter term, with the aim of bootstrapping deeper techniques. In the rest of this section, I will very briefly introduce a range of different types of CL applications, and then describe one illustrative application in more detail.

Most of the early practically usable CL systems (other than the limited domain database interfaces) were designed to help humans communicate with each other. Spelling and grammar checking has been a standard part of document processing for decades. Predictive text is also a long-standing application: before its use in phones it was employed in some systems for ‘augmentative and alternative communication’ for people with a disability. Machine-aided translation has a long history of practical utility: such systems help a human translator, for instance by storing and retrieving translations of text passages. Useful machine translation (MT) has been available since the 1970s, with the more recent statistical machine translation (SMT) approaches being available for a variety of language pairs and accurate enough to be helpful on a range of different types of text. The best MT allows a reader to get a reasonable idea of the meaning of text written in a language they do not understand, although far more care is needed in interpretation of MT output than is required for a good human translation. The success of MT illustrates the important point that humans can adapt to imperfect NLP systems. There are also a variety of tools for language teaching and assessment: for instance to help first and second language learners by detecting various types of errors in their writing. Such a system should not mark something as an error when it is actually correct, but it is acceptable to miss some errors. That is, error detection systems emphasize precision over recall, in contrast to an MT system, which is designed to attempt translation for nearly all its input.

Another very important group of NLP applications involve changing the medium in which the language is expressed. For instance, some speech recognition systems are designed to transcribe speech to produce text (others are used in dialogue systems). Text-to-speech, optical character recognition (OCR) and handwriting recognition can also be described as converting the medium of the input, since OCR and handwriting recognition convert visual data to machine-readable text. Models of language are required to do this accurately: in the case of speech and handwriting recognition, the initial processing produces a set of possibilities and models trained on large amounts of text are used to select between them (as discussed below). For text-to-speech, the models are designed to predict intonation and the pronunciation of homonyms which are not homophones.

Internet search was originally based on approaches from Information Retrieval, combined with metrics of webpage popularity, and as such used little language processing. More recent techniques are more advanced, taking into account synonymy and near-synonymy, for instance. The need to process and index vast quantities of text is a strong constraint on what techniques are practical, but various forms of analysis of language are now being used at web scale. More targeted search and extraction of particular classes of information is known as text-mining, which is now widely used, especially on scientific texts. Question answering systems use a combination of text-mining techniques and online databases to provide specific responses to certain types of question.
Many applications involve different types of text classification. A simple example is detection of spam email. Sentiment analysis, which involves deciding whether opinions expressed are positive or negative is a form of classification described in more detail in §3.2. Automated assessment can be modelled as classification of text according to some marking scheme. It is now regularly used for some high volume exams, although typically only in conjunction with human assessment. Exam marking might be seen as a bad application for an inherently imperfect technology, but in some cases automated assessment has been found to be as reliable as human marking.

Finally, there are dialogue systems, including ‘intelligent’ personal assistants, such as Siri. These combine search functionality, question answering and interfaces to databases, including diaries and so on. These systems have some model of dialogue context and can adapt to individual users. Just as importantly, users adapt to the systems, rephrasing queries which are not understood and reusing queries which are successful.

Various properties are required for a successful NLP application. Since NLP systems are imperfect, applications have to tolerate imperfection in processing. In extreme cases, even a simple application such as spelling correction could require arbitrary amounts of reasoning, beyond the reach of any current approach, but in successful applications these situations are rare enough that the absence of deep understanding does not prevent them being useful. Nearly all practical systems now exploit statistical information extracted from corpora, as we will see in more detail in the next sections.

31.2 An example application: sentiment analysis

Finding out what people think about politicians, policies, products, services, companies and so on is a huge and lucrative business. Increasingly this is done by automatic analysis of web documents and social media. The full problem involves finding all the references to an entity from some document set (e.g. references to Hillary Clinton in all newspaper articles appearing in September 2013, references to Siri in all tweets with hashtag #apple), and then classifying the references as positive, negative or neutral. Customers who use opinion mining want to see summaries of the data (e.g. to see whether popularity is going up or down), but may also want to see actual examples (text snippets). Companies generally want a fine-grained classification of aspects of their product (e.g. laptop batteries, phone screens).

