NaturalLanguageProcessing-Lecture18

Instructor (Christopher Manning):Okay, the last class of CS 224n for the year. And so today what I thought I would do is talk about one area of some of the applications and natural language processing in the area of question-answering systems, and then some related more general problem of doing textual inference. So there is kind of a modern tradition of doing textural question-answering, which really dates from about the early 1990s. And that's primarily what I am going to talk about today. So really, this modern conception of question-answering didn't so much come from the natural language processing community, but it came from the information retrieval community.

And if you think about it, what's happened with information retrieval as a field is, in a way, a little bit funny. But the name of the field is information retrieval, which suggests that you are meant to be returning information to people. But in practice, the field of information retrieval almost immediately got subverted. So what information retrieval as a field means 99 percent of the time is document retrieval, that people are going to specify an information need, and you are just going to hand them back documents. And so that there is no real attempt to actually find the information people want. So we are just giving people back this big unit.

I mean in – but that's kind of only of limited utility when you have huge collections of full-text documents. Because to the extent that you have these big, long things that are pages long, there is still a lot of work for humans to do in many cases to actually be able to find the answers. And so the suggestion was, well, maybe we should try and come up with a more focused task where the information need was quite richly specified was being specified as a full question. And then the answer wasn't going to be a complete document; the answer was only going to be something short, maybe just a snippet of text, or maybe just the answer.

In a kind of a different way, I think, actually the same thing is emerging as an application with what's happened with modern web search. That although the vast majority of web search is working, they are still type in a few keywords levels, and there is no reason that necessarily has to be anything other than that. But it turns out that for a lot of queries that have a kind of a clear answer these days, that you can actually – that the systems for giving people snippets on the results page, although those snippets were primarily intended, originally, for helping the human being decide which documents were really relevant ones that they can go off and look at, it turns out that these days the snippet selection procedures are sufficiently good.

But quite often, if you have got a question with a factual answer, you can just see the answer to it in the snippet that is displayed, and you never actually have to click through to the original documents. And that is sort of another scenario, which is realizing the same goal of can we give people just the information that they want for an answer rather than returning the full documents. And so this is seen as one way of addressing this general need of now that human beings are sort of effectively flooded with more

information than people can consume as to how to do more of the work for people to make their lives easier, so not all quite from a [inaudible]. I like the internet, really I do.

Any time I need a piece of shareware, or I want to find out the weather in Bogata, I am the first guy to the modem humming. But as a source of information, it sucks. You have got a billion pieces of data struggling to be heard, and seen, and downloaded. And anything that I want to know seems to get trampled underfoot in the crowd. Okay, so these are the kind of questions that people have dealt with in question-answering community which has been dominated by TREC quick-answering where TREC is, effectively, competition that does various IR and question-answering-like tasks that's been run by the National Institute of Standards and Technology in the U.S.

So there are questions like who is the author of the book The Iron Lady a biography of Margaret Thatcher? What was the monetary value of the Nobel Peace Prize in 1989? How much did Mercury spend on advertising in 1993? If you look at these questions, the majority of these questions are what have come to be called in this community factor-weight questions. They are questions that have a very clear answer which is a simple thing, so name of an author, monetary amount, another monetary amount for advertising. And that kind of question has been the one that has dominated a lot of this work. It's never been the only kind that's been involved in this work.

Effectively, it's been a level worth that's been kind of a match between what the technology has provided, and it's perceived as having a certain range of usages. Lots of us do web searches where we are wanting to look up someone's address, or phone number, or whatever. But they have always sprinkled into it other questions which are of rather more difficult kind. So one here is why did David Koresh ask the FBI for a word processor? Which is a much more difficult kind of question than just giving a simple kind of entity as an answer.

Okay, I think it is reasonable to say that people do want to ask questions; that it is actually a natural way of expressing what your information need is just because of English. So if you look at some of the early web search logs that were made publicly available in the very late 1990s, you actually find that there are just lots of questions in them. So people ask who invented certain music; how to make stink bombs; how to copy PSX; how can I eliminate stress; there are just tons of questions in them.

And I mean I think this factor is especially strong given that anyone who has used a search engine for more than three days should realize that it's completely pointless to be typing full questions like this in them. I think it just is something that is very natural for a human being as the way to go about expressing what their information need is. Okay. So if you look at the history of NLP, overall question-answering just isn't a new area. And so I have already mentioned when we talked about compositional semantics that there is tons of work in the early days about doing question-answering where what you were doing question-answering from was things like databases.

And that's when there were famous systems like the Lunar System which had facts in a database about moon rocks. And you could ask questions about which minerals are found in moon rocks or things like that. And there is a modern instantiation of that with spoken dialog systems which are often working off a database. And you are asking about when the – at what times does United fly to Florida or something like that, that's very active still today. But the difference from that and this is that this kind of work is all very domain-specific work which has a lot of handcrafted components.

The thing that's been a much newer focus is doing what's sometimes called open domain QA, which is putting things much more into the information retrieval space of saying, okay, you have just got a large collection of documents, arbitrary documents. And what we would like you to be able to do is intelligently ask the questions based on those documents as your information source. And so in modern times, I think a very influential bit of work was one that was done by Julian Kupiec in 1993 when he built a system, Murax, whose idea was that it just took the text of encyclopedia, and the aim was to then be able to answer questions posed to that encyclopedia.

And then Annette Hershman did work on reading comprehension of the kind that you might remember from high school which is the same kind of gaining information from arbitrary documents. It turns out that this area also isn't really new. It's actually kind of an interesting old paper from 1965 by – his first name is Robert – anyway, Simmons. And really the – Simmons's paper is precisely about taking an encyclopedia, loading it on to a computer, taking a question, parsing it into a logical form, do some kind of information retrieval, get relevant texts, parse those into a logical form, and make them to give an answer.

