NaturalLanguageProcessing-Lecture09

Instructor (Christopher Manning):Hi again, okay so today I'm gonna sort of pop up mainly from the details of the math of the [inaudible] models that we've been looking at, and look in general at the problems of doing n-seng density recognition information extraction for text problems. I guess it must be mid-term time now; my in-person audience is shrinking away like nothing else, oh well.

Okay, so I kind of lost at the n-seng just a little bit about what happens when you do [inaudible] inferences sequences and I kind of still want to kind of rush this rather than get through too many of the details, but I'll rush it twice. So for these few slides here, the problem we're looking at is doing part of speech taggings and the sequences that we want to assign and determine a proper [inaudible] fill and then put these on, and so if at some point that we tagged up for some point, this exactly where we can use our Macset classifier where we can use all of this and all of that to predict a classification of the next thing.

And one way to run a Macset classifier is exactly that. You predict this one, and then you go along and you predict that one and then you go along and predict that one; and that's sort of doable, but the central observation is well, maybe it'd be wise to rather than kind of early committing to each one, maybe it would be wise to delay the decision a bit because maybe knowing what this is or help give you good information about what that should have been.

But you can kind of capture in your sequence features that look at these pair of things, and so people do that and that's where the simplest way of doing that is then to do a beam search so the idea of the beam search is simply okay, we have some candidates for what this is. It could be a number, it could be a symbol, we keep the best candidates and we try out the best candidates for this given each of those best candidates and we keep the best several in a fixed size beam window and chug along.

And you know, normally that actually works just fine for these kinds of problems and it is very easy to write works efficiently and so on. The slightly classier way to do this which guarantees you to find the best sequence is to do the Turby algorithm style in France and I'm kind of gonna skip that still now and say that well, if you want to know more about how we do this kind of dynamic program inference, I'll talk about it in great depth when we do parsing which is the topic that starts for next week's classes.

Okay, so I just want to then say a few slides on the kind of couple of problems that we've been looking at most on the edges of, which are part of speech tagging and [inaudible] recognition. First, so, I mean, I've been in general telling a story of these kind of discriminative classifiers like Macset models or extremely good for doing these kinds of tasks and that's true, but just as, to give you a little bit of background, I mean, the main technology that was being used previously for tasks like part of speech tagging with hidden Markov models, which were extending the kind of generative models that we began with, so we talked about language models and then doing naive based star

classifiers and if you kind of keep on making a generative sequence model, you get to the world of HNM's.

And so a well known HNM tagger that was done quite recently was [inaudible] Barnes who is now working at Google in Mountainview, and it was so carefully designed, carefully smoothed using that smoothing technology, hidden Markov model and it actually just does a very good job as a part of speech tagger and so here are a few results for part of speech tagging.

So some of the earlier work for HNM part of speech tagging was around 94 percent and so moving to using the Macset Markov models throughout the pack in 1996, got 96.6 percent part of speech per tag accuracy but then [inaudible] Barnes showed that actually if you're very careful in how you smooth and design your HNM, you could actually get the same accuracy really with an HNM. More recently, again, this is me and some students at Stanford, we wrote an [inaudible] Markov model tagger and look we did a weensy bit better than those guys did; and so this kind of number is the kind of state of art on accuracy for tagging.

The part of it that I think is more meaningful is that if you look over here, this is the accuracy on words that were never seen during training, so they're new words, and I think by making use of better, clever features for predicting unknown words in context and doing a better job of things like the kind of doing the smoothing, regularization as the Macset model that I talked about last time whereas [inaudible] had used early stopping and count cut-offs, but the accuracy on unknown words actually is noticeably better, that we get about a 33 percent error reduction on tagging unknown words.

Okay, one other thing I should just briefly mention is CRF. CRF is a conditional random fields, and the idea of CRF's is effectively, the model still looks like the one we saw before, apart from what your class is here is no longer the class of an individual word and it's the class – it's the entire sequence of classes that you put over the entire document and doing models in this way have some nice theoretical properties. They fix some brokenness that theoretically exists in doing next of entry Markov model style inference and providing your features are local, you can still dynamic program these system and if you did 228 and did you Markov random field inference, you should know how to did; and if you didn't – so I guess I'm not gonna explain how to do these because it'd take a lot of time and it'd sort of be, people who are done 228 would get it and other people wouldn't. So I think it's kind of not very useful.

