Natural Language Processing: Part II
Overview of Natural Language Processing (L90): Part III/ACS

2016, 12 Lectures, Michaelmas Term
September 26, 2016
Ekaterina Shutova (es407@cam.ac.uk)
http://www.cl.cam.ac.uk/users/es407/

Lecture Synopsis

Aims

This course introduces the fundamental techniques of natural language processing. It aims to explain the potential and the main limitations of these techniques. Some current research issues are introduced and some current and potential applications discussed and evaluated.

1. **Introduction.** Brief history of NLP research, current applications, components of NLP systems.
2. **Finite-state techniques.** Inflectional and derivational morphology, finite-state automata in NLP, finite-state transducers.
3. **Prediction and part-of-speech tagging.** Corpora, simple N-grams, word prediction, stochastic tagging, evaluating system performance.
4. **Context-free grammars and parsing.** Generative grammar, context-free grammars, parsing with context-free grammars, weights and probabilities. Limitations of context-free grammars. Dependencies.
5. **Lexical semantics.** Semantic relations, WordNet, word senses, word sense disambiguation.
6. **Distributional semantics 1.** Representing lexical meaning with distributions. Similarity metrics.
7. **Distributional semantics 2.** Generalisation and clustering. Selectional preference induction. Multimodal semantics.
8. **Compositional semantics.** Compositional semantics with FOPL and lambda calculus. Compositional distributional semantics. Inference and entailment.
9. **Discourse processing.** Anaphora resolution, discourse relations.
10. **Language generation and regeneration.** Components of a generation system. Summarisation.
11. **Applications.** Examples of practical applications of NLP techniques.
12. **Recent trends in NLP research.** Recent trends in NLP research.

Objectives

At the end of the course students should

- be able to discuss the current and likely future performance of several NLP applications;
- be able to describe briefly a fundamental technique for processing language for several subtasks, such as morphological processing, parsing, word sense disambiguation etc.;
- understand how these techniques draw on and relate to other areas of computer science.

Overview

NLP is a large and multidisciplinary field, so this course can only provide a very general introduction. The idea is that this is a 'taster' course that gives an idea of the different subfields and shows a few of the huge range of computational techniques that are used. The first lecture is designed to give an overview including a very brief idea of the main applications and the methodologies which have been employed. The history of NLP is briefly discussed as a way of putting this into perspective. The next nine lectures describe some of the main subdisciplines in more detail. The organisation is mainly based on increased 'depth' of processing, starting with relatively surface-oriented techniques and progressing to considering meaning of sentences and meaning of utterances in context. Most lectures will start off by considering the subarea as a whole and then go on to describe one or more sample algorithms which tackle particular problems. The algorithms have been chosen because they are relatively straightforward to describe and because they illustrate a specific technique which has been shown to be useful, but the idea is to exemplify an approach, not to give a detailed survey (which would be impossible in the time available). Lectures 2-9 are primarily about analysing language: lecture 10 discusses generation. Lecture 11 presents practical applications of NLP techniques. The final lecture is intended to give further context: it will include discussion of recent developments in the field of NLP. The material in Lectures 11 and 12 will not be directly examined. Slides for Lectures 11 and 12 will be made available via the course webpage after the lecture.

There are various themes running throughout the lectures. One theme is the connection to linguistics and the tension that sometimes exists between the predominant view in theoretical linguistics and the approaches adopted within NLP. A somewhat related theme is the distinction between knowledge-based and probabilistic approaches. Evaluation will be discussed in the context of the different algorithms.

Because NLP is such a large area, there are many topics that aren't touched on at all in these lectures. Speech recognition and speech synthesis is almost totally ignored. Information Retrieval is the topic of a separate Part II course.

Feedback on the handout, lists of typos etc, would be greatly appreciated.

Recommended Reading

Recommended Book:

Jurafsky, Daniel and James Martin, *Speech and Language Processing*, Prentice-Hall, 2008 (second edition): referenced as J&M throughout this handout. This doesn't have anything on language generation (lecture 10): there is a chapter on this in the first edition of J&M but it is not useful for this course.

Background:

These books are about linguistics rather that NLP/computational linguistics. They are not necessary to understand the course, but should give readers an idea about some of the properties of human languages that make NLP interesting and challenging, without being technical.

Pinker, S., *The Language Instinct*, Penguin, 1994.

This is a thought-provoking and sometimes controversial 'popular' introduction to linguistics.

Matthews, Peter, *Linguistics: a very short introduction*, OUP, 2003.

The title is accurate ...

Background/reference:

The Internet Grammar of English, `http://www.ucl.ac.uk/internet-grammar/home.htm`

Syntactic concepts and terminology.

Bender, Emily M. 2013. *Linguistic Fundamentals for Natural Language Processing: 100 Essentials from Morphology and Syntax.* Synthesis Lectures on Human Language Technologies #20. Morgan and Claypool Publishers.

This is based on a tutorial which was entitled '100 things you always wanted to know about Linguistics but were afraid to ask'[1]. It is an extremely succinct introduction to the concepts in morphology and syntax most relevant to computational linguistics, looking at a wide range of languages.

[1] '... for fear of being told 1000 more'

Study Guide

These lectures are now available to ACS/Part III students (as part of module L90, 'Overview of Natural Language Processing') as well as to Part II students. The other component of L90 is a practical exercise, organized by Simone Teufel. In the practical the students will have to apply the learned material to a real-world task (sentiment classification). The practical will be described in more detail in a separate document, and the assessment criteria will be published at `http://www.cl.cam.ac.uk/teaching/1617/L90/assessment.html`. Part II students are not expected to do that practical. ACS/Part III students will not take an exam.

The handouts and lectures should contain enough information to enable students to adequately answer the Part II exam questions, but the handout is not intended to substitute for a textbook (or for attending the lectures). In most cases, J&M go into a considerable amount of further detail: rather than put lots of suggestions for further reading in the handout, in general I have assumed that students will look at J&M, and then follow up the references in there if they are interested. The notes at the end of each lecture give details of the sections of J&M that are relevant and details of any discrepancies with these notes.

Supervisors (Part II only) should look at the supervision guide available from the course webpage and familiarise themselves with the relevant parts of Jurafsky and Martin (see notes at the end of each lecture). However, good students should find it quite easy to come up with questions that the supervisors (and the lecturer) can't answer! Language is like that ...

Generally I'm taking a rather informal/example-based approach to concepts such as finite-state automata, context-free grammars etc. The assumption is that students will have already covered this material in other contexts and that this course will illustrate some NLP applications.

This course inevitably assumes some very basic linguistic knowledge, such as the distinction between the major parts of speech. It introduces some linguistic concepts that won't be familiar to all students: since I'll have to go through these quickly, reading the first few chapters of an introductory linguistics textbook may help students understand the material. The idea is to introduce just enough linguistics to motivate the approaches used within NLP rather than to teach the linguistics for its own sake. At the end of this handout, there are some mini-exercises to help students understand the concepts: it would be very useful if these were attempted before the lectures as indicated. There are also some suggested post-lecture exercises: answers to these are made available to supervisors only.

As far as possible, exam questions will be suitable for people who speak English as a second language. For instance, if a question relied on knowledge of the ambiguity of a particular English word, a gloss of the relevant senses would be given.

Of course, I'll be happy to try and answer questions about the course or more general NLP questions, preferably by email.

Changes to the course since previous years.

In 2016-17 the course has seen a number of changes. The material on statistical NLP has been expanded, covering topics such as generalisation and clustering, selectional preference induction, compositional distributional semantics and multimodal semantics. The constraint-based grammar material has been replaced by compositional distributional semantics. The examples in the NLG lecture (lecture 10) have been replaced by a review of summarisation techniques. Lecture 11 will be a guest lecture on statistical dialogue systems.

Generally, past exam questions from 2003 onwards are still relevant, with the exception of the ones addressing Lappin and Leass algorithm and constraint-based grammars, which are no longer taught.

URLs

Nearly all the URLs given in these notes should be linked from:
`http://www.cl.cam.ac.uk/~aac10/stuff.html`
(apart from this one of course ...).

1 Lecture 1: Introduction to NLP

The primary aim of this lecture is to give students some idea of the objectives of NLP. The main subareas of NLP will be introduced, especially those which will be discussed in more detail in the rest of the course. There will be a preliminary discussion of the main problems involved in language processing by means of examples taken from NLP applications. This lecture also introduces some methodological distinctions and puts the applications and methodology into some historical context.

1.1 What is NLP?

Natural language processing (NLP) can be defined as the computational modelling of human language. The term 'NLP' is sometimes used rather more narrowly than that, often excluding information retrieval and sometimes even excluding machine translation. NLP is sometimes contrasted with 'computational linguistics', with NLP being thought of as more applied. Nowadays, alternative terms are often preferred, like 'Language Technology' or 'Language Engineering'. The term 'language' is often used in contrast with 'speech' (e.g., Speech and Language Technology). But I'm going to simply refer to NLP and use the term broadly.

NLP is essentially multidisciplinary: it is closely related to linguistics (although the extent to which NLP overtly draws on linguistic theory varies considerably). Like NLP, formal linguistics deals with the development of models of human languages, but some approaches in linguistics reject the validity of statistical techniques, which are seen as an essential part of computational linguistics. NLP also has links to research in cognitive science, psychology, philosophy and maths (especially logic). Within CS, it relates to formal language theory, compiler techniques, theorem proving, machine learning and human-computer interaction. Of course it is also related to AI, though nowadays it's not generally thought of as part of AI.

1.2 Some linguistic terminology

The course is organised so that there are ten lectures after this one corresponding to different NLP subareas. In the first eight, the discussion moves from relatively 'shallow' processing to areas which involve meaning and connections with the real world. These subareas loosely correspond to some of the standard subdivisions of linguistics:

1. Morphology: the structure of words. For instance, *unusually* can be thought of as composed of a prefix *un-*, a stem *usual*, and an affix *-ly*. *composed* is *compose* plus the inflectional affix *-ed*: a spelling rule means we end up with *composed* rather than *composeed*. Morphology will be discussed in lecture 2.

2. Syntax: the way words are used to form phrases. e.g., it is part of English syntax that a determiner (a word such as *the*) will come before a noun, and also that determiners are obligatory with certain singular nouns. Formal and computational aspects of syntax will be discussed in lectures 3 and 4.

3. Semantics. Compositional semantics is the construction of meaning (often expressed as logic) based on syntax. This is discussed in lecture 8. This is contrasted to lexical semantics, i.e., the meaning of individual words which is the topic of lectures 5, 6 and 7.

4. Pragmatics: meaning in context. This will come into lecture 9, although linguistics and NLP generally have very different perspectives here.

Lecture 10 looks at language generation rather than language analysis, and lecture 11 covers some practical applications of NLP techniques.

1.3 Why is computational language processing difficult?

Consider trying to build a system that would answer email sent by customers to a retailer selling laptops and accessories via the Internet. This might be expected to handle queries such as the following:

- Has my order number 4291 been shipped yet?

- Is FD5 compatible with a 505G?
- What is the speed of the 505G?

Assume the query is to be evaluated against a database containing product and order information, with relations such as the following:

ORDER

Order number	Date ordered	Date shipped
4290	2/2/13	2/2/13
4291	2/2/13	2/2/13
4292	2/2/13	

USER: Has my order number 4291 been shipped yet?
DB QUERY: order(number=4291,date_shipped=?)
RESPONSE TO USER: Order number 4291 was shipped on 2/2/13

It might look quite easy to write patterns in order to build a system to respond to these queries, but very similar strings can mean very different things, while very different strings can mean much the same thing. 1 and 2 below look very similar but mean something completely different, while 2 and 3 look very different but essentially mean the same in this context.

1. How fast is the TZ?
2. How fast will my TZ arrive?
3. Please tell me when I can expect the TZ I ordered.

While some tasks in NLP can be done adequately without having any sort of account of meaning, others require that we can construct detailed representations which will reflect the underlying meaning rather than the superficial string.

In fact, in natural languages (as opposed to programming languages), ambiguity is ubiquitous, so exactly the same string might mean different things. For instance in the query:

> Do you sell Sony laptops and disk drives?

the user may or may not be asking about Sony disk drives. This particular ambiguity may be represented by different bracketings:

> Do you sell (Sony laptops) and (disk drives)?
> Do you sell (Sony (laptops and disk drives))?

We'll see lots of examples of different types of ambiguity in these lectures.

Natural language has properties which are essential to communication which are not found in formal languages, such as predicate calculus, computer programming languages, semantic web languages and so on. Natural language is incredibly flexible. It is learnable, but compact. Natural languages are emergent, evolving systems. Ambiguity and synonymy are inherent to flexibility and learnability. Despite ambiguity, natural language can be indefinitely precise: ambiguity is largely local[2] (at least for humans) and natural languages accommodate (semi-)formal additions.

Often humans have knowledge of the world which resolves a possible ambiguity, probably without the speaker or hearer even being aware that there is a potential ambiguity.[3] But hand-coding such knowledge in NLP applications has turned out to be impossibly hard to do for more than very limited domains: the term *AI-complete* is sometimes used (by analogy to NP-complete), meaning that we'd have to solve the entire problem of representing the world

[2] i.e., immediate context resolves the ambiguity: examples of this will be discussed in later lectures.

[3] I'll use *hearer* generally to mean the person who is on the receiving end, regardless of the modality of the language transmission: i.e., regardless of whether it's spoken, signed or written. Similarly, I'll use *speaker* for the person generating the speech, text etc and *utterance* to mean the speech or text itself. This is the standard linguistic terminology, which recognises that spoken language is primary and text is a later development.

and acquiring world knowledge.[4] The term AI-complete is intended jokingly, but conveys what's probably the most important guiding principle in current NLP: we're looking for applications which don't require AI-complete solutions: i.e., ones where we can either work with very limited domains or approximate full world knowledge and reasoning by relatively simple techniques.

1.4 Some NLP applications

Useful NLP systems have been built for a large range of applications, including (but not limited to):

- spelling and grammar checking
- predictive text
- optical character recognition (OCR)
- screen readers for blind and partially sighted users
- augmentative and alternative communication (i.e., systems to aid people who have difficulty communicating because of disability)
- machine aided translation (i.e., systems which help a human translator, e.g., by storing translations of phrases and providing online dictionaries integrated with word processors, etc)
- lexicographers' tools
- information retrieval
- document classification (filtering, routing)
- document clustering
- information extraction
- sentiment classification
- text mining
- question answering
- summarization
- text segmentation
- exam marking
- language teaching and assessment
- report generation (possibly multilingual)
- machine translation
- natural language interfaces to databases
- email understanding
- dialogue systems

Several of these applications are discussed briefly below. Roughly speaking, the list is ordered according to the complexity of the language technology required. The applications towards the top of the list can be seen simply as aids to human users, while those at the bottom may be perceived as agents in their own right. Perfect performance on any of these applications would be AI-complete, but perfection isn't necessary for utility: in many cases, useful versions of these applications had been built by the late 70s. Commercial success has often been harder to achieve, however.

[4] In this course, I will use *domain* to mean some circumscribed body of knowledge: for instance, information about laptop orders constitutes a limited domain.

1.5 An example application: Sentiment classification

Finding out what people think about politicians, policies, products, services, companies and so on is a huge 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 Daily Mail articles appearing in September 2014, 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 (as 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 analyzing 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 fixed 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 has still been a challenging problem to address. Some of the early research work was done on movie reviews (Pang et al, 2002).[5] The rating associated with each review is known (that is, reviewers give each movie a numerical score), 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.[6] Pang et al only selected review articles which had clear positive or negative ratings, and balanced their corpus so the positive/negative split was 50/50.

The most basic technique is to look at the words in the review in isolation of each other, and to classify the document on the basis of whether those words generally indicate positive or negative reviews. This is a *bag-of-words* technique: we model the document as an unordered collection of words (*bag* rather than set because there will be repetition). A document with more positive words than negative ones should be a positive review. In principle, this could be done by using human judgements of positive/negative words, but using machine learning techniques works better[7] (humans don't consider many words that turn out to be useful indicators). However, Pang et al found that the accuracy of the classification is only around 80% (for a problem where there was a 50% chance success rate). One source of errors is negation: (e.g., *Ridley Scott has never directed a bad film* is a positive statement). Another problem is that the machine learning technique may match the data too closely: e.g., if a machine learner were trained on reviews which include a lot of films from before 2005, it would discover that *Ridley* was a strong positive indicator but it would then tend to misclassify reviews for 'Kingdom of Heaven' (which was panned). More subtle problems arise from not tracking 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' `http://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 isnt necessarily a good thing, since it prevents us from relaxing and enjoying AN AMERICAN WEREWOLF IN PARIS as a completely mindless, campy entertainment experience. Delpys 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 `http://www.imdb.com/reviews/103/10363.html`]

Unfortunately, although in principle NLP techniques can deal with syntax, semantics and discourse and thus address these sort of problems, doing this in a way that can significantly improve performance over the simple system turns

[5] Pang, Lee and Vaithyanatha (2002), *Thumbs up? Sentiment Classification using Machine Learning Techniques* In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[6] A *corpus* (plural *corpora*) is the technical term for a body of text that has been collected for some purpose, see §3.1.

[7] Classifiers are discussed in more detail in lecture 9.

out to be hard. Many 'obvious' techniques for improving the bag-of-words model did not improve performance. For instance, although a good model should include negation, it only applies to a small percentage of examples. Modelling negation requires parsing to determine the scope of negation correctly, but it is easy to introduce errors which make performance worse overall. Nevertheless, systems have been developed that have considerably better performance than the simple methods: for instance, Moilanen and Pulman (2007) [8] demonstrated that parsing and a form of compositional semantics (topics which will be introduced in Lectures 4 and 8) can considerably improve performance over the simple methods. Ultimately to understand whether a statement is really positive or negative in a particular context is AI-complete (think about irony, for instance), but the (application-oriented) question is not whether automatic methods can be perfect but whether they can be good enough to be useful.

1.6 Information retrieval, information extraction and question answering

Information retrieval (IR) involves returning a set of documents in response to a user query: Internet search engines were developed from traditional IR techniques, but use additional methods such as ranking documents according to links (e.g., Google's PageRank).

Information extraction involves trying to discover specific information from a set of documents. The information required can be described as a template. For instance, for company joint ventures, the template might have slots for the companies, the dates, the products, the amount of money involved. The slot fillers are generally strings.

Question answering attempts to find a specific answer to a specific question from a set of documents, or at least a short piece of text that contains the answer.

(1) Q: What is the capital of France?
A: Paris has been the French capital for many centuries.

There have been question-answering systems on the Web since the 1990s, but until recently most used very basic techniques. One common approach involved employing a large staff of people who searched the web to find pages which are answers to potential questions. The question-answering system then performed very limited manipulation on the actual user input to map to a known question. The same basic technique is used in many online help systems. However, with enough resource, impressive results from automatic QA are now possible: most famously an IBM research team created a QA system that beat human champions on the quiz show *Jeopardy!* in 2011 (see Ferrucci et al, AI Magazine, 2010, for an overview). Current QA systems are very successful at answering questions that humans find difficult, but very bad at answering many questions that humans find trivial, essentially because such information is not explicitly stated anywhere online. The problem is that the trivial information is essential for modelling human reasoning and thus for really understanding language.

1.7 Machine translation

MT work started in the US in the early fifties, concentrating on Russian to English. A prototype system was publicly demonstrated in 1954 (remember that the first electronic computer had only been built a few years before that). MT funding was drastically cut in the US in the mid-60s and ceased to be academically respectable in some places, but Systran was providing useful translations by the late 60s using hand-built *transfer rules* to map between source and target languages. Systran is still going (updating it over the years is an amazing feat of software engineering). Systran was used for most of the language pairs available from Google Translate until about 2007/2008, but Google now uses a *statistical MT* (SMT) system which was developed in-house, exploiting Google's access to the huge amount of *parallel text* available on the web (i.e., they mine web pages for pairings of source text and translated text). This works very well for some language pairs and very badly for others, depending firstly on the availability of properly translated parallel text and secondly on the closeness of the languages. (But for some language pairs, MT systems using manually-constructed rules still greatly outperform SMT.)

Until the 80s, the utility of general purpose MT systems was severely limited by the fact that text was not available in electronic form: for instance, in the 1960s and 70s, teams of skilled typists were needed to input Russian documents before the MT system was run.

[8] Moilanen, Karo, and Stephen Pulman. "Sentiment composition." In Proceedings of the Recent Advances in Natural Language Processing International Conference, pp. 378-382. 2007.

None of these systems are a substitute for good human translation: they are useful because they allow users to get an idea of what a document is about, and maybe decide whether it is interesting enough to get translated properly.

1.8 Natural language interfaces and dialogue systems

Natural language interfaces were the 'standard' NLP application in the 70s and 80s. LUNAR is the classic example of a natural language interface to a database (NLID): it concerned lunar rock samples brought back from the Apollo missions. LUNAR is described by Woods (1978) (but the work was done several years earlier): it was capable of translating elaborate natural language expressions into database queries.

SHRDLU (Winograd, 1973) was a system capable of participating in a dialogue about a microworld (the blocks world) and manipulating this world according to commands issued in English by the user. SHRDLU had a big impact on the perception of NLP at the time since it seemed to show that computers could actually 'understand' language: the impossibility of scaling up from the microworld was not realised.

LUNAR and SHRDLU both exploited the limitations of one particular domain to make the natural language understanding problem tractable, particularly with respect to ambiguity. To take a trivial example, if you know your database is about lunar rock, you don't need to consider the music or movement senses of *rock* when you're analysing a query.

There have been many advances in NLP since these systems were built: natural language interface systems have become much easier to build, and somewhat easier to use, but they still haven't become ubiquitous. Natural Language interfaces to databases were commercially available in the late 1970s, but largely died out by the 1990s: porting to new databases and especially to new domains requires very specialist skills and is essentially too expensive (automatic porting was attempted but never successfully developed). Users generally preferred graphical interfaces when these became available.

Modern 'intelligent' personal assistants, such as Siri, use interfaces to databases as part of their functionality, along with search and question answering. 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.

1.9 Some more history

Before the 1970s, most NLP researchers were concentrating on MT as an application (see above). NLP was a very early application of computer science and started about the same time as Chomsky was publishing his first major works in formal linguistics (Chomskyan linguistics quickly became dominant, especially in the US). In the 1950s and early 1960s, there was little distinction between formal linguistics and computational linguistics: ideas about formal grammar were being worked out in linguistics, and algorithms for parsing natural language were being developed at the same time as algorithms for parsing programming languages. However, the approaches that Chomsky and his colleagues developed turned out to be only somewhat indirectly useful for NLP and the two fields became distinct, at least in the US and UK.

NLP in the 1970s and first half of the 1980s was predominantly based on a paradigm where extensive linguistic and real-world knowledge was hand-coded. There was controversy about how much linguistic knowledge was necessary for processing, with some researchers downplaying syntax, in particular, in favour of world knowledge. NLP researchers were very much part of the AI community (especially in the US and the UK), and the debate that went on in AI about the use of logic vs other meaning representations ('neat' vs 'scruffy') also affected NLP. By the 1980s, several linguistic formalisms had appeared which were fully formally grounded and reasonably computationally tractable, and the linguistic/logical paradigm in NLP was firmly established. Unfortunately, this didn't lead to many useful systems, partly because many of the difficult problems (disambiguation etc) were seen as somebody else's job (and mainstream AI was not developing adequate knowledge representation techniques) and partly because most researchers were concentrating on the 'agent-like' applications and neglecting the user aids. Although the symbolic, linguistically-based systems sometimes worked quite well as NLIDs, they proved to be of little use when it came to processing less restricted text, for applications such as IE. It also became apparent that lexical acquisition was a serious bottleneck for serious development of such systems.

Statistical NLP became the most common paradigm in the 1990s, at least in the research community. By this point, there was a huge divide between mainstream linguists and the NLP community. Chomsky had declared:

> But it must be recognized that the notion 'probability of a sentence' is an entirely useless one, under any known interpretation of this term. (Chomsky 1969)

Certain linguistics journals would not even review papers which had a quantitative component. But speech and NLP researchers wanted results:

> Whenever I fire a linguist our system performance improves. (Fred Jelinek, allegedly said at a workshop in 1988, various forms of the quotation have been attested. Jelinek later said he never actually fired anyone, they just left . . .)

Speech recognition had demonstrated that simple statistical techniques worked, given enough training data. NLP systems were built which required very limited hand-coded knowledge, apart from initial training material. Most applications were much shallower than the earlier NLIDs, but the switch to statistical NLP coincided with a change in US funding, which started to emphasise speech recognition and IE. There was also a general realization of the importance of serious evaluation and of reporting results in a way that could be reproduced by other researchers. US funding emphasised competitions with specific tasks and supplied test material, which encouraged this, although there was a downside in that some of the techniques developed were very task-specific. It should be emphasised that there had been computational work on corpora for many years (much of it by linguists): it became much easier to do corpus work by the late 1980s as disk space became cheap and machine-readable text became ubiquitous. Despite the shift in research emphasis to statistical approaches, most commercial systems remained primarily based on hand-coded linguistic information.

Later the symbolic/statistical split became less pronounced, since most researchers are interested in both.[9] Most work involves some type of machine learning, including machine learning for symbolic processing. Linguistically-based NLP made something of a comeback, with increasing availability of open source resources, and the realisation that at least some of the classic statistical techniques seem to be reaching limits on performance, especially because of difficulties of acquiring training data and in adapting to new types of text. However, modern linguistically-based NLP approaches make heavy use of machine learning and statistical processing.

The ubiquity of the Internet has completely changed the space of interesting NLP applications since the early 1990s, and the vast amount of text available can potentially be exploited, especially for statistical techniques. The dotcom boom and bust at the turn of the millennium considerably affected NLP in industry but interest has increased again and at the time of writing (2014), another boom is well under way.

1.10 Combining components in NLP systems

Many NLP applications can be adequately implemented with relatively 'shallow' processing. For instance, English spelling checking only requires a word list and simple morphology to be useful. I'll use the term 'deep' NLP for systems that build a meaning representation (or an elaborate syntactic representation), which is generally agreed to be required for applications such as NLIDs and email question answering.

The input to an NLP system could be speech or text. It could also be gesture (multimodal input or perhaps a Sign Language). The output might be non-linguistic, but most systems need to give some sort of feedback to the user, even if they are simply performing some action (issuing a ticket, paying a bill, etc). However, often the feedback can be very formulaic.

There's general agreement that certain NLP subtasks can be described semi-independently and combined to build an application, although assumptions about the detailed nature of the interfaces between the tasks differ:

- input preprocessing: speech recogniser or text preprocessor (non-trivial in languages like Chinese or for highly structured text for any language) or gesture recogniser. Such systems might themselves be very complex, but I won't discuss them in this course — we'll assume that the input to the main NLP component is segmented text.
- morphological analysis (lecture 2): this is relatively well-understood for the most common languages that NLP has considered, but is complicated for many languages (e.g., Turkish, Basque).

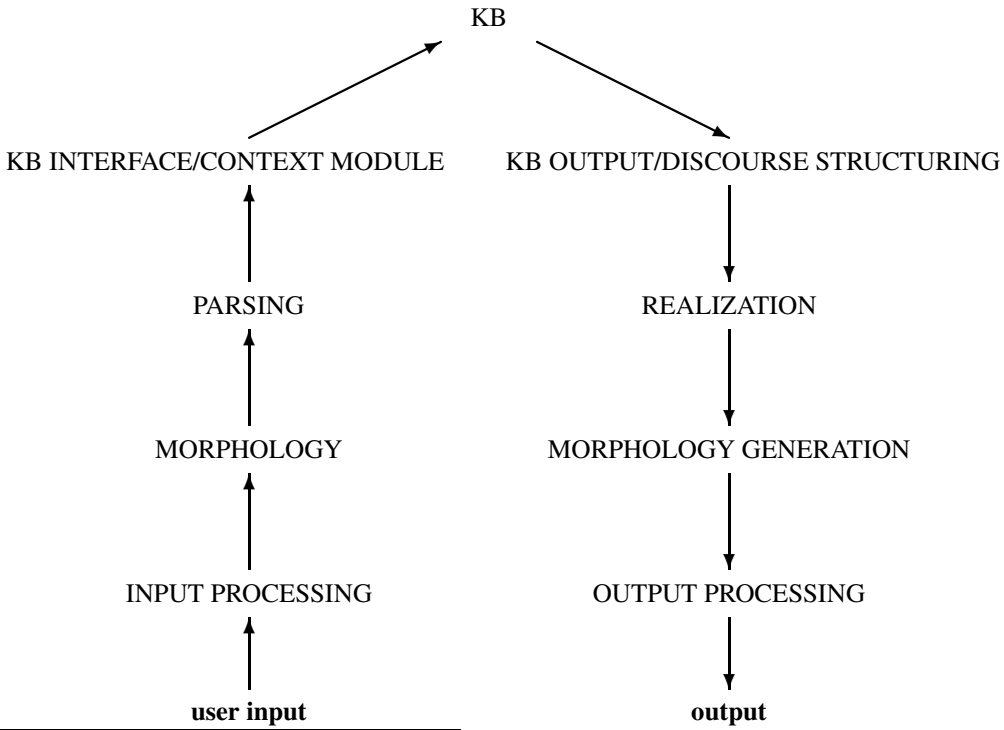
[9] At least, there are now very few researchers who avoid statistical techniques as a matter of principle and all statistical systems have a symbolic component!

- part of speech tagging (lecture 3): (used by deep processing systems as a way of cutting down parser search space).
- parsing (lecture 4): uncovering syntactic structure (also applied in compositional semantics)
- further semantic processing includes word sense disambiguation (lecture 5) and inference (lecture 8): these tasks are supported by lexical meaning representations (lectures 5, 6 and 7).
- context processing: this includes a range of subtasks, including anaphora resolution (lecture 9).
- discourse structuring: the part of language generation that's concerned with deciding what meaning to convey (lecture 10).
- realization (lecture 10): the inverse of parsing, i.e., conversion of meaning representations or syntactic structures to strings. This may use the same grammar and lexicon[10] as the parser.
- morphological generation (lecture 2): as with morphological analysis, this is relatively straightforward for English.
- output processing: text-to-speech, text formatter, etc. As with input processing, this may be simple or complex.

For a system implementing an NLP subtask to correspond to a module in an application, it obviously needs a well-defined set of interfaces. What's less obvious is that it needs its own evaluation strategy and test suites so it can be developed independently. In principle, at least, components are *reusable* in various ways: for instance, a parser could be used with multiple grammars, the same grammar can be processed by different parsers and generators, a parser/grammar combination could be used in MT or in a natural language interface. However, for a variety of reasons, it is not easy to reuse components like this, and generally a lot of work is required for each new application.

We can draw schematic diagrams for applications showing how the subtasks fit together.

Natural language interface to a knowledge base



[10]The term *lexicon* is generally used for the part of the NLP system that contains dictionary-like information — i.e. information about individual words.

However, this is an abstraction. Simple pipelining does not work well because each component makes errors. Much more complex architectures may be used: the IBM Jeopardy playing system has over 100 modules which produce multiple candidate answers — sophisticated probabilistic methods are used to rank these. Nevertheless, the diagram should give some indication of the potential role of each subtask in a full application.

In lectures 2–10, various algorithms will be discussed which correspond to implementations of the various subtasks. Lecture 11 will discuss a practical NLP application in some more detail.

1.11 General comments

- Even 'simple' NLP applications need complex knowledge sources for some problems.
- Applications cannot be 100% perfect, because full real world knowledge and reasoning is not possible.
- Applications that are less than 100% perfect can be useful (humans aren't 100% perfect anyway).
- Applications that aid humans are much easier to construct than applications which replace humans. It is difficult to make the limitations of 'agents' which accept speech or text input clear to naive human users.
- NLP interfaces are nearly always competing with a non-language based approach.
- Currently nearly all applications either do relatively shallow processing on arbitrary input or deep processing on narrow domains. MT can be domain-specific to varying extents: MT on arbitrary text still isn't very good (in general), but can be useful.
- Limited domain systems require extensive and expensive expertise to port. Research that relies on extensive hand-coding of knowledge for small domains is now generally regarded as a dead-end, though reusable hand-coding is a different matter.
- The development of NLP has been driven as much by hardware and software advances, and societal and infrastructure changes as by great new ideas. Improvements in NLP techniques are generally incremental rather than revolutionary.

2 Lecture 2: Morphology and finite-state techniques

This lecture starts with a brief discussion of morphology, concentrating mainly on English morphology. The concept of a lexicon in an NLP system is discussed with respect to morphological processing. Spelling rules are introduced and the use of finite state transducers to implement spelling rules is explained. The lecture concludes with a brief overview of some other uses of finite state techniques in NLP.

2.1 A very brief and simplified introduction to morphology

Morphology concerns the structure of words. Words are assumed to be made up of *morpheme*s, which are the minimal information carrying unit. Morphemes which can only occur in conjunction with other morphemes are *affix*es: words are made up of a stem (more than one in the case of compounds) and zero or more affixes. For instance, *dog* is a stem which may occur with the plural suffix *+s* i.e., *dogs*. The compound *bookshop* has two stems (*book* and *shop*): most English compounds are spelled with a space, however, and therefore not standardly analysed by morphological processors. English only has suffixes (affixes which come after a stem) and prefixes (which come before the stem — in English prefixes are limited to derivational morphology), but other languages have *infixes* (affixes which occur inside the stem) and circumfixes (affixes which go around a stem, such as the *ge-t* in German *gekauft*). For instance, Arabic has stems (root forms) such as *k_t_b*, which are combined with infixes to form words (e.g., *kataba*, he wrote; *kotob*, books). Some English irregular verbs show a relic of inflection by infixation (e.g. *sing*, *sang*, *sung*) but this process is no longer *productive* (i.e., it won't apply to any new words, such as *ping*).[11]

Note the requirement that a morpheme can be regarded as a unit. There are cases where there seems to be a similarity in meaning between some clusters of words with similar spellings: e.g., *slink*, *slide*, *slither*, *slip*. But such examples cannot be decomposed (i.e., there is no *sl-* morpheme) because the rest of the word does not stand as a unit.

2.2 Inflectional vs derivational morphology

Inflectional and derivational morphology can be distinguished, although the dividing line isn't always sharp. The distinction is of some importance in NLP, since it means different representation techniques may be appropriate. Inflectional morphology can be thought of as setting values of slots in some *paradigm* (i.e., there is a fixed set of slots which can be thought of as being filled with simple values). Inflectional morphology concerns properties such as tense, aspect, number, person, gender, and case, although not all languages code all of these: English, for instance, has very little morphological marking of case and gender. Derivational affixes, such as *un-*, *re-*, *anti-* etc, have a broader range of semantic possibilities (there seems no principled limit on what they can mean) and don't fit into neat paradigms. Inflectional affixes may be combined (though not in English). However, there are always obvious limits to this, since once all the possible slot values are 'set', nothing else can happen. In contrast, there are no obvious limitations on the number of derivational affixes (*antidisestablishmentarianism*, *antidisestablishmentarianismization*) and they may even be applied recursively (*antiantimissile*). In some languages, such as the Inuit language(s), derivational morphology is often used where English would use adjectival modification or other syntactic means. This leads to very long 'words' occurring naturally and is responsible for the (misleading/mistaken) claim that 'Eskimo' has hundreds of words for snow.