And really, the scenario that he presents is essentially the kind of scenario that I will be showing later today as to exactly how some more NOP-focused approaches to question-answering without doing question-answering. And it wasn't purely conceptual. This was a system that he started to try and build. But I think it's fair to say in those days that there wasn't the ability to do parsing, generate semantic forms, etc. And it wasn't really anything that could possibly go anywhere, where these days it's gone a lot further. Okay. There are several question-answering systems that you can play with online.

I'll talk later about LCC's work, and they are a start-up company who has dominated the TREC valuations and question-answering. And you can go and play with their system. I think one of the things that you can notice if you play with that is it's just kinda too slow for real-time use. Like you can find to put in your question, and it puts up little messages saying it's parsing your question and searching for documents, and filtering documents, and finding identities. And eventually you'll get your answers back. But I think it's kind of outside the kind of space that most people would want to wait when they are doing their average web search. Okay.

So most of modern question-answering research has been done in the context of this TREC competition, and so I have already shown you a few of the kind of questions that they use. So TREC has been a moving target because every – it started in 1999, and every

year they sort of change in some way how they are doing things. I mean not just being capricious; they just see that as a way to develop the research in interesting new directions. But I guess what I am going to do today is sort of gloss over some of those differences.

And to the extent that I am focusing on something in particular, it's really more like the early years of TREC QA of 1999 to 2004 and not so much the last few years when they have been doing some more particular things that I am not really going to talk about. So in the early days of TREC QA, each year there was a set of 500 questions, mainly but not exclusively fact-based questions. And in the first few years, the goal was that – what you are going to do was something close to IR. You were going to return a piece of text. And the difference was instead of returning a whole document it was going to be returning a very small snippet of text. And they are actually kind of two variants.

One where it was 50 bytes, and the other [inaudible] was 250 bytes. And I'll just quickly mention the scoring system that was used in the first three years. I'll give some numbers on it later. But this was mean reciprocal rank scoring. And so essentially, you only got points if you returned a snippet of text containing the correct answer inside your first five results. And if you got them right, if your first snippet was – ranked snippet had the right answer, you got one point. If your second-ranked snippet had the right answer, you got a half a point, and similarly, with reciprocals down to five.

And then if you are below five, you just got zero. And then from 2002 onwards, they dropped that and just went to exact answers. So your job was just to return the exact answer. And you got points if you got the exact answer right, and you didn't get points if you didn't get the exact answer. Okay. The collection was effectively a bunch of old newswire, so it's buried again between the years. But typically, it's a bunch of newswire sources over a bunch of years. So it's a reasonable size collection of documents – about a thousand – sorry, about a million articles and a few gigabytes of text. Yeah.

Student:[Inaudible].

Instructor (Christopher Manning):That has varied across the years. But they have made attempts to get questions from realistic sources, so one of the sources that they have actually gotten questions from is from web query logs that they then selected things out from that, questions. Some of them have also been created in other means by human beings coming up with questions. Yeah. The crucial thing is, obviously, you want to come up with the questions without looking at the documents. Because if you are looking at the documents, you can look at a document and come up with a question, but you always then always use all the right words and syntactic structures that the document uses.

And that that is isn't an effective way to simulate the use of task. Okay. I think it's fair to say in essentially all of the – all the core work that's been done in the TREC QA community, that the kind of starting point has been, okay, the document collection is big enough that it doesn't make sense to pre-process the entire document collection in

advance using expensive NLP techniques like doing full parsing. So instead what we are going to do is we are going to have information retrieval over the entire document collection. And then when we find questions and candidate answers, we are then going to selectively do deeper processing of that information.

That was maybe a true assumption when this started back in 1999. I think these days that that's just kind of not a true assumption anymore, that there is absolutely no reason for the scale of text. Why, you can't actually just preprocess the whole thing. And I think there is an actually an interesting research avenue there these days as to what extent you could reinvent doing information retrieval style task where rather than doing the main indexing over words, that you are doing main indexing over fragments of much more sophisticated NLP analysis. That's still something that there has been a little bit of work on but hasn't been very explored, certainly not in the academic community.

Okay, so I note at the bottom that many systems – in the early days, everyone just did TREC over the document collection they were given. As time went by, partly for work I'll talk about in the moment, that everyone started supplementing the document collection with other stuff from the web. And maybe I'll talk about that once I get into the context of the Ask MSR work.

So the system that's most dominated the TREC QA competitions has been work that's been done by Sandra Harabagiu and Dan Moldovan, who started off their career at Southern Methodist University and went to University of Texas at Dallas and have been involved in this company LCC, Language Computer Corporation. So really, most years they have had by far the best system in this competition. And I'll show a couple of results later. And their system is something that does quite deep NLP of various sorts using a lot of the technologies that we have talked about, and things like named entity recognition and syntactic parsing.

But overall, if you look at the competition, there has been a wide spread of technologies that have been used from things that have been using very superficial, more IR star technologies, to things that are using much deeper NLP star technologies. But the LCC system factoid questions in the kind of 2002 and 2004 age where you just give the answer. If you can give the right answer to that set of 500 questions about 70 percent of the time, which I actually thing is just really impressive. So that's just a really high level of performance when your task is to get one answer out of these documents, many of which but not all of which are factoids.

And in fact, if you just look at their factoid, their accuracy is much higher than 70 percent. A part of why it comes down to 70 percent is that they don't do so well in harder question types then. Actually, a student of Daphne Kohler's, David DeCree, has been interested NLP. He actually spent a couple of months looking at the results of their system on their factoid questions because he had some ideas about how results might be able to be improved for doing relation extraction for NLP. And I mean he just essentially decided not to pursuit it because if you look at the factoid questions in TREC QA, the

LCC system does nearly always get the answer right. And there is just almost no headroom. Okay.