A full opinion mining system requires that relevant text is retrieved, references to the objects of interest are recognized in that text (generally as ‘named entities’: e.g. Sony 505G, Hillary Clinton), and the parts of the text that refer to those entities are determined. Once this is done, the referring text can be classified for positive or negative sentiment. To be commercially useful, this has to be done very rapidly, especially when analysing trends on social media, so a significant software engineering effort is involved. But academic researchers have looked at a simpler version of the task by starting from a set of documents which are already known to be opinions about a particular topic or entity (e.g. reviews), where the problem is just to work out whether the author is expressing positive or negative opinions. This allows researchers to focus on sentiment classification but this has still been a challenging problem to address. Some of the early research work was done on movie reviews (Pang et al. 2002). The rating associated with each review is known (that is, reviewers give each movie a number of stars), so there is an objective standard as to whether the review is positive or negative. This avoids the need to manually annotate the data before experimenting with it, which is a time-consuming and error-prone process. The research problem is to assign sentiment automatically to each document in the entire corpus to agree with the known ratings.
Sentiment classification can be done by using manually-defined patterns, which are automatically checked against the review to be classified. Such patterns are often expressed as regular expressions. For instance, "great act*" would match great acting, great actor and so on. However, it is often preferable to use machine learning (ML) techniques: humans are good at deciding whether particular words or phrases are positive or negative but tend to miss many useful indicators. I will outline how ML can be applied to sentiment classification, but the general methodology is applicable to many other problems in CL.

The first step is to separate the document collection into training data, which will be used to build the classifier, and test data, which will be used to evaluate it. It is standard to use 90 per cent of the data for training and keep 10 per cent unseen for testing. The separation of the training and test data is crucial to the methodology, since the aim is to construct a model which can be generalized to data not previously seen. The second step is to extract features from each document in the training data. The most basic technique is to use the words in the review in isolation of each other as features: this is known as a bag of words model because the document is modelled as an unordered collection of words. The third step is then to use the features to train a classifier. The system will automatically learn whether particular features tend to indicate positive or negative reviews, and how reliable that indication is. For instance, great would be a strong indicator of a positive review if the ratings are positive for a high percentage of the documents in which it occurs, and peculiar a weak indicator of a negative review if it occurs in both negative- and positive-rated documents with the majority being negative. Finally, the trained classifier is run on the features extracted from each review in the test data in order to assign the review to the sentiment class with the highest probability. The percentage of reviews categorized correctly gives the success rate.

A range of different algorithms for classifiers exist, which make different approximations about the properties of the data being modelled, but it is unnecessary to go into the details here. Indeed, the difference in performance is often quite small between different classifiers. Pang et al. (2002) found that the accuracy of classification with the basic bag of words technique on the movie corpus was around 80 per cent. The set of reviews was artificially balanced and neutral reviews excluded, so chance success rate would be 50 per cent. It will be obvious that the bag of words model does not fully capture the way sentiment is expressed in language. It even ignores the possibility of negation: e.g. Ridley Scott has never directed a bad film is a positive statement, though bad is usually an indicator of a negative review. However, computational linguists have learned to be wary of dismissing simple models as obviously inadequate without testing them, since it has turned out embarrassingly often that simple models outperform models with more linguistic sophistication. Naturally negation is an important attribute, but modelling it requires that techniques are available which capture the scope of negation, and which do not introduce errors which make performance worse overall. The bag of words approach can be taken as a ‘baseline’: that is, a very basic and straightforward approach against which more sophisticated techniques should be evaluated to see if they actually lead to better performance. In fact, initial attempts at better language modelling failed to lead to large improvements in performance on the movie corpus, but some later sentiment analysis systems use much more sophisticated techniques. For instance, Moilanen and Pulman (2007) demonstrated that parsing and a form of compositional semantics can considerably improve performance over the simple methods.

One danger with machine learning techniques is that they may match the training data too closely: the technical term is ‘overfitting’. Any ML approach requires that the training data is reasonably close to the test data. For instance, a sentiment classifier trained on movie
reviews may not perform well on reviews of laptops. Sensitivity to domain and genre is common to all statistical approaches, but there is a particular issue with sentiment analysis because indicators often change over time. For instance, if the classifier were trained on reviews which included a large number of films from before 2005, Ridley might emerge as a strong positive indicator, but the system could then tend to misclassify reviews for ‘Kingdom of Heaven’ (which was panned). Thus, in practical use, it is necessary to retrain systems regularly to ensure that features are not retained after they become unreliable.

More subtle and linguistically interesting problems arise from the contrasts in the discourse. The two extracts below are from Pang et al.’s paper:

This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance. However, it can’t hold up.

(taken from a review by David Wilcock of ‘Cop Land’ www.imdb.com/reviews/101/10185.html)

AN AMERICAN WEREWOLF IN PARIS is a failed attempt … Julie Delpy is far too good for this movie. She imbues Serafine with spirit, spunk, and humanity. This isn’t necessarily a good thing, since it prevents us from relaxing and enjoying AN AMERICAN WEREWOLF IN PARIS as a completely mindless, campy entertainment experience. Delpy’s injection of class into an otherwise classless production raises the specter of what this film could have been with a better script and a better cast … She was radiant, charismatic, and effective.