I think an important thing to know, from the practitioner's view point, I use these models now mainly, so I get a bit more respect form people on the first floor of the Gates Building, but from a practical view point, I mean, they make absolutely no difference. That what you find with these models is that practical performance, it really all depends on having clever features that exploit the observations, so both the word that you're trying to label and the context around it. But that's really where nearly all of the information is, and that the actual – follows non-existence, the information and existence of tags, that information is just a really weak information source compared to the

information that you're getting out of the observations, and so although these are theoretically nicer in a practical sense, that is not gonna make any difference.

So in the assignment that you guys are gonna be doing, if it doesn't work well enough, I would strongly advise you should be thinking of better features and not thinking if I change the MEMM to a CRF, that will make it perform a lot better, because I bet you a lot of money that the amount better it's likely to make it perform would be .1 of a percent or less.

Okay, and so in some sense this comes back to the same issues that are on this slide. So there's been a lot of work in recent years in using these kinds of discriminative sequence classifiers. The kind of theoretical issues of IU building a generative model or IU building a discriminative model, it's not really that they make – they make a little bit of a difference, but the theoretical issues are actually complicated. I mean, the theoretical results are that if you have very little data you should actually do better of the generative model and if you have a lot of data, you should do better with a discriminative model.

At the point of you kind of think of the [inaudible], so different context of NOP observations; you could often think that well, you should do better with the generative model, but the real reason that discriminate models have been very successful and OP is because they provide an easy framework for putting arbitrary observation features into the model and provide a method in which the whites of the observation features can be set in such a way that you don't get these bad effects of multi counting features that you tend to get in generative models.

Okay, and the main disadvantage of them is that compared to just counting and dividing in generative style models, that the estimation of these models is just much more expensive, but at that point, of course, we have all of this wonderful new and cheap computing technology on our side and even in the length of time that I've been building these kind of discriminative models, that once upon a time you had to leave them training overnight and that seemed kind of miserable. But these days, the same kind of models, you can train them in 20 minutes and probably less if you're clever at more de-threading things and so that the situation's enormously improved and that's why most people use them.

Okay, this is my natural juncture to mention, all the work for this class at the moment, so I guess this is the worst moment in the class at the kind of intersection of everything that is due and being handed out. The good news is that the TA's have finished grading the first programming assignment so you can pick those up afterwards, and then just a few notes on the other things.

So programming assignment two is due today. A couple of people have sort of said things about how slow the decoder is. I mean, we think that the decoder shouldn't be too horribly slow to run that if you're – if it's taking 30 minutes to decode one sentence, you're definitely doing something very wrong in the way it's being used. But I mean, just this little clarification that I guess, our main assumption is that you're trying to come up

with a good model by just looking at the AR alignment error rate score and adjusting the model to try and get a good AR, and that at the end of the day, you're gonna be running the decoder against some results to write about in your assignment write up.

We're not really expecting that your hill climbing, your performance on the output of the decoder, because yeah, the decoder is sort of slow. Okay, and then I sent out an e-mail yesterday about the final project so in between assignments two and three, you should save a little bit of brain time to think about what you could do for final project and who you could do it with and that hopefully that can kind of mean that you've at least sort of thought a bit about what data you need and how you can get stuff and what you can do and come up with something interesting for the final project.

And again, do come and talk to me or one of the TA's about final project ideas. We're happy to help and can sometimes give some pointers. And then finally, finally, today is also the day where we hand out the third programming assignment and so the third programming assignment is in some sense the scariest one because it involves doing both name density recognition and passing over the bio-medical domains. I'm gonna say a little bit about bio-medical and AR just now. Yeah, it's hopefully not too scary. I guess part of that is getting your head around what you do have to do and how does the best footwork.

So for the biomedical NER, that really most of the system is there. Essentially all you have to write is kind of the core bit, which corresponds to those scary math slides of working out the objective function for a Macset model, and then we provided the optimization code and we provided the maximum entry Markov model sequence model. So providing you can do the bit on the core ride, the rest should just work for you.