And I'll say more in a few minutes about the LCC system. But before I do that, I thought I'd sort of show a little bit of something that's strongly contrastive and which, in its own way, had a strong influence on what happened in question-answering. And that's the Ask MSR system, so MSR is Microsoft Research. And Ask MSR wasn't really built as a system that's goal was to sort of put maximal resources into doing TREC question-answering.

It was more built for a research science point of saying, well, you can do all of this deep NLP stuff, but how far can you get if you actually do quite shallow stuff, and in particular their interest in the hypothesis that maybe a lot of the time that you don't have to do a lot of difficult stuff to get the answers right. Maybe often you can actually find answers in easy places. And so this is an idea that's gone on to be exploited a lot. And so the basic observation is if you are in the world of the web, there are a lot of documents out there. There are billions of documents out there. Most of the things that you want to know, well, maybe you can find them in some difficult place.

So here is the question of: in what year did Abraham Lincoln die? And well, there are difficult ways that you could find the answer because there can be some piece of text which is this mention of a death date here. And if you are clever, you might be able to connect it up with Abraham Lincoln. There are kind of hard ways you can find the answer. But, you know, on the web, it's reasonable to assume that there are very – there are going to be thousands of pages that say when Abraham Lincoln died. And maybe, if they are just very easy ways, you can find it. Like here is a common pattern that you see in lists of people.

You have the person's name, and then you have their birth and death dates following it, sometimes in brackets, here it has got a comma. I mean that looks like something you could match with a regular expression, so maybe we could just use regular expressions and get the answer out, and then we wouldn't have to do hard work. And so the idea is to sort of exploit the mess of redundancy of the web and find answers in easy places rather than saying, well, we have to be able to do deep NLP to find the answers in difficult places. On Stanford connections, this was actually a project that Andrew Eng worked briefly at while he was an intern at MSR.

He alleges he only worked on it for a couple of weeks. But he is, anyway, one of the co-authors on this work. So here is, basically, how the Ask MSR system works. You start off with a question: where is the Louvre Museum located? You very crudely rewrite it; I'll show you in a minute. You send your rewrites off to a search engine. You get back just the snippet text. They never actually went to the full documents for efficiency reasons; they just sorta straight for search results page; you then kind of collect and count N-grams from those snippet results. You filter them and tile them, and then you get ranked answers back from it, so in my more detail.

They had a simple classification of questions in the seven categories, which they don't actually say much about in the paper. I presume that they used regular expressions to match question types. And for each question type, they did a simple transformation of the questions. But essentially, you can – I think you can kind of ignore, almost, the classification. I mean really what they are wanting to do is just turn the question into something where you have got the wording, as it would be, in the statement form. But rather than kind of parse out the question and do it cleverly, they just work on redundancy. So where is the Louvre Museum located?

Well, we want to move that from it isn't into the middle of the sentence. We could parse it and work out the right place to put it with some accuracy. But rather than do that, why don't we just try sticking it everywhere, so we generate: is the Louvre Museum located; the is Louvre Museum located; the Louvre Museum is located, that's the right one. And boy, you just send them all off to a search engine. And presumably, the ones that are badly ordered will just produce bad results, and it really doesn't matter. Okay. You then also, from these ten classifications, get out a little bit of named entity result information about a result type. Okay. So you get back the snippets of the results.

And from those you just count N-grams of different sizes. So if your question is: who created the character of Scrooge? You get out Dickens 117 times; Christmas Carol 178 times; Charles Dickens 75 times, etc., down. You can [inaudible] kind of hopeful. If you just count how often N-grams occur, though, there is kind of a floor in that because just necessarily any shorter N-gram has to occur more times than any longer N-gram. And that doesn't seem like a good way to do things. So their final – oh, sorry, final step will come after one more slide. Okay, so they use a regular expression to filter – so this is kind of, on the cheap, named entity classification.

They just filter based on N-grams to decide whether the answer is the right type for date, person and location. I guess it'll do nothing in that example, but at least if you want to answer to be year or not be year you can filter out a few things. But then what they do is they have this process of tiling answers. So if one answer is the substring of another answer, that you can kind of tile them across and then get accumulative account. So here, if we have Charles Dickens 20 times – I guess my example is broken since Dickens is here occurring less times than Charles Dickens, whatever. You get the subsequences of them. You can tile them together to get an answer, Mr. Charles Dickens. And then you sum the scores, and that gets your support.

Okay, well, how well did this work? So one experiment that they did was that they tried this out where using just the TREC documents as their corpus and try to run this system. And the answer to that is that they got a mean reciprocal rank of 0.26. So if you remember to what mean reciprocal rank means, that kinda means, you know, on average they were getting the right answer ranks sort of about fourth or fifth. Obviously, in practice, sometimes they will have gotten it first and sometimes not at all. That's not so great. On the other hand, it turns out that actually that performance isn't that bad. Because that – this was actually done as a post hoc experiment.

It wasn't actually entered into the TREC competition. But it turns out that that level of performance would have been good enough to get them ninth place in the TREC competition, which in some sense is rather embarrassing for many of the systems that were ranked 10th through 30th, many of which were presenting systems that were using a lot of clever NLP and named entity recognizers and parsers, and this's and thats's, in a much more complex systems and components. But the more interesting part is, well, then they ran the same system where they didn't use the TREC corpus at all; they just ran the whole system on the web. And, well, then their mean reciprocal rank is massively better.

Now they have got a mean reciprocal rank of 0.42, which means kinda on average that the correct answer is around second in the listing. So you gained way, way better results there, and that's illustrating their point that they can really rely on the enormity of the web to make use of all of the redundant answers and find the answer in easy locations and therefore have it come out well in total in their statistics. This result is even stronger than the numbers indicate. Because it turns out that one of the quirks of TREC, if you remember back to when I gave the list of documents that the documents were always newswire from some past periods. So it was newswire from 1996 to 1999, roughly.

