NaturalLanguageProcessing-Lecture10

Instructor (Christopher Manning):Okay. Hi, everyone. And so today, what I'm gonna be doing is moving on to starting our material and parsing, and this is the start of the section where we move on to actually dealing with whole sentences, so up until now, we've done various kinds of things where we've had language models and information extraction, things like that, but essentially we've been getting or predicting little bits of information about sentences and well, if we really want to [inaudible] that for language understanding, it seems like we really want to know something about the structure and the analysis of entire sentences. And so that's the part of the class that we start on today. And in particular, we start on to the topic of how we can pass, meaning giving a suitable structure the sentences. And that's seen as then a prerequisite for them, learning more about the semantic analysis of sentences. Okay. So parsing is an area that's gone through a huge amount of much success and good change over the last 15 or so years. So there's traditional work on parsing, which was grammar based, [inaudible] based parsing and then in the last decade or so, there's then been this great amount of work in doing statistical parsing. So I just want to quickly say a little bit about the flavor of that and, then mainly, we'll be going into statistical methods of parsing, and I'll say a bit about traditional parsing. So in the 1990s there were – really for the entire history of NLP there's been lots of work on how can you parse natural language sentences, but still in the early 1990s, essentially, what you saw is that there were these parses that hand written grammars which produced detailed, actually usually richer than we often provide today, grammatical analysis of sentences that if you actually tried to run these parses and just pick up this morning New York Times cover page article and start trying to parse it, what you found is that these parses had very uneven and usually poor coverage.

The coverage would be quite poor. In particular, commonly the parses fail to return any analysis at all, but quite a large percentage of sentences. On the flip side when sentences could be parsed by the grammar, normally sentences would have many parses, and most of these parses actually had no good way of dealing with that. Some of them had literally no way of dealing with that and said this sentence has 1,027 parses and others of them would use various kinds of heuristic and adhoc rules to favor certain kinds of analysis over others. Another problem was lots of people had lots of different parses, but there wasn't really way to, apart from kicking the tires and trying it on some things, there really wasn't much of a quantitative way to actually get a sense of how good different parses were, partly, I mean, you could sort of certainly measure on what percentage of sentences the parses failed to return anything, but you kind of obviously want to know more, you want to know if they do return something, is it correct and what's the quality of the analysis, etcetera, etcetera. Okay. But then over the last 15 years there's then been this sort of revolution of people doing statistical parses which are trained from data and, in practice, normally the same data which is made for an easy case of parses being commensurable and you can see exactly how they differ from each other, and so effectively, now there's basically a commodity component of statistical parses and a whole bunch of people have written statistical parses, many of which you can download and run open source. So that's lead to a new generation of people being able to use parses

because now, essentially, there's these good statistical parses, but it doesn't mean that they always get everything right all the time, but most of the time, get most things right.

This can be used and they always return some analysis for every sentence, so they make reliable components for what they deliver and so there's no starting to be a ton of work where people are putting parses inside other bigger systems because it's fairly straight forward to do and so some of the kinds of applications you see are for inside [inaudible] systems and I'll talk about those more later, but normally, they do full parsing on both questions and candidate passages and documents that might contain an answer. You can actually use parses to get a better sense of sentence structure to improve parse [inaudible] recognition that I talked about before. In particular, in biological identity recognition, I mentioned how there's often a lot of ambiguity between where the things were, genes or gene products like DNA and RNA and looking at the sentence relationships they figure in there's actually a good use of information. For sentence compressional summarizations – so if you want to have a shorter version of a sentence to fit on your mobile phone screen or something like that it's good to have an idea of which bits of the sentence structure go together and perhaps kind of parenthetical clauses so you can delete parts and still have a grammatical sentence that conveys the main content. Product mining, opinion mining for products so this is the kinds of things that if you want to evaluate cell phones and know why people like the iPhone that you want to get some kind of idea of sentence structure so you know what opinions are being predicated of the iPhone and about what attributes of the iPhone, etcetera, etcetera, lots of applications. Okay. So we're gonna do parsing, what's difficult about parsing. So this kind of comes back to the very first lecture when I had my whole bunch of headline sentences of funny ambiguities and parsing, which I won't go through in great detail, but I'll just do this one sentence here.