Inflectional morphology is generally close to fully productive, in the sense that a word of a particular class will show all the possible inflections although the actual affix used may vary. For instance, an English verb will have a present tense form, a 3rd person singular present tense form, a past participle and a passive participle (the latter two being the same for regular verbs). This will also apply to any new words which enter the language: e.g., *text* as a verb — *texts*, *texted*. Derivational morphology is less productive and the classes of words to which an affix applies is less clearcut. For instance, the suffix *-ee* is relatively productive (*textee* sounds plausible, meaning the recipient of a text message, for instance), but doesn't apply to all verbs (?*snoree*, ?*jogee*, ?*dropee*). Derivational affixes may change the part of speech of a word (e.g., *-ise*/*-ize* converts nouns into verbs: *plural*, *pluralise*). However, there are also examples of what is sometimes called *zero derivation*, where a similar effect is observed without an affix: e.g. *tango*, *waltz* etc are words which are basically nouns but can be used as verbs.

[11] Arguably, though, spoken English has one productive infixation process, exemplified by *absobloodylutely*.

Stems and affixes can be individually ambiguous. There is also potential for ambiguity in how a word form is split into morphemes. For instance, *unionised* could be *union -ise -ed* or (in chemistry) *un- ion -ise -ed*. This sort of structural ambiguity isn't nearly as common in English morphology as in syntax, however. Note that *un- ion* is not a possible form (because *un-* can't attach to a noun). Furthermore, although there is a prefix *un-* that can attach to verbs, it nearly always denotes a reversal of a process (e.g., *untie*), whereas the *un-* that attaches to adjectives means 'not', which is the meaning in the case of *un- ion -ise -ed*. Hence the internal structure of *un- ion -ise -ed* has to be *(un- ((ion -ise) -ed))*.

2.3 Applications of morphological processing

It is possible to use a *full-form lexicon* for English NLP: i.e., to list all the inflected forms and to treat derivational morphology as non-productive. However, when a new word has to be handled (because the lexicon is incomplete, potentially because a new word has entered the language) it is redundant to have to specify (or learn) the inflected forms as well as the stem, since the vast majority of words in English have regular morphology. So a full-form lexicon is best regarded as a form of compilation. Many other languages have many more inflectional forms, which increases the need to do morphological analysis rather than full-form listing.

Traditional IR systems use *stemming* rather than full morphological analysis. For IR, what is required is to relate forms, not to analyse them compositionally, and this can most easily be achieved by reducing all morphologically complex forms to a canonical form. Although this is referred to as stemming, the canonical form may not be the linguistic stem. The most commonly used algorithm is the *Porter stemmer*, which uses a series of simple rules to strip endings (see J&M, section 3.8) without the need for a lexicon. However, stemming does not necessarily help IR. Search engines now generally do inflectional morphology, but this can be dangerous. For instance, searching for *corpus* as well as *corpora* when given the latter as input (as some search engines sometimes do) can result in a large number of spurious results involving *Corpus Christi* and similar terms.

In most NLP applications, however, morphological analysis is a precursor to some form of parsing. In this case, the requirement is to analyse the form into a stem and affixes so that the necessary syntactic (and possibly semantic) information can be associated with it. Morphological analysis is often called *lemmatization*. For instance, for the part of speech tagging application which I will discuss in the next lecture, *mugged* would be assigned a part of speech tag which indicates it is a verb, though *mug* is ambiguous between verb and noun. For full parsing, as discussed in lecture 4, we need more detailed syntactic and semantic information. Morphological generation takes a stem and some syntactic information and returns the correct form. For some applications, there is a requirement that morphological processing is *bidirectional*: that is, can be used for analysis and generation. The finite state transducers we will look at below have this property.

2.4 Spelling rules

English morphology is essentially concatenative: i.e., we can think of words as a sequence of prefixes, stems and suffixes. Some words have irregular morphology and their inflectional forms simply have to be listed. However, in other cases, there are regular phonological or spelling changes associated with affixation. For instance, the suffix *-s* is pronounced differently when it is added to a stem which ends in *s*, *x* or *z* and the spelling reflects this with the addition of an *e* (*boxes* etc). For the purposes of this course, I'll just talk about spelling effects rather than phonological effects: these effects can be captured by *spelling rules* (also known as *orthographic rules*).

English spelling rules can be described independently of the particular stems and affixes involved, simply in terms of the affix boundary. The 'e-insertion' rule can be described as follows:

$$\varepsilon \rightarrow \mathrm{e} / \left\{ \begin{array}{c} \mathrm{s} \\ \mathrm{x} \\ \mathrm{z} \end{array} \right\} \hat{} \_\_ s$$

In such rules, the mapping is always given from the 'underlying' form to the surface form, the mapping is shown to the left of the slash and the context to the right, with the __ indicating the position in question. ε is used for the empty string and ^ for the affix boundary. This particular rule is read as saying that the empty string maps to 'e' in the context where it is preceded by an s,x, or z and an affix boundary and followed by an s. For instance, this maps *box^s* to *boxes*.

This rule might look as though it is written in a context sensitive grammar formalism, but actually we'll see in §2.7 that it corresponds to a finite state transducer. Because the rule is independent of the particular affix, it applies equally to the plural form of nouns and the 3rd person singular present form of verbs. Other spelling rules in English include consonant doubling (e.g., *rat*, *ratted*, though note, not **auditted*) and y/ie conversion (*party*, *parties*).[12]

2.5 Lexical requirements for morphological processing

There are three sorts of lexical information that are needed for full, high precision morphological processing:

- affixes, plus the associated information conveyed by the affix
- irregular forms, with associated information similar to that for affixes
- stems with syntactic categories (plus more detailed information if derivational morphology is to be treated as productive)

One approach to an affix lexicon is for it to consist of a pairing of affix and some encoding of the syntactic/semantic effect of the affix.[13] For instance, consider the following fragment of a suffix lexicon (we can assume there is a separate lexicon for prefixes):

```
ed PAST_VERB
ed PSP_VERB
s PLURAL_NOUN
```

Here `PAST_VERB`, `PSP_VERB` and `PLURAL_NOUN` are abbreviations for some bundle of syntactic/semantic information and form the interface between morphology and the syntax/semantics.

A lexicon of irregular forms is also needed. One approach is for this to just be a triple consisting of inflected form, 'affix information' and stem, where 'affix information' corresponds to whatever encoding is used for the regular affix. For instance:

```
began PAST_VERB begin
begun PSP_VERB begin
```

Note that this information can be used for generation as well as analysis, as can the affix lexicon.

In most cases, English irregular forms are the same for all senses of a word. For instance, *ran* is the past of *run* whether we are talking about athletes, politicians or noses. This argues for associating irregularity with particular word forms rather than particular senses, especially since compounds also tend to follow the irregular spelling, even non-productively formed ones (e.g., the plural of *dormouse* is *dormice*). However, there are exceptions: e.g., *The washing was hung/*hanged out to dry* vs *the murderer was hanged*.

Morphological analysers also generally have access to a lexicon of regular stems. This is needed for high precision: e.g. to avoid analysing *corpus* as *corpu -s*, we need to know that there isn't a word *corpu*. There are also cases where historically a word was derived, but where the base form is no longer found in the language: we can avoid analysing *unkempt* as *un- kempt*, for instance, simply by not having *kempt* in the stem lexicon. Ideally this lexicon should have syntactic information: for instance, *feed* could be *fee -ed*, but since *fee* is a noun rather than a verb, this isn't a possible analysis. However, in the approach I'll assume, the morphological analyser is split into two stages. The first of these only concerns morpheme forms and returns both *fee -ed* and *feed* given the input *feed*. A second stage which is closely coupled to the syntactic analysis then rules out *fee -ed* because the affix and stem syntactic information are not compatible.

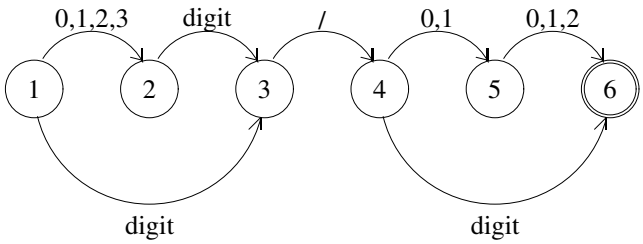
If morphology was purely concatenative, it would be very simple to write an algorithm to split off affixes. Spelling rules complicate this somewhat: in fact, it's still possible to do a reasonable job for English with ad hoc code, but a cleaner and more general approach is to use finite state techniques.

[12] Note the use of * ('star') above: this notation is used in linguistics to indicate a word or sentence which is judged (by the author, at least) to be incorrect. ? is generally used for a sentence which is questionable, or at least doesn't have the intended interpretation. # is used for a pragmatically anomalous sentence.

[13] J&M describe an alternative approach which is to make the syntactic information correspond to a level in a finite state transducer. However, at least for English, this considerably complicates the transducers.

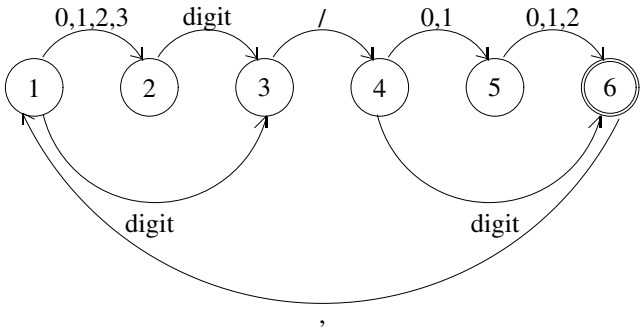
2.6 Finite state automata for recognition

The approach to spelling rules that I'll describe involves the use of finite state transducers (FSTs). Rather than jumping straight into this, I'll briefly consider the simpler finite state automata and how they can be used in a simple recogniser. Suppose we want to recognise dates (just day and month pairs) written in the format day/month. The day and the month may be expressed as one or two digits (e.g. 11/2, 1/12 etc). This format corresponds to the following simple FSA, where each character corresponds to one transition:



Accept states are shown with a double circle. This is a non-deterministic FSA: for instance, an input starting with the digit 3 will move the FSA to both state 2 and state 3. This corresponds to a *local ambiguity*: i.e., one that will be resolved by subsequent context. By convention, there must be no 'left over' characters when the system is in the final state.

To make this a bit more interesting, suppose we want to recognise a comma-separated list of such dates. The FSA, shown below, now has a cycle and can accept a sequence of indefinite length (note that this is iteration and not full recursion, however).



Both these FSAs will accept sequences which are not valid dates, such as 37/00. Conversely, if we use them to generate (random) dates, we will get some invalid output. In general, a system which generates output which is invalid is said to *overgenerate*. In fact, in many language applications, some amount of overgeneration can be tolerated, especially if we are only concerned with analysis.

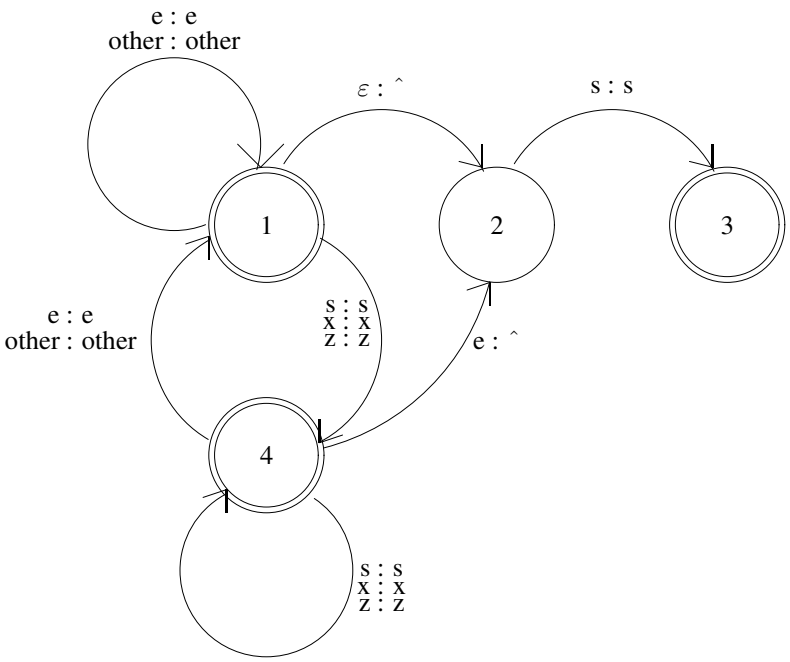
2.7 Finite state transducers

FSAs can be used to recognise particular patterns, but don't, by themselves, allow for any analysis of word forms. Hence for morphology, we use finite state transducers (FSTs) which allow the surface structure to be mapped into the list of morphemes. FSTs are useful for both analysis and generation, since the mapping is bidirectional. This approach is known as *two-level morphology*.

To illustrate two-level morphology, consider the following FST, which recognises the affix *-s* allowing for environments corresponding to the e-insertion spelling rule shown in §2.4 and repeated below.[14]

[14] Actually, I've simplified this slightly so the FST works correctly but the correspondence to the spelling rule is not exact: J&M give a more complex transducer which is an accurate reflection of the spelling rule. They also use an explicit terminating character while I prefer to rely on the 'use all the input' convention, which results in simpler rules.

$$\varepsilon \rightarrow \text{e} / \left\{ \begin{array}{c} \text{s} \\ \text{x} \\ \text{z} \end{array} \right\} \hat{} \_ s$$



Transducers map between two representations, so each transition corresponds to a pair of characters. As with the spelling rule, we use the special character 'ε' to correspond to the empty character and '^' to correspond to an affix boundary. The abbreviation 'other : other' means that any character not mentioned specifically in the FST maps to itself.[15] As with the FSA example, we assume that the FST only accepts an input if the end of the input corresponds to an accept state (i.e., no 'left-over' characters are allowed).

For instance, with this FST, the surface form *cakes* would start from 1 and go through the transitions/states (c:c) 1, (a:a) 1, (k:k) 1, (e:e) 1, (ε:^) 2, (s:s) 3 (accept, underlying *cake^s*) and also (c:c) 1, (a:a) 1, (k:k) 1, (e:e) 1, (s:s) 4 (accept, underlying *cakes*). 'd o g s' maps to 'd o g ^ s', 'f o x e s' maps to 'f o x ^ s' and to 'f o x e ^ s', and 'b u z z e s' maps to 'b u z z ^ s' and 'b u z z e ^ s'.[16] When the transducer is run in analysis mode, this means the system can detect an affix boundary (and hence look up the stem and the affix in the appropriate lexicons). In generation mode, it can construct the correct string. This FST is non-deterministic.

Similar FSTs can be written for the other spelling rules for English (although to do consonant doubling correctly, information about stress and syllable boundaries is required and there are also differences between British and American spelling conventions which complicate matters). Morphology systems are usually implemented so that there is one FST per spelling rule and these operate in parallel.

One issue with this use of FSTs is that they do not allow for any internal structure of the word form. For instance, we can produce a set of FSTs which will result in *unionised* being mapped into *un^ion^ise^ed*, but as we've seen, the affixes actually have to be applied in the right order and the bracketing isn't modelled by the FSTs.

2.8 Some other uses of finite state techniques in NLP

- Grammars for simple spoken dialogue systems. Finite state techniques are not adequate to model grammars of natural languages: I'll discuss this a little in §4.11. However, for very simple spoken dialogue systems, a finite-

[15] The solution notes for the 2003 FST question are slightly wrong in that they should have y : y as well as other : other on one transition.

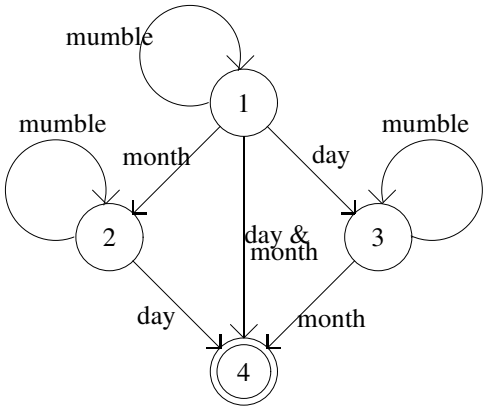
[16] In all cases they also map to themselves: e.g., 'b u z z e s' maps to 'b u z z e s' without the affix marker: this is necessary because words ending in 's' and 'es' are not always inflected forms. e.g., *Moses*

state grammar may be adequate. More complex grammars can be written as context free grammars (CFGs) and compiled into finite state approximations.

- Partial grammars for named entity recognition (briefly discussed in §4.11).
- Dialogue models for spoken dialogue systems (SDS). SDS use dialogue models for a variety of purposes: including controlling the way that the information acquired from the user is instantiated (e.g., the slots that are filled in an underlying database) and limiting the vocabulary to achieve higher recognition rates. FSAs can be used to record possible transitions between states in a simple dialogue. For instance, consider the problem of obtaining a date expressed as a day and a month from a user. There are four possible states, corresponding to the user input recognised so far:

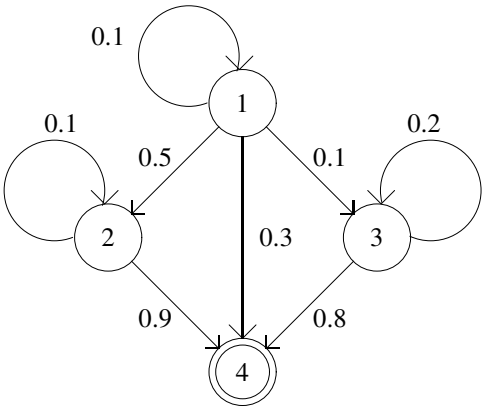
  1. No information. System prompts for month and day.
  2. Month only is known. System prompts for day.
  3. Day only is known. System prompts for month.
  4. Month and day known.

  The FSA is shown below. The loops that stay in a single state correspond to user responses that aren't recognised as containing the required information (*mumble* is the term generally used for an unrecognised input).



2.9 Probabilistic FSAs

In many cases, it is useful to augment the FSA with information about transition probabilities. For instance, in the SDS system described above, it is more likely that a user will specify a month alone than a day alone. A probabilistic FSA for the SDS is shown below. Note that the probabilities on the outgoing arcs from each state must sum to 1.



2.10 Further reading

Chapters 2 and 3 of J&M. Much of Chapter 2 should be familiar from other courses in the CST. Chapter 3 uses more elaborate transducers than I've discussed. See Bender (2013) for a much broader discussion of morphology.

3 Lecture 3: Prediction and part-of-speech tagging

This lecture introduces some simple statistical techniques and illustrates their use in NLP for prediction of words and part-of-speech categories. It starts with a discussion of corpora, then introduces word prediction. Word prediction can be seen as a way of (crudely) modelling some syntactic information (i.e., word order). Similar statistical techniques can also be used to discover parts of speech for uses of words in a corpus. The lecture concludes with some discussion of evaluation.

3.1 Corpora

A *corpus* (corpora is the plural) is simply a body of text that has been collected for some purpose. A *balanced corpus* contains texts which represent different genres (newspapers, fiction, textbooks, parliamentary reports, cooking recipes, scientific papers etc etc): early examples were the Brown corpus (US English: 1960s) and the Lancaster-Oslo-Bergen (LOB) corpus (British English: 1970s) which are each about 1 million words: the more recent British National Corpus (BNC: 1990s) contains approximately 100 million words, including about 10 million words of spoken English. Corpora are important for many types of linguistic research, although mainstream linguists in the past mostly dismissed their use in favour of reliance on intuitive judgements about whether or not an utterance is grammatical (a corpus can only (directly) provide positive evidence about grammaticality). However, many linguists do now use corpora. Corpora are essential for most modern NLP research, though NLP researchers have often used newspaper text (particularly the Wall Street Journal) or text extracted from webpages rather than balanced corpora. Distributed corpora are often annotated in some way: the most important type of annotation for NLP is part-of-speech tagging (POS tagging), which I'll discuss further below. Corpora may also be collected for a specific task. For instance, when implementing an email answering application, it is essential to collect samples of representative emails. For interface applications in particular, collecting a corpus requires a simulation of the actual application: this has often been done by a *Wizard of Oz* experiment, where a human pretends to be a computer.

Corpora are needed in NLP for two reasons. Firstly, we have to evaluate algorithms on real language: corpora are required for this purpose for any style of NLP. Secondly, corpora provide the data source for many machine-learning approaches.

3.2 Prediction

The essential idea of prediction is that, given a sequence of words, we want to determine what's most likely to come next. There are a number of reasons to want to do this: the most important is as a form of *language modelling* for automatic speech recognition (ASR). Speech recognisers cannot accurately determine a word from the sound signal for that word alone, and they cannot reliably tell where each word in an utterance starts and finishes. For instance, *have an ice Dave*, *heaven ice day* and *have a nice day* could easily be confused. [17] For ASR, an initial signal processing phase produces a lattice of hypotheses of the words uttered, which are then ranked and filtered using the probabilities of the possible sequences according to the language model. The language models which are currently most practically effective work on the basis of *n-grams* (a type of *Markov chain*), where the sequence of the prior $n-1$ words is used to derive a probability for the next. Trigram models use the preceding 2 words, bigram models the preceding word and unigram models use no context at all, but simply work on the basis of individual word probabilities. Bigrams are discussed below, though I won't go into details of exactly how they are used in speech recognition.

Word prediction has been used for decades in communication aids: i.e., systems for people who can't speak because of some form of disability. People who use text-to-speech systems to talk because of a non-linguistic disability usually have some form of general motor impairment which also restricts their ability to type at normal rates (stroke, ALS, cerebral palsy etc). Often they use alternative input devices, such as adapted keyboards, puffer switches, mouth sticks or eye trackers. Generally such users can only construct text at a few words a minute, which is too slow for anything like normal communication to be possible (normal speech is around 150 words per minute). As a partial aid, a word prediction system is sometimes helpful: this gives a list of candidate words that changes as the initial letters are entered by the user. The user chooses the desired word from a menu when it appears. The main difficulty with using statistical

[17] In fact, although humans are better at doing this than speech recognisers, we also need context to recognise words, especially words like *the* and *a*. If a recording is made of normal, fluently spoken, speech and the segments corresponding to *the* and *a* are presented to a human subject in isolation, it's generally not possible for them to tell the difference.

prediction models in such applications is in finding enough data: to be useful, the model really has to be trained on an individual speaker's output, but of course very little of this is likely to be available. Training a conversational aid on newspaper text can be worse than using a unigram model from the user's own data. Of course, predictive text is now commonplace on mobile devices, where users are disabled by not having a proper keyboard.

Prediction is important in estimation of entropy, including estimations of the entropy of English. The notion of entropy is important in language modelling because it gives a metric for the difficulty of the prediction problem. For instance, speech recognition is vastly easier in situations where the speaker is only saying two easily distinguishable words (e.g., when a dialogue system prompts by saying *answer 'yes' or 'no'*) than when the vocabulary is unlimited: measurements of entropy can quantify this, but won't be discussed further in this course.

Other applications for prediction include handwriting recognition, spelling correction and text segmentation for languages such as Chinese, which are conventionally written without explicit word boundaries. A form of prediction is used to model the target language in statistical MT systems, and can also be used to rank the output of grammar-based language realizers in symbolic MT or other language generation applications. Some approaches to word sense disambiguation, to be discussed in lecture 5, can also be treated as a form of prediction.

3.3 bigrams

A bigram model assigns a probability to a word based on the previous word alone: i.e., $P(w_n|w_{n-1})$ (the probability of w_n conditional on w_{n-1}) where w_n is the nth word in some string. For application to communication aids, we are simply concerned with predicting the next word: once the user has made their choice, the word can't be changed. However, for speech recognition and similar applications, we require the probability of some string of words $P(w_1^n)$ which is approximated by the product of the bigram probabilities:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

This assumes independence of the individual probabilities, which is clearly wrong, but the approximation nevertheless works reasonably well. 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, the decision between *a* and *the* in a lattice may be influenced by the following noun, which is a much better predictor than the words before the determiner.

We acquire these probabilities from a corpus. The bigram probabilities are given as:

$$\frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

i.e., the count of a particular bigram, normalised by dividing by the total number of bigrams starting with the same word (which is equivalent to the total number of occurrences of that word, except in the case of the last token, a complication which can be ignored for a reasonable size of corpus). For example, suppose we have the following tiny corpus of utterances:

⟨s⟩ good morning ⟨/s⟩ ⟨s⟩ good afternoon ⟨/s⟩ ⟨s⟩ good afternoon ⟨/s⟩ ⟨s⟩ it is very good ⟨/s⟩ ⟨s⟩ it is good ⟨/s⟩

I have used the symbol ⟨s⟩ to indicate the beginning of the utterance and ⟨/s⟩ to indicate the end.

sequence	count	bigram probability
<s>	5	
<s> good	3	.6
<s> it	2	.4
good	5	
good morning	1	.2
good afternoon	2	.4
good </s>	2	.4
morning	1	

```
morning </s>     1       1
afternoon        2
afternoon </s>   2       1
it               2
it is            2       1
is               2
is very          1       .5
is good          1       .5
very             1
very good        1       1
</s>             5
</s><s>          4       1
```

This yields a probability of 0.24 for the string '⟨s⟩ good ⟨/s⟩' and also for '⟨s⟩ good afternoon ⟨/s⟩'.

Notice that we can regard bigrams as comprising a simple deterministic weighted FSA. The *Viterbi algorithm*, a dynamic programming technique for efficiently applying n-grams in speech recognition and other applications to find the highest probability sequence (or sequences), is usually described in terms of an FSA.

With the approach described, the probability of '⟨s⟩ very good ⟨/s⟩' based on this toy corpus is 0: the conditional probability of 'very' given '⟨s⟩' is 0 since there are no examples of this in the training data. Even for realistically sized corpora, zero probabilities are problematic: we will never have enough data to ensure that we will see all possible events and so we don't want to rule out unseen events entirely. To allow for *sparse data* we have to use *smoothing*, which simply means that we make some assumption about the 'real' probability of unseen or very infrequently seen events and distribute that probability appropriately. A common approach is simply to add one to all counts: this is *add-one smoothing* which is not sound theoretically, but is simple to implement. A better approach in the case of bigrams is to *backoff* to the unigram probabilities: i.e., to distribute the unseen probability mass so that it is proportional to the unigram probabilities. This sort of estimation is extremely important to get good results from n-gram techniques, but is not discussed further here.

3.4 Part of speech tagging

Sometimes we are interested in a form of prediction that involves assigning classes to items in a sequence rather than predicting the next item. One important application is to 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. 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 into cans use) and modal verb (the 'possibility' use). 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 are used in the standard tagsets for POS tagging. For the examples in this lecture I'll use the CLAWS 5 (C5) tagset which is given in full in Figure 5.9 in J&M. The tags needed for the example above are:

```
NN1 singular noun
NN2 plural noun
PNP personal pronoun
VM0 modal auxiliary verb
VVB base form of verb (except infinitive)
VVI infinitive form of verb (i.e. occurs with 'to' and in similar contexts)
```

The lexicon which associates the words with the tags includes:

```
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, i.e., for this example:

They_PNP can_VM0 fish_VVI ._PUN

Note the tag for the full stop: punctuation is treated as unambiguous.

The other syntactically possible reading mentioned above corresponds to the following using C5 tags:

They_PNP can_VVB fish_NN2 ._PUN

However, POS taggers (unlike full parsers) don't attempt to produce globally coherent analyses. Thus a POS tagger might also return:

They_PNP can_VM0 fish_NN2 ._PUN

despite the fact that this doesn't correspond to a possible reading of the sentence.

POS tagging can be regarded as a form of very basic, coarse-grained, sense disambiguation. It is useful as a way of annotating a corpus because it makes it easier to extract some types of information (for linguistic research or NLP experiments). It also acts as a basis for more complex forms of annotation. Named entity recognisers (discussed in lecture 4) are generally run on POS-tagged data. As mentioned in lecture 1, POS taggers are sometimes run as preprocessors to full parsing, since this can cut down the search space to be considered by the parser. They can also be used as part of a method for dealing with words which are not in the parser's lexicon (unknown words).

3.5 Stochastic POS tagging using Hidden Markov Models

One form of POS tagging uses a technique known as *Hidden Markov Modelling* (HMM). It involves an n-gram technique, but in this case the n-grams are sequences of POS tags rather than of words. The most common approaches depend on a small amount of manually tagged *training data* from which POS n-grams can be extracted.[18] I'll illustrate this with respect to another trivial corpus:

They used to can fish in those towns. But now few people fish in these areas.

This might be tagged as follows:

```
They_PNP used_VVD to_TO0 can_VVI fish_NN2 in_PRP those_DT0 towns_NN2 ._PUN
But_CJC now_AV0 few_DT0 people_NN2 fish_VVB in_PRP these_DT0 areas_NN2 ._PUN
```

This yields the following counts and probabilities:

sequence	count	bigram probability
AV0	1	
AV0 DT0	1	1
CJC	1	
CJC AV0	1	1
DT0	3	
DT0 NN2	3	1
NN2	4	

[18] It is possible to build POS taggers that work without a hand-tagged corpus, but they don't perform as well as a system trained on even a very small manually-tagged corpus and they still require a lexicon associating possible tags with words. Completely unsupervised approaches also exist, where no lexicon is used, but the categories induced do not correspond to any standard tagset.

```
NN2 PRP          1                0.25
NN2 PUN          2                0.5
NN2 VVB          1                0.25
PNP              1
PNP VVD          1                1
PRP              2
PRP DT0          2                1
PUN              1
PUN CJC          1                1
TO0              1
TO0 VVI          1                1
VVB              1
VVB PRP          1                1
VVD              1
VVD TO0          1                1
VVI              1
VVI NN2          1                1
```

I've used the correct PUN CJC probability, allowing for the final PUN. We also obtain a lexicon from the tagged data:

```
word   tag  count        word   tag  count

they   PNP    1          towns  NN2    1
used   VVD    1          .      PUN    1
to     TO0    1          but    CJC    1
can    VVI    1          now    AV0    1
fish   NN2    1          few    DT0    1
       VVB    1          people NN2    1
in     PRP    2          these  DT0    1
those  DT0    1          areas  NN2    1
```

The idea of stochastic POS tagging is that the tag can be assigned based on consideration of the lexical probability (how likely it is that the word has that tag), plus the sequence of prior tags (for a bigram model, the immediately prior tag). This is more complicated than prediction because we have to take into account both words and tags.

We wish to produce a sequence of tags which have the maximum probability given a sequence of words. I will follow J&M's notation: the hat, $\hat{}$, means "estimate of"; $\hat{t}_1^n$ means "estimate of the sequence of n tags"; $\underset{x}{\text{argmax}} f(x)$ means "the x such that $f(x)$ is maximized". Hence:

$$\hat{t}_1^n = \underset{t_1^n}{\text{argmax}}\, P(t_1^n|w_1^n)$$

We can't estimate this directly (mini-exercise: explain why not). By Bayes theorem:

$$P(t_1^n|w_1^n) = \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$$

Since we're looking at assigning tags to a particular sequence of words, $P(w_1^n)$ is constant, so for a relative measure of probability we can use:

$$\hat{t}_1^n = \underset{t_1^n}{\text{argmax}}\, P(w_1^n|t_1^n)P(t_1^n)$$

We now have to estimate $P(t_1^n)$ and $P(w_1^n|t_1^n)$. If we make the bigram assumption, then the probability of a tag depends on the previous tag, hence the tag sequence is estimated as a product of the probabilities:

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

We will also assume that the probability of the word is independent of the words and tags around it and depends only on its own tag:

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

These values can be estimated from the corpus frequencies. So our final equation for the HMM POS tagger using bigrams is:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Note that we end up multiplying $P(t_i|t_{i-1})$ with $P(w_i|t_i)$ (the probability of the word given the tag) rather than $P(t_i|w_i)$ (the probability of the tag given the word). For instance, if we're trying to choose between the tags NN2 and VVB for *fish* in the sentence *they fish*, we calculate P(NN2|PNP), P(fish|NN2), P(VVB|PNP) and P(fish|VVB) (assuming PNP is the only possible tag for *they*).

As the equation above indicates, in order to POS tag a sentence, we maximise the overall tag sequence probability (again, this can be implemented efficiently using the Viterbi algorithm). So a tag which has high probability considering its individual bigram estimate will not be chosen if it does not form part of the highest probability path. For example, consider:
they_PNP can_VVB fish_NN2
they_PNP can_VM0 fish_VVI
The product of P(VVI|VM0) and P(fish|VVI) may be lower than that of P(NN2|VVB) and P(fish|NN2) but the overall probability depends also on P(can|VVB) versus P(can|VM0) and the latter (modal) use has much higher frequency in a balanced corpus.

In fact, POS taggers generally use trigrams rather than bigrams — the relevant equations are given in J&M, 5.5.4. As with word prediction, backoff (to bigrams) and smoothing are crucial for reasonable performance because of sparse data.

When a POS tagger sees a word which was not in its training data, we need some way of assigning possible tags to the word. One approach is simply to use all possible *open class* tags, with probabilities based on the unigram probabilities of those tags.[19] A better approach is to use a morphological analyser (without a lexicon) to restrict the candidates: e.g., words ending in *-ed* are likely to be VVD (simple past) or VVN (past participle), but can't be VVG (-ing form).

3.6 Evaluation of POS tagging

POS tagging algorithms are evaluated in terms of percentage of correct tags. The standard assumption is that every word should be tagged with exactly one tag, which is scored as correct or incorrect: there are no marks for near misses. 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. Success rates of about 97% are possible for English POS tagging (performance seems to have reached a plateau, probably partly because of errors in the manually-tagged corpora) but the baseline of choosing the most common tag based on the training set often gives about 90% accuracy.[20] Some POS taggers return multiple tags in cases where more than one tag has a similar probability.

Increasing the size of the tagset does not necessarily result in decreased performance: some additional tags could be assigned more-or-less unambiguously and more fine-grained tags can increase performance. For instance, suppose we wanted to distinguish between present tense verbs according to whether they were 1st, 2nd or 3rd person. With the C5 tagset, and the stochastic tagger described, this would be impossible to do with high accuracy, because 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, the n-gram approach would allow some discrimination. In general, predicting on the basis of classes means we have less of a sparse data problem than when predicting on the basis of words, but we have less discriminating power. There is also something of a trade-off between the utility of a set of tags and their effectiveness in POS tagging. For instance, C5 assigns separate tags for the different forms of *be*, which is redundant for many

[19] Open class words are ones for which we can never give a complete list for a living language, since words are always being invented: i.e., verbs, nouns, adjectives and adverbs. Other words (prepositions, determiners, conjunctions and so on) are considered closed class.

[20] Accuracy differs between languages mainly because of differences in morphology: Japanese POS-tagging is almost deterministic but accuracy for Turkish is only around 90%.

purposes, but helps make distinctions between other tags in tagging models such as the HMM described here where the context is given by a tag sequence alone (i.e., rather than considering words prior to the current one).

POS tagging exemplifies some general issues in NLP evaluation:

Training data and test data The assumption in NLP is always that a system should work on novel data, therefore test data must be kept unseen.

For machine learning approaches, such as stochastic POS tagging, the usual technique is to spilt a data set into 90% training and 10% test data. Care needs to be taken that the test data is representative.

For an approach that relies on significant hand-coding, the test data should be literally unseen by the researchers. Development cycles involve looking at some initial data, developing the algorithm, testing on unseen data, revising the algorithm and testing on a new batch of data. The seen data is kept for regression testing.

Baselines Evaluation should be reported with respect to a baseline, which is normally what could be achieved with a very basic approach, given the same training data. For instance, a baseline for POS tagging with training data is to choose the most common tag for a particular word on the basis of the training data (and to simply choose the most frequent tag of all for unseen words).

Ceiling It is often useful to try and compute some sort of ceiling for the performance of an application. This is usually taken to be human performance on that task, where the ceiling is the percentage agreement found between two annotators (*interannotator agreement*). For POS tagging, this has been reported as 96% (which makes existing POS taggers look impressive since some perform at higher accuracy). However this raises lots of questions: relatively untrained human annotators working independently often have quite low agreement, but trained annotators discussing results can achieve much higher performance (approaching 100% for POS tagging). Human performance varies considerably between individuals. Fatigue can cause errors, even with very experienced annotators. In any case, human performance may not be a realistic ceiling on relatively unnatural tasks, such as POS tagging.

Error analysis The error rate on a particular problem will be distributed very unevenly. For instance, a POS tagger will never confuse the tag PUN with the tag VVN (past participle), but might confuse VVN with AJ0 (adjective) because there's 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 specifically interested in finding examples of unusual word uses. Similarly, in text categorisation, some errors are more important than others: e.g. treating an incoming order for an expensive product as junk email is a much worse error than the converse.

Reproducibility If at all possible, evaluation should be done on a generally available corpus so that other researchers can replicate the experiments.

3.7 Further reading

N-grams are described in Chapter 4 of J&M, POS tagging in Chapter 5. The description in the second edition is considerably clearer than that in the first edition.

4 Lecture 4: Context-free grammars and parsing.

In this lecture, I'll discuss syntax in a way which is much closer to the standard notions in formal linguistics than POS-tagging is. To start with, I'll briefly motivate the idea of a generative grammar in linguistics, review the notion of a context-free grammar and then show a context-free grammar for a tiny fragment of English. We'll then see how context free grammars can be used to implement parsers, and discuss chart parsing, which allows efficient processing of strings containing a high degree of ambiguity. Finally we'll briefly touch on probabilistic context-free approaches.

4.1 Generative grammar

Since Chomsky's work in the 1950s, much work in formal linguistics has been concerned with the notion of a *generative grammar* — i.e., a formally specified grammar that can generate all and only the acceptable sentences of a natural language. It's important to realise that nobody has actually written a complete grammar of this type for any living natural language:[21] what most linguists are really interested in is the principles that underlie such grammars, especially to the extent that they apply to all natural languages. NLP researchers, on the other hand, are at least sometimes interested in actually building and using large-scale detailed grammars.

The formalisms which are of interest to us for modelling syntax assign internal structure to the strings of a language, which may be represented by bracketing. We already saw some evidence of this in derivational morphology (the *unionised* example), but here we are concerned with the structure of phrases. For instance, the sentence:

the big dog slept

can be bracketed

((the (big dog)) slept)

The phrase, *big dog*, is an example of a *constituent* (i.e. something that is enclosed in a pair of brackets): *the big dog* is also a constituent, but *the big* and *dog slept* are not. Constituent structure is generally justified by arguments about substitution which I won't go into here: J&M discuss this briefly, as does Bender (2013) but see an introductory syntax book for a full discussion. In this course, I will simply give bracketed structures and hope that the constituents make sense intuitively, rather than trying to justify them.

Two grammars are said to be *weakly-equivalent* if they generate the same strings. Two grammars are *strongly-equivalent* if they assign the same structure to all strings they generate.

In most, but not all, approaches, the internal structures are given labels. For instance, *the big dog* is a *noun phrase* (abbreviated NP), *slept*, *slept in the park* and *licked Sandy* are *verb phrase*s (VPs). The labels such as NP and VP correspond to non-terminal symbols in a grammar. In this lecture, I'll discuss the use of simple context-free grammars for language description.

4.2 Context free grammars

The idea of a context-free grammar (CFG) should be familiar from formal language theory. A CFG has four components, described here as they apply to grammars of natural languages:

1. a set of non-terminal symbols (e.g., S, VP), conventionally written in uppercase;
2. a set of terminal symbols (i.e., the words), conventionally written in lowercase;
3. a set of rules (productions), where the left hand side (the mother) is a single non-terminal and the right hand side is a sequence of one or more non-terminal or terminal symbols (the daughters);
4. a start symbol, conventionally S, which is a member of the set of non-terminal symbols.

[21] There are grammars which cover all recorded utterances of extinct languages.

The formal description of a CFG generally allows productions with an empty right hand side (e.g., Det $\rightarrow \varepsilon$). It is convenient to exclude these however, since they complicate parsing algorithms, and a weakly-equivalent grammar can always be constructed that disallows such *empty productions*.

A grammar in which all nonterminal daughters are the leftmost daughter in a rule (i.e., where all rules are of the form $X \rightarrow Ya*$), is said to be *left-associative*. A grammar where all the nonterminals are rightmost is *right-associative*. Such grammars are weakly-equivalent to regular grammars (i.e., grammars that can be implemented by FSAs), but natural languages seem to require more expressive power than this (see §4.11).

4.3 A simple CFG for a fragment of English

The following tiny fragment is intended to illustrate some of the properties of CFGs so that we can discuss parsing. It has some serious deficiencies as a representation of even this fragment, which I'll ignore for now, though we'll see some of them at the end of the lecture. Notice that for this fragment there is no distinction between main verb *can* and the modal verb *can*.

```
S -> NP VP
VP -> VP PP
VP -> V
VP -> V NP
VP -> V VP
NP -> NP PP
PP -> P NP
;;; lexicon
V -> can
V -> fish
NP -> fish
NP -> rivers
NP -> pools
NP -> December
NP -> Scotland
NP -> it
NP -> they
P -> in
```

The rules with terminal symbols on the right hand side correspond to the lexicon. Here and below, comments are preceded by `;;;`

Here are some strings which this grammar generates, along with their bracketings:

they fish
(S (NP they) (VP (V fish)))

they can fish
(S (NP they) (VP (V can) (VP (V fish))))
;;; the modal verb 'are able to' reading
(S (NP they) (VP (V can) (NP fish)))
;;; the less plausible, put fish in cans, reading

they fish in rivers
(S (NP they) (VP (VP (V fish)) (PP (P in) (NP rivers))))

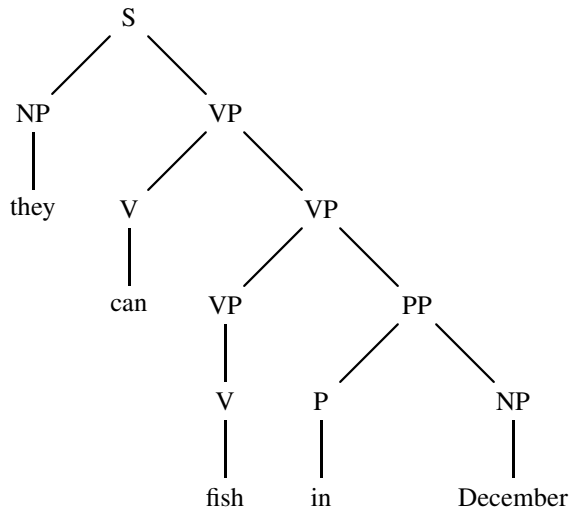
they fish in rivers in December
(S (NP they) (VP (VP (V fish)) (PP (P in) (NP (NP rivers) (PP (P in) (NP December))))))
;;; i.e. the implausible reading where the rivers are in December
;;; (cf rivers in Scotland)
(S (NP they) (VP (VP (VP (V fish)) (PP (P in) (NP rivers))) (PP (P in) (NP December))))
;;; i.e. the fishing is done in December

One important thing to notice about these examples is that there's lots of potential for ambiguity. In the *they can fish* example, this is due to *lexical ambiguity* (it arises from the dual lexical entries for *fish*), but the last example demonstrates purely *structural ambiguity*. In this case, the ambiguity arises from the two possible *attachments* of the prepositional phrase (PP) *in December*: it can attach to the NP (*rivers*) or to the VP. These attachments correspond to different semantics, as indicated by the glosses. PP attachment ambiguities are a major headache in parsing, since sequences of four or more PPs are common in real texts and the number of readings increases as the Catalan series, which is exponential. Other phenomena have similar properties: for instance, compound nouns (e.g. *long-stay car park shuttle bus*). Humans disambiguate such attachments as they hear a sentence, but they're relying on the meaning in context to do this, in a way we cannot currently emulate, except when the sentences are restricted to a very limited domain.

Notice that *fish* could have been entered in the lexicon directly as a VP, but that this would cause problems if we were doing inflectional morphology, because we want to say that suffixes like *-ed* apply to Vs. Making *rivers* etc NPs rather than nouns is a simplification I've adopted here just to keep the example grammar smaller.

4.4 Parse trees

Parse trees are equivalent to bracketed structures, but are easier to read for complex cases. A parse tree and bracketed structure for one reading of *they can fish in December* is shown below. The correspondence should be obvious.




```
(S (NP they)
   (VP (V can)
       (VP (VP (V fish))
           (PP (P in)
               (NP December)))))
```

4.5 Chart parsing

In order to parse with reasonable efficiency, we need to keep a record of the rules that we have applied so that we don't have to backtrack and redo work that we've done before. This works for parsing with CFGs because the rules are independent of their context: a VP can always expand as a V and an NP regardless of whether or not it was preceded by an NP or a V, for instance. (In some cases we may be able to apply techniques that look at the context to cut down the search space, because we can tell that a particular rule application is never going to be part of a sentence, but this is strictly a filter: we're never going to get incorrect results by reusing partial structures.) This record keeping strategy is an application of dynamic programming/memoization which is used in processing formal languages too. In NLP the data structure used for recording partial results is generally known as a *chart* and algorithms for parsing using such

structures are referred to as *chart parsers*.[22] Chart parsing strategies are designed to be *complete*: that is, the chart parser will find all valid analyses according to a grammar (though there are some minor caveats e.g., concerning rules which can apply to their own output).

A chart is a collection of *edges*, usually implemented as a vector of edges, indexed by edge identifiers. In the simplest version of chart parsing, each edge records a rule application and has the following structure:

[*id*,*left_vertex*, *right_vertex*,*mother_category*, *daughters*]

A vertex is an integer representing a point in the input string, as illustrated below:

```
. they . can . fish .
0      1     2      3
```

mother_category refers to the rule that has been applied to create the edge. *daughters* is a list of the edges that acted as the daughters for this particular rule application: it is there purely for record keeping so that the output of parsing can be a labelled bracketing.

For instance, the following edges would be among those found on the chart after a complete parse of *they can fish* according to the grammar given above (id numbering is arbitrary):

```
id  left   right  mother   daughters
3    1       2      V        (can)
4    2       3      NP       (fish)
5    2       3      V        (fish)
6    2       3      VP       (5)
7    1       3      VP       (3 6)
8    1       3      VP       (3 4)
```

The daughters for the terminal rule applications are simply the input word strings.

Note that local ambiguities correspond to situations where a particular span has more than one associated edge. We'll see below that we can *pack* structures so that we never have two edges with the same category and the same span, but we'll ignore this for the moment (see §4.8). Also, in this chart we're only recording complete rule applications: this is *passive* chart parsing. The more efficient *active* chart is discussed below, in §4.9.

4.6 A bottom-up passive chart parser

The following pseudo-code sketch is for a very simple chart parser. Informally, it proceeds by adding the next word (in left to right order), and adding each lexical category possible for that word, doing everything it can immediately after each lexical category is added. The main function is **Add new edge** which is called for each word in the input going left to right. **Add new edge** recursively scans backwards looking for other daughters.

Parse:
  Initialise the chart (i.e., clear previous results)
  For each word *word* in the input sentence, let *from* be the left vertex, *to* be the right vertex and *daughters* be (*word*)
    For each category *category* that is lexically associated with *word*
      Add new edge *from*, *to*, *category*, *daughters*
  Output results for all spanning edges
  (i.e., ones that cover the entire input and which have a mother corresponding to the root category)

Add new edge *from*, *to*, *category*, *daughters*:
  Put edge in chart: [*id*,*from*,*to*, *category*,*daughters*]
  For each *rule* in the grammar of form *lhs* –> $cat_1 \ldots cat_{n-1}$,*category*
    Find set of lists of contiguous edges [id_1,$from_1$,to_1, cat_1,$daughters_1$] … [id_{n-1},$from_{n-1}$,*from*, cat_{n-1},$daughters_{n-1}$]

[22] Natural languages have vastly higher degrees of ambiguity than programming languages: chart parsing is well-suited to this.

(such that $to_1 = from_2$ etc)
(i.e., find all edges that match a rule)
For each list of edges, **Add new edge** $from_1$, *to*, *lhs*, ($id_1 \ldots id$)
(i.e., apply the rule to the edges)

Notice that this means that the grammar rules are indexed by their rightmost category, and that the edges in the chart must be indexed by their *to* vertex (because we scan backward from the rightmost category). Consider:

```
. they . can . fish .
0      1     2      3
```

The following diagram shows the chart edges as they are constructed in order (when there is a choice, taking rules in a priority order according to the order they appear in the grammar):

```
id  left   right  mother   daughters
1    0       1      NP       (they)
2    1       2      V        (can)
3    1       2      VP       (2)
4    0       2      S        (1 3)
5    2       3      V        (fish)
6    2       3      VP       (5)
7    1       3      VP       (2 6)
8    0       3      S        (1 7)
9    2       3      NP       (fish)
10   1       3      VP       (2 9)
11   0       3      S        (1 10)
```

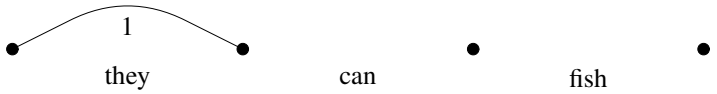
The spanning edges are 11 and 8: the output routine to give bracketed parses simply outputs a left bracket, outputs the category, recurses through each of the daughters and then outputs a right bracket. So, for instance, the output from edge 11 is:

```
(S (NP they) (VP (V can) (NP fish)))
```

This chart parsing algorithm is *complete*: it returns all possible analyses (except in the case where it does not terminate because there is a unary rule that applies to its own output).

4.7 A detailed trace of the simple chart parser

Parse
word = they
categories = NP
Add new edge 0, 1, NP, (they)

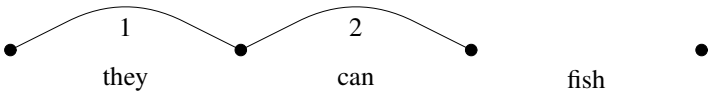


Matching grammar rules are:

```
VP -> V NP
PP -> P NP
```

No matching edges corresponding to V or P

word = can
categories = V
Add new edge 1, 2, V, (can)

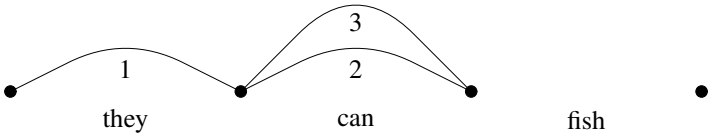


Matching grammar rules are:

```
VP -> V
```

set of edge lists $= \{(2)\}$

Add new edge 1, 2, VP, (2)

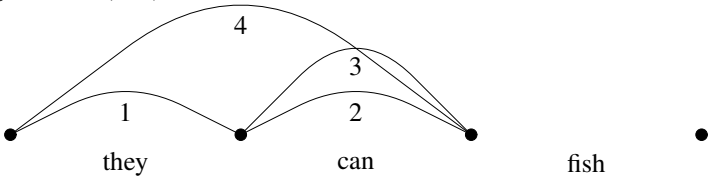


Matching grammar rules are:

```
S -> NP VP
VP -> V VP
```

set of edge lists corresponding to NP VP $= \{(1, 3)\}$

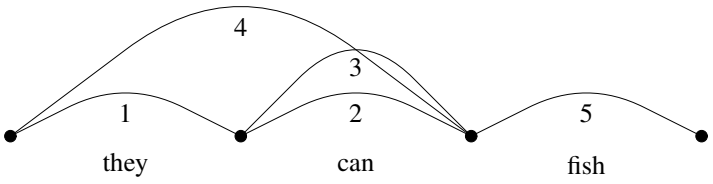
Add new edge 0, 2, S, (1, 3)



No matching grammar rules for S

No edges matching V VP

word = fish
categories = V, NP
Add new edge 2, 3, V, (fish)

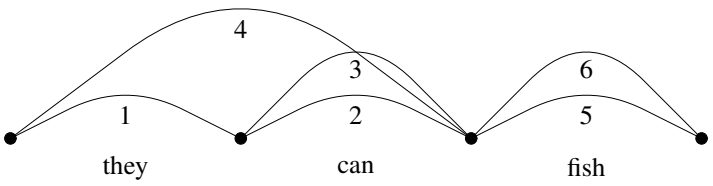


Matching grammar rules are:

```
VP -> V
```

set of edge lists $= \{(5)\}$

Add new edge 2, 3, VP, (5)

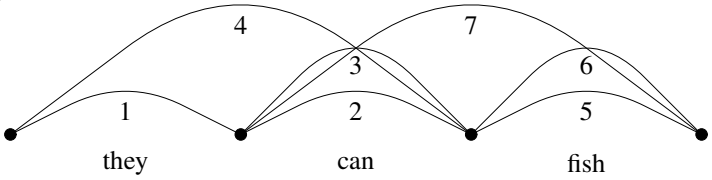


Matching grammar rules are:

```
S -> NP VP
VP -> V VP
```

No edges match NP
set of edge lists for V VP $= \{(2, 6)\}$

Add new edge 1, 3, VP, (2, 6)

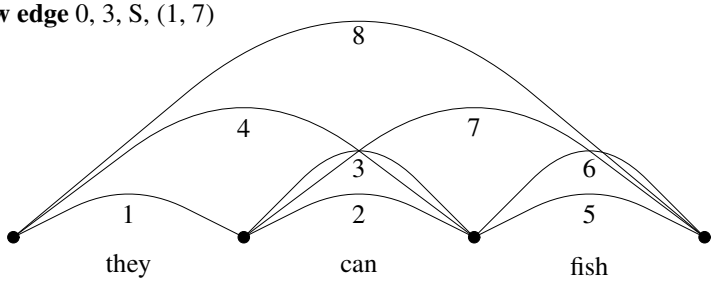


Matching grammar rules are:

```
S -> NP VP
VP -> V VP
```

set of edge lists for NP VP $= \{(1, 7)\}$

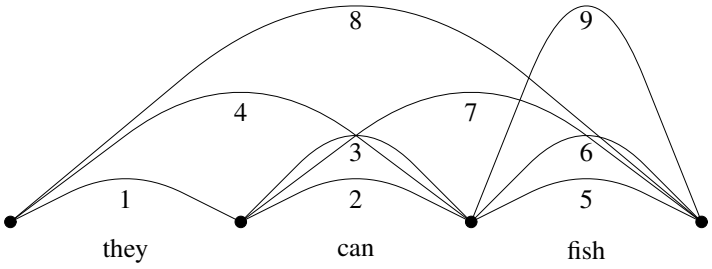
Add new edge 0, 3, S, (1, 7)



No matching grammar rules for S

No edges matching V

Add new edge 2, 3, NP, (fish)

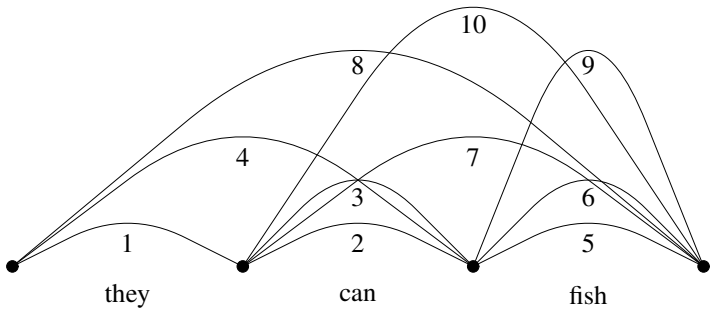


Matching grammar rules are:

```
VP -> V NP
PP -> P NP
```

set of edge lists corresponding to V NP $= \{(2, 9)\}$

Add new edge 1, 3, VP, (2, 9)

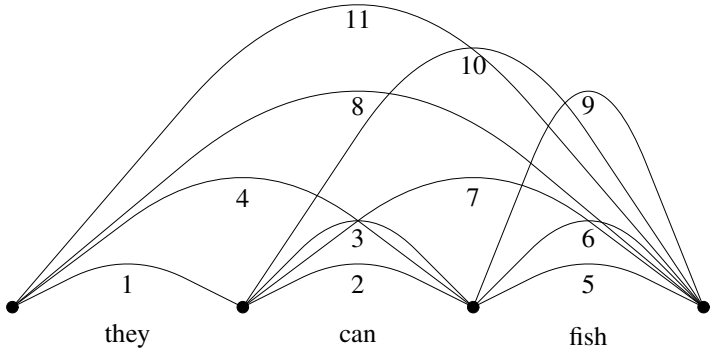


Matching grammar rules are:

```
S -> NP VP
VP -> V VP
```

set of edge lists corresponding to NP VP $= \{(1, 10)\}$

Add new edge 0, 3, S, (1, 10)



No matching grammar rules for S

No edges corresponding to V VP

No edges corresponding to P NP

No further words in input

Spanning edges are 8 and 11: Output results for 8

```
(S (NP they) (VP (V can) (VP (V fish))))
```

Output results for 11

```
(S (NP they) (VP (V can) (NP fish)))
```

4.8 Packing

The algorithm given above is exponential in the case where there are an exponential number of parses. The body of the algorithm can be modified so that it runs in cubic time, though producing the output is still exponential. The modification is simply to change the daughters value on an edge to be a set of lists of daughters and to make an equality check before adding an edge so we don't add one that's equivalent to an existing one. That is, if we are about to add an edge:

[*id*,*left_vertex*, *right_vertex*,*mother_category*, *daughters*]

and there is an existing edge:

[*id-old*,*left_vertex*, *right_vertex*,*mother_category*, *daughters-old*]

we simply modify the old edge to record the new daughters:

[*id-old*,*left_vertex*, *right_vertex*,*mother_category*, *daughters-old* ⊔ *daughters*]

There is no need to recurse with this edge, because we couldn't get any new results: once we've found we can pack an edge, we always stop that part of the search. Thus packing saves computation and in fact leads to cubic time operation, though I won't go through the proof of this.

For the example above, everything proceeds as before up to edge 9:

```
id  left   right  mother   daughters
1    0       1      NP       {(they)}
2    1       2      V        {(can)}
3    1       2      VP       {(2)}
4    0       2      S        {(1 3)}
5    2       3      V        {(fish)}
6    2       3      VP       {(5)}
7    1       3      VP       {(2 6)}
8    0       3      S        {(1 7)}
9    2       3      NP       {(fish)}
```

However, rather than add edge 10, which would be:

```
10   1       3      VP       (2 9)
```

we match this with edge 7, and simply add the new daughters to that.

```
7    1       3      VP       {(2 6), (2 9)}
```

The algorithm then terminates. We only have one spanning edge (edge 8) but the display routine is more complex because we have to consider the alternative sets of daughters for edge 7. (You should go through this to convince yourself that the same results are obtained as before.) Although in this case, the amount of processing saved is small, the effects are much more important with longer sentences (consider *he believes they can fish*, for instance).

4.9 Active chart parsing

A more minor efficiency improvement is obtained by storing the results of partial rule applications. This is *active* chart parsing, so called because the partial edges are considered to be active: i.e. they 'want' more input to make them complete. An active edge records the input it expects as well as the daughters it has already seen. Active edges are stored on the chart as well as passive edges. For instance, with an active chart parser, we might have the following edges when parsing a sentence starting *they fish*:

```
id  left   right  mother    expected      daughters
1    0       1      NP                     (they)
2    0       1      S         VP           (1 ?)
3    0       1      NP        PP           (1,?)
4    1       2      V                      (fish)
5    1       2      VP                     (4)
6    0       2      S                      (2,5)
7    1       2      VP        NP           (4,?)
8    1       2      VP        VP           (4,?)
9    1       2      VP        PP           (5,?)
```

Edge 1 is complete (a passive edge). Edge 2 is active: the daughter marked as ? will be instantiated by the edge corresponding to the VP when it is found (e.g., edge 5 instantiates the active part of edge 2 to give edge 6).

Each word gives rise to a passive edge. Each passive edge of category C gives rise to active edges corresponding to rules with leftmost daughter C (although there are various possible pruning strategies than can be used to cut down on spurious active edges). Every time a passive edge is added, the active edges are searched to see if the new passive edge can complete an active edge.

I will not give full details of the active chart parser here: there are several possible variants. The main thing to note is that active edges may be used to create more than one passive edge. For instance, if we have the string *they fish in Scotland*, edge 2 will be completed by *fish* and also by *fish in Scotland*. Whether this leads to a practical improvement in efficiency depends on whether the saving in time that results because the NP is only combined with the S rule once outweighs the overhead of storing the edge. Active edges may be packed.

4.10 Ordering the search space

In the pseudo-code above, the order of addition of edges to the chart was determined by the recursion. In general, chart parsers make use of an *agenda* of edges, so that the next edges to be operated on are the ones that are first on the agenda. Different parsing algorithms can be implemented by making this agenda a stack or a queue, for instance.

So far, we've considered *bottom up* parsing: an alternative is *top down* parsing, where the initial edges are given by the rules whose mother corresponds to the start symbol.

Some efficiency improvements can be obtained by ordering the search space appropriately, though which version is most efficient depends on properties of the individual grammar. However, the most important reason to use an explicit agenda is when we are returning parses in some sort of priority order, corresponding to weights on different grammar rules or lexical entries.

Weights can be manually assigned to rules and lexical entries in a manually constructed grammar. However, since the beginning of the 1990s, a lot of work has been done on automatically acquiring probabilities from a corpus annotated with syntactic trees (a *treebank*), either as part of a general process of automatic grammar acquisition, or as automatically acquired additions to a manually constructed grammar. Probabilistic CFGs (PCFGs) can be defined quite straightforwardly, if the assumption is made that the probabilities of rules and lexical entries are independent of one another (of course this assumption is not correct, but the orderings given seem to work quite well in practice). The importance of this is that we rarely want to return all parses in a real application, but instead we want to return those which are top-ranked: i.e., the most likely parses. This is especially true when we consider that realistic grammars can easily return many tens of thousands of parses for sentences of quite moderate length (20 words or so). If edges are prioritised by probability, very low priority edges can be completely excluded from consideration if there is a cut-off such that we can be reasonably certain that no edges with a lower priority than the cut-off will contribute to the highest-ranked parse. Limiting the number of analyses under consideration is known as *beam search* (the analogy is that we're looking within a beam of light, corresponding to the highest probability edges). Beam search is linear rather than exponential or cubic. Just as importantly, a good priority ordering from a parser reduces the amount of work that has to be done to filter the results by whatever system is processing the parser's output.

4.11 Why can't we use FSAs to model the syntax of natural languages?

In this lecture, we started using CFGs. This raises the question of why we need this more expressive (and hence computationally expensive) formalism, rather than modelling syntax with FSAs.[23] The usual answer is that the syntax of natural languages cannot be described by an FSA, even in principle, due to the presence of *centre-embedding*, i.e. structures which map to:

$$A \rightarrow \alpha A \beta$$

and which generate grammars of the form $a^n b^n$. For instance:

the students the police arrested complained

[23] Recall that any FSA can be converted to a *regular grammar*, so grammar rules written in the format we've been using could be parsed using an FSA if they met the restrictions for a regular grammar.

has a centre-embedded structure. However, this is not entirely satisfactory, since humans have difficulty processing more than two levels of embedding:

> ? the students the police the journalists criticised arrested complained

If the recursion is finite (no matter how deep), then the strings of the language could be generated by an FSA. So it's not entirely clear whether an FSA might not suffice, despite centre embedding.

There's a fairly extensive discussion of the theoretical issues in J&M , but there are two essential points for our purposes:

1. Grammars written using finite state techniques alone may be very highly redundant, which makes them difficult to build and slow to run.
2. Without internal structure, we can't build up good semantic representations.

These are the main reasons for the use of more powerful formalisms from an NLP perspective (in the next section, I'll discuss whether simple CFGs are inadequate for similar reasons).

However, FSAs are very useful for partial grammars. In particular, for information extraction, we need to recognise *named entities*: e.g. Professor Smith, IBM, 101 Dalmatians, the White House, the Alps and so on. Although NPs are in general recursive (*the man who likes the dog which bites postmen*), relative clauses are not generally part of named entities. Also the internal structure of the names is unimportant for IE. Hence FSAs can be used, with sequences such as 'title surname', 'DT0 PNP' etc

CFGs can be automatically compiled into approximately equivalent FSAs by putting bounds on the recursion. This is particularly important in speech recognition engines.

4.12 Deficiencies in atomic category CFGs

If we consider the sample grammar in §4.3, several problems are apparent. One is that there is no account of subject-verb agreement, so, for instance, **it fish* is allowed by the grammar as well as *they fish*.[24]

We could, of course, allow for agreement by increasing the number of atomic symbols in the CFG, introducing NP-sg, NP-pl, VP-sg and VP-pl, for instance. But this approach would soon become very tedious:

```
S -> NP-sg VP-sg
S -> NP-pl VP-pl
VP-sg -> V-sg NP-sg
VP-sg -> V-sg NP-pl
VP-pl -> V-pl NP-sg
VP-pl -> V-pl NP-pl
NP-sg -> he
NP-sg -> fish
NP-pl -> fish
```

Note that we have to expand out the symbols even when there's no constraint on agreement, since we have no way of saying that we don't care about the value of number for a category (e.g., past tense verbs).

Another linguistic phenomenon that we are failing to deal with is *subcategorization*. This is the lexical property that tells us how many *argument*s a verb can have (among other things). Subcategorization tends to mirror semantics, although there are many complications. A verb such as *adore*, for instance, relates two entities and is transitive: a sentence such as **Kim adored* is strange, while *Kim adored Sandy* is usual. A verb such as *give* is *ditransitive*: *Kim gave Sandy an apple* (or *Kim gave an apple to Sandy*). Without going into details of exactly how subcategorization is defined, or what an argument is, it should be intuitively obvious that we're not encoding this property with our CFG. The grammar in lecture 4 allows the following, for instance:

[24] In English, the subject of a sentence is generally a noun phrase which comes before the verb, in contrast to the object, which follows the verb. The subject and the verb must (usually) either both have singular morphology or both have plural morphology: i.e., they must *agree*. There was also no account of *case*: this is only reflected in a few places in modern English, but **they can they* is clearly ungrammatical (as opposed to *they can them*, which is grammatical with the transitive verb use of *can*).

they fish fish it
(S (NP they) (VP (V fish) (VP (V fish) (NP it))))

Again this could be dealt with by multiplying out symbols (V-intrans, V-ditrans etc), but the grammar becomes extremely cumbersome.

Finally, consider the phenomenon of *long-distance dependencies*, exemplified, for instance, by:

which problem did you say you don't understand?
who do you think Kim asked Sandy to hit?
which kids did you say were making all that noise?

Traditionally, each of these sentences is said to contain a *gap*, corresponding to the place where the noun phrase would normally appear: the gaps are marked by underscores below:

which problem did you say you don't understand _?
who do you think Kim asked Sandy to hit _?
which kids did you say _ were making all that noise?

Notice that, in the third example, the verb *were* shows plural agreement.

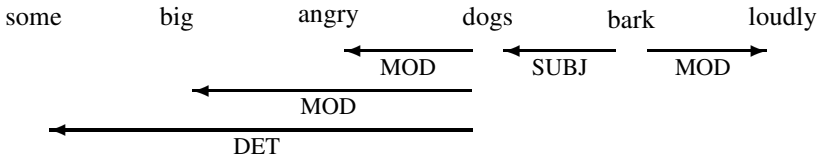
Doing this in standard CFGs is possible, but extremely verbose, potentially leading to trillions of rules. Instead of having simple atomic categories in the CFG, we want to allow for features on the categories, which can have values indicating things like plurality. As the long-distance dependency examples should indicate, the features need to be complex-valued. For instance,

* what kid did you say _ were making all that noise?

is not grammatical. The analysis needs to be able to represent the information that the gap corresponds to a plural noun phrase.

4.13 Dependency structures

An alternative approach to representation which is currently very popular is *dependency structure*. Dependencies relate words in the sentence via a fixed set of relationships. The relationships are usually syntactic (e.g., subject (SUBJ), object (OBJ), modifier (MOD)): in this case, the dependency structure can be seen as an alternative to parse trees, which has the advantage of capturing meaningful relationships more directly. Dependency parsers produce such representations directly, but other systems construct them by converting another representation such as a parse tree. There are a number of different schemes for representing dependency structures: a simplified example is shown below.



Dependencies link a head word in the phrase and a dependent: in the above example the arrows go from the head word, e.g. *dogs*, to the dependent, e.g. its modifiers *big* and *angry*.

4.14 Further reading

This lecture has described material which J&M discuss in chapters 12 and 13, though we also touched on PCFGs (covered in their chapter 14) and issues of language complexity which they discuss in chapter 16. I chose to concentrate on bottom-up chart parsing in this lecture, mainly because I find it easier to describe than the Earley algorithm and the full version of chart parsing given in J&M, but also because it is easier to see how to extend this to PCFGs.

Bender 2013 (listed in the introduction) has a succinct discussion of constituency and other syntactic issues alluded to here.

5 Lecture 5: Lexical semantics

Lexical semantics concerns word meaning. The traditional way of representing lexical meaning in formal semantics is the use of meaning postulates that define concepts by listing their salient attributes. The standard example is:

$$\forall x[\text{bachelor}'(x) \leftrightarrow \text{man}'(x) \wedge \text{unmarried}'(x)]$$

Linguistically and philosophically, any such approach has clear problems. Take the bachelor example: is the current Pope a bachelor? Technically presumably yes, but *bachelor* seems to imply someone who could be married: it's a strange word to apply to the Pope under current assumptions about celibacy. Meaning postulates are also too unconstrained: I could construct a predicate 'bachelor-weds-thurs' to correspond to someone who was unmarried on Wednesday and married on Thursday, but this isn't going to correspond to a word in any natural language. In any case, very few words are as simple to define as *bachelor*: consider how you might start to define *table*, *tomato* or *thought*, for instance.[25]

Rather than try and build complete and precise representations of word meaning therefore, computational linguists work with partial or approximate representations. In this lecture, I will discuss the classical lexical semantic relations and then discuss polysemy and word sense disambiguation. In the following two lectures, we will look at a statistical approach to representing meaning.

5.1 Hyponymy: IS-A

Hyponymy is the classical IS-A relation: e.g. *dog* is a *hyponym* of *animal* and *animal* is the *hypernym* of *dog*. To be more precise, the relevant sense of *dog* is the hyponym of the relevant sense of *animal* (*dog* can also be a verb or used in a metaphorical and derogatory way to refer to a human). As nearly everything said in this lecture is about word senses rather than words, I will avoid explicitly qualifying all statements in this way, but this should be globally understood. In turn, *dog* is the hypernym of *spaniel*, *terrier* and so on. Hyponyms can be arranged into *taxonomies*: classically these are tree-structured: i.e., each term has only one *hypernym*.

Hyponymy relates to ideas which have been formalised in description logics and used in ontologies and the semantic web. However, formalising the linguistic notion of hyponymy is difficult. Despite the fact that hyponymy is by far the most important meaning relationship assumed in NLP, many questions arise which don't have very good answers:

1. What classes of words can be categorised by hyponymy? Some nouns, classically biological taxonomies, but also human artefacts, professions etc work reasonably well. Abstract nouns, such as *truth*, don't really work very well (they are either not in hyponymic relationships at all, or very shallow ones). Some verbs can be treated as being hyponyms of one another — e.g. *murder* is a hyponym of *kill*, but this is not nearly as clear as it is for concrete nouns. Event-denoting nouns are similar to verbs in this respect. Hyponymy is almost useless for adjectives.

2. Do differences in quantisation and individuation matter? For instance, is *chair* a hyponym of *furniture*? is *beer* a hyponym of *drink*? is *coin* a hyponym of *money*?

3. Is multiple inheritance allowed? Intuitively, multiple parents might be possible: e.g. *coin* might be *metal* (or *object*?) and also *money*. Artefacts in general can often be described either in terms of their form or their function.

4. What should the top of the hierarchy look like? The best answer seems to be to say that there is no single top but that there are a series of hierarchies.

Despite this lack of clarity, hyponymy relationships are a good source of inference rules in language-based inference, which we will discuss in lecture 8.

[25] There has been a court case that hinged on the precise meaning of *table* and also one that depended on whether tomatoes were fruits or vegetables.

5.2 Other lexical semantic relations

Meronymy i.e., PART-OF

The standard examples of meronymy apply to physical relationships: e.g., *arm* is part of a *body* (*arm* is a *meronym* of *body*); *steering wheel* is a meronym of *car*. Note the distinction between 'part' and 'piece': if I attack a car with a chainsaw, I get pieces rather than parts!

Synonymy i.e., two words with the same meaning (or nearly the same meaning)

True synonyms are relatively uncommon: most cases of true synonymy are correlated with dialect differences (e.g., *eggplant* / *aubergine*, *boot* / *trunk*). Often synonymy involves register distinctions, slang or jargons: e.g., *policeman*, *cop*, *rozzer* . . . Near-synonyms convey nuances of meaning: *thin*, *slim*, *slender*, *skinny*.

Antonymy i.e., opposite meaning

Antonymy is mostly discussed with respect to adjectives: e.g., *big*/*little*, though it's only relevant for some classes of adjectives.

5.3 WordNet

WordNet is the main resource for lexical semantics for English that is used in NLP — primarily because of its very large coverage and the fact that it's freely available. WordNets are under development for many other languages, though so far none are as extensive as the original.

The primary organisation of WordNet is into *synsets*: synonym sets (near-synonyms). To illustrate this, the following is part of what WordNet returns as an 'overview' of *red*:

```
wn red -over

Overview of adj red

The adj red has 6 senses (first 5 from tagged texts)

1. (43) red, reddish, ruddy, blood-red, carmine,
cerise, cherry, cherry-red, crimson, ruby, ruby-red,
scarlet -- (having any of numerous bright or strong
colors reminiscent of the color of blood or cherries
or tomatoes or rubies)
2. (8) red, reddish -- ((used of hair or fur) of a
reddish brown color; "red deer"; reddish hair")
```

Nouns in WordNet are organised by hyponymy, as illustrated by the fragment below:

```
Sense 6
big cat, cat
       => leopard, Panthera pardus
           => leopardess
           => panther
       => snow leopard, ounce, Panthera uncia
       => jaguar, panther, Panthera onca, Felis onca
       => lion, king of beasts, Panthera leo
           => lioness
           => lionet
       => tiger, Panthera tigris
           => Bengal tiger
           => tigress
       => liger
```

```
=> tiglon, tigon
=> cheetah, chetah, Acinonyx jubatus
=> saber-toothed tiger, sabertooth
    => Smiledon californicus
    => false saber-toothed tiger
```

Taxonomies have also been extracted from machine-readable dictionaries: Microsoft's MindNet is the best known example. There has been considerable work on extracting taxonomic relationships from corpora, including some aimed at automatically extending WordNet.

5.4 Using lexical semantics

The most commonly used lexical relations are hyponymy and (near-)synonymy. Hyponymy relations can be used in many ways, for instance:

- Semantic classification: e.g., for selectional restrictions (e.g., the object of *eat* has to be something edible) and for named entity recognition.
- Shallow inference: 'X murdered Y' implies 'X killed Y' etc.
- Back-off to semantic classes in some statistical approaches (for instance, WordNet classes can be used in document classification).
- Word-sense disambiguation.
- Query expansion for information retrieval: if a search doesn't return enough results, one option is to replace an over-specific term with a hypernym.

Synonymy or near-synonymy is relevant for some of these reasons and also for generation. (However dialect and register haven't been investigated much in NLP, so the possible relevance of different classes of synonym for customising text hasn't really been looked at.)

5.5 Polysemy

Polysemy refers to the state of a word having more than one sense: the standard example is *bank* (river bank) vs *bank* (financial institution).

This is *homonymy* — the two senses are unrelated (not entirely true for *bank*, in fact, but historical relatedness isn't important — it's whether ordinary speakers of the language feel there's a relationship). Homonymy is the most obvious case of polysemy, but is relatively infrequent compared to uses which have different but related meanings, such as *bank* (financial institution) vs *bank* (in a casino).

If polysemy were always homonymy, word senses would be discrete: two senses would be no more likely to share characteristics than would morphologically unrelated words. But most senses are actually related. Regular or systematic polysemy (zero derivation, as mentioned in §2.2) concerns related but distinct usages of words, often with associated syntactic effects. For instance, *strawberry, cherry* (fruit / plant), *rabbit, turkey, halibut* (meat / animal), *tango, waltz* (dance (noun) / dance (verb)).

There are a lot of complicated issues in deciding whether a word is polysemous or simply general/vague. For instance, *teacher* is intuitively general between male and female teachers rather than ambiguous, but giving good criteria as a basis of this distinction is difficult. Dictionaries are not much help, since their decisions as to whether to split a sense or to provide a general definition are very often contingent on external factors such as the size of the dictionary or the intended audience, and even when these factors are relatively constant, lexicographers often make different decisions about whether and how to split up senses.

5.6 Word sense disambiguation

Word sense disambiguation (WSD) is needed for most NL applications that involve semantics (explicitly or implicitly). In limited domains, WSD is not too big a problem, but for large coverage text processing it's a serious bottleneck.

WSD needs depend on the application, but in order to experiment with WSD as a standalone module, there has to be a standard: most commonly WordNet, because for decades it was the only extensive modern resource for English that was freely available. This is controversial, because WordNet has a very fine granularity of senses and the senses often overlap, but there's no clear alternative. Various WSD 'competitions' have been organised (SENSEVAL).

WSD up to the early 1990s was mostly done by hand-constructed rules (still used in some MT systems). Dahlgren investigated WSD in a fairly broad domain in the 1980s. Reasonably broad-coverage WSD generally depends on:

- frequency
- collocations
- selectional restrictions/preferences

What's changed since the 1980s is that various statistical or machine-learning techniques have been used to avoid hand-crafting rules.

- supervised learning. Requires a sense-tagged corpus, which is extremely time-consuming to construct systematically (examples are the Semcor and SENSEVAL corpora, but both are really too small). Often experiments have been done with a small set of words which can be sense-tagged by the experimenter. Supervised learning techniques do not carry over well from one corpus to another.
- minimally-supervised learning (see below)
- Machine readable dictionaries (MRDs) and, more recently, online dictionaries. Disambiguating dictionary definitions according to the internal data in dictionaries is necessary to build taxonomies from MRDs. MRDs have also been used as a source of selectional preference and collocation information for general WSD (quite successfully).
- Wikipedia disambiguation pages.

Until recently, most of the statistical or machine-learning techniques have been evaluated on homonyms: these are relatively easy to disambiguate. So 95% disambiguation in e.g., Yarowsky's experiments sounds good (see below), but doesn't translate into high precision on all words when target is WordNet senses (in SENSEVAL 2 the best system was around 70%).

There have also been some attempts at automatic *sense induction*, where an attempt is made to determine the clusters of usages in texts that correspond to senses. In principle, this is a very good idea, since the whole notion of a word sense is fuzzy: word senses can be argued to be artefacts of dictionary publishing.

5.7 Collocations

Informally, a collocation is a group of two or more words that occur together more often than would be expected by chance (there are other definitions — this is not really a precise notion). Collocations have always been the most useful source of information for WSD, even in Dahlgren's early experiments. For instance:

(2) Striped bass are common.

(3) Bass guitars are common.

striped is a good indication that we're talking about the fish (because it's a particular sort of bass), similarly with *guitar* and music. In both *bass guitar* and *striped bass*, we've arguably got a *multiword expression* (i.e., a conventional phrase that might be listed in a dictionary), but the principle holds for any sort of collocation. The best collocates for

WSD tend to be syntactically related in the sentence to the word to be disambiguated, but many techniques simply use a window of words.

The term collocation is sometimes restricted to the situation where there is a syntactic relationship between the words. J&M (second edition) define collocation as a position-specific relationship (in contrast to *bag-of-words*, where position is ignored) but this is not a standard definition.

5.8 Yarowsky's minimally-supervised learning approach to WSD

Yarowsky (1995) describes a technique for minimally supervised learning using collocates. A few seed collocates (possibly position-specific) are chosen for each sense (manually or via an MRD), then these are used to accurately identify distinct senses. The sentences in which the disambiguated senses occur can then be used to learn other discriminating collocates automatically, producing a decision list. The process can then be iterated. The algorithm allows bad collocates to be overridden. This works because of the general principle of 'one sense per collocation' (experimentally demonstrated by Yarowsky — it's not absolute, but there are very strong preferences).

In a bit more detail, using Yarowsky's example of disambiguating *plant* (which is homonymous between factory vs vegetation senses):

1. Identify all examples of the word to be disambiguated in the training corpus and store their contexts.

sense	training example
?	company said that the *plant* is still operating
?	although thousands of *plant* and animal species
?	zonal distribution of *plant* life
?	company manufacturing *plant* is in Orlando
	etc

2. Identify some seeds which reliably disambiguate a few of these uses. Tag the disambiguated senses automatically and count the rest as residual. For instance, choosing 'plant life' as a seed for the vegetation sense of plant (sense A) and 'manufacturing plant' as the seed for the factory sense (sense B):

sense	training example
?	company said that the *plant* is still operating
?	although thousands of *plant* and animal species
A	zonal distribution of *plant* life
B	company manufacturing *plant* is in Orlando
	etc

This disambiguated 2% of uses in Yarowsky's corpus, leaving 98% residual.

3. Train a *decision list* classifier on the Sense A/Sense B examples. A decision list approach gives a list of criteria which are tried in order until an applicable test is found: this is then applied. The decision list classifier takes a set of already classified examples and returns criteria which distinguish them (e.g., word before / after / within window). The tests are each associated with a reliability metric. The original seeds are likely to be at the top of the decision list that is returned, followed by other discriminating terms. e.g. the decision list might include:

reliability	criterion	sense
8.10	*plant* life	A
7.58	manufacturing *plant*	B
6.27	*animal* within 10 words of *plant*	A
	etc	

Here '*animal* within 10 words of *plant*' is a new criterion, learned by the classifier.

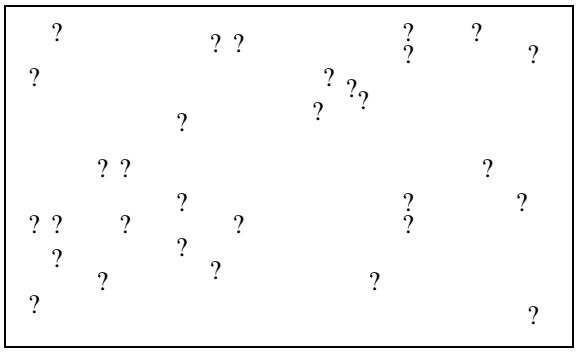
4. Apply the decision list classifier to the training set and add all examples which are tagged with greater than a threshold reliability to the Sense A and Sense B sets.

sense	training example
?	company said that the *plant* is still operating
A	although thousands of *plant* and animal species
A	zonal distribution of *plant* life
B	company manufacturing *plant* is in Orlando
	etc

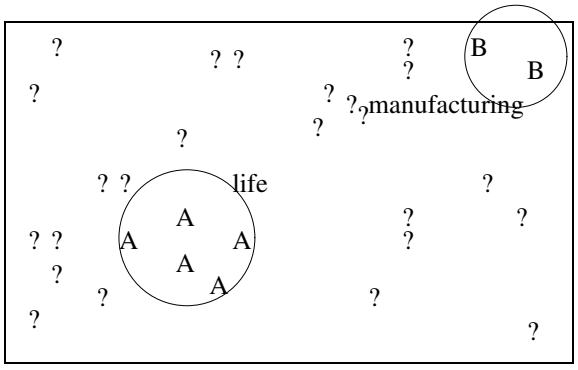
5. Iterate the previous steps 3 and 4 until convergence
6. Apply the classifier to the unseen test data

The following schematic diagrams may help:

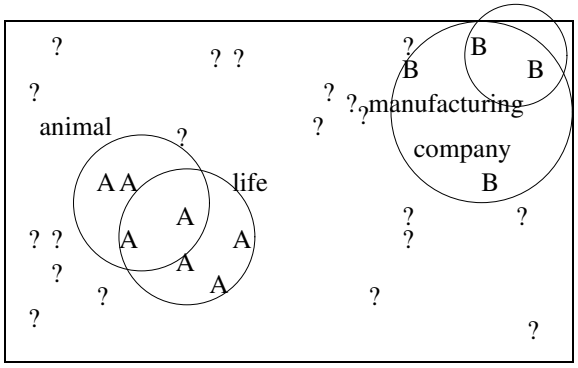
Initial state:



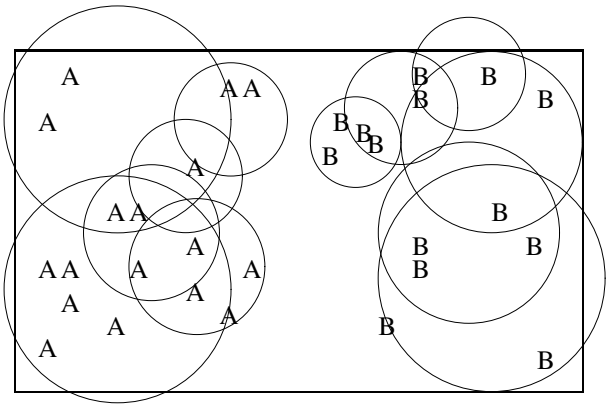
Seeds



Iterating:



Final:



Yarowsky also demonstrated the principle of 'one sense per discourse'. For instance, if *plant* is used in the botanical sense in a particular text, then subsequent instances of *plant* in the same tense will also tend to be used in the botanical sense. Again, this is a very strong, but not absolute effect. This can be used as an additional refinement for the algorithm above, assuming we have a way of detecting the boundaries between distinct texts in the corpus.

Decision list classifiers can be thought of as automatically trained case statements. The experimenter decides on the classes of test (e.g., word next to word to be disambiguated; word within window 10). The system automatically generates and orders the specific tests based on the training data.

Yarowsky argues that decision lists work better than many other statistical frameworks because no attempt is made to combine probabilities. This would be complex, because the criteria are not independent of each other. More details of this approach are in J&M (section 20.5).

Yarowsky's experiments were nearly all on homonyms: these principles probably don't hold as well for sense extension.

5.9 Evaluation of WSD

The baseline for WSD is generally 'pick the most frequent' sense: this is hard to beat! However, in many applications, we don't know the frequency of senses.

SENSEVAL and SENSEVAL-2 evaluated WSD in multiple languages, with various criteria, but generally using WordNet senses for English. The human ceiling for this task varies considerably between words: probably partly because of inherent differences in semantic distance between groups of uses and partly because of WordNet itself, which sometimes makes very fine-grained distinctions. An interesting variant in SENSEVAL-2 was to do one experiment on WSD where the disambiguation was with respect to uses requiring different translations into Japanese. This has the advantage that it is useful and relatively objective, but sometimes this task requires splitting terms which aren't polysemous in English (e.g., *water* — hot vs cold). Performance of WSD on this task seems a bit better than the general WSD task.

5.10 Further reading and background

WordNet is freely downloadable: the website has pointers to several papers which provide a good introduction. Recent initiatives have made many WordNets for other languages freely available: many are cross-linked to each other (generally via the English WordNet).

WSD is out of fashion as a standalone NLP subtask: these are several reasons for this, but the most important is that it seems unhelpful to consider it in isolation from applications. WSD is important in speech synthesis, for example, but only for a relatively small number of words where the sense distinction indicates a difference in pronunciation (e.g., *bass* but not *bank* or *plant*). In SMT, the necessary disambiguation happens as part of the general model rather than being a separate step, even conceptually. Nevertheless, it is useful to look at WSD because the important features in WSD models are important in these other contexts, and because it is a relatively straightforward illustration of an idea which applies to a class of NLP problems. Yarowsky's approach is a good illustration of a family of seed-based, semi-supervised methods used in other contexts and the paper is well-written and should be understandable:
Yarowsky, David (1995)

Unsupervised word sense disambiguation rivalling supervised methods,
Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-95) MIT, 189–196

Like many other NLP papers, this can be downloaded via the ACL Anthology
`http://aclweb.org/anthology-new/`.

6 Lecture 6: Distributional semantics

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

> 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 you don't know the word *scrumpy*, you can get a good idea of its meaning from contexts like this. Humans typically learn word meanings from context rather than explicit definition: sometimes these meanings are perceptually grounded (e.g., someone gives you a glass of scrumpy), sometimes not.

It is to a large extent an open question how word meanings are represented in the brain.[26] Distributional semantics uses linguistic context to represent meaning and this is likely to have some relationship to mental meaning representation. It can only be a partial representation of lexical meaning (perceptual grounding is clearly important too, and more-or-less explicit definition may also play a role) but, as we'll see, distributions based on language alone are good enough for some tasks. In these models, meaning is seen as a space, with dimensions corresponding to elements in the context (*features*). Computational techniques generally use vectors to represent the space and the terms *semantic space models* and *vector space models* are sometimes used instead of distributional semantics. [27] Schematically:

	feature_1	feature_2	...	feature_n
word_1	$f_{1,1}$	$f_{2,1}$		$f_{n,1}$
word_2	$f_{1,2}$	$f_{2,2}$		$f_{n,2}$
...				
word_m	$f_{1,m}$	$f_{2,m}$		$f_{n,m}$

There are many different possible notions of **features:** co-occur with word_n in some window, co-occur with word_n as a syntactic dependent, occur in paragraph_n, occur in document_n ...

The main use of distributional models has been in measuring similarity between pairs of words: such measurements can be exploited in a variety of ways to model language: similarity measurements allow clustering of words so can be used as a way of getting at unlabelled semantic classes. In the rest of this lecture, we will first discuss some possible models illustrating the choices that must be made when designing a distributional semantics system and go through a step-by-step example. We'll then look at some real distributions and then describe how distributions can be used in measuring similarity between words. We'll briefly describe how polysemy affects distributions and conclude with a discussion of the relationship between distributional models and the classic lexical semantic relations of synonymy, antonymy and hyponymy.

6.1 Models

Distributions are vectors in a multidimensional semantic space, that is, objects with a magnitude (length) and a direction. The *semantic space* has dimensions which correspond to possible contexts. For our purposes, a distribution can be seen as a point in that space (the vector being defined with respect to the origin of that space). e.g., cat [...dog 0.8, eat 0.7, joke 0.01, mansion 0.2, zebra 0.1...], where 'dog', 'joke' and so on are the dimensions.

Context:
Different models adopt different notions of context:

- Word windows (unfiltered): n words on either side of the item under consideration (unparsed text).
  Example: n=2 (5 words window):

  > ... the prime **minister** acknowledged that ...

[26] In fact, it's common to talk about concepts rather than word meanings when discussing mental representation, but while some researchers treat some (or all) concepts as equivalent to word senses (or phrases), others think they are somehow distinct. We will return to this in lecture 11.

[27] Vector space models in IR are directly related to distributional models. Some more complex techniques use tensors of different orders rather than simply using vectors, but we won't discuss them in this lecture.

- Word windows (filtered): n words on either side of the item under consideration (unparsed text). Some words will be in a stoplist and not considered part of the context. It is common to put function words and some very frequent content words in the stoplist. The stoplist may be constructed manually, but often the corpus is POS-tagged and only certain POS tags are considered part of the context.
  Example: n=2 (5 words window), underlined words are in the stop list.

  ... the prime **minister** acknowledged that ...

- Lexeme windows: as above, but a morphological processor is applied first that converts the words to their stems.

- Dependencies: syntactic or semantic. The corpus is converted into a list of directed links between heads and dependents. The context for a lexeme is constructed based on the dependencies it belongs to. The length of the dependency path varies according to the implementation.

Example of distributions for *word* comparing unparsed and parsed data:

unparsed data		**parsed data**	
meaning_n	ipa_n	or_c+phrase_n	and_c+deed_n
derive_v	verb_n	and_c+phrase_n	meaning_n+of_p
dictionary_n	mean_v	syllable_n+of_p	from_p+language_n
pronounce_v	hebrew_n	play_n+on_p	pron_rel_+utter_v
phrase_n	usage_n	etymology_n+of_p	for_p+word_n
latin_j	literally_r	portmanteau_n+of_p	in_p+sentence_n

Context weighting:
Different models use different methods of weighting the context elements:

- Binary model: if context c co-occurs with word w, value of vector $\vec{w}$ for dimension c is 1, 0 otherwise.

  ... [a long long long **example** for a distributional semantics] model... (n=4)

  ... {a 1} {dog 0} {long 1} {sell 0} {semantics 1}...

- Basic frequency model: the value of vector $\vec{w}$ for dimension c is the number of times that context c co-occurs with w.

  ... [a long long long **example** for a distributional semantics] model... (n=4)

  ... {a 2} {dog 0} {long 3} {sell 0} {semantics 1}...

- Characteristic model: the weights given to the vector components express how characteristic a given context is for w. Functions used include:

  – Pointwise Mutual Information (PMI), with or without discounting factor.

  $$pmi_{wc} = log(\frac{f_{wc} * f_{total}}{f_w * f_c}) \tag{4}$$

  where f_{wc} is the frequency with which word w occurs in context c, f_{total} is the total frequency of all the possible contexts, f_w is the frequency of the word w and f_c is the overall frequency of the context item. i.e., if we use words as dimensions, f_{wc} is the frequency with which word w and word c cooccur, f_c is the overall frequency of word c and f_{total} is the total frequency of all the context words.

  – Positive PMI (PPMI): as PMI but 0 if PMI < 0.

  – Derivatives such as Mitchell and Lapata's (2010) weighting function (PMI without the log).

Most work uses some form of characteristic model in order to give most weight to frequently cooccuring features, but allowing for the overall frequency of the terms in the context. Note that PMI is one of the measures used for finding collocations (see previous lecture): the distributional models can be seen as combining the collocations for words.

Semantic space:
Once the contexts and weights have been decided on, models also vary in which elements are included in the final vectors: i.e., what the total semantic space consists of. The main options are as follows (with positive and negative aspects indicated):

- Entire vocabulary. All the information is included – even rare contexts may be important. However using many dimensions (many hundreds of thousands) makes it slow. The dimensions will include a lot of noise: e.g. 002.png—thumb—right—200px—graph_n
- Top n words with highest frequencies. This is more efficient (5000-10000 dimensions are usual) and only 'real' words will be included. However, the model may miss infrequent but relevant contexts.
- Singular Value Decomposition and other dimensionality reduction techniques. SVD was used in LSA – Landauer and Dumais (1997). The number of dimensions is reduced by exploiting redundancies in the data. A new dimension might correspond to a generalisation over several of the original dimensions (e.g. the dimensions for car and vehicle are collapsed into one). This can be very efficient (200-500 dimensions are often used) and it should capture generalisations in the data. The problem is that SVD matrices are not interpretable. Arguably, this is a theoretical problem, but it is more obviously a practical problem: using SVD makes it impossible to debug the feature space by manual inspection. It is therefore unsuitable for initial experiments.

But there are several other variants.

6.2 Getting distributions from text

In this section, we illustrate a word-window model using PMI weightings with a stop list consisting of all closed-class words. We use the following example text (from Douglas Adams, Mostly harmless):

> The major difference between a thing that might go wrong and a thing that cannot possibly go wrong is that when a thing that cannot possibly go wrong goes wrong it usually turns out to be impossible to get at or repair.

Naturally, real distributions are calculated from much larger corpora!

Dimensions (all the open class words in the text, without lemmatization):

difference	major	usually
get	possibly	wrong
go	repair	
goes	thing	
impossible	turns	

Overall frequency counts (needed for the PMI calculations):

difference 1	major 1	usually 1
get 1	possibly 2	wrong 4
go 3	repair 1	
goes 1	thing 3	
impossible 1	turns 1	

Conversion into 5-word windows:

- ∅ ∅ **the** major difference

- ∅ the **major** difference between
- the major **difference** between a
- major difference **between** a thing
- ...

Context windows for wrong:
The major difference between a thing that [might go wrong and a] thing that cannot [possibly go wrong is that] when a thing that cannot [possibly go [wrong goes wrong] it usually] turns out to be impossible to get at or repair.

Distribution for *wrong* (raw frequencies)

difference 0	major 0	usually 1
get 0	possibly 2	wrong 2
go 3	repair 0	
goes 2	thing 0	
impossible 0	turns 0	

Note that the single token of *goes* is counted twice, because it occurs with two different tokens of *wrong*.

Distribution for wrong (PPMI):

difference 0	major 0	usually 0.70
get 0	possibly 0.70	wrong 0.40
go 0.70	repair 0	
goes 1	thing 0	
impossible 0	turns 0	

For instance, for the context *possibly*, f_{wc} is 2 (as in the raw frequency distribution table), f_c is the total count of *possibly* which is also 2 (as in the overall frequency count table), f_w is 4 (again, as in the overall frequency count table) and f_{total} is 20 (i.e., the sum of the frequencies in the overall frequency count table), so PMI is log(5).

6.3 Real distributions

In this section we give some more realistic examples of distributions, which have been derived using a methodology that we have adopted for our own research. The corpus (WikiWoods: Flickinger et al 2010: `http://moin.delph-in.net/WikiWoods`) is based on a dump of the entire English Wikipedia parsed with the English Resource Grammar (Flickinger, 2000, see lecture 5) and converted into semantic dependencies (see Lecture 6), though the results would be expected to be similar with syntactic dependencies. The dependencies considered include:

- For nouns: head verbs (+ any other argument of the verb), modifying adjectives, head prepositions (+ any other argument of the preposition).
  e.g. cat: chase_v+mouse_n, black_a, of_p+neighbour_n
- For verbs: arguments (NPs and PPs), adverbial modifiers.
  e.g. eat: cat_n+mouse_n, in_p+kitchen_n, fast_a
- For adjectives: modified nouns; rest as for nouns (assuming intersective composition).
  e.g. black: cat_n, chase_v+mouse_n

The model uses a semantic space of the top 100,000 contexts (because we wanted to include the rare terms) with a variant of PMI (Bouma 2007) for weighting:

$$pmi_{wc} = \frac{log(\frac{f_{wc} * f_{total}}{f_w * f_c})}{-log(\frac{f_{wc}}{f_{total}})} \quad (5)$$

An example noun, language:

0.54::other+than_p()+English_n
0.53::English_n+as_p()
0.52::English_n+be_v
0.49::english_a
0.48::and_c+literature_n
0.48::people_n+speak_v
0.47::French_n+be_v
0.46::Spanish_n+be_v
0.46::and_c+dialects_n
0.45::grammar_n+of_p()
0.45::foreign_a
0.45::germanic_a
0.44::German_n+be_v
0.44::of_p()+instruction_n
0.44::speaker_n+of_p()
0.42::generic_entity_rel_+speak_v
0.42::pron_rel_+speak_v
0.42::colon_v+English_n
0.42::be_v+English_n
0.42::language_n+be_v
0.42::and_c+culture_n
0.41::arabic_a
0.41::dialects_n+of_p()
0.40::part_of_rel_+speak_v
0.40::percent_n+speak_v
0.39::spanish_a
0.39::welsh_a
0.39::tonal_a

An example adjective, academic:

0.52::Decathlon_n
0.51::excellence_n
0.45::dishonesty_n
0.45::rigor_n
0.43::achievement_n
0.42::discipline_n
0.40::vice_president_n+for_p()
0.39::institution_n
0.39::credentials_n
0.38::journal_n
0.37::journal_n+be_v
0.37::vocational_a
0.37::student_n+achieve_v
0.36::athletic_a
0.36::reputation_n+for_p()
0.35::regalia_n
0.35::program_n
0.35::freedom_n
0.35::student_n+with_p()
0.35::curriculum_n
0.34::standard_n
0.34::at_p()+institution_n
0.34::career_n
0.34::Career_n
0.33::dress_n
0.33::scholarship_n
0.33::prepare_v+student_n
0.33::qualification_n

Corpus choice is another parameter that has to be considered in building models. Some research suggests that one should use as much data as possible. Some commonly used corpora:

- British National Corpus (BNC): 100 m words
- Wikipedia dump used in Wikiwoods: 897 m words
- UKWac (obtained from web-crawling): 2 bn words

In general, more data does give better models but the domain has to be considered: for instance, huge corpora of financial news won't give models that work well with other text types. Furthermore, more data is not realistic from a psycholinguistic point of view. We encounter perhaps 50,000 words a day (although nobody actually has good estimates of this!) so the BNC, which is very small by the standards of current experiments, corresponds to approximately 5 years' exposure.

It is clear that sparse data is a problem for relatively rare words. For instance, consider the following distribution for unicycle, as obtained from Wikiwoods:

0.45::motorized_a
0.40::pron_rel_+ride_v
0.24::for_p()+entertainment_n
0.24::half_n+be_v
0.24::unwieldy_a
0.23::earn_v+point_n
0.22::pron_rel_+crash_v
0.19::man_n+on_p()
0.19::on_p()+stage_n
0.19::position_n+on_p()
0.17::slip_v
0.16::and_c+1_n
0.16::autonomous_a
0.16::balance_v
0.13::tall_a
0.12::fast_a
0.11::red_a
0.07::come_v
0.06::high_a

Note that humans exploit a lot more information from context and can get a good idea of word meanings from a small number of examples.

Distributions are generally constructed without any form of sense disambiguation. The semantic space can be thought of as consisting of subspaces for different senses, with homonyms (presumably) relatively distinct. For instance, consider the following distribution for pot:

0.57::melt_v
0.44::pron_rel_+smoke_v
0.43::of_p()+gold_n
0.41::porous_a
0.40::of_p()+tea_n
0.39::player_n+win_v
0.39::money_n+in_p()
0.38::of_p()+coffee_n
0.33::amount_n+in_p()
0.33::ceramic_a
0.33::hot_a
0.32::boil_v
0.31::bowl_n+and_c
0.31::ingredient_n+in_p()
0.30::plant_n+in_p()
0.30::simmer_v
0.29::pot_n+and_c
0.28::bottom_n+of_p()
0.28::of_p()+flower_n
0.28::of_p()+water_n
0.28::food_n+in_p()

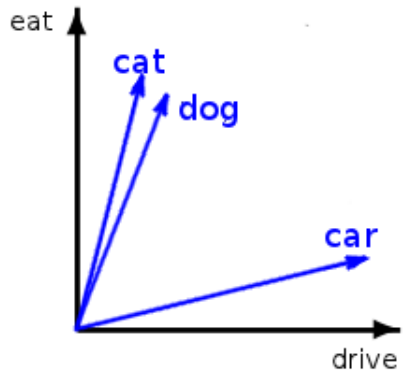
Finally, note that distributions contain many contexts which arise from multiword expressions of various types, and these often have high weights. The distribution of *pot* contains several examples, as does the following distribution for time:

0.46::of_p()+death_n
0.45::same_a
0.45::1_n+at_p(temp)
0.45::Nick_n+of_p()
0.42::spare_a
0.42::playoffs_n+for_p()
0.42::of_p()+retirement_n
0.41::of_p()+release_n
0.40::pron_rel_+spend_v
0.39::sand_n+of_p()
0.39::pron_rel_+waste_v
0.38::place_n+around_p()
0.38::of_p()+arrival_n
0.38::of_p()+completion_n
0.37::after_p()+time_n
0.37::of_p()+arrest_n
0.37::country_n+at_p()
0.37::age_n+at_p()
0.37::space_n+and_c
0.37::in_p()+career_n
0.37::world_n+at_p()

To sum up, there is a wide range of choice in constructing distributional models. Manually examining the characteristic contexts gives us a good idea of how sensible different weighting measures are, for instance, but we need to look at how distributions are actually used to evaluate how well they model meaning.

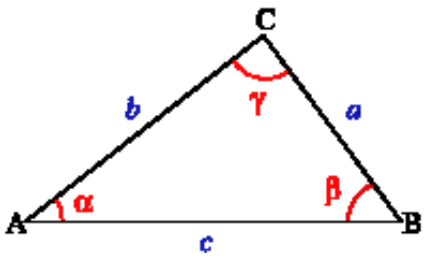
6.4 Similarity

Calculating similarity in a distributional space is done by calculating the distance between the vectors.



The most common method is *cosine similarity*.

- Law of cosines: $c^2 = a^2 + b^2 - 2ab\cos\gamma$



Cosine similarity:

$$\frac{\sum v1_k * v2_k}{\sqrt{\sum v1_k^2} * \sqrt{\sum v2_k^2}} \tag{6}$$

This measure calculates the angle between two vectors and is therefore length-independent. This is important, as frequent words have longer vectors than less frequent ones.

The following examples should give some idea of the scales of similarity found:

house – building 0.43
gem – jewel 0.31
capitalism – communism 0.29
motorcycle – bike 0.29
test – exam 0.27
school – student 0.25
singer – academic 0.17
horse – farm 0.13
man –accident 0.09
tree – auction 0.02
cat –county 0.007

Note that perfect similarity gives a cosine of 1, but that even near-synonyms like *gem* and *jewel* have much lower cosine similarity.

Words most similar to cat, as chosen from the 5000 most frequent nouns in Wikipedia.

1 cat	0.29 goat	0.24 squirrel	0.22 mammal
0.45 dog	0.28 snake	0.24 dragon	0.21 bat
0.36 animal	0.28 bear	0.24 frog	0.21 duck
0.34 rat	0.28 man	0.23 baby	0.21 cattle
0.33 rabbit	0.28 cow	0.23 child	0.21 dinosaur
0.33 pig	0.26 fox	0.23 lion	0.21 character
0.31 monkey	0.26 girl	0.23 person	0.21 kid
0.31 bird	0.26 sheep	0.23 pet	0.21 turtle
0.30 horse	0.26 boy	0.23 lizard	0.20 robot
0.29 mouse	0.26 elephant	0.23 chicken	
0.29 wolf	0.25 deer	0.22 monster	
0.29 creature	0.25 woman	0.22 people	
0.29 human	0.25 fish	0.22 tiger	

This notion of similarity is very broad. It includes synonyms, near-synonyms, hyponyms, taxonomical siblings, antonyms and so on. But it does correlate with a psychological reality. One of the favourite tests of the distributional semantics community is the calculation of rank correlation between a distributional similarity system and human judgements on the Miller & Charles (1991) test set shown below:

3.92 automobile-car	3.05 bird-cock	0.84 forest-graveyard
3.84 journey-voyage	2.97 bird-crane	0.55 monk-slave
3.84 gem-jewel	2.95 implement-tool	0.42 lad-wizard
3.76 boy-lad	2.82 brother-monk	0.42 coast-forest
3.7 coast-shore	1.68 crane-implement	0.13 cord-smile
3.61 asylum-madhouse	1.66 brother-lad	0.11 glass-magician
3.5 magician-wizard	1.16 car-journey	0.08 rooster-voyage
3.42 midday-noon	1.1 monk-oracle	0.08 noon-string
3.11 furnace-stove	0.89 food-rooster	
3.08 food-fruit	0.87 coast-hill	

The human similarity results can be replicated: the Miller & Charles experiment is a re-run of Rubenstein & Goodenough (1965): the correlation coefficient between them is 0.97. A good distributional similarity system can have a correlation of 0.8 or better with the human data (although there is a danger the reported results are unreasonably high, because this data has been used in so many experiments).

Another frequently used dataset is the TOEFL (Test of English as a Foreign Language) synonym test. For example:

Stem: levied
Choices: (a) imposed
(b) believed
(c) requested
(d) correlated
Solution: (a) imposed

Non-native English speakers are reported to average around 65% on this test (US college applicants): the best corpus-based results are 100% (Bullinaria and Levy, 2012) . But note that the authors who got this result suggest the test is not very reliable — one reason is probably that the data includes some extremely rare words.

Similarity measures can be applied as a type of backoff technique in a range of tasks. For instance, in sentiment analysis (discussed in lecture 1), an initial bag of words acquired from the training data can be expanded by including distributionally similar words.

6.5 Distributions and classic lexical semantic relationships

Distributions are a usage representation: they are corpus-dependent, culture-dependent and register-dependent. Synonyms with different registers often don't have a very high similarity. For example, the similarity between policeman and cop is 0.23 and the reason for this relatively low number becomes clear if one examines the highest weighted features:

policeman	0.36::uniformed_a	0.28::incompetent_a	0.26::on_p()+duty_n
0.59::ball_n+poss_rel	0.35::uniform_n+poss_rel	0.28::pron_rel_+shoot_v	0.25::salary_n+poss_rel
0.48::and_c+civilian_n	0.35::civilian_n+and_c	0.28::hat_n+poss_rel	0.25::on_p()+horseback_n
0.42::soldier_n+and_c	0.31::iraqi_a	0.28::terrorist_n+and_c	0.25::armed_a
0.41::and_c+soldier_n	0.31::lot_n+poss_rel	0.27::and_c+crowd_n	0.24::and_c+nurse_n
0.38::secret_a	0.31::chechen_a	0.27::military_a	0.24::job_n+as_p()
0.37::people_n+include_v	0.30::laugh_v	0.27::helmet_n+poss_rel	0.24::open_v+fire_n
0.37::corrupt_a	0.29::and_c+criminal_n	0.27::father_n+be_v	
cop	0.33::pron_rel_+call_v	0.27::investigate_v+murder_n	0.23::and_c+interference_n
0.45::crooked_a	0.32::funky_a	0.26::on_p()+force_n	0.23::arrive_v
0.45::corrupt_a	0.32::bad_a	0.25::parody_n+of_p()	0.23::and_c+detective_n
0.44::maniac_a	0.29::veteran_a	0.25::Mason_n+and_c	0.22::look_v+way_n
0.38::dirty_a	0.29::and_c+robot_n	0.25::pron_rel_+kill_v	0.22::dead_a
0.37::honest_a	0.28::and_c+criminal_n	0.25::racist_a	0.22::pron_rel_+stab_v
0.36::uniformed_a	0.28::bogus_a	0.24::addicted_a	0.21::pron_rel_+evade_v
0.35::tough_a	0.28::talk_v+to_p()+pron_rel_	0.23::gritty_a	

Synonyms and similarity: some further examples:

- Similarity between eggplant/aubergine: 0.11
  These are true synonyms but have relatively low cosine similarity. This is partly due to frequency (222 for eggplant, 56 for aubergine).
- Similarity between policeman/cop: 0.23 (as discussed)
- Similarity between city/town: 0.73

In general, true synonymy does not correspond to higher similarity scores than near-synonymy.
Antonyms have high similarity, as indicated by the examples below:

- cold/hot 0.29
- dead/alive 0.24
- large/small 0.68
- colonel/general 0.33

It is possible to automatically distinguish antonyms from (near-)synonyms using corpus-based techniques, but this requires additional heuristics. For instance, it has been observed that antonyms are frequently coordinated while synonyms are not:

- a selection of cold and hot drinks
- wanted dead or alive
- lecturers, readers and professors are invited to attend

Similarly, it is possible to acquire hyponymy relationships from distributions, but this is much less effective than looking for explicit taxonomic relationships in Wikipedia text.

6.6 Historical background