And so one of the quirky rules – since you are meant to be answering the TREC questions over that corpus that if the question is something like: who is the president of Colombia? You are meant to answer who is the president of Colombia during that time period; whereas, if you ask that question on the web, the dominant answer will be the current president of Colombia. So in some sense this answer actually slightly under-represents the strength of the system. Okay. So the thing that MSR conclusively proved that there was just enormous value in looking for answers on the web.

And so in more recent years, since this system, essentially all of the – essentially all of the good TREC systems also go out and look on the web for answers to questions. In terms of the TREC competition, that, to my mind, has kind of made it end up a little bit false and bogus because – I mean so the observation – so the way TREC is constructed is you have to return a source document inside the TREC collection that supports your answer. But what people are exploiting with the web is if you can find the right answer on the web, it's a much easier task once you know the answer to then find the document in the TREC collection that supports the answer.

Then using the TREC collection to – yeah, so that's kind of, to my mind, made the whole thing a little bit funny and bogus at that point. So that part seems a little bit less interesting to me. There you go. Okay. All right, so clearly that's a very powerful technique exploiting the information that's available on the web. And lots of work since then has gone on into in various ways to make use of the fact that you just got a huge magnitude of data to make certain tasks possible. I mean of course it is worthwhile of pointing out that there are lots of scenarios in which that just isn't available to you.

So if you want to be doing something like answering customer e-mails or something like that, well, then you have got one document. And you want to answer a question, and you just don't have the same kind of web redundancy to work with. It's also the case that

what they provided mainly worked for these factoid star questions. It didn't really give a lot of power for more difficult question kinds. And finally, obviously, really a system that they presented, the whole goal of it was to show how much of it you could do with very little.

I mean obviously you can point out all of the areas in which is kind of hokey in this seven question categories, and then recognizing the answer ties to the regular expression. There are kind of obvious ways in which you could tweak it a little bit further. Okay, so now I'll move to the opposite extreme and talk for a little about the LCC system for question-answering. And so this picture shows a picture of their architecture about 2003. It's not that I think it's fundamentally changed a huge amount since then, it's just I have a slide of their system architecture that dates to then, so if I just go through this for moment, and then we'll go through some of the parts again in more detail.

So you have a question come in. And so like this is partly showing how with TREC QA having gotten more complex that they have kind, in this period, had three kinds of questions: factoid, [inaudible], definition questions which are processed a bit differently. But I'll largely ignore that. So a question comes in. They parse up the question with a statistical parser. They then do what's called here as semantic transformation, which effectively just turns it into a dependency parse of the corresponding statement with a little bit of additional stuff for how you treat question operators.

You then – this is a very important part – you work out the expected answer types, so is the answer going to be a temperature of first name or whatever. And then you work out keywords of the question which are going to be fed from information retrieval. Okay. So the information retrieval, you have a document index. You do basically IR, but it's at the level of passages. So you retrieve candidate passages. And then somewhere from here, you work out – somewhere over here – all right, I have wanted to do this [inaudible]. Okay. So in the candidate passages, you find plausible answers by information extraction, tile style methods.

You then do more complex theorem prover-based methods to re-rank the answers to find the most likely ones. And eventually, you return them. Okay. But I'll just start off with the results here. I mean this is from one of the earliest TRECs. But basically – and these are, again, numbers that are showing mean reciprocal rank numbers. Overall, the LCC system really, really works for them. So I mean if you kind of look at these results, I mean I guess the picture you get is these are the systems that were really bad. These are the systems that kind of, sort of give information retrieval and a bit of extra stuff, and you get your mean reciprocal rank of 0.2.

These are the systems that are somewhat better, and maybe they have some stuff in them like named entity recognizers and various other things. And then these are the LCC results. And in terms of mean reciprocal rank, there is [inaudible] actually almost double anybody else's. And in those days that they were the only group that really was trying to apply deeper NLP and AI-star theorem proving methods to this problem of question-answering and really worked for them to get the answer, as you see from this. I mean the

truth is a bit more complex. It's not all of the value comes from the higher level methods. Quite a lot of the value comes from the lower level stuff.

So I mean from some of their own analysis of when the system has errors, I mean one of the most important things is to do the information retrieval well. I guess this is just like any kind of feed-forward system, that unless you have got the right candidate answers out of the information retrieval system, you are just sunk before you get started. And so that's actually when they – when the system loses, one of the biggest single places the system loses is where they just never find the right documents that have the answers. And a lot of their other work has also been in having a very good named entity recognition system, as I'll talk about in a minute. Okay.

So a kind of – in terms of modern QA systems, and you particularly see this in the LCC system, one of the really, really crucial parts is having this jewel system between determining answer types and just – and labeling named entity. So essentially you want to have a rich typology in which you can determine what semantic type of thing you want an answer of, and then you'll be able to find entities of that semantic type in documents. So when a question comes in, you build – you have got to classifier here, which is an answer-type predictor. I think I suggested that as a possible 224n project.

There is not stuff that LCC uses, but there is a source of data from work of Leann Roth where there is a fairly large corpus of questions classified for their answer type available. So that's a fairly easy thing to work on for building a classifier. And you map onto an answer-type hierarchy, which will then be used later. And then simultaneously, you take the question, you do any kind of the query expansion things you think are relevant, and then you have keywords that you send off into the information retrieval system where you then get back passages. Okay. So determining the answer type isn't trivial.

I mean the first level is who questions ask for a person, and how many questions ask for a number, and things like that. But there is quite a lot of subtlety in how questions get asked and what they are asking for. So if your question is: who sells the most hybrid cars? That's not asking for a person; that's asking for a company name, which is meant to ask about things if you don't think about it too much. But as soon as it is modifying other words like which president went to war with Mexico, that's now asking for a person name. So there is a fair bit of subtlety in working out the right answer type. Okay. Yeah. So then they do information retrieval.