"I saw that gasoline can explode." So that has two readings, right? One is that there's a gasoline can, and I saw that gasoline can explode, and the other one is with the sort of fact compliment, "I saw that gasoline can explode," as a concept. And the important thing to notice here, for what we're gonna do, is that a sentence like this is just globally ambiguous in terms of syntactic structure. It's not like what we have in programming languages. The programming languages will have local ambiguity that sometimes you need to look at the next token to be able to decide how to parse something out. The programming languages are very carefully designed so they only have local ambiguities, in particular, older standard programming language, syntaxes are such that you just need to look at the one next token and that gives you sufficient information to resolve all ambiguities. Well, here, you get to the end of the sentence, there's still nothing syntactically to resolve the ambiguity. The only thing that allows humans to resolve the ambiguity is using their context of how the world is. And there are interesting cycle linguistic angles there which people still debate as to human beings, how much they're doing syntactic structure processing before recovering the semantics versus how quickly their integrating the semantics as they go along. And there's some evidence both ways, but I think more and more people are thinking that there's a lot of early semantic integration as humans parse. Okay. So in classical NLP parsing we wrote down a symbolic grammar and lexicons so here's my very little baby grammar. You can essentially say that what we then did was use the proof system to prove that certain parses

follow from the words, or if you would prefer you can think of it as a search system in a classic AI search system so we search for possible parses from the words. But the problem with either of those techniques is that they scaled really, really badly. So I didn't actually put it up here, but if you remember, again, from the first lecture there was that sentence where I showed some parses of the Feds raises interest rates half a percent in order to lessen inflation or something like that, but it seems like if you kind of write the smallest grammar that you can that will reduce the parses that I showed for that sentence, it actually produces 36 parses for that sentence, which already seems a kind of a frightening number. I tried it out on a small context free grammar that I'd written for other purposes, which only had 10 rules in that and that 10 rule context free grammar gave 592 parses to the sentence.

In fact, as we'll discuss later for certain ways of building tree banks from building grammars from tree banks, but technically, the kind of grammar you're using actually licenses an infinite number of parses for sentences because it can just kind of build infinite structure up above sentences. Okay. So the problem with non-probabilistic parsing was that there essentially wasn't a good way to solve that problem. So if you looked at your 592 parses, the obvious thing to think was, "Well, maybe I can put more constraints on when rules can apply so I can get rid some of those weird and unlikely parses," but the problem was that on the other hand the more that you put in those kinds of constraints the more there are sentences that you failed to parse at all, and so you had this coverage problem. And going along with that there's then some clever algorithms and programming to make sure that despite the fact that you, theoretically, have an infinite number of pauses, in practice you can actually find the good parses very quickly. Okay. And so what's enabled statistical parsing? Part of you can obviously go back to the whole history of AI and [inaudible] tradition and people weren't very imperial in ways of thinking that dominated AI and were reflected with what people did with NLP, but as well as that, I mean a huge requirement of being able to do probabilistic parsing is actually to have the date to make it possible. So you look at the history of statistical methods and parsing in and around the late 80s and early 90s there was already an intense interest in using probabilistic and machine learning methods for NLP and people started to do some work looking at unsupervised methods of learning parses. But the problem is that those methods just didn't work very well. Now, much more recently, I myself, with students have done some work on unsupervised parsing methods that work a lot better and I can mention that a weeks hence, but the least of that time the attempts to do unsupervised parsing were just really, really poor. And so what really transformed the ability to do statistical parsing research was actually having supervised data, and to this day, if what you want is a high performance statistical parsing, it's trained on supervised data, and so it's really the building of treebanks that enabled this whole line of enterprise to flourish. And so what is a treebank? This is one sentence from a tree bank. So what a treebank is a first approximation, you get people who are willing to work long hours for low pay and you give them sentences and they have to put syntactic structures over sentences. So this was the most famous – there are now kind of a lot of treebanks at different places, but the most famous treebank is still by far the most widely used one is the pin treebank, which was done at the University of Pennsylvania starting in the late 80s and it kind of became generally available to people in 1993. So they spent several

years with several students and other research staff putting tree analyses over about 50,000 sentences.