Distributional semantics has been discussed since at least the 1950s: the first computational work published was probably Karen Spärck Jones (1964) Cambridge PhD thesis 'Synonymy and Semantic Classification' which used dictionaries for context. The first experiments on sentential contexts I am aware of were by Harper (1965) (inspired by the work of the linguist Zellig Harris) which were refined to use a more motivated notion of similarity by Spärck Jones (1967). Salton's early work on vector space models in IR was also ongoing from the early 1960s. The early distributional work not followed up within computational linguistics, and in fact was almost entirely forgotten. This was partly because of the limitations of computers and available corpora, but also because the 1966 ALPAC report led to greatly diminished funding for CL and because the dominant Chomskyan approach in linguistics was highly hostile to any quantitative methodology. Spärck Jones switched to working on Information Retrieval, but the early classification experiments influenced her approach to IR, in particular the development of tf*idf. By the late 1980s and early 1990s there were sufficiently large corpora, computer memory and disk space to make simple distributional semantic techniques practical: much of the research at that point was heavily influenced by the IR techniques. By the early 2000s, large scale, robust parsing made more complex notions of distributional context practical and there has been a huge proliferation of CL research in recent years.

7 Lecture 7: Distributional semantics (continued)

7.1 Distributional word clustering

Clustering refers to a class of machine learning techniques, whose goal is to group a set of objects into clusters, such that the objects within a cluster are similar to each other (given a definition of similarity) and objects in different clusters are dissimilar. Word clustering in particular is commonly used in NLP to identify words with similar meaning. For instance, *apple, grape* and *avocado* would belong to one cluster and *car, bike* and *motorcycle* to another. As in the case of supervised classification, clustering algorithms perform generalisations about the properties of concepts and their similarity based on the data represented in the form of features and a (distance) function to be optimized. The features typically used in word clustering experiments include lexico-syntactic contexts in which the words appear (typically in the form of dependencies), as well as information about subcategorization and semantic roles. However, in principle even context windows extracted from unparsed data could be used as features. Such window-based features lead to clusters of words that are topically similar (such as *bike* and *handlebar*) rather than belonging to the same category (such as *bike* and *car*) as in the case of dependency features. In contrast to supervised classification, the clusters are usually inferred in an unsupervised way, without any need for manually labelled data.

Let us consider a simple experiment, grouping nouns into clusters using dependency features. Specifically, we will use the verbs that take the noun as a direct object or a subject to construct our feature vectors. In order to do this, we first need to extract all verb-subject and verb-direct object dependencies from a large corpus (we will use the Gigaword corpus in our experiment). For each noun in our dataset, we then create a feature vector where each feature is a verb lemma indexed by the type of dependency it was found in. The resulting feature vectors (the top-ranked features) for the nouns *tree* and *crop* are shown below as an example.[28]

tree	26 fell_v_Subj	**crop**	9 yield_v_Dobj
131 grow_v_Subj	25 look_v_Subj	78 grow_v_Subj	9 protect_v_Dobj
85 plant_v_Dobj	23 make_v_Subj	76 grow_v_Dobj	9 fail_v_Subj
82 climb_v_Dobj	23 make_v_Dobj	44 produce_v_Dobj	9 destroy_v_Dobj
49 plant_v_Subj	23 grow_v_Dobj	23 yield_v_Subj	8 plant_v_Subj
48 see_v_Dobj	22 use_v_Dobj	16 harvest_v_Dobj	7 spray_v_Subj
46 cut_v_Dobj	22 surround_v_Subj	12 plant_v_Dobj	7 spray_v_Dobj
40 stand_v_Subj	22 round_v_Dobj	10 sow_v_Subj	7 lose_v_Dobj
27 fall_v_Dobj	20 overhang_v_Subj	10 ensure_v_Dobj	6 feed_v_Subj
26 like_v_Dobj	...	10 cut_v_Dobj	...

One can see that these two feature vectors share many prominent features, e.g. *grow_v_Subj, plant_v_Dobj, grow_v_Dobj, cut_v_Dobj* etc., suggesting that the meanings of the nouns may be related. The feature values are the corpus frequencies of the respective dependencies, typically normalised before the clustering algorithm is applied.

There are many different clustering algorithms available and used in word clustering, each making different assumptions about the data and thus yielding slightly different results. Popular algorithms include, for instance, expectation maximization (EM) clustering, Gaussian mixture model, spectral clustering and K-means. Below, I will briefly describe an example technique for clustering nouns using the K-means algorithm.

Given a set of data points $\{x_1, x_2, ..., x_N\}$, where each data point is represented by a D-dimensional feature vector, K-means clustering aims to partition the N data points into K clusters $C = \{C_1, C_2, ..., C_K\}$. The objective is to find the partitioning that minimizes the sum of the squares of the distances of each data point to the cluster mean vector μ_i:

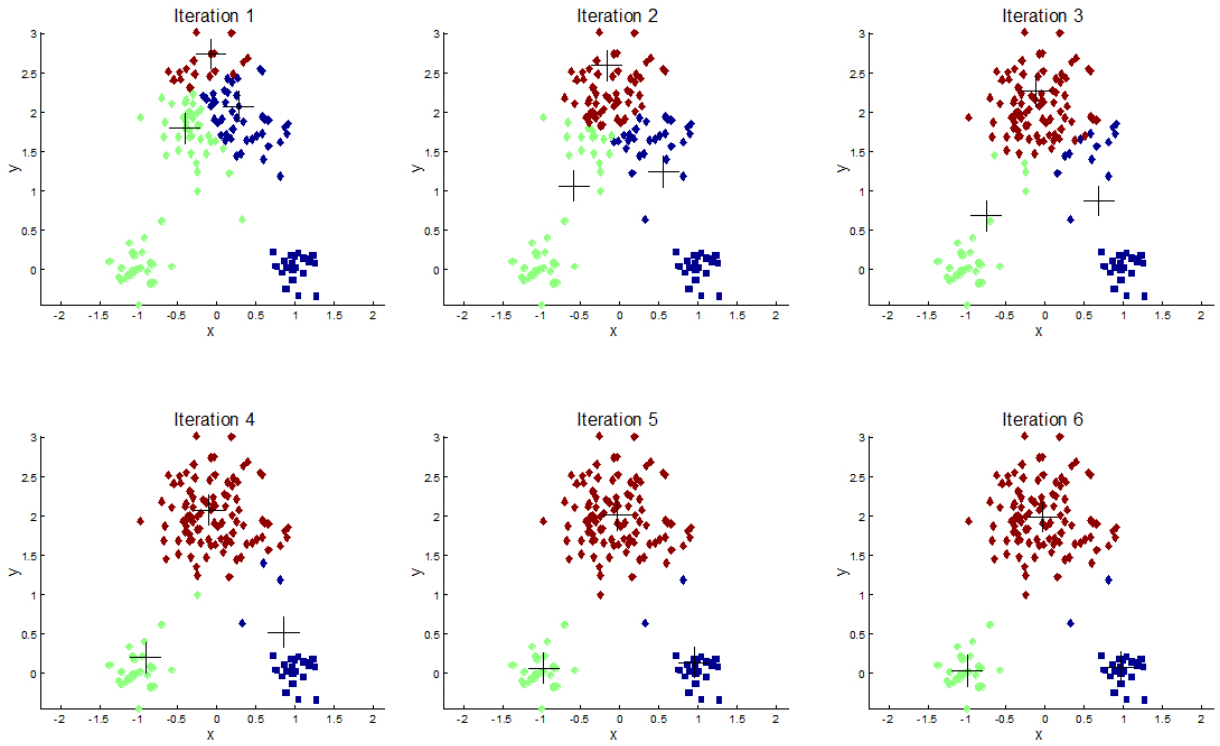
$$\arg\min_{C} \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \tag{7}$$

where μ_i is computed as the mean of the data points in C_i.

The below figure[29] aims to informally convey the intuition behind K-means clustering.

[28] In reality, such feature vectors typically contain thousands of features, covering all verbs that can occur with the nouns in the dataset

[29] Figure source: `https://apandre.wordpress.com/visible-data/cluster-analysis/`



The algorithm starts by randomly initialising the K cluster means. In this example, we initialise three mean vectors as the goal is to group the data points into three clusters. The data points are then assigned to clusters (as shown by the color highlighting in the figure) in a way that minimizes the overall sum of the squares of their distances to the given means. In the next iteration, the new cluster means are computed given the current data assignment. The data points are then again re-assigned to clusters given the new means, so as to optimize the above objective. These two steps, i.e. the new mean computation and the respective re-assignment of objects to clusters, are iteratively repeated until convergence.

In our example experiment, we clustered 2000 frequent nouns into 200 clusters, using verb-subject and verb-object dependencies as features, as described above. Some examples of the resulting clusters are shown below:

Cluster1 'tree' 'crop' 'flower' 'plant' 'root' 'leaf' 'seed' 'rose' 'wood' 'grain' 'stem' 'forest' 'garden'
Cluster2 'consent' 'permission' 'concession' 'injunction' 'licence' 'approval'
Cluster21 'lifetime' 'quarter' 'period' 'century' 'succession' 'stage' 'generation' 'decade' 'phase' 'interval' 'future'
Cluster22 'subsidy' 'compensation' 'damages' 'allowance' 'payment' 'pension' 'grant'
Cluster109 'official' 'officer' 'inspector' 'journalist' 'detective' 'constable' 'police' 'policeman' 'reporter'
Cluster111 'girl' 'other' 'woman' 'child' 'person' 'people'
Cluster112 'length' 'past' 'mile' 'metre' 'distance' 'inch' 'yard'
Cluster113 'tide' 'breeze' 'flood' 'wind' 'rain' 'storm' 'weather' 'wave' 'current' 'heat'
Cluster117 'sister' 'daughter' 'parent' 'relative' 'lover' 'cousin' 'friend' 'wife' 'mother' 'husband' 'brother' 'father'

One can see that the clusters tend to represent generalisations over words, capturing some of the underlying concepts. For instance, Cluster 1 in the figure contains various kinds of plants, i.e. things that can *grow, be planted, be watered* etc. Other clusters cover concepts such as *time* (Cluster 21), *distance* (Cluster 112), *weather* and *natural phenomena* (Cluster 113) or *people* (Clusters 109, 111, 117) grouped based on different attributes. Hopefully, these examples intuitively show how similar context distributions may lead to the clusters of nouns that form coherent concepts.

Verb clustering is also commonly used in NLP, along with noun clustering. There are approaches that have shown that it is also possible to cluster adjectives and even adverbs using distributional features. However, the latter are not typically applied in wider NLP. Noun and verb clustering are, on the contrary, commonly used in lexical acquisition, an area of NLP that aims to automatically derive lexical information from corpora and build large-scale lexical resources

that can support other NLP tasks. Obtaining generalisations over words can help in tasks such as automatically acquiring subcategorization information and parsing. Word clustering of some form is also sometimes used as a smoothing technique, to reduce the effects of data sparsity in other statistical learning tasks.

7.2 Selectional preferences

Selectional preferences are the semantic constraints that a predicate places onto its arguments. This means that certain classes of entities are more likely to fill the predicate's argument slot than others. For instance, while the sentences "*The authors wrote a new paper.*" and "*The cat is eating your sausage!*" sound natural and describe plausible real-life situations, the sentences "*The carrot ate the keys.*" and "*The law sang a driveway.*" appear implausible and difficult to interpret, as the arguments do not satisfy the verbs' common preferences.

Selectional preferences provide generalisations about word meaning and use and find a wide range of applications in NLP. These include word sense disambiguation, resolving ambiguous syntactic attachments, natural language inference and detection of multi-word expressions, among others. Automatic acquisition of selectional preferences from linguistic data has thus become an active area of research in NLP. One of the challenges it faces is data sparsity: a rarely attested predicate-argument pair may be either an infelicitous relation or a felicitous but rarely used one. A selectional preference acquisition method, therefore, needs to be able to generalise from frequently manifested predicate-argument pairs to the infrequent ones by abstracting from lexical units to classes of arguments via some notion of similarity. The NLP community has investigated a range of techniques to perform generalisation from observed arguments to their underlying types, including the use of WordNet synsets as argument classes, word clustering, distributional similarity metrics and latent variable models.

The widespread interest in acquisition of selectional preferences was triggered by the work of Resnik (1997)[30], who was the first to combine the knowledge of semantic classes (at that stage predefined by an ontology) with the statistical methods from information theory. He viewed selectional preferences as probability distributions over all potential arguments of a predicate, rather than a single argument class (or a limited set of argument classes) assigned to the predicate. This new setting enabled corpus-based statistical learning of selectional preferences, mainly focused on the preferences of verbs for their nominal arguments.

7.3 Resnik's selectional preference model

Resnik modeled selectional preferences of a verb in probabilistic terms as the difference between the posterior distribution of noun classes in a particular grammatical relation with the verb and their prior distribution in that syntactic position regardless of the identity of the verb. He quantified this difference using the Kullbach-Leibler divergence and defined *selectional preference strength* of the verb as follows:

$$S_R(v) = D(P(c|v)||P(c)) = \sum_c P(c|v) \log \frac{P(c|v)}{P(c)}, \tag{8}$$

where $P(c)$ is the prior probability of the noun class, $P(c|v)$ is the posterior probability of the noun class given the verb and R is the grammatical relation in question. In order to quantify how well a particular argument class fits the verb, Resnik defined another measure called *selectional association*:

$$A_R(v, c) = \frac{1}{S_R(v)} P(c|v) \log \frac{P(c|v)}{P(c)}, \tag{9}$$

which stands for the relative contribution of a particular argument class to the overall selectional preference strength of the verb.

The individual probabilities can be straightforwardly calculated from corpus frequencies:

$$P(c) = \frac{f(c)}{\sum_k f(c_k)},$$

[30] Philip Resnik. 1997. Selectional preference and sense disambiguation. In *ACL SIGLEX Workshop on Tagging Text with Lexical Semantics,* Washington, D.C.

$$P(c|v) = \frac{f(v,c)}{f(v)},$$

$$f(c) = \sum_{n_i \in c} f(n_i),$$

where $f(v,c)$ is the frequency of the verb v co-occurring with the noun class c; $f(v)$ is the total frequency of the verb v with all noun classes and $f(c)$ is the total frequency of the noun class c.

Resnik used WordNet to define the argument classes, as well as to map the words in the corpus to those classes. As I already mentioned above, more modern methods tend to acquire argument classes automatically from corpus data. This can be done, for instance, using clustering techniques.

7.4 Examples of selectional preference distributions

I will now show some examples of selectional preference distributions acquired using Resnik's model and our noun clustering method described in section 7.1. We will compute preferences of verbs for their subjects and direct objects. We will use the syntactically-parsed version of the British National Corpus (BNC)[31] to extract co-occurrence frequencies of verbs with their subjects and direct objects. The co-occurence frequency of a verb with a given noun cluster can be computed by summing up its co-occurrence frequencies with the individual nouns in the cluster. The resulting frequencies can then be used to compute $P(c|v)$ in Resnik's formulae. The frequencies of the noun clusters on their own and the resulting $P(c)$ values are computed similarly.

The highest-ranking direct object classes of the verb *kill* acquired by this model are shown below.

Selectional preferences of ***kill*** (Dobj)
0.3874 girl other woman child person people
0.2009 being species sheep animal creature horse baby human fish male lamb bird rabbit female insect cattle mouse monster
0.1977 sister daughter parent relative lover cousin friend wife mother husband brother father
0.0426 thousand citizen inhabitant resident minority youngster refugee peasant miner hundred
0.0378 gene tissue cell particle fragment bacterium protein acid complex compound molecule organism
0.0336 fleet soldier knight force rebel guard troops crew army pilot
0.0335 official officer inspector journalist detective constable police policeman reporter
0.0322 victim bull teenager prisoner hero gang enemy rider offender youth killer thief driver defender hell
0.0136 week month year
0.0129 staff volunteer worker teacher employee member personnel manager
...

One can see from the figure that nearly all of the arguments represent humans or other living beings, with the exception of the *time* cluster which corresponds to the metaphorical use of *kill*. The direct object preferences of *drink* below are similar in the sense that the top arguments represent *liquids* followed by a set of other clusters that are metaphorically used or are less suitable arguments otherwise.

[31] http://www.natcorp.ox.ac.uk/

Selectional preferences of ***drink*** (Dobj)
0.5831 drink coffee champagne pint wine beer
0.2778 drop tear sweat paint blood water juice
0.1084 mixture salt dose ingredient sugar substance drug milk cream alcohol fibre chemical
0.0515 brush bowl bucket receiver barrel dish glass container plate basket bottle tray
0.0069 couple minute night morning hour time evening afternoon
0.0041 stability efficiency security prospects health welfare survival safety
0.0025 recording music tape song tune radio guitar trick album football organ stuff
0.0005 rage excitement panic anger terror flame laughter
0.0004 ball shot kick arrow stroke bullet punch bomb shell blow missile
0.0003 lunch dinner breakfast meal
...

However, note the difference in the selectional association values assigned to potable and non-potable arguments, with the latter being very low. While the model reflects selectional preferences of verbs as distributions over all noun clusters, the number of suitable argument clusters varies from verb to verb depending on the verb's semantics. The model reflects this variation though the distribution of selectional association values.

Highly polysemous verbs, such as *run*, have a greater number of plausible arguments corresponding to their different senses. The subject preference distribution of *run* below reflects this:

Selectional preferences of ***run*** (Subj)
0.2125 drop tear sweat paint blood water juice
0.1665 technology architecture program system product version interface software tool computer network processor chip package
0.1657 tunnel road path trail lane route track street bridge
0.1166 carriage bike vehicle train truck lorry coach taxi
0.0919 tide breeze flood wind rain storm weather wave current heat
0.0865 tube lock tank circuit joint filter battery engine device disk furniture machine mine seal equipment machinery wheel motor slide disc instrument
0.0792 ocean canal stream bath river waters pond pool lake
0.0497 rope hook cable wire thread ring knot belt chain string
0.0469 arrangement policy measure reform proposal project programme scheme plan course
0.0352 week month year
0.0351 couple minute night morning hour time evening afternoon
0.0341 criticism appeal charge application allegation claim objection suggestion case complaint
0.0253 championship open tournament league final round race match competition game contest
0.0218 desire hostility anxiety passion doubt fear curiosity enthusiasm impulse instinct emotion feeling suspicion
0.0183 expenditure cost risk expense emission budget spending
0.0136 competitor rival team club champion star winner squad county player liverpool partner leeds
0.0102 being species sheep animal creature horse baby human fish male lamb bird rabbit female insect cattle mouse monster
...

Note that the more typical (and possibly more cognitively salient) subjects of *run*, such as *humans* or *animals*, are ranked lower than many abstract or metaphorically used arguments (such as a *computer* or a *plan*). This showcases an important problem for selectional preference acquisition, and distributional semantics in general, i.e. the biases that stem from the data source. As we discussed in Lecture 6, the choice of text corpus from which distributions are acquired directly influences what we see in the distributions. Any data source comes with its own set of biases, and in case of text corpora the topic bias and the dialect bias are among the more prominent ones. One reason for this is that we typically write about abstract topics and events, resulting in high coverage of abstract word senses and comparatively lower coverage of the original physical senses. This is further demonstrated by the verb *cut*, which is used predominantly in the domains of economics and finance and its highest-ranked direct objects are *cost* and *price*, as shown in the distribution below.

Selectional preferences of ***cut*** (Dobj)
0.2845 expenditure cost risk expense emission budget spending
0.1527 dividend price rate premium rent rating salary wages
0.0832 employment investment growth supplies sale import export production consumption traffic input spread supply flow
0.0738 potato apple slice food cake meat bread fruit
0.0407 stitch brick metal bone strip cluster coffin stone piece tile fabric rock layer remains block
0.0379 excess deficit inflation unemployment pollution inequality poverty delay discrimination symptom shortage
0.0366 tree crop flower plant root leaf seed rose wood grain stem forest garden
0.0330 tail collar strand skirt trousers hair curtain sleeve
0.0244 rope hook cable wire thread ring knot belt chain string
...

Predicate-argument distributions acquired from text thus tend to be skewed in favour of abstract domains and figurative uses, inadequately reflecting our daily experiences with cutting, which guide human acquisition of meaning. One of the ways in which the NLP community has addressed this problem is by integrating further data sources, such as images, videos and audio, in order to be able to acquire the properties of concepts and the relations between them observed in the physical world. This area of semantics is known as multimodal semantics, and we will discuss it in more detail below.

7.5 Multimodal distributional semantics

Much research in cognitive science and neuroscience suggests that human meaning representations are not merely a product of our linguistic exposure, but are also grounded in our perceptual system and sensori-motor experience (see, for instance, the paper by Barsalou (2008) cited below). This suggests that word meaning and relational knowledge are acquired not only from linguistic input, such as text or speech, but also from other modalities, such as vision, taste, smell, touch and motor activity. Multimodal models of word meaning have thus enjoyed a growing interest in semantics, outperforming purely text-based models in tasks such as semantic similarity estimation, predicting compositionality, and concept categorization. However, the field has so far focused on combining linguistic and visual information, with other modalities receiving little attention.

The first distributional models combining linguistic and visual information are the so-called Bag-of-Visual-Words (BoVW) models. Bag-of-visual-words is an image analysis technique, whose goal to break down the images into discrete parts, known as visual words. This allows us to represent images in a similar manner as we represent text, i.e. in the form of discrete visual contexts, and extract distributional vectors from the images using a similar pipeline. Creating a visual distributional vector for a word would thus involve collecting co-occurrence counts of this word and the visual words in a given corpus of images. The corpora typically used to implement this technique include ImageNet,[32] ESP-game dataset[33] and Yahoo! Webscope Flickr 100M dataset,[34] where images have been manually annotated for the concepts (i.e. words) that they depict.

As in case of the bag-of-words text representation, bag-of-visual-words models also ignore the ordering of visual words in the image, i.e. the structure of the image, and only take into account the presence of a particular visual word in the image. This allows us to represent the image dataset as a collection of discrete visual contexts with fixed dimensionality, and thus map all of our word meaning representations onto the same visual feature space.

Given a corpus of images, BoVW methods first identify local interest points in the images, known as *keypoints*, which are salient image patches that contain rich local information about the image. An example image retrieved for the word *bike* with the keypoints identified is shown in Figure 1.[35] Each keypoint is then represented as a vector of low-level descriptor features, such as color gradients of the surrounding sample regions. Such descriptor features can be obtained using Scale-Invariant Feature Transform (SIFT).[36] The keypoints are then clustered in this descriptor space

[32] `http://image-net.org/`

[33] `http://www.cs.cmu.edu/~biglou/resources/`

[34] `http://yfcc100m.appspot.com/`

[35] Figure source: `http://www-scf.usc.edu/~boqinggo/Feature.htm`

[36] Since SIFT is a computer vision topic, I will not discuss it in detail in this course. Some of you may be familiar with it from a computer vision course, however, I do not expect you to know the specific details of SIFT for the exam. The goal of this lecture is rather to give you an intuition of the kinds of information that can be extracted from the images and how it can be used to enhance semantic representations.

Figure 1: Local interest points (keypoints) in an image retrieved for the word *bike*

using a clustering algorithm, such as k-means, to form visual words. The number of clusters is typically set to a value between 1000 and 5000, which determines the dimensionality of the visual word space. A linguistic concept, such as *dog*, can then be represented as a vector of its co-occurrence counts with each of the visual words in the image corpus, resulting in a visual distributional vector for *dog*.

To build a multimodal representation, we then need to combine the linguistic and visual distributional vectors. There are two common ways of doing this:

- **Feature level fusion.** In feature level fusion, also known as *early* fusion,[37] the linguistic and visual vectors of a word are simply concatenated, resulting in a high-dimensional, merged distributional space. Some approaches perform dimensionality reduction, i.e. map the linguistic and visual features into the same low dimensional space, e.g. using SVD or NMF. The resulting multimodal vectors are then used similarly to the traditional linguistic distributional vectors, that we discussed earlier. For instance, if our task is to estimate similarity between two words (such as *cathedral – church, dog – race, boat – fishing*), then we can compute the cosine distance between the multimodal vectors of these two words. We can also cluster words based on their multimodal representations, by using the multimodal vectors as feature vectors in clustering.

- **Scoring level fusion.** In scoring level fusion or *late* fusion, the scores are first computed independently based on the words' linguistic and visual representations and then later combined, for instance, by taking their mean. In case of the semantic similarity task, we would first compute linguistic and visual similarity scores and then take their weighted mean to obtain the final similarity score.

Multimodal models are typically evaluated in semantic similarity and relatedness tasks. The most popular similarity datasets include MEN[38] and WordSim.[39] In the vast majority of experiments, multimodal models outperform the purely text-based ones (e.g. word windows). However, it should be noted that visual modality so far appears to be more useful in representing the semantics of nouns and adjectives than that of verbs. It is not straightforward to extract high-quality visual features for verbs from static images. The researchers are thus increasingly interested in extracting visual features from videos.

Recent research has shown that convolutional neural networks (CNN), that led to considerable performance gains in computer vision, are equally beneficial in multimodal semantic tasks (see e.g. a recent paper by Kiela et al. (2016)

[37] Sometimes also referred to as *middle* fusion

[38] `http://clic.cimec.unitn.it/~elia.bruni/MEN`

[39] `http://www.cl.cam.ac.uk/users/fh295/simlex.html`

referred to below). CNN-based multimodal models outperform the traditional SIFT feature-based models, such as BoVW. However, a detailed discussion of CNNs is outside the scope of our course.

Some approaches extracted word co-occurrence information from image descriptions, i.e. natural language tags or captions, without performing image processing. The rationale behind these approaches is that the use of natural language descriptions allows to capture predicate-argument relations between e.g. actions and objects depicted in the image. Besides, such "visual" features can be mapped onto the same semantic space as textual features, i.e. words. The drawback is, however, the reliance on manual annotation of the captions and the inability of the model to generalise any information from the image beyond the manually assigned descriptions.

The predicate–argument co-occurrence information extracted from image and video descriptions in such a way can be integrated into the model of selectional preferences, discussed in the previous section. For instance, we can extract verb–noun co-occurrences in the visual dataset and compute the "visual" $P(c)$ and $P(c|v)$, which can be inserted in Resnik's formula. This would allow us to compute the visual preferences of the verbs and compare them to the linguistic ones, that we discussed in the previous section. In order to combine the linguistic and visual information, we can then interpolate the linguistic and visual probabilities and re-compute the verb preferences. The figure below shows the direct object preferences of the verb *cut* computed using the linguistic only, the visual only and the interpolated models.

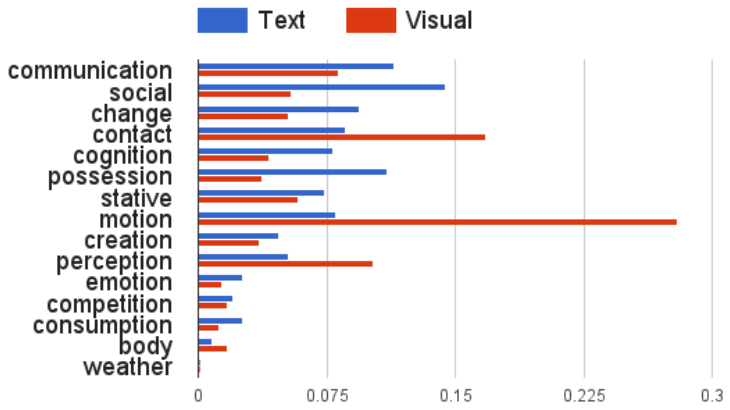
Linguistic (only)
(1) 0.284 expenditure cost risk expense emission budget spending;
(2) 0.152 dividend price rate premium rent rating salary wages;
(3) 0.083 employment investment growth supplies sale import export production [..]

Interpolated
(1) 0.346 expenditure cost risk expense emission budget spending;
(2) 0.211 dividend price rate premium rent rating salary wages;
(3) 0.126 tail collar strand skirt trousers hair curtain sleeve

Visual (only)
(1) 0.224 tail collar strand skirt trousers hair curtain sleeve;
(2) 0.098 expenditure cost risk expense emission budget spending;
(3) 0.090 management delivery maintenance transport service housing

One can see from the figure that the physical argument of *cut* is the dominant one in the output of the visual model. When combined with linguistic information it is ranked third, which is higher than its position in the output of the purely linguistic model (see previous section).

It is important to remember that, like any textual dataset, visual datasets also have their biases. The figure below shows statistics over different verb classes as they are represented in the Yahoo! Webscope Flickr-100M dataset. The verb tags, which were manually assigned to the images in the dataset, were mapped to high-level verb classes in the WordNet hierarchy, such as *communication, motion* etc. This was compared to the distribution of the same verb classes in the British National Corpus (in blue).



While the visual data is dominated by the more physical verbs of motion, perception and contact, the textual data is more representative of the abstract classes of verbs, such as possession, cognition, change and so on. Therefore, when designing a semantic model, it is important to consider which data sources are more suitable for the task, given the nature of the task and the datasets' individual biases.

7.6 Further reading

Philip Resnik. 1997. Selectional preference and sense disambiguation. In *ACL SIGLEX Workshop on Tagging Text with Lexical Semantics,* Washington, D.C.

Lawrence W. Barsalou. 2008. Grounded cognition. *Annual Review of Psychology*, 59(1):617–645.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Douwe Kiela, Anita Vero and Stephen Clark. Comparing Data Sources and Architectures for Deep Visual Representation Learning in Semantics. In *Proceedings of EMNLP 2016*, Austin, TX.

8 Lecture 8: Compositional semantics

Compositional semantics is the study of how meaning is conveyed by the structure of a sentence (as opposed to lexical semantics, which is primarily about word meaning). In lecture 4, we've looked at grammars primarily as a way to describe a language: i.e., to say which strings are part of the language and which are not, or (equivalently) as devices that, in principle, could generate all the strings of a language. However, what we usually want for language analysis is some idea of the meaning of a sentence. At its most basic, this is 'who did what to whom?' but clearly there is much more information that is implied by the structure of most sentences. The parse trees we saw in lecture 4 have some of this information, but it is implicit rather than explicit. In the simple examples covered by those grammars, syntax and semantics are very closely related, but if we look at more complex examples, this is not always the case.

Consider the following examples:

(10) a Kitty chased Rover.
b Rover was chased by Kitty.

The meaning these two sentences convey is essentially the same (what differs is the emphasis) but the parse trees are quite different.[40] A possible logical representation would be chase$'(k, r)$, assuming that k and r are constants corresponding to *Kitty* and *Rover* and chase$'$ is the two place predicate corresponding to the verb *chase*.[41] Note the convention that a predicate corresponding to a lexeme is written using the stem of the lexeme followed by $'$: chase$'$. A logical meaning representation constructed for a sentence is called the *logical form* of the sentence. Here and in what follows I am ignoring tense for simplicity although this is an important aspect of sentence meaning.

Another relatively straightforward example that shows a syntax/semantics mismatch is *pleonastic pronouns*: i.e., pronouns that do not refer to actual entities. See the examples below (with indicative logical forms).

(11) a It barked.
b $\exists x[\text{bark}'(x) \wedge \text{PRON}(x)]$

(12) a It rained.
b rain$'$

In *it rains*, the *it* does not refer to a real entity (it will not be resolved, see §9.8), so the semantics should not involve any representation for the *it*.

More complex examples include verbs like *seem*: for instance *Kim seems to sleep* means much the same thing as *it seems that Kim sleeps* (contrast this with the behaviour of *believe*). An indicative representation for both would be seem$'$(sleep$'(k)$) — but note that there is no straightforward way of representing this in a first order logic.

There are many more examples of this sort that make the syntax/semantics interface much more complex than it first appears and demonstrate that we cannot simply read the compositional semantics off a parse tree or other syntactic representation.

Grammars that produce explicit representations for compositional semantics are often referred to as *deep grammars*. Deep grammars that do not overgenerate (much) are said to be *bidirectional*. This means they can be used in the realization step in a Natural Language Generation system to produce text from an input logical form (see lecture 10). This generally requires somewhat different algorithms from parsing (although *chart generation* is a variant of parsing), but this will not be discussed in this course.

In this lecture, I will start off by showing how simple logical representations can be produced from CFGs and then discuss how they can be integrated with ideas from distributional semantics.

8.1 Compositional semantics using lambda calculus

The assumption behind compositional semantics is that the meaning of each whole phrase must relate to the meaning of its parts. For instance, to supply a meaning for the phrase *chased Rover*, we need to combine a meaning for *chased*

[40] Very broadly speaking, the second sentence is more appropriate if Rover is the topic of the discourse: we'll very briefly return to this issue in lectures 9 and 10.

[41] Although introductory logic books invariably assume proper names correspond to constants, this does not work well for a broad coverage system. Another option is: $\exists x, y[\text{chase}'(x, y) \wedge \text{Kitty}'(x) \wedge \text{Rover}'(y)]$.

with a meaning for *Rover* in some way.

To enforce a notion of compositionality, we can require that each syntactic rule in a grammar has a corresponding semantic rule which shows how the meaning of the daughters is combined. In linguistics, this is usually done using lambda calculus, following the work of Montague. The notion of lambda expression should be familiar from previous courses (e.g., Computation Theory, Discrete Maths). Informally, lambda calculus gives us a logical notation to express the argument requirements of predicates. For instance, we can represent the fact that a predicate like bark′ is 'looking for' a single argument by:

$$\lambda x[\text{bark}'(x)]$$

Syntactically, the lambda behaves like a quantifier in FOPC: the *lambda variable* x is said to be within the scope of the *lambda operator* in the same way that a variable is syntactically within the scope of a quantifier. But *lambda expressions* correspond to functions, not propositions. The lambda variable indicates a variable that will be bound by function application. Applying a lambda expression to a term will yield a new term, with the lambda variable replaced by the term. For instance, to build the semantics for the phrase *Kitty barks* we can apply the semantics for *barks* to the semantics for *Kitty*:

$$\lambda x[\text{bark}'(x)](k) = \text{bark}'(k)$$

Replacement of the lambda variable is known as *lambda-conversion*. If the lambda variable is repeated, both instances are instantiated: $\lambda x[\text{bark}'(x) \wedge \text{sleep}'(x)]$ denotes the set of things that bark and sleep

$$\lambda x[\text{bark}'(x) \wedge \text{sleep}'(x)](r) = \text{bark}'(r) \wedge \text{sleep}'(r)$$

A partially instantiated transitive verb predicate has one uninstantiated variable, as does an intransitive verb: e.g., $\lambda x[\text{chase}'(x, r)]$ — the set of things that chase Rover.

$$\lambda x[\text{chase}'(x, r)](k) = \text{chase}'(k, r)$$

Lambdas can be nested: this lets us represent transitive verbs so that they apply to only one argument at once. For instance: $\lambda x[\lambda y[\text{chase}'(y, x)]]$ (which is often written $\lambda x \lambda y[\text{chase}'(y, x)]$ where there can be no confusion). For instance:

$$\lambda x[\lambda y[\text{chase}'(y, x)]](r) = \lambda y[\text{chase}'(y, r)]$$

That is, applying the semantics of *chase* to the semantics of *Rover* gives us a lambda expression equivalent to the set of things that chase Rover.

The following example illustrates that bracketing shows the order of application in the conventional way:

$$(\lambda x[\lambda y[\text{chase}'(y, x)]](r))(k) = \lambda y[\text{chase}'(y, r)](k) = \text{chase}'(k, r)$$

In other words, we work out the value of the bracketed expression first (the innermost bracketed expression if there is more than one), and then apply the result, and so on until we're finished.

A grammar fragment In this fragment, I'm using X' to indicate the semantics of the constituent X (e.g. NP′ means the semantics of the NP), where the semantics may correspond to a function: e.g., VP′(NP′) means the application of the semantics of the VP to the semantics of the NP. The numbers are not really part of the CFG — they are just there to identify different constituents.