So they have got a system of effectively working out a set of keywords to send to the information retrieval system. And so it's kind of like just standard IR, but they do various kinds of filtering where they basically send nouns and verbs that aren't stock words off to the information retrieval system. And then they have this kind of complex system as to which words to send off. I mean the LCC system has always had a lot of kind of feedback loop kind of things where you are kind of able to attenuate the specificity of how much stuff you send, and how much stuff you get back, so that you can kind of start with only particular stuff and see how many results you get.

And if it doesn't seem very good, you send more stuff and get more results. The LCC people believe that's a very important part of their system. I have never been a hundred percent convinced of that myself. It always sort of seems like somehow you should just be – more be able to globally do one ranking. But anyway, this is the kind of procedure that they use to work out which keywords to send off to the IR system. Okay. And so then they have a passage extraction loop where you do passage retrieval where the passage size is not pre-fixed; it's actually dynamic. And so there is this loop that I was just mentioning. So you, first of all, use some of the keywords heuristics.

You ask for some passages. If you are not getting enough passages back, you should – you back off and drop some of the keywords. If you are getting too many passages, you then refine by adding more keywords. And then you end up with a collection of passages. Okay. So then what you want to do is you rank the passages. And there is a sort of a fairly obvious passage-ranking system. So number of words that are recognized from the question, how close together they are, how many unrecognized words there are. Okay. So that the kind of picture you have is like this. So the question is name the first private citizen to fly into space.

And so we have worked out an answer type person. And so crucially, once we have worked on answer type, we can then use named entity recognizer to work out candidate answers. So we can then look for person names, and so we have Christa McAuliffe, Karen Allen, and Mike Smith, and Brian Kerwin; they are the person names in this. And so then you are finding a passage. And then to work out what's the best candidate answer, you are then working out which candidate answer is closest to the words and the queries. So you have first private citizen, clients, space, that's kind of pretty easy. And so Christa McAuliffe is then being selected as the best candidate answer.

All right, so a crucial part of the system is this pairing of answer types and named entity recognition of those types. In fact, that's a kind of a huge proportion of what the LCC system does. So in the 2003 system, the LCC system got 289 correct answers to factoid questions, so a little bit under 60 percent correct. And then of those 289 questions, 234 of them were being gotten correct, essentially, by this pairing of answer type prediction and named entity recognition to find the right answer inside a passage. And so there is a bit of a listing below, so there are 55 quantity questions; 45 number questions; 35 dates; 31 persons.

And then heading down into products, continents, universities, addresses, URI's, prices. And so a central tool that LCC has is a named entity recognition system that works over a large number of named entity types. And so a distinction you see is that in kind of academic named entity recognition research that, by and large, people have these sort of fairly small number of classes which are useful. You have things like persons and companies and so on.

But those are the kind of publicly available resources, where a number of private companies have built up data to do very fine-grained named entity recognition where you are getting of the order of 100, even 200, subclasses of entities for various needs. Okay.

But nevertheless, this is only 234 out of the 289. And crucially, the LCC system has always gotten a quite substantial lift from the fact that they can actually do quite a bit more than that. That other people – lots of people can do passive IR and run a named entity recognition system. And the LCC system gets about another 25 percent performance from doing more than that. Okay.

And so I then wanted to say a bit about the more complex NLP they do. Okay. So the general observation is that there just are questions in which you want to do more in terms of syntactic parsing or working out predicate argument structures with semantic [inaudible] and being able to do certain kinds of word similarities, and inferencing, and things like this. And so you have examples like these. So when was Microsoft established? What are the content keywords there for passage retrieval? They are basically just Microsoft and established, and you know that you are looking for a date. But, well, the problem is that Microsoft establishes all kinds of things.

So Microsoft plans to establish manufacturing partnerships in Brazil and Mexico in May. So you could return the answer May. Or if you knew what year that article was published, you might be able to say May 2004. And, you know, that's clearly not the answer. So you really need to be able to work out – well, let me actually try and find the sentence where Microsoft is the thing that is being established because that is really what we need. So we need a sentence like this: Microsoft [inaudible] was found in the U.S. in 1975, incorporated in 1981 and established in the U.K. in 1982.

But wait, this is still hard because if you just mention the word established, or you are likely to return 1982 which isn't the right answer. What you then want to do is have enough semantic technology that you can work out that founded is equivalent to established, and that this is – the passive form of Microsoft was established where Microsoft is the object of establishing. And so the right answer is 1975 where this is talking about a subsidiary in the United Kingdom. Okay. So how do they go about doing that? Okay, so essentially they just take a standard statistical parser of the kind that we talked about extensively. And so you start off with a question.

How many dogs pull a sled in the Iditarod? You parse it up with the use of Collins star parser. And then effectively, then turning it into a dependency style parse representation in the kind of way that I briefly talked about where you percolate the headwords, and you read off the dependencies. And the only thing in there that they do in addition to that is kind of a little bit of interpretation of question stuff. So how many dogs is being turned into counted dogs, so we have got the verb pull; subject is the dogs. You want to count the dogs, what is being called a sled, and it's being pulled in the Iditarod. Okay.

And so then what you can do is be trying to match on documents for that kind of semantic logical form, as they refer to it. And in particular, the way their system is set up is to do a form of loose logical abductive inference. So effectively what they do is having done this representation, they turn this representation into something which are predicates connected together, so that they are, literally, having a form of this. And X-17, let's see, Iditarod; X-18 is a sled; and if X-18 is in X-17, and there is a pulling event, E-31. And E-

31's object of the pulling is the sled. So they make a conjunction of logical terms out of this, which is actually then used in a loose theorem prover.

It's kind of a loose theorem prover because it kind of does this form of weighted deduction where you can effectively license any non-exact proof by sort of just having it have a lower likelihood. Okay. They make extensive use of lexical resources of the kind I talked about last time. And so they build from WordNet and extensions to WordNet that they build lexical chains of relationship. And this works very effectively for them in working – when you have to work out answers to things that just don't match well at an information retrieval level. So I have just got a kind of a couple of examples of the kind of questions that they have managed to get right by using these kind of technologies.