(taken from a review by James Berardinelli www.imdb.com/reviews/103/10363.html)

Techniques exist in CL for modelling contrasts in discourse (i.e. in connected text more than one sentence long), and these can be exploited in sentiment analysis. In fact, in the related task of modelling citations in scientific text, contrast is an important indicator of whether the authors of a paper claim to improve on the cited work: for instance ‘X’s work demonstrated … but …’. Scientific text also provides good examples of the differences in terminology which make it necessary to adapt sentiment analysis techniques to different domains. For instance, in chemistry papers describing the synthesis of compounds, strong is often a negative term, especially in the phrase strong conditions, indicating that a synthesis is difficult to carry out. Naturally, there are also dialect differences: quite good is positive in American English but (usually) very faint praise in British English. Indeed, deciding whether words like sick are being used in a positive or negative way can tax many human readers.

The sentiment classification approach described here is an example of supervised learning because the classifier was trained on data which was labelled according to whether the sentiment was positive or negative. However, the ratings were obtained from the existing reviews, so there was no need for an additional annotation step. Often CL experiments require specially annotated data, potentially in large quantities if it is to be used for training. There are also unsupervised approaches to machine learning, where structure is induced from unlabelled data, but even in this case some type of labelling is usually required for evaluation. When annotation is carried out by humans for an experiment, it is now usual to use multiple annotators and to make sure they agree with each other to a reasonable extent, but it is very rare to obtain complete agreement in annotation. The degree of agreement achieved between the human annotators is often used as a ceiling on expected performance of a computational model.
31.3 Statistical models

In the introduction, I argued that statistical models play a crucial role in modern computational linguistics. Sentiment analysis is just one example of this. In this section I will outline three types of model which are relevant to a range of practical applications, but also have theoretical implications. This discussion is general and relatively non-technical since the linguistic interest is in the type of features used and the way that data is acquired to train the model rather than the details of the models themselves.

31.3.1 N-gram language models

A simple but extremely important class of statistical models of language are based on n-grams, where a sequence of the \( n - 1 \) words is used to give a probability for the \( n \)th word. For instance, trigram models use the preceding two words, bigram models use the preceding word and unigram models use no context at all, but simply work on the basis of individual word probabilities. Such models are classically used in automatic speech recognition (ASR) but also form a component of other applications mentioned in the previous section, including augmentative and alternative communication systems for people with disability and statistical machine translation. The need for language-based prediction in ASR arises because recognizers cannot accurately determine which words are uttered from the sound signals alone, and they cannot reliably tell where each word starts and finishes. For instance, \textit{have an ice Dave}, \textit{heaven ice day} and \textit{have a nice day} could easily be confused. In fact, humans also need context to recognize words, especially words like \textit{the} and \textit{a}. If a recording is made of normal fluent speech and isolated segments corresponding to \textit{the} and \textit{a} are presented to a human subject, it is generally not possible for them to tell the difference. Similarly, humans are bad at transcribing speech in a language they do not understand. For ASR, an initial signal processing phase produces multiple hypotheses for the words uttered, which can then be ranked and filtered using a model of the probabilities of the alternative possible word sequences. The term ‘language model’ is standardly used for this type of prediction, where the problem is to choose the word sequence with the highest probability, because the words are being modelled rather than the sounds. However, this terminology is unfortunate, because it is used to refer to a very particular type of model of language.

Identifying the originator of a particular technique is often difficult, but there is no doubt that it was Fred Jelinek who pioneered this approach to speech recognition in the 1970s and that Jelinek’s group at IBM built the first usable ASR system. To make the n-gram idea more concrete, consider the bigram model probabilities: \( P(w_n|w_{n-1}) \) is the probability of \( w_n \) conditional on \( w_{n-1} \), where \( w_n \) is the \( n \)th word in some sequence. The probability of a sequence of words \( P(w_1^m) \) may then be approximated by the product of the corresponding bigram probabilities:

\[
P(w_1^m) \approx \prod_{k=1}^{m} P(w_k|w_{k-1})
\]

This approximation assumes independence of the individual probabilities, an assumption which is clearly wrong, but nevertheless works well enough for the estimate to be useful. Note that, although the n-gram probabilities are based only on the preceding words, the effect of this combination of probabilities is that the choice between possibilities at any point is sensitive to both preceding and following contexts. For instance, this means that the
decision between a and the may be influenced by the following noun, which is a much better predictor than the words before the determiner. A naive implementation of this approximation would be hopelessly intractable, but there are efficient algorithms for finding the most likely sequence of words given n-gram probability estimates.