Most of the work was actually done by classic student. It turns out the classic students, I guess some combination of they have a good understanding of traditional grammar, you know, nouns and adverbs, they don't have a lot of other employment opportunities and it turns out linguistics argue too much. They're always dissatisfied with the analysis of different things and want it done differently and better, whereas classic students are more willing to go with okay, these are the rules and this is how you're meant to do things. So, essentially, what a pin-tree bank tree gives you is this context free grammar or phrase structure grammar analysis. It actually gives you a little bit of extra stuff, and so it's worth just being aware of some of the extra stuff. So there's basic patterns for sentences and noun phrases and PP here. Sometimes those categories are given a little bit of extra information about their function, so this is saying, this is an adverbial sentence so it sort of adverbially modifying sentence to the main clause. This noun phrase here is the subject of the sentence. This PP here is [inaudible] prepositional phrase, so those are referred to as functional tags. Most of the time when people are doing basic context free grammar parsing they just strip them off as they read in the sentences and that's what our pre-processing will do for the parses that you guys build. But then the second bit is there's then under the NP subject there's this non star because, actually, if you read this sentence the move followed…reflecting a continuing decline in that market, but the point here is there isn't actually any direct subject for the verb reflecting that this is part of the whole sentence, and really what the subject is is this whole big sentence up here. So the pin treebank also [inaudible] various empty nodes, which are kind of scene as syntactic structural elements that aren't directly pronounced words. Okay. So in retrospect, building treebanks has just been a marvelous thing for the field of natural language processing and it's kind of not clear why it didn't get done two, three decades ago because even if you weren't thinking very probabilistically, in retrospect, it seems like well, solely having treebanks would just be a marvelous idea because then everybody could evaluate their parses and see how good their coverage was and compare it. We have a good sample of English so we can see what phenomena parses do and don't cover and where there grammar should be extended. It seems just a generally useful concept, but I guess a lot of things seem a good idea in retrospect, but somehow weren't obvious to people at that start. Treebank building seems like a very inefficient, painful and expensive process to understand, but in retrospect, it's really turned out to be the opposite because if you look at the history of NLP that's sort from 1965 to 1990, approximately, but if you add it all up in person years, you can kind of identify dozens of projects were people built grammars and parses to go with those grammars and in total spent 10s of persons years on those activities of coming up with good parses, but the legacy of that work is essentially zero.

There's some conceptual legacy, but in terms of having something useful for what we do today, the legacy is basically zero because every one of these grammars were written in some different formalizing by some different person and have a lot of nothingness in [inaudible]. But more and more these days there are people doing linguistics when you want to investigate a phenomena and it doesn't answer all questions. One of the easiest

and best ways to get a reasonable idea of what other possible variants of this construction that occur in actual usage is actually to go and look in the pin treebank and find examples of that structure. So there are various tools that are then available including [inaudible] which are effectively then pattern languages where you can search over pin treebank trees. Okay. Well, what is linguistics structure? I'll come back to the alternative later, but I just want to mention the two main kinds of linguistic structure people use. The kind I've already drawn and shown you what the pin treebank natively has is constituency structure, otherwise known as phrase structure, otherwise known as a context free grammar. So if you draw quickly as a tree, you get something like this, "Fed raises interest rates." Okay. So this kind of structure is all about making claims that certain things are constituent, so it's claiming "interest rates" is a constituent. And constituent just means something that goes together and acts as a unit and so if you really want to know a lot about this you have to do a linguistics class, and that's the fun. You can go into a syntax class and learn all about how people work out constituency and why different theories do different things, but in very brief – you know – these are the kind of evidence that people used for constituency so that if you have groups of words that kind of sort of move around as a unit that's evidence that they are constituents. So you can say, "John talked to the children about drugs or John talked about drugs to the children," so the fact that you can swap those two units makes it look like both of them are constituents. Other ideas of constituency are that if things are constituent that normally you can expand it and contract it in various ways with things that are the same semantic types. So from, "I sat on the box," you can expand into a bigger propositional phrase, "I sat right on top of the box," or you can shrink it down to a single word that can act in the place of a prepositional phrase as, "I sat there." So these kinds of words are called proforms.