So here the question is: when was the internal combustion engine invented? And the candidate answer they used was the first internal combustion engine was built in 1867. So I mean the internal combustion engine part matches well in modular the hyphen appearing here, which goes to show that there are kind of lots of tricks to this matching. But the central part is they want to support that built is connected with invented. And while built and invented aren't synonyms, but nevertheless they are able to trace a path of inferences through mainly WordNet star links from building depending on creating something; and inventing is creating something mentally, which is allowing you to create.

So effectively, they are in this sort of funny middle ground between kind of word association and lexical inference where you are kind of creating a kind of a plausibility story that is reasonable to say that building something supports the fact that it's been invented. Okay. Here is one more example. How hot does the inside of an active volcano get? So they convert that into a semantic form. And the potential answer is lava fragments belched out of the mountain were as hot as 300 degrees Fahrenheit. And so this is kind of a tricky one for information retrieval system because there is actually almost no word overlap.

I think it's true that the – apart from the, I think the only word that actually overlaps is the word hot, and that's a little bit useful, but it's not actually very useful. And so they are actually able to sort of use their lexical resources and inference system to say that a volcano is a kind of mountain, so they can make that connection. That lava is part of a volcano, and so therefore fragments of lava have the properties of lava. And if lava is at a certain temperature inside the volcano, then the inside of the volcano, it's plausible, should be that temperature. And so that they can get that out as a kind of a loose proof and answer 300 degrees Fahrenheit, which is kind of cool. Okay.

So that's mainly what I was going to say precisely about question-answering, although there are a couple more slides about question-answering at the end. I thought I would vector from that to say a little bit about the more general task of textual inference that actually I have worked on with a bunch of people at Stanford for a number of years. And it's also having some of its own competitions. So the idea of textual inference task was to sort of generalize this task of can we actually do robust textual inference in the kind of

the way LCC is doing to see whether some piece of text supports having the answer to a question.

And so this idea was proposed by Ido Dagan, who is kind of an NLP researcher, who is not really kind of by background a computational semanticist precisely; he was really more someone who did sort of large-scale, more heuristic NLP processing. But his idea was, well really, if you think about it, lots of the tasks that people would like to do in NLP, the core technology that is missing to be able to do richer, deeper tasks than is available today is essentially being able to do this core textual inference step of being able to say does this piece of text provide an answer to this information need? And question-answering is one obvious place that that occurs.

But it occurs in lots of other places as well. I mean in terms of the general vision of just being able to do higher quality semantic search – you can think of semantic search as an instance of this vision. If you would like to be able to automatically look for answers and FAQs. If you want to be able to answer people's questions in the e-mail, but there is sort of this big space of sort of doing semantic stuff with text. And so the way he formulated the task was that there is going to be a passage, which is like a passage in a passage retrieval system, Sydney was the host city of the 2000 Olympics. And then there would be another piece of text that was the hypothesis.

And the way he formulated the hypothesis was phrased as a statement that you could have equivalently phrased it as a question. The Olympics have been held in Sydney. And so what your job is to say: does this piece of text support this hypothesis or not? So in this case you want to say the answer is yes. And so this maps on directly to the final phase of the LCC question-answering system, which is you have got candidate answer passages which you have ranked based on what you could do with answer types in NERs. And then you want to do a final richer stage of processing. And you want to say, okay, here is my best candidate answer passage according to my machine one ranker.

Does this answer – does this passage actually provide an answer to the question by doing richer processing? And so that's what people have been working on. Now I mean if I just say to you there is a passage, and there is a hypothesis, I mean really that formulation is so general that you can encode any form of human knowledge and reasoning in it, all right. My passage could be the function G of X equals log of sin of one over the exponential of X-squared blah, blah, blah, blah. And my hypothesis is that the derivative of G is something else. And you would have to say whether that follows from the text or not, right.

You just take any piece of – any piece of knowledge and problem solving; you can encode it up in this format. But that's not the way it's been used. I mean it has been used for things like this where there is a fairly direct, if not trivial connection between what's in the text and the hypothesis you are meant to answer. Okay. I guess I got ahead of myself saying applications. But this is the kind of semantic search application that you would like to be able to say find documents that concern a certain topic, even if they

don’t use particular words. So find the documents talking about lobbyists attempting to bribe U.S. legislators. Well, here is a document.

The AP named two more senators who received contributions engineered by lobbyist Jack Abramoff in return for political favors. Well again, in terms of standard IR keyword overlap, it ain’t a very good match. It has precisely one word in common with the search string, which is the word lobbyist. But, well, it is obviously a good answer to that semantic search, indeed. It’s talking about a thing of relevance. So that gives us a text hypothesis pair. Can we determine it’s a good answer? So obviously this question – this task is also just very similar to what people do with reading comprehension. These are the examples I take from CNN.

It turns out CNN are doing their share to try and improve the awful state of education in America. Every day they stick up reading comprehension questions for some of their articles, so you can see if you can work out the answer. Where is the country of Somalia located? Yeah. Here is another fun application that Dan Roth suggested – who is someone else who has been involved in textual inference. I guess all of the time for people involved in a lot of the computing industry for consulting and so on, people give you these six-page documents of terms and conditions. And by and large, you wouldn’t – you wish you didn’t have to read it all.

And there are really just a few simple things that you would like to know whether this document is bad or good for, such as does this document have very restrictive terms, or does it say that information is to be shown as not confidential unless people explicitly indicate that it is. And well, you can at least imagine that you should have to have a system answer that for you. Obviously, that’s well beyond the current technology. So this is kind of a picture of the system that we have been using in the main for this task. In some sense it’s not very unlike what you saw on the LCC question-answering system. So we start with a text and a hypothesis, and we do a whole bunch of linguistic analysis.