So in particular that means in terms of getting it working, you should first of all just get it working as a simple classifier and so we've also given you a trivial two-class classification data set and that's the right place to get started. Don't forget this section on Friday that we'll then also be talking about stuff that's relevant for that.

Okay, biomedical NER; so in general in biomedicine there's this enormous problem that is well suited for NOP people being useful, because there's this enormous exponential growth of rate in which scientific knowledge is being produced to [inaudible] rate, in which pieces of paper that have the outputs of scientists are being produced and although there have been some attempts to kind of encourage information production in various kinds of structured forms, that the reality, certainly up until now, is basically what scientists do is they write down their results in articles and so with complications of formulas and so on , that what you're getting is pieces of text that contain all the information about a domain.

And that's a lot of the time true for interpretive stuff even for things in more symbolic forms, so things like genome sequences, yes you can just get sequences of base pairs and people use some of these same kind of sequence models on those for other purposes, but assumes you actually want to know what do bits for genomes do, while all the

information is again, in natural language text sentences. And I guess just recently, even for your U.S. government taxpayer dollars at work, that the NIH has declared that all publicly funded biomedical research has to be made available free online in [inaudible] mid-line, which is then this huge repository of text that has all the information.

There are also a whole bunch of biological databases. There's gene banks, [inaudible], fly base, [inaudible], there are all of these kinds of resources that are available that do give various kinds of structured information for biomedical research as well. The first question is how are those resources made? And the answer is that those resources are made by hand, so that there are a whole bunch of projects and there are places like the National Library of Medicine, where literally they're employing dozens of people with good qualifications in biology where what they're doing is reading these sheets of paper in articles in Nature or Cell or whatever journal it is and putting bits of structured data into databases.

So that seems like a – kind of an opportunity where could some of these tasks be done more automatically using techniques like name identity recognition and information extraction. And so now rather than having kind of persons, organizations and locations, we want to do the same kind of NER tasks, where we have proteins and DNA and cells and things like that. So is this a difficult task? It turns out that this is actually a really difficult task. It turns out that it's just much easier recognizing person names and place names and organizations that it is recognizing biological entities. And so why is that? So the first key observation is that we're certainly not dealing with a closed class list of words, that there are all of these newly invented terms that are being used all the time for newly invented things. I guess that's what scientists are doing and as well as for truly newly invented things, people will also use different and novel terms for the same thing; so there's this huge space of entities. How things are named varies a huge amount, among different communities, that you can literally go into different biological communities and there are the yeast people, and there are the C-Elegans people, and they kind of have their own little cultures just on how the go about naming things. But in most of those communities, there isn't actually a lot systematicity in how things are named. So in particular, common English words are quite often used as gene names. So there are genes that are called period, curve and FOR, sounds kind of tricky, all use these gene names.

And so a lot of the time, genes are effectively given kind of descriptive names or kind of joke names, so there's one fly gene that's called Iless because if you take it out the fly, it doesn't grow any eyes and things like that; so you get all of these silly names. Names can also be ambiguous so in fly based is COK, is used as a symbol for the clock gene, but there's another gene, which is the period gene, and I guess periods remind people of clocks or something like that, because it's like when you do computer hardware. And so COK is also used as a synonym for the period gene, which is different than the clock gene. Another big source of ambiguity is often there are these systematic [inaudible] in which the same name is used for things that are associated in different states, so you'll have a piece of a genome that's a gene that codes for something and what it codes for will produce a DNA, which can be transformed into an RNA and that the things that go along that pathway will often have been given the use, often the same name will be used for

them and you have to be able to tell from context as to what state they're in and what they are. And it turns out that that can actually be difficult for human beings. I mean, one person, a bunch of years back published this result of only 69 percent human accuracy agreement on these things. I think that that's a really low-ball number and it's the way this person kind of set up, their measurement task just wasn't a very good one and I think other people have been able to set up tasks for human labeling where they've been able to get accuracies up into the 80's, but nevertheless, it's not a trivial task for humans even, to be able to do this sort of labeling task, and I mean, I think if we kind of go back to that bit of text and this is the kind of stuff you're reading, entire independent trans-activation by tap in cells derived from the CNS and the whole mechanism of HIV one gene regulation. There's a lot of hard to understand stuff that's in there.