The most common and ubiquitous fall of proform is pronouns, which you can use instead of noun phrases, so that's he, she, it, kinds of words. If you also get proforms like [inaudible] like prepositional phrases. And there are lots of other tests, coordination, blah, blah. Okay. So there are two claims here. I forgot the second one. The first one is that this acts as a unit, but the second claim is then with the we can have types of these units. So the claim is this is a noun phrase and that means that there's a class of constituents that behave the same, meaning that they can appear in the same kinds of a positions. So the claim is that interest rates is a noun phrase and fed is a noun phrase so that they should be substitutable without mauling the syntactic structure. So you should be able to say interest rates raises fed, and it's kind of hard to make semantic sense of that, but it's syntactically well formed with fed being a proper noun. It's a good syntactic structure. Then here's the other alternative, which is dependency structure. So the idea of dependency structure is that it shows you which words are direct arguments or modifiers of others words. There are at least a couple of ways of representing dependency structure. This is actually the original and traditional way of doing it when it was proposed with errors. And here's a way of doing dependency structure as a tree. So what you're doing – here we're following out arrows to their points to the tail of an arrow is where the head is and the pointy bits point to the dependents. So here "put" has three dependents, boy, [inaudible], on and so they are the three arguments of the putting of [inaudible]. Boy has a dependent of the, because it's "the boy" as does tortoise and then on has a dependent of

rug, which has a dependent of the, but you can also represent like this, so put has its three dependents and then they have their dependents under them.

Student:[Inaudible]

Instructor (Christopher Manning):So that is a good question and a good thing to think about and the answer goes like this. If you look at this graph, its right property is that there are no dependencies that cross. If you have no dependences that cross, that means your dependencies are such, that means for the implied constituency of that dependency graph is always a tree and so yes, you can convert it into a tree. So your question comes down to do you get dependencies that cross. And the answer to that is yes, you do, but not very much. And you might think that that's an unsatisfying answer or not, so the kinds of place that you get dependencies that cross is with special constructions like questions. So if you have a sentence like, "What did you buy at the store yesterday," and you say, okay, what, that's serving as the object of what was bought, and therefore bought will be taking a subject of you and an object of what, but then it will also take an argument of where, at the store, and an argument of when, yesterday then the what are is gonna cross over the when and where archs and you have crossing dependencies, which means that you can't represent it as a tree, but you know, essentially – in natural languages, crossing dependencies are rare. They occur in these special kinds of circumstances so they appear in English when you've question words like that are fronted to the front of the sentence, which is sort in some sense is a weird function of English. That in terms of sort of the normal placement of words what you should say is "You bought what at the store yesterday?" Right? That'd be kind of maintaining the normal placement of words. And you can say that kind of question with some question intimation and people sometimes do, but it's not the usual way to say. The questions in English have this funny, mock syntactic word ordering. What do people do about that? It varies, but what the pin treebank does about that is it represents it as a tree anyway. So in the pin treebank and context free grammar parsing, everything is always represented as a tree, and things that don't fit into that tree structure are handled by other mechanisms.

So if you have something like what did you buy at the store yesterday, the what would be just attached to the sentence as okay, here's a WH word, start at the sentence and it would be given tree structure and then the connecting between what and the object of by would be done with these kind of empty nodes and co indexing that I was mentioning earlier. It turns out that you can use fairly simple statistics to work out these attachment decisions. The first way in which that was explored, which to a first approximation turned out to be the wrong way, was to say maybe we can use straightforward syntactic factors to decide what are likely attachments or even what is the attachment that a human being will choose. So this works was actually [inaudible] in a psycho linguistics community, and so early on, people proposed structural factors of sentence structure which would determine how humans would resolve these syntactic ambiguities of PP attachment, so in particular, Kimble in 73 proposed the principal of right association, which said that you should attach a PP low or near, so that gave you early closure. So Kimble would say okay we've got with our telescope how you should attach it is to the low nearby thing, the man, and prefer this structure. Not that different in time. There was actually then a Lyn Frazier