So we build up these kind of semantic dependency graphs. And India [inaudible] missiles, subject, object, we get information about words. We do named entity recognition, so we know that India is a location. And we have got parts of speech. The first document frequencies – we have been doing alignment between the two texts as a graph alignment. It turns out – I mean this [inaudible] is a graph alignment, so it somehow sounds less deep than saying that you are doing abductive theorem proving as in the LCC system. But it turns out that essentially, just as they did returning sentences into these kind of dependency graphs.

And if you then just really do a trivial transformation where you take each word here as a predicate, each word and each relation as a predicate and connect them together with arbitrary variables, it’s really, then, isomorphic doing theorem proving versus graph matching between all of these nodes. And then the third phase after that is – so doing this alignment gives us a basic quality of an alignment. And we then have this follow-on stage where you have a bunch of specialist inferrers that try and determine various attributes of

sentences that either support or go against there being an entailment. And I'll say more about that. Okay. So this is the kind of graph alignment idea.

So there is kind of an interesting tension in this phase. On the one hand, you would like to kind of match words that match including kind of doing all of the same kind of WordNet stuff, so troops can match soldiers, and killed – well, ideally you would want to match killed with this idiom of lost their lives. Truth be told, our system isn't clever enough to do that, but it can tell, nevertheless, that killed and lost are somehow connected with each other, verbal meanings, and so that they can get aligned. So partly, you want to do it on the lexical level, but partly you also want to do it at the syntactic level and look for syntactic parallels of several troops were killed.

It is partly syntactically parallel to 13 soldiers lost. Although, actually, in this case it's tricky because this one is passivized, whereas this one isn't passivized, but it's still following a – it's still filling a similar semantic role. So there is sort of a trickiness about how much to pay attention to word matches versus syntactic matches of various kinds. At any rate, what our system does is tries to come up with kind of doing a give-searchy kind of matching, find the best match between the text and the hypothesis. I think it's fair to say that for any current system that you really want part of the system to be doing something that's kind of loose-ish; IR plus plus kind of matching.

There has been one [inaudible] system in the RT space which has really tried to do full deep inference, that they convert the text and the passages into logical forms and do first-order theorem proving of the kind that I talked about when I talked about compositional semantics. That system has essentially always had almost trivial coverage, that in the RT competitions of this, the data sets each year have been 800 pairs that you are meant to do judgments of. And typically, the number of cases in which that system has been able to come up with a proof or a disproof of the answer has been about 30 out of 800, only about 4 percent of the cases.

But there are just all sorts of ways in which it – you can just sort of not have enough coverage of things at either a lexical or syntactic level to be able to process them. Now it turns out that even for the 30 examples where it thinks it has a proof or a disproof, that actually the percent it gets right is only about 75 percent. As it turns out, there are other places where it gets the wrong answer anyway. But I think this is an example I kind of like, which kind of indicates why you want to have some kind of looser matching. So here is the passage.

Today's best estimate of giant panda numbers in the wild is about 1,100 individuals living in up to 32 separate populations, mostly in China's Sichuan province but also in Shaanxi and Gansu provinces. Apparently pandas did really well in the earthquake. No pandas died. It goes to show you are better of not being in a building. Okay. And then the hypothesis is there are 32 pandas in the wild in China. Now – well, sorry, sorry, I should say I am contrasting two hypotheses. This one is false. And here, there are about 1,100 pandas in the wild in China. And this one is true. And the interesting thing, really, is how it's just obvious that this one is true.

And as a human being you say, oh, well, it's obviously true. Look at the passage; it says there are 1,100 pandas in the wild in China. But if you actually look hard at the passage, man this is kind of complex because, well, the 1,100 modified individuals. It's not 1,100 pandas; it's 1,100 individuals. And, well, where do the pandas come in? Well, there are some pandas over here, but pandas are actually a modifier of the word numbers. So it's panda numbers, whatever those are, that depends on your compound noun processing. But the panda numbers, it's not that it's saying that panda numbers is 1,100 individuals. The panda numbers, in turn, is the kind of prepositional phrase modifier of an estimate.

So it's the estimate that is 1,100 individuals. And man this is a lot of complex stuff. I mean in a perfect NLP system, well, maybe you would have a system that could understand all of that perfectly and wouldn't know that from best estimate of giant panda numbers in the wild is about 1,100 is equivalent to there are about 1,100 pandas. But, you know, our system certainly, we just can't kind of do this level of inference. If we have simple stuff where we have things like 13 soldiers ambushed and 13 troops ambushed, okay, we got it there. Oh, and if it's more than ten troops ambushed, we can do that because there is just enough easiness of the parallelism of syntactic structure.

But we can get this right in the sort of a principle deep way. But on the other hand, if you don't try and go too deep, and you just say, well, look at this; 1,100 individuals, kind of close to the word giant panda numbers, looks pretty promising. Well, then, you are going to get the answer right. And I think that this, for a lot of the systems that try and do this task, that that's the tension. That if you try and actually say, look, I am interested in NLP. I really want to understand these things well and be dead certain things are correct.

Your accuracy goes down because it turns out that there are a lot of things that you'll get right if you just say, you know, I found a number of individuals that's within a few words of what I was looking for, giant pandas, go with it; then you'll actually get the answers right. So effectively what the kind of current state-of-the-art is, I think, is to sort of try and explore this middle ground between IR and deep NLP where your are able to use deep NLP to increase your confidence that answers are right because you can work out what words modify or predicated what other words, to rule out answers that are wrong because you can tell the syntactic structure is such that a certain candidate just couldn't possibly be right.