The estimates for the n-gram probabilities are acquired automatically from a corpus of written text. For instance, bigram probabilities are estimated as:

\[
P(w_n | w_{n-1}) \approx \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}
\]

i.e. the count \( C \) of a particular bigram \( w_{n-1}w_n \) in the corpus divided by the sum of the counts of all bigrams starting with \( w_{n-1} \). This is equivalent to the total number of occurrences of \( w_{n-1} \), except in the case of the last token in a corpus, a complication which can be ignored for all practical purposes. That is, we actually use:

\[
P(w_n | w_{n-1}) \approx \frac{C(w_{n-1}w_n)}{C(w_{n-1})}
\]

For speech recognition it is common to use 4-gram or 5-gram models. However, even if using bigrams, there will be cases where a sequence occurs in the test data that has never been seen in the training data. To give such an event a probability of zero, as implied by the estimate given above, would rule it out entirely, so it is necessary to smooth the probabilities. This means that we make some assumption about the ‘real’ probability of unseen or very infrequently seen events and distribute that probability appropriately: there are a variety of techniques for doing this. In the case of n-grams, it is possible to backoff to the probabilities of shorter sequences. For instance, in a trigram model, if a particular trigram has a very low or zero frequency, we can backoff to the bigram probabilities, and from bigram probabilities to unigram probabilities. This sort of estimation is essential to get good results from n-gram techniques. If Chomsky had not used *Colorless green ideas sleep furiously* as an example, it is likely that none of its constituent trigrams would occur even in a very large corpus, but these techniques would still allow a probability estimate.

Modelling language using n-grams might seem a very crude technique, but it was only adopted for speech recognition after more linguistically sophisticated techniques were tried and failed. Fred Jelinek’s (in)famous remark, ‘Every time I fire a linguist, the performance of our speech recognition system goes up’ should be seen in this context. Jelinek himself was fully aware that n-gram models were inadequate for many natural language phenomena, such as long-distance dependencies, and indeed he described n-gram models as ‘almost moronic’, but he challenged those who objected to find models that performed better. This is partly an engineering issue: n-gram models are outperformed by other statistical approaches for a fixed quantity of training data, but training data for language models is simply ordinary text and is therefore available in indefinitely large quantities for languages such as English. Since n-gram models can be trained more efficiently than the competing techniques, their performance can be further improved simply by adding more data.

One could say that the challenge for linguists is to explain what factors are responsible for n-gram models performing so well, given their obvious inadequacies, but there is no clear answer to this question. To an extent, n-grams act as a crude surrogate for ‘real’ intelligence. For instance, in the context of a talk on computational linguistics, it would be possible for a
human to reason that ‘computers recognize speech’ is a more likely utterance than ‘computers wreck a nice beach’, but the corpus-derived probabilities would also give the same decision automatically. As we saw in the introduction, old-fashioned AI-style reasoning about the world has proved impossible to incorporate in general CL models, while the n-gram approach is straightforward, given enough data. The explanation that n-grams work because they stand in for full reasoning has appealed to linguists, because it would mean that the success of n-gram models was due to modelling the world, rather than modelling language. However this is not the only factor. To an extent, n-gram models do capture syntax, at least for languages like English where word order is fairly fixed, even though they do not involve explicit syntactic categories. For instance, an n-gram model will contain vast numbers of sequences such as ‘the bell’, ‘a bell’, ‘a mouse’, ‘that mouse’ and so on, and do not have sequences ‘a have’, ‘the died’ and so on, and this gives an indication of the distribution of the and a. Indeed, it is possible to automatically induce syntactic categories corresponding to determiner and noun from n-grams, although this turns out not to be practically helpful for speech recognition.4 It is also clear that various types of fixed and semi-fixed multiword expressions play an important role in natural language and n-gram models allow these to be used for prediction. The possible role of n-grams in modelling phenomena such as adjective ordering, where there are soft constraints which are not captured by conventional syntax, is discussed in §31.4.

31.3.2 Part-of-speech tagging

Sometimes we are interested in a statistical model that involves assigning classes to words in a sequence rather than predicting the next word. One important case is part-of-speech tagging (POS tagging), where the words in a corpus are associated with a tag indicating some syntactic information that applies to that particular use of the word. The tags used are generally rather more specific than the conventional notion of part of speech. For instance, consider the example sentence below:

They can fish.