```
S -> NP VP
```
VP′(NP′)
```
VP -> Vditrans NP1 NP2
```
(Vditrans′(NP1′))(NP2′)
```
VP -> Vtrans NP
```
Vtrans′(NP′)
```
VP -> Vintrans
```
Vintrans′
```
Vditrans -> gives
```
$\lambda x \lambda y \lambda z[\text{give}'(z, y, x)]$

`Vtrans -> chases`
$\lambda x \lambda y[\text{chase}'(y, x)]$
`Vintrans -> barks`
$\lambda z[\text{bark}'(z)]$
`Vintrans -> sleeps`
$\lambda w[\text{sleep}'(w)]$
`NP -> Kitty`
k
`NP -> Lynx`
l
`NP -> Rover`
r

The post-lecture exercises ask you to work through some examples using this fragment.

8.2 Logical representations in broad coverage grammars

It is possible to extend this style of compositional semantics so that some sort of representation is produced for all sentences of a language covered by a broad coverage grammar. However, the extent to which it is possible to do this within first order predicate calculus (FOPC) is a matter of debate, and even the most optimistic researcher would not claim that the representations currently being produced are fully adequate to capture the meaning of the expressions.

In some cases, a FOPC representation is possible by careful choice of representation. For example, to represent an adverbial modifier such as *loudly*, rather than make the adverbial take the verb as an argument (e.g., $\text{loud}'(\text{bark}'(r))$), it is common to introduce an extra variable for all verbs to represent the event.

(13) a Rover barked.
b $\exists e[\text{bark}'(e, r)]$

(14) a Rover barked loudly.
b $\exists e[\text{bark}'(e, r) \wedge \text{loud}'(e)]$

This can be roughly glossed as: 'there was a barking event, Rover did the barking, and the barking was loud'. The events are said to be *reified*: literally 'made into things'.

Another issue is that FOPC forces quantifiers to be in a particular scopal relationship, and this information is not (generally) overt in NL sentences.

Every dog chased a cat.

is ambiguous between:

$$\forall x[\text{dog}'(x) \implies \exists y[\text{cat}'(y) \wedge \text{chase}'(x, y)]]$$

and the less-likely, 'one specific cat' reading:

$$\exists y[\text{cat}'(y) \wedge \forall x[\text{dog}'(x) \implies \text{chase}'(x, y)]]$$

Some current NLP systems construct an underspecified representation which is neutral between these readings, if they represent quantifier scope at all. There are several different alternative formalisms for underspecification.

There are a range of natural language examples which FOPC cannot handle, however. FOPC only has two quantifiers (roughly corresponding to English *some* and *every*), but it is possible to represent some other English quantifiers in FOPC, though representing numbers, for instance, is cumbersome. However, the quantifier *most* cannot be represented in a first order logic. The use of event variables does not help with the representation of adverbs such as *perhaps* or *maybe*, and there are also problems with the representation of modal verbs, like *can*. In these cases, there are known logics which can provide a suitable formalization, but the inference problem may not be tractable and the different logics proposed are not necessarily compatible.

There are other cases where the correct formalization is quite unclear. Consider sentences involving *bare plural* subjects, such as:

(15) Dogs are mammals.

(16) Birds fly.

(17) Ducks lay eggs.

(18) Voters rejected AV in 2011.

What quantifier could you use to capture the meaning of each of these examples? In some cases, researchers have argued that the correct interpretation must involve non-monotonicity or defaults: e.g., birds fly unless they are penguins, ostriches, baby birds . . . But no fully satisfactory solution has been developed along these lines.

Another issue is that not all phrases have meanings which can be determined compositionally. For instance, *puce sphere* is compositional: it refers to something which is both puce and a sphere. In contrast, *red tape* may be compositional (as in 19), but may also refer to bureaucracy (as in 20), in which case it is a non-compositional *multiword expression* (MWE).

(19) She tied red tape round the parcel.

(20) The deputy head of department was dismayed at the endless red tape.

Clear cases of MWEs, like *red tape*, can be accounted for explicitly in the grammar (so *red tape* is treated as being ambiguous between the compositional reading and the MWE), but the problem is that many phrases are somewhere in between being an MWE and being fully compositional. Consider the phrase *real pleasure* (as in *it was a real pleasure to work with you*): this isn't non-compositional enough to be listed in most dictionaries, but is still a conventional expression particularly appropriate in certain social contexts. It appears that something else is necessary besides compositional semantics for such cases. Compositional distributional approaches, which I discuss below, in principle have the machinery to model graded compositionality. However, modelling idiomaticity in this framework is still an issue that requires further research.

Finally, although there is an extensive linguistic literature on formal semantics which describes logical representations for different constructions, the coverage is still very incomplete, even for well-studied languages such as English. Furthermore, the analyses which are suggested are often incompatible with the goals of broad-coverage parsing, which requires that ambiguity be avoided as much as possible. Overall, there are many unresolved puzzles in deciding on logical representations, so even approaches which do use logical representations are actually using some form of approximate meaning representation. Under many circumstances, it is better to see the logical representation as an annotation that captures some of the meaning, rather than a complete representation.

8.3 Compositional distributional semantics

Given the success of distributional semantics in modeling the meaning of words, it is natural to ask whether this approach can be extended to account for the meaning of phrases and sentences. Language can, however, have an infinite number of sentences, given a limited vocabulary. This suggests that it is not even in principle possible to learn distributional vectors for all phrases and sentences, since even a very large corpus is always finite. Therefore, to account for their meaning, we need to be able to do composition in a distributional space.

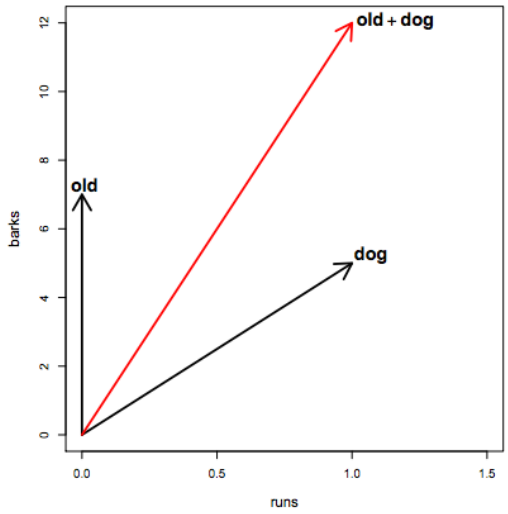
This area of research is known as compositional distributional semantics. The central insight of compositional distributional semantics is to model the composition of words as algebraic operations on their vector representations, as provided by a conventional distributional model. Popular approaches to performing such composition include *vector mixture models* and *lexical function models*, which we will look at in more detail below.

Vector mixture models In their influential paper *Composition in Distributional Models of Semantics*, Mitchell and Lapata (2010) proposed two kinds of composition models: additive and multiplicative. Within the additive paradigm, each word is represented by its distributional vector, and the semantics of a phrase is computed by summing up the vectors of the words (i.e. summing up the values in each of their components). In case of the multiplicative model, the

values are multiplied rather than added. A toy example of vector addition and multiplication for the phrases *old dog* and *old cat* is shown below.[42]

				additive		multiplicative	
	dog	**cat**	**old**	**old + dog**	**old + cat**	**old ⊙ dog**	**old ⊙ cat**
runs	1	4	0	1	4	0	0
barks	5	0	7	12	7	35	0

The resulting additive vector for the phrase *old dog* would look as follows:



Mitchell and Lapata argue that additive and multiplicative models each address a different kind of phenomena. The components of additive vectors inherit the cumulative value from the corresponding components of the word vectors. So if a word vector has a high value in a component, the same high value will appear in the composed vector, even if the same component was low or 0 in the vector of the other word. For example, **old + cat** inherits a relatively high *barks* score from **old**. In contract, multiplication captures the interaction between the values in the components of the word vectors. For instance, since cat has a 0 *barks* value, **old** ⊙ **cat** has 0 for this component irrespective of the *barks* value in the vector of **old**. When the vectors of both words have high values in a given component, the composed vector will get a very high value out of their product, as illustrated for the second **old** ⊙ **dog** component in the table above (Baroni et al., 2014). Mitchell and Lapata suggest that these interaction properties of the multiplicative model can be thought of as a form of "feature intersection".

Both of these models are symmetric, in the sense that the vector of the noun and that of the adjective contribute the the composed vector equally. However, semantic composition is rarely symmetric and the head word in a phrase often contributes more important information to its meaning than the dependent word. For instance, an *old dog* is in the first place a dog. Mitchell and Lapata addressed this by proposing a *weighted additive model*, where the contribution of the vector components of the head word may receive a higher weight.

Mitchell and Lapata evaluated their models in the task of predicting human judgements of phrase similarity for adjective-noun, noun-noun and verb-noun pairs. They have shown that the weighted additive and the multiplicative models exhibit correlation with human judgements. However, it should be noted that these kinds of models do have some fundamental limitations. Firstly, addition and multiplication are both commutative operations, and hence the models do not account for word order, e.g.the vectors produced for *John hit the ball* and *The ball hit John* would be identical. In addition, vector mixture models are more suitable for modelling content words, but are unlikely to port well to function words to model the meaning of phrases such as *some dogs; lice and dogs; lice on dogs*. This suggests

[42]This and the below figures are taken from Marco Baroni, Raffaella Bernardi and Roberto Zamparelli. 2014. Frege in space: A program for compositional distributional semantics. In the special issue on *Perspectives on Semantic Representations for Textual Inference of LiLT*. Volume 9, pp 241-346, on which this section is based.

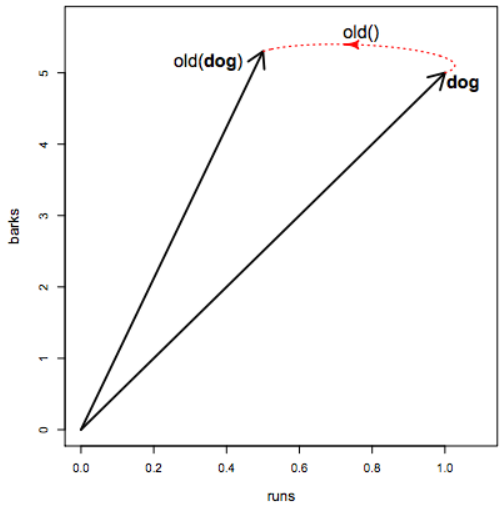
that these models can only account for simple phrases and are not suitable to model the meaning of real sentences. An alternative that takes the above problems into account is lexical function models, which I discuss below.

Lexical function models Lexical function models adopt the view from formal semantics that composition can be implemented as function application. This class of models distinguish between words whose meaning is directly determined by their distributional behaviour, e.g. nouns, and words that act as *distributional functions* transforming the distributional profile of other words, e.g., verbs, adjectives and prepositions. In these models, nouns, noun phrases and sentences are represented as vectors, adjectives as matrices that act on the noun or noun phrase vectors, and transitive verbs as third-order tensors that act on noun or noun phrase vectors. The meaning of a phrase is then derived by composing these lexical representations.

We will consider the case of adjective representation as lexical functions and their application to derive the meaning of adjective-noun phrases. Baroni and Zamparelli (2010) in their paper *Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space* were the first to to represent adjectives as functions that act on noun vectors, e.g. $old\ dog = old(dog)$. Such functions on vectors are known in linear algebra as *linear transformations* or *linear maps*. Each adjective is represented as a linear transformation from a set of noun vectors to a set of vectors of the respective adjective-noun phrases. In practice, this means that the adjective is represented in a form of a matrix of parameters, e.g. $\mathbf{A}_{old}$, $\mathbf{A}_{furry}$, etc. Nouns are represented by their traditional distributional vectors (e.g. **house, dog**). Composition is then defined simply as the product of the adjective matrix and the noun vector:

$$\mathbf{old\ dog} = \mathbf{A}_{old} \times \mathbf{dog}. \tag{21}$$

Applying the transformation $\mathbf{A}_{old}$ to the vector of **dog** changes its direction to yield a representation of the phase *old dog*, as shown in the figure below:



The matrices are learned from corpus data using regression techniques. Regression is a class of machine learning methods whose goal is to predict a continuous numerical value from a set of features. This contrasts with classification, whose goal is to assign a discrete class to an instance given a feature vector. In our case, we need to learn a set of parameters (elements in our adjective matrix) that would allow us to predict the values of the components of the output vector for adjective-noun phrases. We train the model for a given adjective (e.g. *old*) using a set of distributional vectors for the adjective–noun phrases observed in the corpus (e.g. *old dog, old house, old cat, old toy* etc.) and a set of vectors of the respective nouns (e.g. *dog, house, cat, toy* etc.) as input. The goal is to learn a set of parameters that would allow us to construct accurate predictions for vector components of the unseen phrases, e.g. *old elephant, old mercedes*.

In summary, we need to follow these steps to learn adjective matrices from corpus data:

1. Obtain a distributional vector $\mathbf{n}_j$ for each noun n_j in the lexicon.

2. Collect adjective noun pairs (a_i, n_j) from the corpus.
3. Obtain a distributional vector $\mathbf{p}_{ij}$ of each bi-gram (a_i, n_j) from the same corpus using a conventional DSM.
4. The set of tuples $\{(\mathbf{n}_j, \mathbf{p}_{ij})\}_j$ then represents a dataset $\mathcal{D}(a_i)$ for the adjective a_i.
5. Learn matrix $\mathbf{A}_i$ from $\mathcal{D}(a_i)$ using linear regression.

In other words, given a data set of noun and phrase vectors $\mathcal{D}(a_i) = \{(\mathbf{n}_j, \mathbf{p}_{ij})\}_{j=1}^{N}$ for adjective-noun phrases involving adjective a_i (extracted using a conventional DSM), our goal is to learn a linear transformation $\mathbf{A}_i$ between them. This can be treated as an optimization problem, of learning an estimate $\hat{\mathbf{A}}_i$ that minimizes a specified loss function. The loss function typically used is the squared error loss, defined as follows:

$$L(\mathbf{A}_i) = \sum_{j \in \mathcal{D}(a_i)} \|\mathbf{p}_{ij} - \mathbf{A}_i \mathbf{n}_j\|^2 \tag{22}$$

And the optimal solution can be found using ordinary least-squares regression, which has been shown to perform well in this task.

The table below demonstrates the composition process for the phrase *old dog* on the toy example that we have already looked at above.

OLD	runs	barks
runs	0.5	0
barks	0.3	1

$\times$

	dog
runs	1
barks	5

$=$

I	**OLD(dog)**
runs	$(0.5 \times 1) + (0 \times 5) = 0.5$
barks	$(0.3 \times 1) + (5 \times 1) = 5.3$

In essence, the adjective matrix encodes the interaction of different features. The labels of the rows and columns in the matrix indicate the role played by each matrix element in mapping from the noun to the noun phrase vector. For example, the first element of the second row in the toy OLD matrix indicates that the *runs*-labeled component of the noun vector will contribute 30% of its value to the *barks*-labeled component of the resulting noun phrase.

Some adjectives, such as *old*, may have a smaller effect on the modified noun than others. For instance, it is unlikely that *old* changes the direction of the noun vector dramatically — an *old dog* still barks, eats, runs and so on, it is still a dog in every practical sense. This will be reflected in a matrix that has values close to 1 on the diagonal and values of other elements close to 0, reflecting little "interference" from other features. In contrast, an adjective such as *dead* that alters the meaning of the noun it modifies more radically could have 0 or even negative values on the diagonal and large negative or positive values of many non-diagonal elements, reflecting the stronger effect it has on the noun (Baroni et al., 2014). See the paper by Baroni et al. (listed below) for a more in-depth interpretation of lexical functions.

As I have already mentioned above, verbs can also be modelled as linear transformations in this framework, with transitive verbs typically represented in the form of *third-order tensors*. A tensor is a general term referring to an object in a vector space. A vector is an example of a first-order tensor (with the dimensionality of N) and a matrix is a second-order tensor (with the dimensionality of $N \times M$). The term tensor is typically used to refer to higher-order tensors, starting with third-order tensors with the dimensionality of $N \times M \times K$. Representing a verb as a third-order tensor allows us to model its relationship with its subject and object simultaneously. The subject and object are noun phrases, and hence they are represented as vectors. Composing a third-order tensor with a vector for the object yields a matrix. This matrix can then be composed with the vector for the subject, which results in a vector for the phrase (or sentence). This also demonstrates the way in which lexical function models follow the rules of syntactic composition.

Lexical function models are, however, generally applied to short phrases or particular types of composition (e.g. adjective noun phrases, noun compounds etc.) independently. They have been evaluated in semantic similarity tasks, as well as adjective clustering and judging semantic plausibility of short phrases, where they beat vector-based methods. These models have also been applied in morphology to learn composition of morphemes: e.g. computing the function $f(f(shame, less), ness)$ to model the meaning of *shamelessness*.

Handling polysemy The vast majority of compositional distributional models build a single representation for all senses of a word, collapsing distinct senses together. Several researchers argue that terms with ambiguous senses can be handled by such models without any recourse to additional disambiguation steps, as long as contextual information is available. Baroni and colleagues suggest that the models might largely avoid problems handling adjectives with multiple senses because the matrices for adjectives implicitly incorporate contextual information. However, they do draw a distinction between two ways in which the meaning of a term can vary. Continuous polysemy — the subtle and continuous variations in meaning resulting from the different contexts in which a word appears — is relatively tractable, in their opinion. This contrasts with discrete homonymy — the association of a single term with completely independent meanings, as we discussed in lecture 5. Baroni et al. admit that homonymy is more difficult to handle in compositional distributional models. Unfortunately, they do not propose a way to automatically determine whether any given variation in meaning is polysemy or homonymy, and offer no account of regular polysemy (i.e., metaphor and metonymy) or whether it would pose similar problems as homonymy for this type of models. However, one study (by Kartsaklis and Sadrzadeh (2013) referred to below) used a clustering method to disambiguate the senses of verbs, and then trained separate functions (tensors) for each sense. They found that prior disambiguation resulted in semantic similarity measures that correlated more closely with human judgments. However, this is an area that requires further research to draw definitive conclusions.

8.4 Inference

There are two distinct ways of thinking about reasoning and inference in language processing. The first can be thought of as inference on an explicit formally-represented knowledge base, while the second is essentially language-based. It is potentially possible to use theorem provers with either approach, although relatively few researchers currently do this: it is more common to use shallower, more robust, techniques.

Inference on a knowledge base This approach assumes that there is an explicit underlying knowledge base, which might be represented in FOPC or some other formal language. For instance, we might have a set of axioms such as:

$C(k, b)$
$C(r, b)$
$H(r)$
$U(k)$
$\forall x[C(k, x) \implies H(x)]$

There is also a link between the natural language terms appropriate for the domain covered and the constants and predicates in the knowledge base: e.g. *chase* might correspond to C, *happy* to H, *unhappy* to U, *Bernard* to b and so on. The approaches to compositional semantics discussed in sections 8.1 and 8.2 are one way to do this. Under these assumptions, a natural language question can be converted to an expression using the relevant predicates and answered either by direct match to the knowledge base or via inference using the meaning representation in the knowledge base. For instance *Is Bernard happy?* corresponds to querying the knowledge base with $H(b)$. As mentioned in Lecture 1, the properties of such a knowledge base can be exploited to reduce ambiguity. This is, of course, a trivial example: the complexity of the system depends on the complexity of the domain (and the way in which out-of-domain queries are handled) but in all cases the interpretation is controlled by the formal model of the domain.

Under these assumptions, the valid inferences are given by the knowledge base: language is just seen as a way of accessing that knowledge. This approach relies on the mapping being adequate for any way that the user might choose to phrase a question (e.g., *Is Kitty sad?* should be interpreted in the same way as *Is Kitty unhappy?*) and acquiring such mappings is an issue, although machine learning methods can be used, given the right sort of training data. But the most important difficulty is the limitations of the knowledge base representation itself. This sort of system works very well for cases where there are clear boundaries to the domain and the reasoning is tractable, which is the case for most types of spoken dialogue system. It may even turn out to be possible to approach mathematical knowledge in this way, or at least some areas of mathematics. However, although some researchers in the 1970s and 80s thought that it would be possible to extend this type of approach to common sense knowledge, few people would now seriously advocate this. Many researchers would also question whether it is helpful to think of human reasoning as operating in this way: i.e., as translating natural language into a symbolic mental representation.

Language-based inference The alternative way of thinking about inference is purely in terms of language: e.g., deciding whether one natural language statement follows from another. For instance,

> Philadelphia is to the east of Pittsburgh.
> Pittsburgh is to the west of Philadelphia.

The difference from the approach above is that the inference is entirely expressed in natural language and its validity is determined by human intuition, rather than validity in any particular logic. We may use logic as a way of helping us model these inferences correctly, but the basic notion of correctness is the human judgement, not logical correctness. If an explicit meaning representation is used, it is best seen as an annotation of the natural language that captures some of the meaning, not as a complete replacement for the text. Such language-based reasoning is not tied to a particular knowledge base or domain. With such an approach we can proceed without having a perfect meaning representation for a natural language statement, as long as we are prepared for the possibility that we will make some incorrect decisions. Since humans are not always right, this is acceptable in principle.

The Recognising Textual Entailment task, discussed below, is a clear example of this methodology, but implicitly or explicitly this is the most common approach in current NLP in general. It can be seen as underlying a number of areas of research, including question answering and semantic search, and is relevant to many others, including summarization. A major limitation, at least in the vast majority of the research so far, is that the inferences are made out of context. There will eventually be a requirement to include some form of modelling of the entities referred to, to represent anaphora for instance, although again this is probably best thought of as an annotation of the natural language text, rather than an entirely distinct domain model.

The distinction between these two approaches can be seen as fundamental in terms of the philosophy of meaning. However NLP research can and does combine the approaches. For example, a conversational agent will use an explicit domain model for some types of interaction and rely on language-based inference for others. While logic and compositional semantics is traditionally associated with the first approach, it can also be useful in the second.

8.5 Recognising Textual Entailment

A series of shared tasks concerning *Recognising Textual Entailment*, the RTE challenge, were carried out starting in 2004 (Dagan et al, 2005). The data consists of a series of texts, generally single sentences presented without context (labelled T below), and some hypothesis (H) which may or may not follow from that text.

(23) T: The girl was found in Drummondville earlier this month.
H: The girl was discovered in Drummondville.

The task is to label the pairs as TRUE (when the entailment follows) or FALSE (when it doesn't follow, rather than when it's known to be an untrue statement) in a way which matches human judgements. The example above was labelled TRUE by the human annotator.

Examples of this sort can be dealt with using a logical form generated from a grammar with compositional semantics combined with inference. To show this in detail would require a lot more discussion of semantics than we had space for above, but I'll give a sketch of the approach here. Assume that the T sentence has logical form T′ and the H sentence has logical form H′. Then if T′ $\implies$ H′ we conclude TRUE, and otherwise we conclude FALSE. For example, the logical form for the T sentence above is approximately as shown below:

(24) T The girl was found in Drummondville earlier this month.
T′ $\exists x, u, e[\text{girl}'(x) \wedge \text{find}'(e, u, x) \wedge \text{in}'(e, \text{Drummondville}) \wedge \text{earlier-this-month}'(e)]$

The verb has the additional argument, e, corresponding to the event as discussed in §8.2 above. So the semantics can be glossed as: 'there was a finding event, the entity found was a girl, the finding event was in Drummondville, and the finding event happened earlier this month'. (The real representation would use a combination of predicates instead of earlier-this-month′, but that's not important for the current discussion.)

The hypothesis text is then:

(25) H The girl was discovered in Drummondville.

H′ $\exists x, u, e[\text{girl}'(x) \wedge \text{discover}'(e, u, x) \wedge \text{in}'(e, \text{Drummondville})]$

$A \wedge B \implies A$, which licences dropping *earlier this month*, so assuming a meaning postulate:

$$[\text{find}'(x, y, z) \implies \text{discover}'(x, y, z)]$$

the inference T′ $\implies$ H′ would go through.

An alternative technique is based on matching dependency structures. Instead of an explicit meaning postulate, a similarity metric can be used to relate *find* and *discover*. The similarity could be extracted from a lexical resource such as WordNet, or from a corpus using distributional methods.

More robust methods can be used which do not require parsing of any type. The crudest technique is to use a bag-of-words method, analogous to that discussed for sentiment detection in lecture 1: if there is a large enough overlap between the words in T and H, the entailment goes though. Note that whether such a method works well or not crucially depends on the H texts: it would trivially be fooled by hypotheses like:

(26) H: The girl was discovered by Drummondville.

The RTE test set was actually deliberately constructed in such a way that word overlap works quite well.

Further examples (all discussed by Bos and Markert, 2005):

(27) T: Clinton's book is not a big seller here.

H: Clinton's book is a big seller.

a FALSE

(28) T: After the war the city was briefly occupied by the Allies and then was returned to the Dutch.

H: After the war, the city was returned to the Dutch.

a TRUE

(29) T: Four Venezuelan firefighters who were traveling to a training course in Texas were killed when their sport utility vehicle drifted onto the shoulder of a highway and struck a parked truck.

H: Four firefighters were killed in a car accident.

a TRUE

(30) T: Lyon is actually the gastronomic capital of France.

H: Lyon is the capital of France.

FALSE

(31) T: US presence puts Qatar in a delicate spot.

H: Qatar is located in a delicate spot.

a FALSE

(32) T: The first settlements on the site of Jakarta were established at the mouth of the Ciliwung, perhaps as early as the 5th century AD.

H: The first settlements on the site of Jakarta were established as early as the 5th century AD.

a TRUE (sic)

The post-lecture exercises suggest that you try and work out how a logical approach might handle (or fail to handle) these examples.

8.6 Further reading

J&M go into quite a lot of detail about formal compositional semantics including underspecification.

The compositional distributional semantics section is based on the paper by Baroni and colleagues:

Marco Baroni, Raffaella Bernardi and Roberto Zamparelli. 2014. Frege in space: A program for compositional distributional semantics. In *Linguistic Issues in Language Technology*, special issue on Perspectives on Semantic Representations for Textual Inference. Volume 9, pp 241–346.

I highly recommend reading the whole paper, where the authors provide both a review of compositional models and an in-depth analysis of the mathematical representations they produce and what kinds of linguistic intuitions they capture.

Other papers referred to in the lecture include:

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013b. Separating disambiguation from composition in distributional semantics. In *Proceedings of the 2013 Conference on Computational Natural Language Learning*, pages 114–123.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science* 34(8):1388–1429.

Bos, Johan and Katja Markert, 2005. Recognising textual entailment with logical inference techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2005)*. Vancouver, Canada.

9 Lecture 9: Discourse processing

The techniques we have seen in lectures 2–8 relate to the interpretation of words and individual sentences, but utterances are always understood in a particular context. Context-dependent situations include:

1. Referring expressions: pronouns, definite expressions etc.
2. Universe of discourse: *every dog barked*, doesn't mean every dog in the world but only every dog in some explicit or implicit contextual set.
3. Responses to questions, etc: only make sense in a context: *Who came to the party? Not Sandy.*
4. Implicit relationships between events: *Max fell. John pushed him* — the second sentence is (usually) understood as providing a causal explanation.

In the first part of this lecture, I give a brief overview of *rhetorical relations* which can be seen as structuring text at a level above the sentence. I'll then go on to talk about one particular case of context-dependent interpretation — anaphor resolution.

9.1 Rhetorical relations and coherence

Consider the following discourse:

(33) Max fell. John pushed him.

This discourse can be interpreted in at least two ways:

(34) Max fell because John pushed him.

(35) Max fell and then John pushed him.

This is yet another form of ambiguity: there are two different interpretations for (33) but there is no syntactic or semantic ambiguity in the interpretation of the two individual sentences in it. There seems to be an implicit relationship between the two sentences in (33): a *discourse relation* or *rhetorical relation*. (I will use the terms interchangeably here, though different theories use different terminology, and rhetorical relation tends to refer to a more surfacy concept than discourse relation.) In (34) the link between the second and first part of the sentence is explicitly an explanation, while (35) is an explicit narration: *because* and *and then* are said to be *cue phrases*. Theories of discourse/rhetorical relations try to reify this intuition using link types such as *Explanation* and *Narration*.

9.2 Coherence

Discourses have to have connectivity to be coherent:

(36) Kim got into her car. Sandy likes apples.

Both of these sentences make perfect sense in isolation, but taken together they are incoherent. Adding context can restore coherence:

(37) Kim got into her car. Sandy likes apples, so Kim thought she'd go to the farm shop and see if she could get some.

The second sentence can be interpreted as an explanation of the first. In many cases, this will also work if the context is known, even if it isn't expressed.

Language generation requires a way of implementing coherence. For example, consider a system that reports share prices. This might generate:

In trading yesterday: Dell was up 4.2%, Safeway was down 3.2%, HP was up 3.1%.

This is much less acceptable than a connected discourse:

> Computer manufacturers gained in trading yesterday: Dell was up 4.2% and HP was up 3.1%. But retail stocks suffered: Safeway was down 3.2%.

Here *but* indicates a Contrast. Not much actual information has been added (assuming we know what sort of company Dell, HP and Safeway are), but the discourse is easier to follow.

Discourse coherence assumptions can affect interpretation:

> John likes Bill. He gave him an expensive Christmas present.

If we interpret this as Explanation, then 'he' is most likely Bill. But if it is Justification (i.e., the speaker is providing evidence to justify the first sentence), then 'he' is John.

9.3 Factors influencing discourse interpretation

1. Cue phrases. These are sometimes unambiguous, but not usually. e.g. *and* is a cue phrase when used in sentential or VP conjunction.
2. Punctuation (or the way the sentence is said — intonation etc) and text structure. For instance, parenthetical information cannot be related to a main clause by Narration (it is generally Explanation), but a list is often interpreted as Narration:

   > Max fell (John pushed him) and Kim laughed.
   > Max fell, John pushed him and Kim laughed.

   Similarly, enumerated lists can indicate a form of narration.
3. Real world content:

   > Max fell. John pushed him as he lay on the ground.

4. Tense and aspect.

   > Max fell. John had pushed him.
   > Max was falling. John pushed him.

It should be clear that it is potentially very hard to identify rhetorical relations. In fact, recent research that simply uses cue phrases and punctuation is quite promising. This can be done by hand-coding a series of finite-state patterns, or by supervised learning.

9.4 Discourse structure and summarization

If we consider a discourse relation as a relationship between two phrases, we get a binary branching tree structure for the discourse. In many relationships, such as Explanation, one phrase depends on the other: e.g., the phrase being explained is the main one and the other is subsidiary. In fact we can get rid of the subsidiary phrases and still have a reasonably coherent discourse. (The main phrase is sometimes called the *nucleus* and the subsidiary one is the *satellite*.) This can be exploited in summarization.

For instance, suppose we remove the satellites in the first three sentences of this subsection:

> We get a binary branching tree structure for the discourse. In many relationships one phrase depends on the other. In fact we can get rid of the subsidiary phrases and still have a reasonably coherent discourse.

Other relationships, such as Narration, give equal weight to both elements, so don't give any clues for summarization.

Rather than trying to find rhetorical relations for arbitrary text, genre-specific cues can be exploited, for instance for scientific texts. This allows more detailed summaries to be constructed.

9.5 Referring expressions

I'll now move on to talking about another form of discourse structure, specifically the link between referring expressions. The following example will be used to illustrate referring expressions and anaphora resolution:

> Niall Ferguson is prolific, well-paid and a snappy dresser. Stephen Moss hated him — at least until he spent an hour being charmed in the historian's Oxford study. (quote taken from the Guardian)

Some terminology:

referent a real world entity that some piece of text (or speech) refers to. e.g., the two people who are mentioned in this quote.

referring expressions bits of language used to perform reference by a speaker. In, the paragraph above, *Niall Ferguson*, *him* and *the historian* are all being used to refer to the same person (they *corefer*).

antecedent the text initially evoking a referent. *Niall Ferguson* is the antecedent of *him* and *the historian*

anaphora the phenomenon of referring to an antecedent: *him* and *the historian* are *anaphoric* because they refer to a previously introduced entity.

What about *a snappy dresser*? Traditionally, this would be described as predicative: that is, it is a property of some entity (similar to adjectival behaviour) rather than being a referring expression itself.