another psycho linguist, she proposed the principal of minimal attachment, and minimal attachment, what it predicts actually depends on the details of your grammar structure, but for the particular kinds of grammars that she was considering – low attachment it turns out actually predicts, in this case, exactly the opposite, so it would be predicting that the PP should be modifying the verb phrase. So, obviously, both of them can't be right in what their predictions are and so the obvious question is to say, "Which of them is right?" And the answer is sort of sometimes one, sometimes the other. There is a structural preference so if you're just gonna have this as the only factor in your model, that the actual statistics run in favor or Kimble. It's more common to do attachment low to the nearest thing. But that's not a very strong predictor. It depends on what corpus that you count over and what the nature of the writing is about etcetera, etcetera, but it turns out that most of the time, in English, it's somewhere between 55 and 66 percent of PP's attach low to the nearest thing next to them.

So that's a little bit of a preference. That helps a little, but it's clearly not a very strong preference. I guess there was a certain blindness I guess in the kind of work that was done in these days where basically everyone was very infatuated with Chauncey's grammars and looking at grammar structures and trying to come up with structural factors that would predict things, but no one was paying very much attention to words. And so it really wasn't until about 1980 that people made the, in retrospect, kind of obvious observation. You can get a lot of information just by looking at the words, so even with a kind of a made up example like this, "The children ate the cake with a spoon," versus, "The children at the cake with frosting," you can kind of think, "eat spoon," that's probably pretty common, "eat cake spoon," probably not so common – I guess it occurs, but overall, eats spoons probably more common, cake frosting, that's very natural, eat frosting, possible, probably happens less. It almost seems like the words should be used, "more useful," and really this is the kind of example that's made to be ambiguous. If you actually kind of take where real text sentences that most of them seem that the words are much more strongly determined things. Okay. So that suggests a really simple way to work these things out so maybe we can just do a likelihood ratio. So we can have a likelihood ratio of the probability of seeing a certain proposition given the verbs so that's kind of breached with, sounds unlikely, versus the probability of the preposition given the noun, agreement with, that sounds pretty good and if you do some data counts, you find that out that you've got a lot more of agreement with so your likelihood ratio favors about seven to one going with a noun attachment.

So that works out fine. Does this always work? Well, just doing that doesn't always work, so here's an example where it goes wrong. "Chrysler confirmed that it would end its troubled venture with Maserati," so I got some statistics where I calculated from a fairly big corpus so the counts of the words concurring with with I do my divisions to get kind of maximum likelihood estimate probabilities and if I do that it's kind of close, but winning by a hair is, "end with," occurs more often than, "venture with," and that's because the phrase end with does occur a lot in context like theater or musical shows. The performance ends with whatever it is. And so if I just use my likelihood ratio statistic I'd get it wrong. I'd predict that in this example that with Maserati would be modifying ends rather than venture and that's clearly wrong. Now, I mean, of course you could just react

to them and say, “Well, you know, that’s life in statistics, you make a prediction and sometimes you get it, sometimes you get it wrong. You found an example of where you got it wrong, what’s the problem? And well, to some extent, that’s true, but it seems like the problem is more – there is a deeper problem here that suggests that we want to do something different because yes, it’s true that, “ends with,” occurs quite a lot with my examples, like, “The show ends with an amazing spectacle,” but if you kind of think about those sentences, basically, they’re actually in transit of sentences. Those are sentences were end isn’t taking any direct object at all. It’s just this, “ends with,” standing ovation that you’re getting kind of an intrinsic use of ends within some PP after it. So it seems like somehow we should be using that bit of information. It turns out that there’s actually another good bit of information that we should be using as well.