This has two readings: one (the most likely) about ability to fish and other about putting fish in cans. Fish is ambiguous between a singular noun, plural noun and a verb, while can is ambiguous between singular noun, verb (the ‘put in cans’ use) and modal verb. However, they is unambiguously a pronoun. (I am ignoring some less likely possibilities, such as proper names.) These distinctions could be indicated by POS tags:

They_pronoun can_modal fish_verb.

They_pronoun can_verb fish_plural-noun.

In fact, much less mnemonic tag names make up the standard tagsets used in corpora and in POS tagging experiments: in CLAWS 5 (C5), which is very widely used, the tags needed for the example above are:

NN1 singular noun
NN2 plural noun
PNP personal pronoun VM0 modal auxiliary verb
A. Copestake

VVB base form of verb (except infinitive)
VVI infinitive form of verb

The corresponding lexicon associating words with C5 tags would be:

they PNP
can VM0 VVB VVI NN1
fish NN1 NN2 VVB VVI

A POS tagger resolves the lexical ambiguities to give the most likely set of tags for the sentence. In the case of They can fish, the correct tagging is most likely to be the one with the modal use of can:

They_PNP can_VM0 fish_VVI ._PUN

Note the tag for the full stop: punctuation is an important part of the model when POS tagging text. The other syntactically possible reading is:

They_PNP can_VVB fish_NN2 ._PUN

However, POS taggers (unlike full parsers) are not constrained to produce globally coherent analyses. Thus a POS tagger could potentially return the following sequence:

They_PNP can_VM0 fish_NN2 ._PUN

despite the fact that this does not correspond to a possible reading of the sentence.

Automatic POS tagging is useful as a way of annotating a corpus because it makes it easier to extract some types of information (for linguistic research or CL experiments) while being faster and more robust than full parsing. It also acts as a basis for more complex forms of annotation. For instance, named entity recognizers (mentioned in §31.2) are generally run on POS-tagged data. POS taggers are sometimes run as preprocessors to full parsing, since this can cut down the search space to be considered by the parser, although it is more effective to use tags which correspond directly to categories used by the parser rather than a general-purpose tagset. POS tagging can also be used as part of a method for dealing with words which are not in a parser’s lexicon (unknown words).

POS taggers using manually constructed rules were first developed in the early 1960s, when the first experiments with stochastic POS tagging using probabilities were also carried out. Stochastic approaches based on the type of statistical models used for speech recognition became popular in the late 1980s. Some of these models involve n-grams, but in this case the n-grams are sequences of POS tags rather than of words. The most commonly-used approaches are supervised and depend on a small amount of manually tagged training data from which the lexicon and POS n-grams can be extracted. It is also possible to build unsupervised POS taggers for a tagset but these still require a lexicon associating (some of) the words with their possible tags and they do not perform as well as a system trained on even a small manually annotated corpus. As mentioned above, syntactic categories can be induced from untagged data without a lexicon, but the categories will not correspond directly to those in standard tagsets.
Tags can be assigned to words in a sentence based on consideration of the lexical probability (how likely it is that a word has a particular tag), plus the sequence of prior tags. For a bigram model, we only look at a single previous tag, for the trigram the two previous tags. As with word prediction, the aim is to find the highest probability tag sequence for sequence of words (usually a sentence) rather than to look at tags in isolation. However, this is a more complex calculation than for prediction, and I will not go through the details here. Since annotated training data is limited, n-grams of length greater than three are ineffective and appropriate backoff and smoothing are crucial for reasonable performance. However, note that, with these models, the frequencies of the word sequences do not play a role. Tagging *Colorless green ideas sleep furiously* uses the same tag sequence probabilities as *Little white lies spread quickly* although the lexical probabilities will differ.

For effective performance, some method is needed to assign possible tags to words not in the training data. One approach is simply to use all possible open class tags, with probabilities based on the unigram probabilities of those tags. A better approach is to use a morphological analyser to restrict this set: e.g. words ending in *–ed* are likely to be VVD (simple past) or VVN (past participle), but cannot be VVG (*–ing* form).

POS tagging algorithms may be evaluated in terms of percentage of correct tags, checked against the annotated test data. The standard assumption is that every word should be tagged with exactly one tag, which is scored as correct or incorrect and that there are no marks for near misses, although some POS taggers return multiple tags in cases where more than one tag has a similar probability, which complicates evaluation. Generally there are some words which can be tagged in only one way, so are automatically counted as correct. Punctuation is generally given an unambiguous tag. The best taggers for English have success rates of 97 per cent when trained and tested on newspaper text, but the baseline of choosing the most common tag based on the training set gives around 90 per cent accuracy.