Generally, entities are introduced in a discourse (technically, *evoked*) by indefinite noun phrases or proper names. Demonstratives (e.g., *this*) and pronouns are generally anaphoric. Definite noun phrases are often anaphoric (as above), but often used to bring a mutually known and uniquely identifiable entity into the current discourse. e.g., *the president of the US*.

Sometimes, pronouns appear before their referents are introduced by a proper name or definite description: this is *cataphora*. E.g., at the start of a discourse:

> Although she couldn't see any dogs, Kim was sure she'd heard barking.

both cases of *she* refer to Kim - the first is a *cataphor*.

9.6 Pronoun agreement

Pronouns generally have to agree in number and gender with their antecedents. In cases where there's a choice of pronoun, such as *he*/*she* or *it* for an animal (or a baby, in some dialects), then the choice has to be consistent.

(38) A little girl is at the door — see what she wants, please?

(39) My dog has hurt his foot — he is in a lot of pain.

(40) * My dog has hurt his foot — it is in a lot of pain.

Complications include the gender neutral *they* (some dialects), use of *they* with *everybody*, group nouns, conjunctions and discontinuous sets:

(41) Somebody's at the door — see what they want, will you?

(42) I don't know who the new teacher will be, but I'm sure they'll make changes to the course.

(43) Everybody's coming to the party, aren't they?

(44) The team played really well, but now they are all very tired.

(45) Kim and Sandy are asleep: they are very tired.

(46) Kim is snoring and Sandy can't keep her eyes open: they are both exhausted.

9.7 Reflexives

(47) John$_i$ cut himself$_i$ shaving. (himself = John, subscript notation used to indicate this)

(48) # John$_i$ cut him$_j$ shaving. (i $\neq$ j — a very odd sentence)

The informal and not fully adequate generalisation is that reflexive pronouns must be co-referential with a preceding argument of the same verb (i.e., something it subcategorises for), while non-reflexive pronouns cannot be. In linguistics, the study of inter-sentential anaphora is known as *binding theory*.

9.8 Pleonastic pronouns

Pleonastic pronouns are semantically empty, and don't refer:

(49) It is snowing

(50) It is not easy to think of good examples.

(51) It is obvious that Kim snores.

(52) It bothers Sandy that Kim snores.

Note also:

(53) They are digging up the street again

This is an (informal) use of *they* which, though probably not technically pleonastic, doesn't apparently refer in the standard way (they = 'the authorities'??).

9.9 Salience

There are a number of effects related to the structure of the discourse which cause particular pronoun antecedents to be preferred, after all the hard constraints discussed above are taken into consideration.

Recency More recent antecedents are preferred. Only relatively recently referred to entities are accessible.

(54) Kim has a big car. Sandy has a small one. Lee likes to drive it.

it preferentially refers to Sandy's car, rather than Kim's.

Grammatical role Subjects $>$ objects $>$ everything else:

(55) Fred went to the Grafton Centre with Bill. He bought a CD.

he is more likely to be interpreted as Fred than as Bill.

Repeated mention Entities that have been mentioned more frequently are preferred:

(56) Fred was getting bored. He decided to go shopping. Bill went to the Grafton Centre with Fred. He bought a CD.

He=Fred (maybe) despite the general preference for subjects.

Parallelism Entities which share the same role as the pronoun in the same sort of sentence are preferred:

(57) Bill went with Fred to the Grafton Centre. Kim went with him to Lion Yard.

Him=Fred, because the parallel interpretation is preferred.

Coherence effects The pronoun resolution may depend on the rhetorical/discourse relation that is inferred.

(58) Bill likes Fred. He has a great sense of humour.

He = Fred preferentially, possibly because the second sentence is interpreted as an explanation of the first, and having a sense of humour is seen as a reason to like someone.

9.10 Lexical semantics and world knowledge effects

The made-up examples above were chosen so that the meaning of the utterance did not determine the way the pronoun was resolved. In real examples, world knowledge may override salience effects. For instance (from Radio 5):

(59) Andrew Strauss again blamed the batting after England lost to Australia last night. They now lead the series three-nil.

Here *they* has to refer to Australia, despite the general preference for subjects as antecedents. The analysis required to work this out is actually non-trivial: you might like to try writing down some plausible meaning postulates which would block the inference that *they* refers to England. (Note also the plural pronoun with singular antecedent, which is normal for sports teams, in British English at least.)

Note, however, that violation of salience effects can easily lead to an odd discourse:

(60) The England football team won last night. Scotland lost. ? They have qualified for the World Cup with a 100% record.

Systems which output natural language discourses, such as summarization systems, have to keep track of anaphora to avoid such problems.

9.11 Algorithms for resolving anaphora

NLP researchers are interested in all types of coreference, but most work has gone into the problem of finding antecedents for pronouns. As well as discourse understanding, this is often important in MT. For instance, English *it* usually has to be resolved to produce a high-quality translation into German because German has grammatical gender (although if all the candidate antecedents have the same gender, we don't need to do any further resolution). I will outline an approach to anaphora resolution using a statistical classifier, but there are many other approaches.

We can formulate pronoun resolution as a classification problem, which can be implemented using one of the standard machine learning approaches to supervised classification (examples of approaches include Naive Bayes, perceptron, k-nearest neighbour), assuming that we have a suitable set of training data. For each pairing of a (non-pleonastic) pronoun and a candidate antecedent, the classifier has to make a binary decision as to whether the candidate is an actual antecedent, based on some features associated with the pairing. For simplicity, we can assume that the candidate antecedents for a pronoun are all the noun phrases within a window of the surrounding text consisting of the current sentence and the preceding 5 sentences (excluding pleonastic pronouns). For example:

> Niall Ferguson is prolific, well-paid and a snappy dresser. Stephen Moss hated him — at least until he spent an hour being charmed in the historian's Oxford study.
>
> Pronoun *he*, candidate antecedents: *Niall Ferguson*, *a snappy dresser*, *Stephen Moss*, *him*, *an hour*, *the historian*, *the historian's Oxford study*.

Notice that this simple approach leads to *a snappy dresser* being included as a candidate antecedent and that a choice had to be made as to how to treat the possessive. I've included the possibility of cataphors, although these are sufficiently rare that they are often excluded.

For each such pairing, we build a *feature vector*[43] using features corresponding to some of the factors discussed in the previous sections. For instance (using t/f rather than 1/0 for binary features for readability):

Cataphoric Binary: t if the pronoun occurs before the candidate antecedent.

Number agreement Binary: t if the pronoun agrees in number with the candidate antecedent.

Gender agreement Binary: t if the pronoun agrees in gender with the candidate antecedent.

Same verb Binary: t if the pronoun and the candidate antecedent are arguments of the same verb (for binding theory).

[43] The term 'instance' is sometimes used in AI, but I prefer 'feature vector', because we're mainly interested in the nature of the features.

Sentence distance Discrete: { 0, 1, 2 . . . } The number of sentences between pronoun and candidate.

Grammatical role Discrete: { subject, object, other } The role of the potential antecedent.

Parallel Binary: t if the potential antecedent and the pronoun share the same grammatical role.

Linguistic form Discrete: { proper, definite, indefinite, pronoun } This indicates something about the syntax of the potential antecedent noun phrase.

Taking some pairings from the example above:

pronoun	antecedent	cataphoric	num	gen	same	distance	role	parallel	form
him	*Niall Ferguson*	f	t	t	f	1	subj	f	prop
him	*Stephen Moss*	f	t	t	t	0	subj	f	prop
him	*he*	t	t	t	f	0	subj	f	pron
he	*Niall Ferguson*	f	t	t	f	1	subj	t	prop
he	*Stephen Moss*	f	t	t	f	0	subj	t	prop
he	*him*	f	t	t	f	0	obj	f	pron

Notice that with this set of features, we cannot model the "repeated mention" effect mentioned in §9.9. It would be possible to model it with a classifier-based system, but it requires that we keep track of the coreferences that have been assigned and thus that we maintain a model of the discourse as individual pronouns are resolved. I will return to the issue of discourse models below. Coherence effects are very complex to model and world knowledge effects are indefinitely difficult (AI-complete in the limit), so both of these are excluded from this simple feature set. Realistic systems use many more features and values than shown here and can approximate some partial world knowledge via classification of named entities, for instance.

To implement the classifier, we require some knowledge of syntactic structure, but not necessarily full parsing. We could approximately determine noun phrases and grammatical role by means of a series of regular expressions over POS-tagged data instead of using a full parser. Even if a full syntactic parser is available, it may be necessary to augment it with special purpose rules to detect pleonastic pronouns.

The training data for this task is produced from a corpus which is marked up by humans with pairings between pronouns and antecedent phrases. The classifier uses the marked-up pairings as positive examples (class TRUE), and all other possible pairings between the pronoun and candidate antecendant as negative examples (class FALSE). For instance, if the pairings above were used as training data, we would have:

class	cataphoric	num	gen	same	distance	role	parallel	form
TRUE	f	t	t	f	1	subj	f	prop
FALSE	f	t	t	t	0	subj	f	prop
FALSE	t	t	t	f	0	subj	f	pron
FALSE	f	t	t	f	1	subj	t	prop
TRUE	f	t	t	f	0	subj	t	prop
FALSE	f	t	t	f	0	obj	f	pron

Note the pre-lecture exercise which suggests that you participate in an online experiment to collect training data. If you do this, you will discover a number of complexities that I have ignored in this account.

In very general terms, a supervised classifier uses the training data to determine an appropriate mapping (i.e., *hypothesis* in the terminology used in the Part 1B AI course) from feature vectors to classes. This mapping is then used when classifying the test data. To make this more concrete, if we are using a probabilistic approach, we want to choose the class c out of the set of classes C ({ TRUE, FALSE } here) which is most probable given a feature vector $\vec{f}$:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|\vec{f})$$

(See §3.5 for the explanation of argmax and $\hat{c}$.) As with the POS tagging problem, for a realistic feature space, we will be unable to model this directly. The Naive Bayes classifier is based on the assumption that we rewrite this formula

using Bayes Theorem and then treat the features as conditionally independent (the independence assumption is the "naive" part). That is:

$$P(c|\vec{f}) = \frac{P(\vec{f}|c)P(c)}{P(\vec{f})}$$

As with the models discussed in Lecture 3, we can ignore the denominator because it is constant, hence:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(\vec{f}|c)P(c)$$

Treating the features as independent means taking the product of the probabilities of the individual features in $\vec{f}$ for the class:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i=1}^{n} P(f_i|c)$$

In practice, the Naive Bayes model is often found to perform well even with a set of features that are clearly not independent.

There are fundamental limitations on performance caused by treating the problem as classification of individual pronoun-antecedent pairs rather than as building a discourse model including all the coreferences. Inability to implement 'repeated mention' is one such limitation, another is the inability to use information gained from one linkage in resolving further pronouns. Consider yet another 'team' example:

(61) Sturt think they can perform better in Twenty20 cricket. It requires additional skills compared with older forms of the limited over game.

A classifier which treats each pronoun entirely separately might well end up resolving the *it* at the start of the second sentence to *Sturt* rather than the correct *Twenty20 cricket*. However, if we already know that *they* corefers with *Sturt*, coreference with *it* will be dispreferred because number agreement does not match (recall from §9.6 that pronoun agreement has to be consistent). This type of effect is especially relevant when general coreference resolution is considered. One approach is to run a simple classifier initially to acquire probabilities of links and to use those results as the input to a second system which clusters the entities to find an optimal solution. I will not discuss this further here, however.

9.12 Evaluation of pronoun resolution

At first sight it seems that we could require that every (non-pleonastic) pronoun is linked to an antecedent, and just measure the accuracy of the links found compared to the test data. One issue which complicates this concerns the identification of the pronouns (some may be pleonastic, others may refer to concepts which aren't expressed in the text as noun phrases) and also identification of the target noun phrases, with embedded noun phrases being a particular issue. We could treat this as a separate problem and assume we're given data with the non-pleonastic pronouns and the candidate antecedents identified, but this isn't fully realistic.

A further range of problems arise essentially because we are using the identification of some piece of text as an antecedent for the pronoun as a surrogate for the real problem, which is identification of references to real world entities. For instance, suppose that, in the example below, our algorithm links *him* to *Andrew* and also links *he* to *Andrew*, but the training data has linked *him* to *Andrew* and *he* to *him*.

> Sally met Andrew in town and took him to the new restaurant. He was impressed.

Our algorithm has successfully linked the coreferring expressions, but if we consider the evaluation approach of comparing the individual links to the test material, it will be penalised. Of course it is trivial to take the transitive closure of the links, but it is not easy to develop an evaluation metric that correctly allows for this and does not, for example, unfairly reward algorithms that link all the pronouns together into one cluster. As a consequence of this sort of issue, it has been difficult to develop agreed metrics for evaluation.

9.13 Statistical classification in language processing

Many problems in natural language can be treated as classification problems: besides pronoun resolution, we have seen sentiment classification and word sense disambiguation, which are straightforward examples of classification. POS-tagging is also a form of classification, but there we take the tag sequence of highest probability rather than considering each tag separately. As we have seen above, we actually need to consider relationships between coreferences to model some discourse effects.

Pronoun resolution has a more complex feature set than the previous examples of classification that we've seen and determination of some of the features requires considerable processing, which is itself error prone. A statistical classifier is somewhat robust to this, assuming that the training data features have been assigned by the same mechanism as used in the test system. For example, if the grammatical role assignment is unreliable, the weight assigned to that feature might be less than if it were perfect.

One serious disadvantage of supervised classification is reliance on training data, which is often expensive and difficult to obtain and may not generalise across domains. Research on unsupervised methods is therefore popular.

There are no hard and fast rules for choosing which statistical approach to classification to use on a given task. Many NLP researchers are only interested in classifiers as tools for investigating problems: they may either simply use the same classifier that previous researchers have tried or experiment with a range of classifiers using a toolkit such as WEKA.[44]

Performance considerations may involve speed as well as accuracy: if a lot of training data is available, then a classifier with faster performance in the training phase may enable one to use more of the available data. The research issues in developing a classifier-based algorithm for an NLP problem generally center around specification of the problem, development of the labelling scheme and determination of the feature set to be used.

9.14 Further reading

J&M discuss the most popular approach to rhetorical relations, *rhetorical structure theory* or RST (section 21.2.1). I haven't discussed it in detail here, partly because I find the theory very unclear: attempts to annotate text using RST approaches tend not to yield good interannotator agreement (see comments on evaluation in lecture 3), although to be fair, this is a problem with all approaches to rhetorical relations. The discussion of the factors influencing anaphora resolution and the description of the classifier approach that I've given here are partly based on J&M's account in Chapter 21: they discuss a log-linear classifier there, but Naive Bayes is described in 20.2.2 and I have followed that description.

[44] `http://www.cs.waikato.ac.nz/ml/weka/` Ian H. Witten and Eibe Frank (2005) "Data Mining: Practical machine learning tools and techniques", 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

10 Lecture 10: Language generation and regeneration

> "Generation from what?!" (attributed to Yorick Wilks)

In the lectures so far, we have concentrated on analysis, but there is also work on generating text. Although Natural Language Generation (NLG) is a recognised subfield of NLP, there are far fewer researchers working in this area than on analysis. There are some recognised subproblems within NLG, some of which will be discussed below, but it is not always easy to break it down into subareas and in some cases, systems which create text are not considered to be NLG system (e.g., when they are a component of an MT system). The main problem, as the quotation above indicates, is there is no commonly agreed starting point for NLG. The possible starting points include:

- Logical form or other sentence meaning representation.
  This is the inverse of (deep) parsing. Sometimes called *realization* when part of a bigger system (but realization is often from a syntax tree).

  One special case of this is as a component of an MT system using *syntactic transfer* or *semantic transfer*.
- Formally-defined data: databases, knowledge bases, semantic web ontologies, etc.
- Semi-structured data: tables, graphs etc.
- Numerical data: e.g., weather reports.
- User input (plus other data sources) in assistive communication.

Such systems can be contrasted with **regeneration** systems, which start from text and produce reformulated text. Here the options include:

- Constructing coherent sentences from partially ordered sets of words: e.g., in statistical MT (although this might better be considered as a form of realization).
- Paraphrase.
- Summarization
- Article construction from text fragments (e.g., automatic construction of Wikipedia articles),
- Text simplification.

There are also mixed generation and regeneration systems.

10.1 Tasks in generation

This categorization is broadly taken from Dale and Mellish (1998):

Content determination deciding what information to convey. This often involves selecting information from a body of possible pieces of information. e.g., weather reports. There is often a mass of information, not all of which is relevant. The content may vary according to the type of user (e.g., expert/non-expert). This step often involves domain experts: i.e., people who know how to interpret the raw data.

Discourse structuring the broad structure of the text or dialogue (Dale and Mellish refer to document structuring). For instance, scientific articles may have an abstract, introduction, methods, results, comparison, conclusion: rearranging the text makes it incoherent. In a dialogue system, there may be multiple messages to convey to the user, and the order may be determined by factors such as urgency.

Aggregation deciding how information may be split into sentence-sized chunks: i.e., finer-grained than document structuring.

Referring expression generation deciding when to use pronouns, how many modifiers to include and so on.

Lexical choice deciding which lexical items to use to convey a given concept. This may be straightforward for many applications — in a limited-domain system, there will be a preferred knowledge base to lexical item mapping, for instance, but it may be useful to vary this.

Surface realization mapping from a meaning representation for an individual sentence (or a detailed syntactic representation) to a string (or speech output). This is generally taken to include morphological generation.

Fluency ranking this is not included by Dale and Mellish but is very important in modern approaches. A large grammar will generate many strings for a given meaning representation. In fact, in most approaches, the grammar will overgenerate and produce a mixture of grammatical and ungrammatical strings. A fluency ranking component, which at its simplest is based on n-grams (compare the discussion of prediction in lecture 2), is used to rank such outputs. In the extreme case, no grammar is used, and the fluency ranking is performed on a partially ordered set of words (for instance in SMT).

When Dale and Mellish produced this classification, there was very little statistical work within NLG (work on SMT had started but wasn't very successful at that point) and NLG was essentially about limited domains. In recent years, there have been statistical approaches to all these subtasks. Most NLG systems are still limited domain however, although regeneration systems are usually not so restricted.

Many of the approaches we have described in the previous lectures can be adapted to be useful in generation. Deep grammars may be bidirectional, and therefore usable for surface realization as well as for parsing. Distributions are relevant for lexical choice: e.g., in deciding whether one word is substitutable for another. However, many practical NLG systems are based on extensive use of templates (i.e., fixed text with slots which can be filled in) and this is satisfactory for many tasks, although it tends to lead to rather stilted output.

10.2 Summarisation

Automatic summarisation is a text regeneration task, whose goal is to produce a short version of a text that contains the most important or relevant information. Depending on the nature of the input, one can distinguish between *single-document summarisation*, where we are given a single document and need to produce its summary (e.g. an abstract or a headline), and *multi-document summarisation*, where we need to aggregate content from multiple documents (e.g. several news stories about the same event) into one cohesive summary. These two tasks are quite different in practice and require different processing steps, as we will see below.

The task of simply identifying important information in the document(s) and presenting it in a short summary is typically referred to as generic summarisation or just summarisation. This contrasts *query-focused summarisation*, whose goal is to summarise a document in order to answer a specific query from a user. This involves retrieving a document that contains the answer to the query, and then creating a short text snippet summarising its content and providing the answer. A simple example of query-focused summarisation is shown in Figure 2. In this query the user asks what NLP is, and we can see that the output of Google's search engine provides an automatically generated answer (above the ranked web pages). To generate the summary, Google's system simply selected the first two sentences from the Wikipedia entry (first hit in the search). This is indeed a common, and surprisingly useful, technique. However, when addressing more complex queries, a set of more sophisticated techniques are needed, as we will see below. For instance, some questions require extracting information from multiple documents and then combining it into one cohesive answer.

Whether single-document or multi-document, there exist two different approaches to summarisation: *extractive* and *abstractive summarisation*. Extractive summarisation methods first extract important or relevant sentences from the document(s) and then combine them into a summary. The summary then represents a list of sentences or phrases already in the documents, ordered in the most cohesive way possible. In contrast, the idea behind abstractive summarisation is to interpret the content of the document (semantics, discourse etc.) and then generate the summary based on this interpretation. The summary in this case is formulated using other words than in the document, and is in some sense a true language generation task. However, due to the amount of interpretation required this is very difficult to implement, and most research has focused on extractive summarisation, which I discuss below.

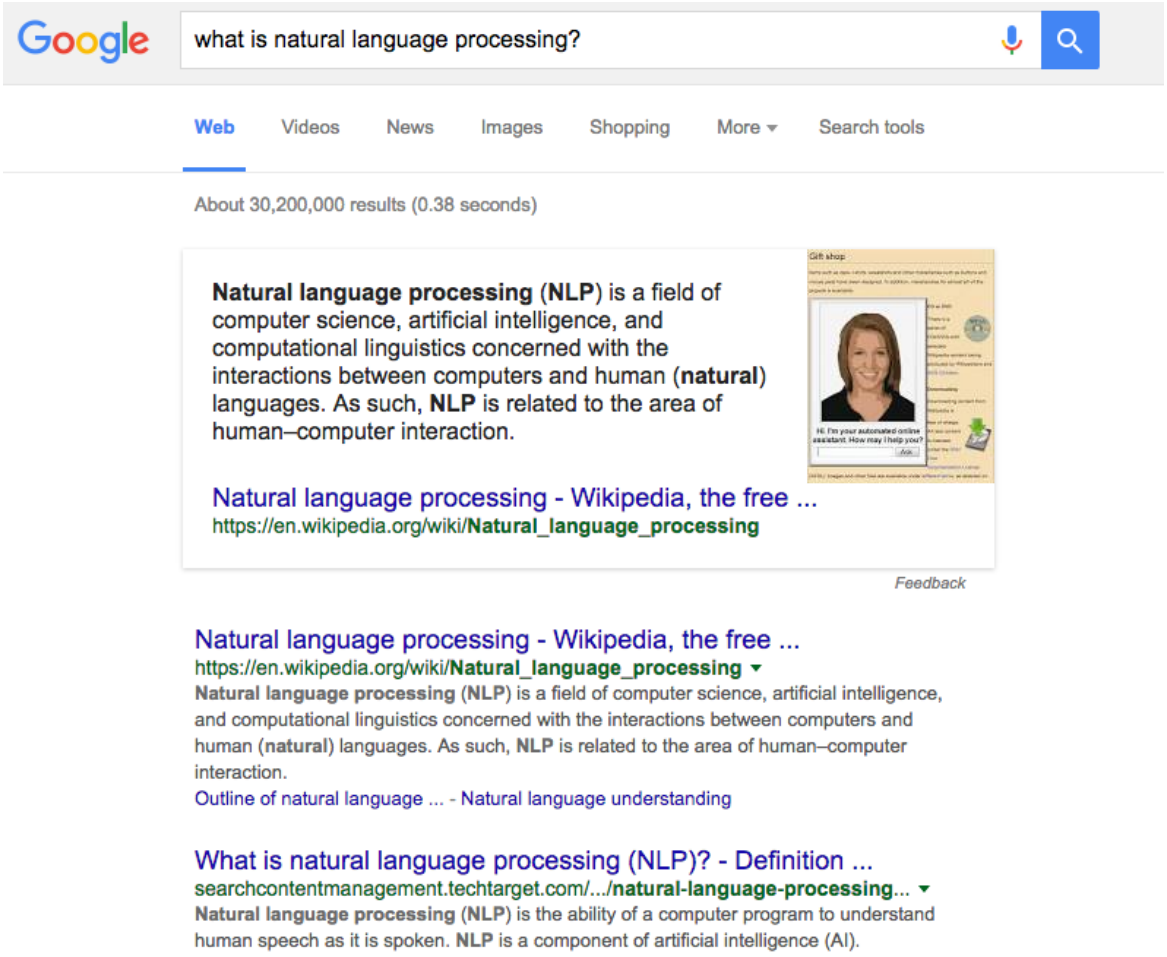


Figure 2: Query-focused summarisation

10.3 Content selection in extractive summarisation

Extractive summarisation systems typically have three main components:

1. Content selection: identify important sentences to extract from the document
2. Information ordering: order the identified sentences within the summary
3. Sentence realisation (optional): perform sentence simplification to remove less relevant information and make the sentences shorter and easier to read.

Content selection is at the heart of all language generation systems, including summarisation, and a range of approaches to this task, both supervised and unsupervised, have been proposed. A typical unsupervised approach to content selection involves using statistical measures to determine which words in the document(s) are *informative*, ranking sentences based on the presence of informative words and selecting the highest ranked sentences for the summary. Informativeness can be measured, for instance, using mutual information, log-likelihood ratio (LLR) or *tf-idf* (term frequency / inverse document frequency).[45] Let us consider ranking informative words with *tf-idf*. *tf-idf* assigns a weight to each word i in the document j as follows:

$$weight(w_i) = tf_{ij} * idf_i$$

[45] *tf-idf* is a measure frequently used in information retrieval.

tf_{ij} is the frequency of word i in doc j; idf_i is the inverse document frequency computed as

$$idf_i = \log \frac{N}{n_i},$$

where N is the total number of documents; and n_i is the number of documents containing the word w_i. This calculation is performed against a background corpus. Essentially, *tf-idf* emphasizes the words that are characteristic to this document, i.e. appear frequently in it and less frequently in the rest of the corpus. The higher *tf-idf*, the more informative this word is considered.

Supervised approaches to content selection start with a training set of documents and their summaries. In the training set the sentences are aligned in the summaries and the documents. The goal is then to learn a set of useful features that characterize the sentences as potentially suitable summary candidates. The investigated features include position of the sentence within the document (e.g. first sentence), the length of the sentence, informative words, cue phrases and so on. The feature representations of the document are then used to train a binary classifier that decides if the sentence should be included in the summary or not.

While the supervised approach appears to be more sophisticated and make use of more knowledge than just the informative words, it is surprising that it does not outperform the unsupervised approach in practice. In addition, it is quite difficult to obtain a sufficiently large training dataset for such an approach. Besides obtaining human-produced summaries, this also requires manual alignment of these summaries with sentences in the document(s). This is non-trivial, since human-produced summaries rarely contain the exact same sentences that appear in the documents. Thus, an unsupervised approach may offer a better alternative, at least for now.

Once the sentences have been selected, they need to be ordered to generate a summary. In case of single-document summarisation, a very simple approach may be adopted, which orders the sentences following their order in the original document.

Below is an example summary of an article describing the arrest of Augusto Pinochet (Nenkova and McKeown, 2011):

> *As his lawyers in London tried to quash a Spanish arrest warrant for Gen. Augusto Pinochet, the former Chilean Dictator, efforts began in Geneva and Paris to have him extradited. Britain has defended its arrest of Gen. Augusto Pinochet, with one lawmaker saying that Chile's claim that the former Chilean Dictator has diplomatic immunity is ridiculous. Margaret Thatcher entertained former Chilean Dictator Gen. Augusto Pinochet at her home two weeks before he was arrested in his bed in a London hospital, the ex-prime minister's office said Tuesday, amid growing diplomatic and domestic controversy over the move.*

Mini-exercise: When reading the summary, think about what it shows about the usability and the limitations of extractive summarisation. And would this summary benefit from simplifying the sentences further?

10.4 Query-focused multi-document summarisation

Query-focused multi-document summarisation aims to answer a query from a user, where the complete answer is not available in any of the individual documents and information from multiple documents needs to be aggregated. For example, a query such as "*Describe the coal mine accidents in China and actions taken*" is likely to require information from different sources. Given a query, query-focused multi-document summarisation systems typically incorporate the following steps:

1. find a set of documents relevant to the query
2. simplify sentences in these documents
3. identify informative sentences in the documents
4. order the sentences into a summary
5. modify the sentences as needed

For our purposes, we will assume that we have a set of relevant documents and will discuss how they are summarised to produce an answer. Information retrieval is a large field in its own right, and it is covered by another Part II course (Information Retrieval) in Lent term.

Sentence simplification Starting with a given set of documents, we first need to perform sentence simplification in these documents in order to remove less relevant information (such as elaborations and clarifications) and retain the most important facts. To do this, the sentences first need to be parsed by a syntactic parser, which identifies prepositional phrases, relative clauses and other syntactic constructions that may become candidates for removal. We know from discourse analysis (lecture 9) that particular syntactic constructions tend to play the role of satellites. We can, therefore, create a set of rules for pruning these satellites off the parse tree. Potential candidates to prune are, for instance:

- appositives: e.g. *Also on display was a painting by Sandor Landeau, ~~an artist who was living in Paris at the time.~~*
- attribution clauses: e.g. *Eating too much bacon can lead to cancer, ~~the WHO reported on Monday.~~*
- PPs without proper names: e.g. *Electoral support for Plaid Cymru increased ~~to a new level.~~*
- initial adverbials: e.g. ~~*For example,*~~ ~~*On the other hand,*~~

It is also possible to develop a classifier for satellite identification and removal, but as with content selection, creating training data for such a system is expensive and non-trivial.

Content selection from multiple documents After the sentences have been simplified we need to perform content selection. The task of content selection from multiple documents is different from the case of a single document, in that we need to ensure that the selected sentences are not merely informative, but also non-redundant. As our aim is to perform query-focused summarisation, the selected sentences also need to contain information relevant to the query. As in the single-document case, we start by ranking sentences in all of the documents in the set by informativeness. The informativeness of a sentence can be calculated based on the number of informative words in it. The words that appear in the query are by definition considered informative, and additional informative words may be identified using e.g. tf-idf.

In order to construct the summary, we start by choosing the most informative sentence in all of the documents and placing it into the summary. We then iteratively add sentences to the summary that are both informative and non-redundant with the summary so far. A popular method for adding sentences to the summary while balancing these two factors is Maximal Marginal Relevance (MMR).

MMR measures both the relevance of the sentence to the query and its novelty with respect to the summary so far.

- **Relevance** to the query is defined as a high similarity between the sentence s_i and the query Q (e.g. cosine similarity can be used)
- **Novelty** with respect to the summary so far is defined as a low similarity with the summary sentences.

At each iteration, we a do pass through all of our documents and choose the next sentence to be added to the summary by maximising

$$\hat{s} = \underset{s_i \in D}{\operatorname{argmax}} \left[\lambda sim(s_i, Q) - (1 - \lambda) \max_{s_j \in S} sim(s_i, s_j) \right],$$

where λ is the weight balancing the contributions of these two factors. Using $\max_{s_j \in S} sim(s_i, s_j)$ ensures that we take a similarity of the candidate sentence with its most similar sentence in the summary. To add another sentence, we do another pass through the whole document set and maximise this measure with respect to the new summary. The algorithm terminates when the summary has reached the desired length.

It should be noted that the similarity between the sentences implied here is not the same as the distributional similarity between word and phrase vectors that we discussed in our lectures on semantics. Sentence similarity in summarisation is typically calculated as follows: each sentence is represented as a vector in the space of all possible words in the

corpus; the dimensions corresponding to the words present in the sentence are assigned a value of 1 or their count in the sentence; other dimensions are assigned the value of 0. Representing sentences as vectors then allows us to calculate cosine similarity between them.

Sentence ordering in the summary Once we've selected the sentences for the summary, we need to order them so that the summary is cohesive. The simplest way of ordering them is chronologically, e.g. by the date of the document. The expectation is then that the sentences about earlier events come first. However, this does not necessarily guarantee coherence of the summary. Other methods that are more coherence-focused order sentences based on their similarity with each other (so that the sentences next to each other are similar, e.g. as determined by their cosine similarity); or order them so that the sentences next to each other discuss the same entity or referent. More recent methods perform topical ordering, e.g. by learning a set of topics present in the documents using LDA topic modelling, and then ordering the sentences by topic. However, this involves a lot of additional processing and is generally a difficult task.

Below is an example summary produced by the system of Li and Li (2013) for the query "*Describe the coal mine accidents in China and actions taken*". The sentences have been automatically ordered by topic, using LDA.

> *(1) In the first eight months, the death toll of coal mine accidents across China rose 8.5 percent from the same period last year. (2)China will close down a number of ill-operated coal mines at the end of this month, said a work safety official here Monday. (3) Li Yizhong, director of the National Bureau of Production Safety Supervision and Administration, has said the collusion between mine owners and officials is to be condemned. (4)from January to September this year, 4,228 people were killed in 2,337 coal mine accidents. (5) Chen said officials who refused to register their stakes in coal mines within the required time*

10.5 Further reading

Unfortunately the second edition of J&M has almost nothing about NLG. There is a chapter in the first edition, but it is now rather dated. Reiter and Dale (2000) *Building Natural Language Generation Systems* is a textbook that discusses the general concepts, but is also now somewhat dated.

Blogging Birds (`http://redkite.abdn.ac.uk/`) is a great example of a system that generates text based on data: see `http://aclweb.org/anthology/P/P13/P13-4029.pdf` for a short description.

The summarisation part of the lecture is based on Dan Jurafsky's summarisation lecture, and is (quite appropriately) a summary thereof. The full lecture can viewed online at `https://class.coursera.org/nlp/lecture/preview`