So that if we construe with Maserati as modifying “end” and if we take out that we don’t have a grammar that is a context free grammar that’s tree structured so we don’t have any of these crossing dependencies, well then, if we take that analysis we’re also making an implicit decision that venture doesn’t have any material flowing because then it couldn’t have any material falling out that wouldn’t involve a crossing dependency. So it seems like for this example what we’re really doing if we an attachment of with to end is we’re implicitly also making a decision that venture doesn’t have any modifiers coming after it. So, essentially, when we do parsing, what we’re doing is saying, well, it’s not just enough to take one particular fact like what should this prepositional phrase modify and count up steps and do some likelihood ratios and make the best guess because, actually, coming up with a parse for a sentence is a big joint decision and any decisions that we make about one thing influence determine a lot of other things so that if we’re saying that with modifies ends then we’re implicitly making a decision that nothing modifies venture and that’s maybe unlikely. There are all of these decisions meshed together to come up with the right analysis of the sentence. So, essentially, what we want to do is come up with some kind of joint model where you can make all of these parsing decisions at the same time and come up with the best global analysis for a sentence. Okay. And so, I mean, part of what we’ll want to do is use more information. So, typically, if you want to do a better prepositional phrase, attachment model, that we’ll want to know what the verb is, what this noun head is, what this preposition is and also what that noun head is. That’s kind of the minimal amount of information that you can have to make a sort of a semi decent prepositional phrase attachment analysis. Here’s just one example from a real sentence to show that prepositional phrase attachments and practice actually get very complex so if you, not every sentence, but if you just look through a bunch of news wise sentences you can go and read the paper when you go home and feel good about it that there’s just tons of sentences that, essentially, just have series of prepositional phrases. So here’s one that I picked out from a boring article where what we find is there’s just this sequence of four prepositional phrases after the basic verb. The thing to notice is that they actually kind of have this sort of weird pattern of dependency.

So the board approved its acquisition, so Bi-World Trustco Limited, theoretically, couldn’t modify either the verb or the noun, it modifies the nearby nouns, so it goes with Kimble – right, of Toronto then modifies the noun phrase inside the prepositional phrase, that also goes with Kimble. That modifies the nearest thing. So we’ve sort of showed lots

of examples that if you had verb/noun prepositional phrase you got two parses, which correspond to the verb and the noun attachment. If you have verb/noun/prepositional phrase/prepositional phrase – how many parses do you get? So that's up to here. So this one can modify [inaudible] and then this one can modify here or here or here. Then there's this slight trickiness because we're insisting that it's a context free grammar so the modifications have to be nested. If you had no constraints, how many possible modification structures could we have?

Student:[Inaudible]

Instructor (Christopher Manning):No. Well, is it? No, you're right. We do. Sorry. Yes. Yes. Okay. Right. You're right. Sorry. If there are no constraints, this one has two, this one had three, this one has four, this one has five, so yes, imperatorial. Sorry. Okay. We do get a constraint because, effectively, here because of the nesting relationships that we effectively lose one, so that for two PPs, the number that we actually get is five. And if you want you can try drawing that on a sheet of paper and check to see if you get five of them and then if you go to then have three PPs, the number that you get is then 14 and it goes up like that. So, actually, the number that you get is the [inaudible] numbers, which effectively, is a series, which you can look up on Wikipedia if you're into these sorts of things. It's a series of numbers that turns up in a lot of places where you have tree structuring. So somehow we want to deal with having all of those attachments and still parse efficiently. So parsing, basically, means take grammar, take sentence. Now, the grammar is written down to sort of generate sentences. It says it goes noun phrase/verb phrase – noun phrase goes to determine a noun. So what we want to do is run the grammar backwards and say, "Okay, here's a sentence, what underlying structures could be behind the sentence?" And so you can think of that as a search problem, you can think of it as a hidden data inference problem. So here's our simple little phrase structure grammar. Normally, by convention, people call S the start category of their phrase structure grammar and when it's not written down you assume that. Actually, if you look at the pin treebank, it has an extra node above S, which is normally referred to root or top, which is actually really useful in practice because it turns out that not all sentences and newswire are sentences.