It is worth noting that increasing the size of the tagset does not necessarily result in decreased performance: this depends on whether the tags that are added can generally be assigned unambiguously or not. Potentially, adding more fine-grained tags could increase performance. For instance, suppose we wanted to distinguish between verbs according to whether they were first, second or third person. If we were to try and do this simply by adding more categories for verbs to the C5 tagset and used a stochastic tagger as described above, the accuracy would be low: all pronouns are tagged PRP, hence they provide no discriminating power. On the other hand, if we tagged *I* and *we* as PRP1, *you* as PRP2 and so on, which, of course can be done with 100 per cent accuracy, the n-gram approach would allow discrimination between first, second or third person verbs. In general, predicting on the basis of classes means there is less of a sparse data problem than when predicting on the basis of words, but we also lose discriminating power. In fact, C5 assigns separate tags for the different forms of *be*, which is redundant for most purposes, but helps make distinctions between other tags.

The error rate of a POS tagger will be distributed very unevenly. For instance, the tag PUN will never be confused with VVN (past participle), but VVN might be confused with AJ0 (adjective) because there is a systematic ambiguity for many forms (e.g. *given*). For a particular application, some errors may be more important than others. For instance, if one is looking for relatively low frequency cases of denominal verbs (that is verbs derived from nouns – e.g. *canoe, tango, fork* used as verbs), then POS tagging is not directly useful in general, because a verbal use without a characteristic affix is likely to be mistagged. This makes POS tagging less useful for lexicographers, who are often interested in finding examples of unusual word uses.
The initial human annotation of text with POS tags is not entirely straightforward. Many of the experiments with POS tagging for English have been done on data from The Wall Street Journal distributed as part of the Penn Treebank. This uses a much smaller tagset than C5 but the manual describing how to assign tags runs to thirty-four pages and the tagged data is nevertheless known to contain many errors. This is presumably partly responsible for the plateau in performance of POS taggers at around 97 per cent accuracy.

It is also important to note that error rates for POS tagging differ between languages. For instance, POS tagging for Japanese is almost deterministic but the accuracy is much lower for Turkish, at around 90 per cent, because it is an agglutinative language.

POS tagging is an example of shallow syntactic analysis. Much more elaborate analyses can be produced from parsers which are trained on a treebank: i.e. a corpus of sentences manually associated with the correct syntactic analysis. The paradigm example of this is the Penn Treebank which has been used in one way or another in the majority of experiments on the automatic parsing of English. The analyses assigned may be syntactic trees with POS tags as lexical categories, or syntactic dependencies, as mentioned in the introduction. However, there are alternative approaches which provide more detailed analyses, and these are discussed in §31.4.

### 31.3.3 Distributional semantics

Distributional semantics refers to a family of techniques for representing word (and phrase) meaning based on contexts of use. Consider the following examples (from the British National Corpus):

- it was authentic scrumpy, rather sharp and very strong
- we could taste a famous local product — scrumpy
- spending hours in the pub drinking scrumpy

Even if one does not know the word *scrumpy*, it is possible to get a good idea of its meaning from contexts like this. Distributional semantics has been discussed since at least the early 1950s: the idea is often credited to J.R. Firth or to Zellig Harris, but Firth was actually concerned to emphasize that meaning is context-dependent and Harris’s work on the distributional hypothesis at that point was primarily concerned with morphology and syntax. The psychologist Charles E. Osgood described a distributional approach to semantics in 1952. The first computational work was done in the early 1960s, by Karen Spärck Jones in Cambridge (Spärck Jones 1964) and by Kenneth Harper at the RAND Corporation, whose work was inspired by Harris (Harper 1965). It seems it was then almost forgotten in CL, though the vector space models which have been used in Information Retrieval since the 1960s are a form of distributional model. Distributional techniques started gaining popularity within CL in the 1990s, after being reintroduced from Information Retrieval, and are now a major topic of research. They are especially attractive because they allow representations of word meaning to be constructed without the need for any manually created taxonomies or manual data annotation and they may offer a partial solution for the failure of symbolic AI in the earlier approaches. They are also widely used in psycholinguistics.