The papers referred to in this lecture are:

Ani Nenkova and Kathleen McKeown, Automatic Summarization *Foundations and Trends in Information Retrieval*, Vol 5, No 2-3, pp. 103-233.

Jiwei Li and Sujian Li. 2013. A Novel Feature-based Bayesian Model for Query Focused Multi-document Summarization. *Transactions of the Association for Computational Linguistics*, Vol. 1.

11 Lecture 11: Applications

No notes: copies of slides will be made available after the lecture.

12 Lecture 12: Recent trends in NLP research

No notes: copies of slides will be made available after the lecture.

A glossary/index of some of the terms used in the lectures

This is primarily intended to cover concepts which are mentioned in more than one lecture. The lecture where the term is explained in most detail is generally indicated. In some cases, I have just given a pointer to the section in the lectures where the term is defined. Note that IGE stands for *The Internet Grammar of English* (`http://www.ucl.ac.uk/internet-grammar/home.htm`). There are a few cases where this uses a term in a slightly different way from these course notes: I have tried to indicate these.

active chart See §4.9.

adjective See IGE or notes for pre-lecture exercises in lecture 3.

adjunct See **argument** and also IGE.

adverb See IGE or notes for pre-lecture exercises in lecture 3.

affix A morpheme which can only occur in conjunction with other morphemes (lecture 2).

AI-complete A half-joking term, applied to problems that would require a solution to the problem of representing the world and acquiring world knowledge (lecture 1).

agreement The requirement for two phrases to have compatible values for grammatical features such as number and gender. For instance, in English, *dogs bark* is grammatical but *dog bark* and *dogs barks* are not. See IGE.

ambiguity The same string (or sequence of sounds) meaning different things. Contrasted with **vagueness**.

anaphora The phenomenon of referring to something that was mentioned previously in a text. An anaphor is an expression which does this, such as a pronoun (see §9.5).

antonymy Opposite meaning: such as *clean* and *dirty* (§5.2).

argument In syntax, the phrases which are lexically required to be present by a particular word (prototypically a verb). This is as opposed to **adjunct**s, which modify a word or phrase but are not required. For instance, in:

> Kim saw Sandy on Tuesday

Sandy is an argument but *on Tuesday* is an adjunct. Arguments are specified by the **subcategorization** of a verb etc. Also see the IGE.

aspect A term used to cover distinctions such as whether a verb suggests an event has been completed or not (as opposed to tense, which refers to the time of an event). For instance, *she was writing a book* vs *she wrote a book*.

backoff Usually used to refer to techniques for dealing with data sparseness in probabilistic systems: using a more general classification rather than a more specific one. For instance, using unigram probabilities instead of bigrams; using word classes instead of individual words (lecture 3).

bag of words Unordered collection of words in some text.

baseline In evaluation, the performance produced by a simple system against which the experimental technique is compared (§3.6).

bidirectional Usable for both analysis and generation (lecture 2).

case Distinctions between nominals indicating their syntactic role in a sentence. In English, some pronouns show a distinction: e.g., *she* is used for subjects, while *her* is used for objects. e.g., *she likes her* vs **her likes she*. Languages such as German and Latin mark case much more extensively.

ceiling In evaluation, the performance produced by a 'perfect' system (such as human annotation) against which the experimental technique is compared (§3.6).

CFG context-free grammar.

chart parsing See §4.5.

Chomsky Noam Chomsky, professor at MIT. His work underlies most modern approaches to syntax in linguistics. Not so hot on probability theory.

classifier A system which assigns classes to items, usually using a machine learning approach.

closed class Refers to parts of speech, such as conjunction, for which all the members could potentially be enumerated (lecture 3).

coherence See §9.2

collocation See §5.7

complement For the purposes of this course, an **argument** other than the subject.

compositionality The idea that the meaning of a phrase is a function of the meaning of its parts. **compositional semantics** is the study of how meaning can be built up by semantic rules which mirror syntactic structure (lecture 8).

constituent A sequence of words which is considered as a unit in a particular grammar (lecture 4).

constraint-based grammar A formalism which describes a language using a set of independently stated constraints, without imposing any conditions on processing or processing order (lecture 4).

context The situation in which an utterance occurs: includes prior utterances, the physical environment, background knowledge of the speaker and hearer(s), etc etc. Nothing to do with context-free grammar.

corpus A body of text used in experiments (plural *corpora*). See §3.1.

cue phrases Phrases which indicates particular **rhetorical relations**.

denominal Something derived from a noun: e.g., the verb *tango* is a denominal verb.

dependency structure A syntactic or semantic representation that links words via relations

derivational morphology See §2.2

determiner See IGE or notes for pre-lecture exercises in lecture 3.

deverbal Something derived from a verb: e.g., the adjective *surprised*.

direct object See IGE. Contrast **indirect object**.

distributional semantics Representing word meaning by context of use (lecture 6).

discourse In NLP, a piece of connected text.

discourse relations See **rhetorical relations**.

domain Not a precise term, but I use it to mean some restricted set of knowledge appropriate for an application.

error analysis In evaluation, working out what sort of errors are found for a given approach (§3.6).

expletive pronoun Another term for **pleonastic pronoun**: see §9.8.

feature a characteristic property used in machine learning.

FSA Finite state automaton

FST Finite state transducer

full-form lexicon A lexicon where all morphological variants are explicitly listed (lecture 2).

generation The process of constructing text (or speech) from some input representation (lecture 10).

generative grammar The family of approaches to linguistics where a natural language is treated as governed by rules which can produce all and only the well-formed utterances. Lecture 4.

genre Type of text: e.g., newspaper, novel, textbook, lecture notes, scientific paper. Note the difference to **domain** (which is about the type of knowledge): it's possible to have texts in different genre discussing the same domain (e.g., discussion of human genome in newspaper vs textbook vs paper).

gloss An explanation/translation of an obscure/foreign word or phrase.

grammar Formally, in the generative tradition, the set of rules and the lexicon. Lecture 4.

head In syntax, the most important element of a phrase.

hearer Anyone on the receiving end of an utterance (spoken, written or signed). §1.3.

Hidden Markov Model See §3.5

HMM Hidden Markov Model

homonymy Instances of **polysemy** where the two senses are unrelated (§5.5).

hyponymy An 'IS-A' relationship (§5.1) More general terms are **hypernym**s, more specific **hyponym**s.

indirect object The beneficiary in verb phrases like *give a present to Sandy* or *give Sandy a present*. In this case the indirect object is *Sandy* and the **direct object** is *a present*.

interannotator agreement The degree of agreement between the decisions of two or more humans with respect to some categorisation (§3.6).

language model A term generally used in speech recognition, for a statistical model of a natural language (lecture 3).

lemmatization Finding the stem and affixes for words (lecture 2).

lexical ambiguity Ambiguity caused because of multiple senses for a word.

lexicon The part of an NLP system that contains information about individual words (lecture 1).

local ambiguity Ambiguity that arises during analysis etc, but which will be resolved when the utterance is completely processed.

logical form The semantic representation constructed for an utterance (lecture 8).

long-distance dependency See §4.12

meaning postulates Inference rules that capture some aspects of the meaning of a word.

meronymy The 'part-of' lexical semantic relation (§5.2).

modifier Something that further specifies a particular entity or event: e.g., *big house*, *shout loudly*.

morpheme Minimal information carrying units within a word (§2.1).

morphology See §1.2

multiword expression A fixed phrase with a non-compositional meaning (lecture 8).

MT Machine translation.

multiword expression A conventional phrase that has something idiosyncratic about it and therefore might be listed in a dictionary.

mumble input Any unrecognised input in a spoken dialogue system (lecture 2).

n-gram A sequence of n words (§3.2).

named entity recognition Recognition and categorisation of person names, names of places, dates etc (lecture 4).

NL Natural language.

NLG Natural language generation (lecture 10).

NLID Natural language interface to a database.

nominal In grammar terminology, noun-like (can be used to describe a word or a phrase).

noun See IGE or notes for pre-lecture exercises in lecture 3.

noun phrase (NP) A phrase which has a noun as syntactic **head**. See IGE.

ontology In NLP and AI, a specification of the entities in a particular domain and (sometimes) the relationships between them. Often hierarchically structured.

open class Opposite of **closed class**.

orthographic rules Same as **spelling rules** (§2.4)

overgenerate Of a grammar, to produce strings which are invalid, e.g., because they are not grammatical according to human judgements.

packing See §4.8

passive chart parsing See §4.6

parse tree See §4.4

part of speech The main syntactic categories: noun, verb, adjective, adverb, preposition, conjunction etc.

part of speech tagging Automatic assignment of syntactic categories to the words in a text. The set of categories used is actually generally more fine-grained than traditional parts of speech.

pleonastic Non-referring (esp. of pronouns): see §9.8

polysemy The phenomenon of words having different senses (§5.5).

POS Part of speech (in the context of POS tagging).

pragmatics See §1.2

predicate In logic, something that takes zero or more arguments and returns a truth value. (Used in IGE for the verb phrase following the subject in a sentence, but I don't use that terminology.)

prefix An **affix** that precedes the **stem**.

probabilistic context free grammars (PCFGs) CFGs with probabilities associated with rules (lecture 4).

realization Construction of a string from a meaning representation for a sentence or a syntax tree (lecture 10).

referring expression See §9.5

relative clause See IGE.

A **restrictive relative clause** is one which limits the interpretation of a noun to a subset: e.g. *the students who sleep in lectures are obviously overworking* refers to a subset of students. Contrast **non-restrictive**, which is a form of parenthetical comment: e.g. *the students, who sleep in lectures, are obviously overworking* means all (or nearly all) are sleeping.

selectional restrictions Constraints on the semantic classes of arguments to verbs etc (e.g., the subject of *think* is restricted to being sentient). The term **selectional preference** is used for non-absolute restrictions.

semantics See §1.2

smoothing Redistributing observed probabilities to allow for **sparse data**, especially to give a non-zero probability to unseen events (lecture 2).

SMT Statistical machine translation.

sparse data Especially in statistical techniques, data concerning rare events which isn't adequate to give good probability estimates (lecture 2).

speaker Someone who makes an **utterance** (§1.3).

spelling rules §2.4

stem A **morpheme** which is a central component of a word (contrast **affix**). §2.1.

stemming Stripping **affix**es (see §2.3).

strong equivalence Of grammars, accepting/rejecting exactly the same strings and assigning the same bracketings (contrast **weak equivalence**). Lecture 4.

structural ambiguity The situation where the same string corresponds to multiple bracketings.

subcategorization The lexical property that tells us how many **argument**s a verb etc can have.

suffix An **affix** that follows the **stem**.

summarization Producing a shorter piece of text (or speech) that captures the essential information in the original.

synonymy Having the same meaning (§5.2).

syntax See §1.2

taxonomy Traditionally, the scheme of classification of biological organisms. Extended in NLP to mean a hierarchical classification of word senses. The term **ontology** is sometimes used in a rather similar way, but ontologies tend to be classifications of domain-knowledge, without necessarily having a direct link to words, and may have a richer structure than a taxonomy.

tense Past, present, future etc.

training data Data used to train any sort of machine-learning system. Must be separated from test data which is kept unseen. Manually-constructed systems should also use strictly unseen data for evaluation.

treebank a corpus annotated with trees (lecture 4).

weak equivalence Of grammars, accepting/rejecting exactly the same strings (contrast **strong equivalence**). Lecture 4.

Wizard of Oz experiment An experiment where data is collected, generally for a dialogue system, by asking users to interact with a mock-up of a real system, where some or all of the 'processing' is actually being done by a human rather than automatically.

WordNet See §5.3

word-sense disambiguation See §5.6

WSD Word-sense disambiguation

utterance A piece of speech or text (sentence or fragment) generated by a speaker in a particular context.

vagueness Of word meanings, contrasted with **ambiguity** : see §5.5.

verb See IGE or notes for pre-lecture exercises in lecture 3.

verb phrase (VP) A phrase headed by a verb.

Acknowledgement

These notes are based on the NLP notes originally written by Ann Copestake and Aurelie Herbelot. I would like to thank Dan Jurafsky for sharing his summarisation material and Ekaterina Kochmar for proofreading the new lectures.

Exercises for NLP course, 2016

Notes on exercises

These exercises are organised by lecture. They are divided into two classes: pre-lecture and post-lecture. The pre-lecture exercises are intended to review the basic concepts that you'll need to fully understand the lecture. Depending on your background, you may find these trivial or you may need to read the notes, but in either case they shouldn't take more than a few minutes. The first one or two examples generally come with answers, other answers are at the end (where appropriate).

Answers to the post-lecture exercises are available to supervisors (where appropriate). These are mostly intended as quick exercises to check understanding of the lecture, though some are more open-ended.

A Lecture 1

A.1 Post-lecture exercises

Without looking at any film reviews beforehand, write down 10 words which you think would be good indications of a positive review (when taken in isolation) and 10 words which you think would be negative. Then go through a review of a film and see whether you find there are more of your positive words than the negative ones. Are there words in the review which you think you should have added to your initial lists?

Have a look at `http://www.cl.cam.ac.uk/~aac10/stuff.html` for pointers to sentiment analysis data used in experiments.

B Lecture 2

B.1 Pre-lecture exercises

1. Split the following words into morphological units, labelling each as stem, suffix or prefix. If there is any ambiguity, give all possible splits.

   (a) dries
   answer: dry (stem), -s (suffix)

   (b) cartwheel
   answer: cart (stem), wheel (stem)

   (c) carries

   (d) running

   (e) uncaring

   (f) intruders

   (g) bookshelves

   (h) reattaches

   (i) anticipated

2. List the simple past and past/passive participle forms of the following verbs:

   (a) sing
   Answer: simple past *sang*, participle *sung*

   (b) carry

   (c) sleep

(d) see

Note that the simple past is used by itself (e.g., *Kim sang well*) while the participle form is used with an auxiliary (e.g., *Kim had sung well*). The passive participle is always the same as the past participle in English: (e.g., *Kim began the lecture early*, *Kim had begun the lecture early*, *The lecture was begun early*).

B.2 Post-lecture exercises

1. For each of the following surface forms, give a list of the states that the FST given in the lecture notes for e-insertion passes through, and the corresponding underlying forms:

   (a) c a t s

   (b) c o r p u s

   (c) a s s e s

   (d) a s s e s s

   (e) a x e s

2. Modify the FSA for dates so that it only accepts valid months. Turn your revised FSA into a FST which maps between the numerical representation of months and their abbreviations (Jan . . . Dec).

3. Earlier dialects of English used expressions like "four and twenty" instead of "twenty four" (the same pattern occurs in modern German, French has something similar but mostly omits the 'and'). This pattern applies for numbers from 21 to 99, with the exception of 30, 40 etc. Assume a numerical version of this, where '4&20' corresponds to '24', '3&30' to '33' etc. Write a FST that represents this mapping for numbers from 1 to 99 ('14', '4' etc should correspond to themselves). What does this illustrate about the FST formalism?

4. The lecture talks about morphological paradigms: suggest appropriate slots for English verbs (you may ignore *be*) and give an example of a regular and irregular verb using this paradigm.

5. Look up the rules for the original Porter stemmer and give 5 examples of words where these could give significantly different results on a task like sentiment analysis compared to finding the linguistic stem.

6. Outline how you might write a FSA corresponding to dialogue states for booking a table in a known restaurant (the full FSA is not required).

C Lecture 3

C.1 Pre-lecture

Label each of the words in the following sentences with their part of speech, distinguishing between nouns, proper nouns, verbs, adjectives, adverbs, determiners, prepositions, pronouns and others. (Traditional classifications often distinguish between a large number of additional parts of speech, but the finer distinctions won't be important here.) There are notes on part of speech distinctions below, if you have problems.

1. The brown fox could jump quickly over the dog, Rover. Answer: The/Det brown/Adj fox/Noun could/Verb(modal) jump/Verb quickly/Adverb over/Preposition the/Det dog/Noun, Rover/Proper noun.

2. The big cat chased the small dog into the barn.

3. Those barns have red roofs.

4. Dogs often bark loudly.

5. Further discussion seems useless.

6. Kim did not like him.

7. Time flies.

Notes on parts of speech. These notes are English-specific and are just intended to help with the lectures and the exercises: see a linguistics textbook for definitions! Some categories have fuzzy boundaries, but none of the complicated cases will be important for this course.

Noun prototypically, nouns refer to physical objects or substances: e.g., *aardvark*, *chainsaw*, *rice*. But they can also be abstract (e.g. *truth*, *beauty*) or refer to events, states or processes (e.g., *decision*). If you can say *the X* and have a sensible phrase, that's a good indication that X is a noun.

Pronoun something that can stand in for a noun: e.g., *him*, *his*

Proper noun / Proper name a name of a person, place etc: e.g., *Elizabeth*, *Paris*

Verb Verbs refer to events, processes or states but since nouns and adjectives can do this as well, the distinction between the categories is based on distribution, not semantics. For instance, nouns can occur with determiners like *the* (e.g., *the decision*) whereas verbs can't (e.g., * *the decide*). In English, verbs are often found with auxiliaries (*be*, *have* or *do*) indicating tense and aspect, and sometime occur with modals, like *can*, *could* etc. Auxiliaries and modals are themselves generally treated as subclasses of verbs.

Adjective a word that modifies a noun: e.g., *big*, *loud*. Most adjectives can also occur after the verb *be* and a few other verbs: e.g., *the students are unhappy*. Numbers are sometimes treated as a type of adjective by linguists but generally given their own category in traditional grammars. Past participle forms of verbs can also often be used as adjectives (e.g., *worried* in *the very worried man*). Sometimes it's impossible to tell whether something is a participle or an adjective (e.g., *the man was worried*).

Adverb a word that modifies a verb: e.g. *quickly*, *probably*.

Determiner these precede nouns e.g., *the*, *every*, *this*. It is not always clear whether a word is a determiner or some type of adjective.

Preposition e.g., *in*, *at*, *with*

Nouns, proper nouns, verbs, adjectives and adverbs are the *open classes*: new words can occur in any of these categories. Determiners, prepositions and pronouns are closed classes (as are auxiliary and modal verbs).

C.2 Post-lecture

1. Try out one or more of the following POS tagging sites:
   `http://alias-i.com/lingpipe/web/demos.html`
   `http://www.lingsoft.fi/demos.html`
   `http://ucrel.lancs.ac.uk/claws/trial.html`
   `http://l2r.cs.uiuc.edu/~cogcomp/pos_demo.php`
   The Lingpipe tagger uses an HMM approach as described in the lecture, the others use different techniques. Lingsoft give considerably more information than the POS tag: their system uses hand-written rules.

   Find two short pieces of naturally occurring English text, one of which you think should be relatively easy to tag correctly and one which you predict to be difficult. Look at the tagged output and estimate the percentage of correct tags in each case, concentrating on the open-class words. You might like to get another student to look at the same output and see if you agree on which tags are correct.

2. (OPEN-ENDED) The notes briefly discuss word prediction for use in an AAC system. Describe how such an approach might be enhanced by the use of POS tags. What type of issues might you expect to arise in developing such a system?

3. The notes mention that verbal uses of words that are mainly nouns (e.g., *tango*) are quite likely to be mistagged. Investigate this with an online part-of-speech tagger and try and explain your results on the basis of the HMM model discussed in the lectures.

D Lecture 4

D.1 Pre-lecture

Put brackets round the noun phrases and the verb phrases in the following sentences (if there is ambiguity, give two bracketings):

1. The cat with white fur chased the small dog into the barn.
   Answer: ((The cat)$_{np}$ with (white fur)$_{np}$)$_{np}$ chased (the small dog)$_{np}$ into (the barn)$_{np}$
   The cat with white fur (chased the small dog into the barn)$_{vp}$

2. The big cat with black fur chased the dog which barked.

3. Three dogs barked at him.

4. Kim saw the birdwatcher with the binoculars.

Note that noun phrases consist of the noun, the determiner (if present) and any modifiers of the noun (adjective, prepositional phrase, relative clause). This means that noun phrases may be nested. Verb phrases include the verb and any auxiliaries, plus the object and indirect object etc (in general, the complements of the verb) and any adverbial modifiers.[46] The verb phrase does not include the subject.

D.2 Post-lecture

1. Using the CFG given in the lecture notes (section 4.3):

   (a) show the edges generated when parsing *they fish in rivers in December* with the simple chart parser in 4.7

   (b) show the edges generated for this sentence if packing is used (as described in 4.9)

   (c) show the edges generated for *they fish in rivers* if an active chart parser is used (as in 4.10)

2. Modify the CFG given in section 4.3 so that intransitive, transitive and ditransitive verbs are distinguished (add suitable lexical entries to illustrate this).

3. Modify the CFG to allow for adverbs.

4. Suppose the following rules are added to the grammar:

   ```
   VP -> VP Adv
   S -> S Adv
   ```

   What would this imply for the analysis of a sentence like: *Kim barked loudly*? If only the first rule were present, would the resulting grammar be weakly- or strongly- equivalent to the original? Could there be reasons to want both rules?

E Lecture 5

E.1 Pre-lecture

Without looking at a dictionary, write down brief definitions for as many senses as you can think of for the following words:

1. plant

[46] A *modifier* is something that further specifies a particular entity or event: e.g., *big house*, *shout loudly*.

2. shower
3. bass

If possible, compare your answers with another student's and with a dictionary.

Using the BNC Simple search (or another suitable corpus search tool: i.e., one that doesn't weight the results returned in any way), go through at least 10 sentences that include the verb *find* and consider whether it could have been replaced by *discover*. You might like to distinguish between cases where the example 'sounds strange' but where meaning is preserved from cases where the meaning changes significantly.

E.2 Post-lecture

1. Give hypernyms and (if possible) hyponyms for the nominal senses of the following words:
   (a) horse
   (b) rice
   (c) curtain
2. List some possible seeds for Yarowsky's algorithm that would distinguish between the senses of *shower* and *bass* that you gave in the pre-lecture exercise.
3. Choose three nouns from WordNet with between two and five senses. For each noun, find 10 or more sentences in the BNC which use that noun and assign a sense to each occurrence. Swap the unannotated data with one or two other people and ask them to repeat the exercise independently. Calculate the percentage agreement between you. Discuss each case where you disagreed (if you've got a lot of disagreements, just look at a subset of the cases) and see if you can work out where your assumptions differed. Can you come to an agreement you all feel happy with on any of these cases? If you've done this exercise with three people, how many of these agreed decisions correspond to the original majority decision?
4. Why is it easier to disambiguate homonyms with Yarowsky's algorithm than related word senses?

F Lecture 6

F.1 Pre-lecture

Without looking at a dictionary, write down brief definitions for as many senses as you can think of for the following words:

1. give
2. run

If possible, compare your answers with another student's and with a dictionary. How does this exercise compare with the pre-lecture exercise for lecture 5?

F.2 Post-lecture

1. Using the BNC Simple search (or another suitable corpus search tool: i.e., one that doesn't weight the results returned in any way), find 10 or more sentential contexts for *shower*. For each of the different notions of context described in the lecture, find the features which a distributional model might associate with *shower*. You may want to use an online dependency parser: the Stanford dependency format is one of the most popular approaches (see `nlp.stanford.edu/software/stanford-dependencies.shtml`, online demo at `http://nlp.stanford.edu:8080/corenlp/`). If you use an online parser, note that the output is unlikely to be perfectly accurate.

2. There are a number of online demonstrations of distributional similarity:

   - http://swoogle.umbc.edu/SimService/
   - http://www.linguatools.de/disco/wortsurfer.html
     This is described in http://www.linguatools.de/disco/disco_en.html. The interface to the demo seems to be German only, but should be obvious (you can choose 'Englisch' searches).

   Search for the words that you wrote down definitions for in the pre-lecture exercises for this lecture and lecture 5. Do the similarities make sense? Do they suggest any senses (usages) that you missed? Were any of these also missing from the dictionaries you looked at?

3. List some possible advantages and disadvantages of using Wikipedia as a corpus for experiments on distributional semantics compared with:

   (a) The BNC

   (b) A 2 billion word corpus of American newspaper text

   (c) The UKWac corpus (see http://www.sketchengine.co.uk/documentation/wiki/Corpora/UKWaC)

   (d) The Google 5-gram corpus http://googleresearch.blogspot.co.uk/2006/08/all-our-n-gram-are-html

G Lecture 7

G.1 Pre-lecture

G.2 Post-lecture

The goal of distributional word clustering is to obtain clusters of words with similar or related meanings. The following clusters have been produced in two different noun clustering experiments:

Experiment 1:

carriage bike vehicle train truck lorry coach taxi
official officer inspector journalist detective constable policeman reporter
sister daughter parent relative lover cousin friend wife mother husband brother father

Experiment 2:

car engine petrol road driver wheel trip steering seat highway sign speed
concert singer stage light music show audience performance ticket
experiment research scientist paper result publication laboratory finding

1. How are the clusters produced in the two experiments different with respect to the similarity they capture? What lexico-semantic relations do the clusters exhibit?

2. The same clustering algorithm, K-means, was used in both experiments. What was different in the setup of the two experiments that resulted in the different kinds of similarity captured by the clusters?

H Lecture 8

H.1 Pre-lecture

A very simple form of semantic representation corresponds to making verbs one-, two- or three- place logical predicates. Proper names are assumed to correspond to constants. The first argument should always correspond to the subject of the active sentence, the second to the object (if there is one) and the third to the indirect object (i.e., the beneficiary, if there is one). Give representations for the following examples:

1. Kim likes Sandy
   Answer: like(Kim, Sandy)
2. Kim sleeps
3. Sandy adores Kim
4. Kim is adored by Sandy (note, this is passive: the *by* should not be represented)
5. Kim gave Rover to Sandy (the *to* is not represented)
6. Kim gave Sandy Rover

H.2 Post-lecture

1. Using the sample grammar provided, produce a derivation for the semantics of:
   - Kitty sleeps.
   - Kitty gives Lynx Rover.
2. Extend the grammar so that *Kitty gives Rover to Lynx* gets exactly the same semantics as *Kitty gives Lynx Rover*. You can assume that *to* is semantically empty in this use.
3. Go through the RTE examples given in the lecture notes, and decide what would be required to handle these inferences correctly.

I Lecture 9

I.1 Pre-lecture

There is an online experiment to collect training data for anaphor resolution at `http://anawiki.essex.ac.uk/phrasedetectives/`. Spending a few minutes on this will give you an idea of the issues that arise in anaphora resolution: there are a series of tasks which are intended to train new participants which take you through progressively more complex cases. Note that you have to register but that you don't have to give an email address unless you want to be eligible for a prize.

I.2 Post-lecture

1. Take a few sentences of real text and work out the values you would obtain for the features discussed in the lecture. See if you can identify some other easy-to-implement features that might help resolution.
2. Try out the Lingpipe coreference system at `http://alias-i.com/lingpipe/web/demos.html`
3. The following text is taken from the Degree Confluence Project (note, it was written by a German speaker and is not fully grammatically correct).

28-Jul-2013 – A trip into the beautiful region Spreewald in the state of Brandenburg gave opportunity to visit the Confluence 52° north – 14° east.

By taking the interstate road B87 from southwest you reach the village Biebersdorf about 6 km after leaving Lübben. In Biebersdorf the in the beginning paved "Groß-Luethener Weg" leads quite directly northeast to the Confluence. On unpaved paths you can reach about 180m from the CP.

From now on it was walk by foot on a tighter path. The last 78 m must be done by walking through a tight planted pine forest. At luck the distance between the rows is wide enough to walk nearly unobstructed to the point. Only a grass snake and many little white butterfly moths were crossing the way.

After all that I reached the Confluence Point at 12.10pm in a dry heat of about 33°C. Despite the blue sky the GPS showed a precision of 5m by use of GPS- and GLONASS-signal. Surely the tight vegetation was the reason for the dilution of precision.

On the way back a pleasant visit of the city Lübben with its beautiful island "Schlossinsel" was taken in, instead of a Bratwurst you can have a perfect gherkin (kind of spicy cucumber) there.

Give rhetorical relations between each sentence pair using the set: NARRATION, BACKGROUND, ELABORATION, CONTINUATION, RESULT, EXPLANATION. Examples of each relation are given below:

NARRATION: Klose got up. He entered the game.
ELABORATION: Klose pushed the Serbian midfielder. He knew him from school.
BACKGROUND: Klose entered the game.The pitch was very wet.
EXPLANATION: Klose received a red card. He pushed the Serbian midfielder.
RESULT: Klose pushed the Serbian midfielder. He received a red card.
CONTINUATION: Klose received a red card. Ronaldo received a yellow card.

Compare your annotations with the annotations done by one or more other people. What is the percentage agreement? Can you find a way to resolve disagreements?

J Lecture 10

J.1 Pre-lecture

J.2 Post-lecture exercises

1. Suppose a town is equipped with a sensor system on its major commuter routes which monitors vehicles passing. A NLG system is to be designed to provide brief reports to motorists on current traffic conditions. Describe the tasks involved in such a system, using the categories in the lecture notes, giving appropriate examples.

2. What other NLP techniques that we discussed in the course could be useful in summarisation, and how?

K General questions

K.1 Ambiguity

Many jokes depend on ambiguity: for each of the following examples, specify the type of ambiguity involved and give as detailed an analysis as you can of the different readings.

1. A: Your dog's chasing a man on a bicycle.
   B: Don't be silly, my dog can't ride a bicycle.

2. Drunk man: I'm going to buy a drink for everyone in the bar.
   Other man: That's good of you, but don't you think they'll argue over it?

3. Haughty lady in a post office: Must I stick the stamp on myself?
   Post-office employee: I think you'll accomplish more, madam, if you stick it on the package.

4. Customer: Will my burger be long?
   Waiter: No sir, it will be round and flat.

5. What did the barman say to the ghost?
   Sorry sir, we don't serve spirits here.

L Answers to some of the exercises

L.1 Lecture 1 (post-lecture)

Something like this experiment was tried by Pang et al (2002) to provide a baseline for their machine learning system. The table below shows the accuracy they obtained on movie reviews by counting the positive and negative terms in the document. The third set was obtained with the help of preliminary frequency data: note the inclusion of '?' and '!'.

	Terms	Accuracy
Human 1	positive: *dazzling, brilliant, phenomenal, excellent, fantastic* negative: *suck, terrible, awful, unwatchable, hideous*	58%
Human 2	positive: *gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting* negative: *bad, cliched, sucks, boring, stupid, slow*	64%
Human 3 (with stats)	positive: *love, wonderful, best, great, superb, still, beautiful* negative: *bad, worst, stupid, waste, boring, ?, !*	69%

L.2 Lecture 2 (pre-lecture)

1. (a) carries
   carry (stem) s (suffix)

   (b) running
   run (stem) ing (suffix)

   (c) uncaring
   un (prefix) care (stem) ing (suffix)

   (d) intruders
   intrude (stem) er (suffix) s (suffix)
   Note that in- is not a real prefix here

   (e) bookshelves
   book (stem) shelf (stem) s (suffix)

   (f) reattaches
   re (prefix) attach (stem) s (suffix)

   (g) anticipated
   anticipate (stem) ed (suffix)

2. (a) carry
   Answer: simple past *carried*, past participle *carried*

   (b) sleep
   Answer: simple past *slept*, past participle *slept*

   (c) see
   Answer: simple past *saw*, past participle *seen*

L.3 Lecture 3 (pre-lecture)

1. The/Det big/Adj cat/Noun chased/Verb the/Det small/Adj dog/Noun into/Prep the/Det barn/Noun.
2. Those/Det barns/Noun have/Verb red/Adj roofs/Noun.
3. Dogs/Noun often/Adverb bark/Verb loudly/Adverb.
4. Further/Adj discussion/Noun seems/Verb useless/Adj.
5. Kim/Proper noun did/Verb(aux) not/Adverb(or Other) like/Verb him/Pronoun.
6. Time/Noun flies/Verb.
   Time/Verb flies/Noun. (the imperative!)

L.4 Lecture 4 (pre-lecture)

1. The big cat with black fur chased the dog which barked.
   ((The big cat)$_{np}$ with (black fur)$_{np}$)$_{np}$ chased (the dog which barked)$_{np}$
   The big cat with black fur (chased the dog which barked)$_{vp}$
2. Three dogs barked at him. (Three dogs)$_{np}$ barked at (him)$_{np}$ Three dogs (barked at him)$_{vp}$
3. Kim saw the birdwatcher with the binoculars.
   Analysis 1 (the birdwatcher has the binoculars) (Kim)$_{np}$ saw ((the birdwatcher)$_{np}$ with (the binoculars)$_{np}$)$_{np}$
   Kim (saw the birdwatcher with the binoculars)$_{vp}$
   Analysis 2 (the seeing was with the binoculars) (Kim)$_{np}$ saw (the birdwatcher)$_{np}$ with (the binoculars)$_{np}$
   Kim (saw the birdwatcher with the binoculars)$_{vp}$

L.5 Lecture 8 (pre-lecture)

1. Kim sleeps
   sleep(Kim)
2. Sandy adores Kim
   adore(Sandy, Kim)
3. Kim is adored by Sandy
   adore(Sandy, Kim)
4. Kim gave Rover to Sandy
   give(Kim, Rover, Sandy)
5. Kim gave Sandy Rover
   give(Kim, Rover, Sandy)