Sometimes they're just a noun phrase or a fragment like a prepositional phrase so you'll have a headline that's something like, "Off with his head," or something and that's just a prepositional phrase. This is stuff you should've seen in some class where we have context free grammar with a set of terminals, a set of non-terminals, a start symbol and a set of production rules that we write them. I will mention only one qualification on that is in practice for the way natural language grammars are written because of the way natural languages are, is that people always make a very clear distinction that isn't in the traditional CFG formalism between pre-terminals, which are categories that rewrite as actual words versus the higher-level structural units. So the pre-terminals are our parts of speech like noun and verb and adjective and pronoun. So pre-terminals are non-terminals so they're a subset of N, but they always rewrite as a single word and then you have other categories like noun phrase/verb phrase/sentence, which will rewrite as some combination of other non-terminals and pre-terminals. And people do that just because

it's a manifest fact of natural languages. These are the traditional properties of grammars so you have – I'll say they're parses, a parse is sound if it doesn't screw it. Everything it returns is something that should be built by the grammar. It terminates if it doesn't go into an infinite loop and it's complete if you can give it a grammar in a sentence and it produces every valid parse for the sentence and terminates. Okay. So I'm just gonna kind of really, really quickly say how you do top down and bottom up parsing as search processes and then I'm gonna get into how people really do parsing as a dynamic programming process. Has everyone seen the top down, bottom up parsing in some class? Yeah. Are there people who haven't? If you haven't, you can read all about [inaudible] and I won't do it.

Okay. Essentially what I just want to clue people into are a few kind of [inaudible] compilers book, you get taught about top down parsing and you get information about bottom up parsing and then practice people, you know, normally use bottom up, [inaudible], that's not really effective for what we want to do. So I just want to kind of clue people into what the issues are there. The reason it's not effective is this issue of global ambiguities. Okay. So top down parsing, here's my little picture that's too small to read, but you just kind of have to see the shape of it. So what I have here in the outer tree is my search tree, and then the inner trees, which are too small to read, are then a representation of my state in the search. So I start with just my top, my start role and I then expand in every way that's licensed by the grammar, which is only one for the first rule, but then I expand the left noun phrase and I can expand it in three ways. I keep expanding down and I would eventually start producing words, so here I produce cats and here I produce people and I do this top down search through the grammar. Okay. So this is a possible thing to do. Actually, running top down until you actually produce words is obviously horrendously inefficient. No one ever did that. People would top down parse until you got to pre-terminals and then you do dictionary look up to see, "Okay, is the first word in the sentence clause, can it be parsed as a noun or something like that." So top down parsing has various problems.

A first problem is [inaudible] rules so if you have [inaudible] rules so if you have [inaudible] rules in your grammar, which natural language grammars just do, so basically everyone wants to have a role in their grammar that is noun phrase that rewrites as noun phrase/conjunction/noun phrase because you want to have, "The boy," as a noun phrase and you want to have, "The boy and his dog," as a noun phrase. And so that's a [inaudible] rule because the head category noun phrase is also the left most category. In programming languages, you're allowed to have shift reduced choices, which are always resolved in favor of shifting and that's what gives you [inaudible] goes with the closest [inaudible], but you're not allowed to have reduce conflicts. If you have those, your grammar compiler tells you that you messed up badly and go and fix that. And a reduced conflict then corresponds to what we're calling attachment ambiguity. So whether you have the prepositional phrase, go with the noun or the verb, then that's then a reduced conflict. What we want to do here is say, "Well, both of those are good analyses and we want to find all of them," which you could do by exploring the full search tree. And bottom up parsing, it also has various properties of good and bad things, but essentially, what I want to just get to is this repeated work property. So here is my wonderful picture

that I slaved on PowerPoint, which is meant to show you, in very small print, the problem of repeated work. It's a little hard to get a sense of the problem of repeated work and why it's the killer problem when you're just looking at five word sentences. So I've labored a bit further and I now have this seven-word sentence, "Cats scratch people with cats with claws," which is licensed by my same teeny grammar. It just starts to be big enough to show the problem of repeated work. But, essentially, as your sentences get longer, the average length of a sentence in something like the New York Times or the Wall Street Journal is around 23 words. There's sort of this skewed distribution. You start getting quite a lot of probability matches around 15 words and then it stays pretty high through about 50, and then it kind of really tails off when there's some sentences that go out into a 100, 200 words. So as you start getting into longer real sentences like those 20 or 30 word sentences, what completely kills you is this problem of repeated work.