In distributional models, the meaning of a word is treated as if it were located in a multidimensional space. The dimensions correspond to elements from the various contexts in which the word does or does not occur. To make this more explicit, we can consider the simplest type of distributional model, where the context corresponds to the words in a
‘window’ on either side of the word to be modelled. For instance, for the example above, the context for *scrumpy* would include ‘authentic’, ‘rather’, ‘sharp’, ‘local’, ‘product’, ‘pub’, ‘drinking’ if the distribution were calculated on the basis of co-occurrence in a word window of five (i.e. including two words on either side of ‘scrumpy’). It is usual to exclude very common closed-class words like *was*. A context set such as this is collected for a large number of words, and the dimensions for the overall space correspond to the more frequent terms found. The usual method is then to represent each word as a vector with those dimensions. To illustrate this, let us assume we have only six dimensions (real systems use thousands of dimensions) and that the value for each dimension is 1 if the word has been seen with that context item and 0 if it has not. See Table 31.1. Thus the meaning of each word is represented as a point in a six-dimensional space. The words *cider* and *scrumpy* are closer together in that space than *elephant* and *scrumpy* or *elephant* and *cider* and thus predicted to be more similar in meaning.

In modern work, the elements in the vector are usually weighted depending on how ‘characteristic’ the context term is. This is based on the probability of co-occurrence normalized by the probability that the co-occurring term occurs elsewhere in the text, so frequently co-occurring words with relatively low overall frequency are the most characteristic. For instance, in a distribution for the word *language* extracted from Wikipedia, *English*, *grammar* and *germanic* are all highly characteristic. Instead of dimensions being words in some window, they may be words related by a syntactic dependency (in which case, in the scrumpy example, *rather* would be excluded), or the dimensions may themselves correspond to parts of syntactic or semantic dependencies. For instance, one dimension could be ‘drink OBJ’ to indicate that in the context the word was the object of *drink* (as for *scrumpy*) and another ‘drink SUBJ’ (for *elephant*). Even with these refinements, a lot of information is lost from the individual contexts, but the corpora used are sufficiently large that each vector amalgamates information from many thousands of contexts for even moderate frequency words.

As indicated with the simple example, vectors for different words can be compared to see how close the points they correspond to are in the semantic space. For instance, the ten words most similar to *cat* (again using distributions extracted from Wikipedia) are: *dog*, *animal*, *rat*, *rabbit*, *pig*, *monkey*, *bird*, *horse*, *mouse* and *wolf* (obviously the exact set returned depends on precisely how the distributions are defined). This notion of similarity is very broad. It includes synonyms, near-synonyms, hyponyms, taxonomic siblings, antonyms and so on. But it seems to reflect psychological reality in that it correlates very well with human similarity judgements. While there is clearly more to word meaning than similarity, it does allow words to be clustered into semantic classes and permits analogical processes to be modelled. Variants of this approach have been tried to extract words in more specific relationships, such as hyponyms and hypernyms.

Two very recent developments in distributional models can be mentioned briefly. One is the research on compositional approaches, where distributions corresponding to phrases

<table>
<thead>
<tr>
<th></th>
<th>apple</th>
<th>drinking</th>
<th>ear</th>
<th>product</th>
<th>sharp</th>
<th>zoo</th>
</tr>
</thead>
<tbody>
<tr>
<td>scrumpy</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>cider</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>elephant</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
are constructed from the distributions of the corresponding words, guided by syntax. For instance, the phrase green banana can be modelled by combining the vector for green and for banana. Some phrases are non-compositional multiword expressions, of course, but others are somewhat compositional but have additional meaning aspects: for instance, green bananas are unripe and therefore not good to eat. A second development is models which combine textual context with information automatically extracted from images. Given some training data, in which images are associated with words that describe them (e.g. a picture of a car is paired with the word car), a mapping can be induced between visual features automatically extracted from the image and the features of the textual distributional model acquired from a corpus. This mapping can then be used to label new images. Crucially, this works (to an extent) not just with words in the training data, but also with new words. For instance, given data for pictures corresponding to cars, vans and motorcycles, the word bus can be associated with a picture of a bus, despite never having been seen in the training data (Lazaridou et al. 2014). While this type of research is in its infancy, it holds out hope that the semantics of words can be grounded in perceptual features corresponding to their referents.

### 31.4 Computational grammars

I have spent most of this chapter describing various forms of statistical models, because these are highly characteristic of modern computational linguistics. However, there is still a place for manually constructed grammars of languages. While in the statistical work there is a discontinuity between the 1960s research, which was largely forgotten, and the reintroduction of statistical models in the 1980s and 1990s, there is more continuity in the history of research on computational grammars and parsing. However, the story is very complex: for instance, formal grammars are also used in computer science to describe the syntax of programming languages, and the first techniques for parsing human languages automatically were developed at the same time as the techniques for parsing these artificial languages. What is clear is that there was eventually a considerable degree of cooperation between linguists and computational linguists in the development of frameworks for describing grammars of natural languages that were declarative (i.e. could be formally specified without describing the procedures used to run the grammar), more powerful than context-free grammars, linguistically adequate for a wide range of languages and yet sufficiently tractable that they could be used in automatic parsing.