But you also can kind of fall back on more imprecise forms of IR style matching. Okay. But it's also obvious that if you just sort of say, well, let's do this kind of loose matching of word overlap proximity kind of similar syntactic relations, there are all kinds of ways that you can be completely fooled, all right. So if your sentence is: the Army acknowledged interrogators had desecrated the Koran. And you just do straight matching of the words and the hypothesis, yes, you get the answer right. But, well, then what happens as soon as this word gets changed, and it's the Army denied the interrogators had desecrated the Koran. Well, I guess that depends on your trust in believing the Army.

But then you shouldn't be getting the right conclusion. And so just having a perfect alignment of a sub-graph isn't enough to guarantee that you are getting the answer right.

And that can then even continue into higher long level things. Okay. So what our system does is after the level of graph alignment, that there are then these sort of system of inferrers that try and do deeper NLP focused on particular kinds of tasks which are effectively looking for evidence that certain kinds of answers are supported in a more precise way where evidence for things that says that they are not supported.

Some of their inferrers find positive evidence that give greater support of an answer, but more of them are in the form of no, we can find counterevidence for this being right. Because I guess it's easier to find – it's easier to find things that are signals of badness than it is to find signals that are of goodness. And so this means that we kind of have imprecise, but not non-existent, crude semantic theories that deal with a lot of phenomenon of natural language. So adjuncts, modals, quantifiers, implicatives, antonymity tense; all of this stuff is in there in various ways.

And effectively what happens then is the quality of the alignment and all of this other stuff is then fed into a final logistic regression classifier. So I am just going to mention a couple of the kind of things that we look for and how they give us more value. So one of them is just looking much more closely at the structure of the sentence and looking at arguments of verbs. So here is the kind of example that you don't want to get wrong. So Ahmadinejad attacked the threat to bring the issue of Iran's nuclear activity to the U.N. Security Council by the U.S., France, Britain and Germany. Ahmadinejad attacked the U.N. Security Council.

Well, this is exactly the kind of example where you get kind of every keyword matches. The syntactic structure of quite a lot of it is similar, so Ahmadinejad is the agent of attacking in both of them. But you really just want to notice that the object of attacking is something different. So that's the kind of thing that we can do by matching more precise patterns in using syntactic parsers. I already gave one example where that if you were embedding in the context like somebody denied something, it shouldn't go through. That actually happens quite a lot in real examples.

So you have a sentence like: Libya has tried with limited success to develop its own indigenous missile and to extend the range of its aging scud force for many years under the alpha tower and other missile programs – that you just don't want to be able to conclude from that that Libya has developed its own domestic missile program. But again, you kind of have kind of excellent matching of syntactic structure because, syntactically, Libya is the subject of develop. You get this control relationship that I talked about briefly. And it's developing an indigenous missile, so the object is right. So the syntactic structure match is really basically perfect.

But what you want to do is notice that "has tried to" isn't what linguists call a non-factor verb; that the thing that is complemented something that you can't to assume to have happened. And so that contrasts with an example like this where scientists have discovered that drinking tea protects against heart disease by improving the function of the artery walls. From that one, you want to be able to conclude that tea protects from disease because discovered is something that is a factor verb that means its complement

is to be taken as true. And so we look at these embedding contexts and then a classification of verbs, as to what kind of verb it is to decide what we can conclude. Okay.

And to give one more example of that, if you think of the general process of taking a hypothesis and matching it against the text, normally a hypothesis only matches some of the text. And sort of the basis level assumption is the fact that there is other stuff in the text can just be ignored. It will just be extra information that we don't need, and so several of these kinds of inferrers are pointing out that that's not always the case. You certainly need to look if you are embedded under a non-factor context, but in more detail depending on whether you are in a negative or a positive context it turns out that it's actually okay or not okay to drop various other bits of information. And so we try to check that.

And so this is one sub-case of this for restrictive adjuncts. So the first sentence is: In all Ceraq bough 422 million worth of oil from Iraq according to the Volca Committee. So that, by itself, doesn't support Ceraq bought oil from Iraq during the embargo because there is extra text that is added to the hypothesis that isn't matched in the text. And therefore, we don't want to conclude it. That's the easy case which you naturally get. But it turns out that if you are in a negative context, like didn't buy oil, things work the opposite way. And these are what have been referred to as downward-entailing context.

So if you have the sentence: Ceraq didn't buy any oil from Iraq according to the Volca Committee; then you are entitled to conclude Ceraq didn't buy oil from Iraq during the embargo. Because since you are in this negative downward-entailing context, you can add a specification of the hypothesis, and that's necessarily true because the broader hypothesis – Ceraq didn't buy any oil from Iraq – is true. And so you get this funny reversal that, in general it's bad to have stuff in the hypothesis that you can't match to the text. But once you are in this kind of downward-entailing negative context, then it is okay to add stuff.

And the classic downward-entailing context is just an overt negative like this. But it turns out that there are lots of things introduced downward-entailing contexts, various kinds of verbs like forbid and deny; things like various quantifiers like few, etc. Okay. So that's basically it. I just note here that there are all kinds of other uses of question-answering that go beyond different things that aren't just TREC question-answering. But from my very final slide, here are the funny examples of question-answering going wrong. So these are actual answers of systems in TREC question-answering where they didn't work so well. Where do lobsters like to live? On a Canadian airline.

Where are zebras most likely found? Near dumps and in the dictionary. Why can't ostriches fly? Because of American economic sanctions. What is the population of Mexico? Three. What can trigger an allergic reaction? Something that can trigger an allergic reaction. Okay. And that's the end. Okay. So that's the end of everything. Apart from what you guys need to do, which is finish the final project and then turn up at Monday for the brief final project presentations. And so I have sent an e-mail about that.

But just again, quick, short, sharp, fun presentations that explain the main idea is kinda what we want. And it can hopefully be an interesting fun thing for everybody.

And we are going to supply a little bit of breakfast even, so you can get a free doughnut. Okay, thanks a lot.

[End of Audio]

Duration: 75 minutes