Okay. Well, what we should do is come up with a dynamic programming algorithm to do parsing and that just is the standard solution and I'm gonna present, in particular, the method of parsing which is called CKY Parsing. This is an old traditional method of parsing invented in the 1960s. This parsing is the later science. Of course, it's not the only way to solve this problem and I'll mention, quickly, a couple of alternatives, so from AI class you could say, "Well, you're doing tree structured search. If you were doing a graph structured search then you could recognize these common sub structures and do things only once." And that's possible. A simpler way of doing it, where it's kind of nicely is to use the idea of memorization so if you make sure you remember work that you've already done, you can avoid the problem that way. Just one slide to talk about human parsing before we get back to machine parsing. Human parsing is kind of interesting and some people find it so interesting they do psycho linguistics as their profession. But if you think of different structures of sentences, well, humans have this problems [inaudible] exponential number of parses. What do they do? And it sort of seems like there's an interesting combination. On the one hand, there's lot of cases in which you have starts of sentences, which have many different continuations with different grammatical structures, and it seems like human beings just never notice. So if you start off saying the words "Have the police," well, those three words can have several different syntactic structures as these continuations show. So if the sentence says, "Have the police eaten their supper," well, that's a question where "have they," is an auxiliary and, "the police," is the subject.

Whereas if you have the sentence, "Have the police come in and look around," there the, "have," is an imperative verb and the, "police," is the object of having and if you have, "Have the police taken out and shot," that one's also an imperative style. So you get these different syntactic structures, but it doesn't seem to worry human beings when they start to listen to words. Okay. So I've only got a few minutes left now, so I'm probably not really gonna have time to show CKY dynamic programming algorithm for parsing this time, so I'll do that at the start of the next time. But I'll just start to introduce probabilistic context free grammars before we finish up. Okay. So most of what we're gonna be doing is parsing with probabilistic context free grammars and so, essentially, they're exactly the same as context free grammars that I mentioned before, apart from we add one thing to it. That is we've got a whole bunch of rules like in context free grammar and for each of

those rules we then give a probability to that rule. Whether these are probabilities of rules given their left-hand side, so the property that we have for these probabilities is that you have some category like verb phrase and the probabilities of all of the ways of rewriting verb phrase add up to one. So we might have verb phrase goes to verb and transfer verb, verb phrase goes to verb noun phrase transfer to verb, verb phrase goes verb noun prepositional phrase, it's all those and you take all of these rules for rewriting verb phrase and then some of their probabilities will add up to one. It's almost always true that we get a probability distribution and that means that our probabilistic context free grammar also defines a language model in precisely the sense of language models that we saw earlier, namely, that if you take the sum of the probabilities of all the sentences generated by the grammar, given this restriction, that what you get out is a language model. Okay. So what we're gonna do then is we've got words, we've got non-terminals dominating these words and we're gonna put probabilities on them and so I'll just show you at the end. So what we want to do is work out the probability of a string, and the probability of a string is going to be the sum of the probabilities of all trees that have that string as their terminals, which is often referred to as the yield of the tree. So if we have this baby example here of a probabilistic context free grammar and the NCNF means In Chauncey Normal Form, and I'll come back to what that means at the start of next time.

Here's a little PCFG, and so if we have a sentence like, "Astronomers saw stars with ears," this is a possible parse that we can give to the sentence. And I've just written next to each node the probability of the rewrite of what that node was according to the grammar then here's the other possible parse of the sentence. I always use these little PCFG examples, and so if we want to work out the probability of this word sequence as a language model, then this probability will be just the sum of the probabilities of the two trees. The probability of the tree is just taken by multiplying the probability of all the rules according to the grammar. So you get these two small numbers and you add them together and this is the language model probability of that sentence according to this PCFG. Okay. So I'll stop there for now and then right at the start of next time we'll then talk about how we can then work with a PCFG and do a dynamic program parse, which is an exponential.

[End of Audio]

Duration: 77 minutes