Two such frameworks which have active communities of both linguists and computational linguists are Lexical Functional Grammar (LFG), initially developed by Bresnan and Kaplan in the 1970s and Head-driven Phrase Structure Grammar (HPSG), due to Pollard and Sag, which started in the 1980s. Much of the recent work involves two large collaborations, PARGRAM for LFG and DELPH-IN for HPSG. Both frameworks allow the specification of formal grammars (including morphology, syntax and compositional semantics) which can be used for both parsing and generation. They have both been used to develop efficient computational grammars with high coverage for English and a small number of other languages. Compared with the alternative paradigm of inducing grammars based on a manually annotated treebank, manually constructed grammars are more time-consuming to develop (even if the initial treebank creation is taken into account), but provide richer annotations, including a representation of compositional semantics, and a higher degree of precision. The large grammars are competitive with the treebank grammars for some applications and are more robust to changes in domain. Furthermore, in both LFG and
HPSG, illustrative grammars have been developed for a very wide range of human languages and various cross-linguistic generalizations can be captured.

The linguistically-motivated grammar building approach has benefited from statistical techniques, which are used for ranking parses and generated sentences, and for developing lexicons and mechanisms for handling unknown words, for instance. Conversely, these approaches provide resources for statistical techniques. For instance, the DELPH-IN Redwoods Treebanks can be used to train grammars, and the WikiWoods corpus, which contains compositional semantic analyses automatically generated from Wikipedia, can be used for distributional semantics.

The use of statistical techniques with such grammars is more than simple convenience: they model aspects of language which the symbolic grammars do not capture. For a simple example, consider adjective and conjunction orderings, which often show strong preferences.

In the following examples, there is strong preference for the first pair in all cases: big red cube / red big cube, cold wet weather / wet cold weather, near and far / far and near, brandy and soda / soda and brandy. Various hypotheses have been put forward to explain these effects, involving features such as relative power. But perhaps the simplest hypothesis is that when an individual speaks or writes, their ordering of a particular pair is based on their previous exposure to the phrase, with the ordering of novel pairs determined by their similarity to the previously seen pairs (see e.g. Copestake and Herbelot 2011). Although some techniques in use in computational linguistics involve complex approaches and high-end computer resources, experiments to investigate a hypothesis such as this can be carried out with an ordinary laptop with little programming being required.

31.5 Conclusion

Computational linguists have developed models both for engineering purposes and as methodology for investigating language. The fact that the models work in applications is an indication that some real properties of language are being captured. However, all the approaches I have outlined here have well-known limitations, and it may be the case that for performance to continue to improve, fundamentally different models will have to be tried.

One area where there is much more scope for computational linguists and linguists to collaborate, even if their theoretical assumptions are very different, is in the development of better corpora for more realistic language modelling. Although very large text corpora are readily available, with billions of words being used in some applications, their use is problematic in experiments which aim to model human behaviour. Accurate estimates for the number of words people are exposed to over the course of their lifetimes are hard to find, but listening to constant speech for six hours every day would correspond to around 20 million words a year. This implies that it should be possible to model adult speakers using corpora of perhaps 300 million words. However, the available data does not correspond to day-to-day use of language. Although part of the British National Corpus contains transcribed conversation, this is very small in comparison to text corpora and other available corpora of conversations are much less varied. The ideal corpus would also have associated video and audio, or at least be annotated in enough detail to get some idea of the context of an interaction, but even the more modest aim of acquiring transcribed conversational data would be very helpful in enabling computational linguists and linguists to improve their models of human language.
Notes

1 There is considerable debate about measuring parsing accuracy, since different approaches yield different types of output. Many current parsers produce dependency structures (see Chapter 8), which specify subjects, objects, modifiers, coordinated terms and so on. The best parsers for English get about 90 per cent of dependencies correct when trained and tested on newspaper text. This accuracy generally drops off markedly for other text types. Parser performance with other languages is often considerably worse. Thus CL systems that incorporate syntactic parsers have to be robust to their inaccuracies.

2 A corpus (plural corpora) is simply a body of machine-readable text. See also Chapter 32 on corpus linguistics, but note that the requirements of computational linguistics are often very different.

3 See Liberman’s (2010) obituary for Jelinek for a detailed discussion of the background.

4 Generalizations over categories can be useful for stochastic approaches in making predictions about rare events, but in the case of syntactic categories, this would require that lexical information is available, which is not the case for low frequency words.

5 https://pargram.b.uib.no/

6 www.delph-in.net

Further Reading

Bender (2013); Erk (2012); Jurafsky and Martin (2008); Manning and Schütze (1999).

References