Okay, so what you guys will be doing is building these kinds of Macset models to try and predict what these things are. So here we have – this one's going to be a DNA. Okay and I just want to say a little bit as a bit of hint is well, what are some of – one of the kinds of things for you guys to be thinking of is thinking about some of these interesting classes of predictive features that give quite a bit of power. So one class I mentioned before were these word shaped features where you can map words onto some kind of equivalent class that's preserving some information about them. So here it's preserving, capitalization, it's preserving whether something is a digit or a letter and certain punctuation like minus and there's actually a bit of clever coding here which does some preservation of length and the way that length is being represented is it remembers length up to three and after that, it'll compress them. So it's a kind of a clever shaped coding and these kind of – given that there are going to be lots of unknown words or words that you've only seen once, having these kind of equivalence class features usefully improve accuracy. They're not the kind of feature that's just very effective in a lot of tasks. It's simply using substrings at different sizes of whole words, and in general for a lot of NER tasks, these kinds of substring features are very effective.

So I can try you guys out on this. Cotramoxisol, what is that? A drug, yeah, it sounds like it must be a drug, right. I mean, I don't know and I don't know if you know but probably most of you don't know, but it just looks like a drug name, right? Witherfield, what does that look like? City, place name, alien fury, countdown to invasion. It could be a computer game, I guess what I was thinking was movie, but it could be a computer game, right; and so, I mean, one way you can have a sense of this and humans sort of pick up a sense of this, is just by kind of these distinctive character sequences. So this was a collection of names which were in five classes. Drugs, companies, movies, places and persons and in this collection, if you saw the sequence OXA, if it was categorical, 100 percent of the time, that was a drug name and in general X's become somewhat of a kind of iconic letter to put in drug names. I mean, it turns out that this is actually a circular problem, right. So there are several branding companies whose whole business is doing the linguistic side of coming up with brand where they're doing the reverse job of looking at these letter sequences and telling you what letter sequences will convey certain attributes to you product and they will say that well, if you put in letters like X and Z, that gives the right aura for a drug name and if you want to make it sound kind of futuristic,

you end up with those kind of names that Intel comes up with like Centrino and ones like that, but I digress.

Right, I mean, in this data set seeing a colon almost always meant it was a drug – sorry, a movie name, a couple of exceptions that were drug names, but the most typical case is this one, which is worlds with field in it. So words with field in it, you know it's not unambiguous; you can basically get everything, so you can get company names, you can get movie names, you can get person names, but nevertheless, it just turns out that following historical etymology, that field is just a strong indicator of being a place name. So almost three-quarters of the instances are place names and lots of the kind of character sequence features you can have of that kind of productivity in it's useful to use them. Okay, there are lots of other more advanced features of various kinds you can also try and use to predict things. Part of speech tags, using pile structures, looking on the web for information about things has been quite popular and quite successful in recent years. If you have things like some of these hand-built list of names, obviously that's useful. For things like biomedical, the identity recognition, doing special things for abbreviations can be useful. I think things on this slider, though, are probably sort of too much work and beyond what you should be doing for the third programming assignment, but you should certainly try out some other things like character subsequence and shape features. Okay and here are just a couple of figures. This was some work me and some students did a few years ago on biomedical name identity recognition. Biomedical name identity recognition is still a hard problem, so for a kind of personal location, entity recognition, numbers are in the 90's but – depends on a lot of the details, but normally people are quoting numbers that are somewhere between 90 and 95 percent F1 measure on entities.

Whereas if you look at biomedical NER results, the results are well down, so but this data set we're getting 83 percent and then for a different data set, which is actually the one we're giving you guys, we were only getting 70 percent F measure. So your goal is to see if you can get better than 70 percent F measure and if you can do that, you are doing well. Okay, so then for the rest of the time, I sort of want to start heading in the direction that heads kind of above doing name density recognition and more into the space of various kinds of information extraction, and this is then gonna be kind of just sort of a survey of that space and then we'll go on to passing next time. Okay, so I've already said a little bit about what the general game plan is, that it's a game plan of well, we have all of, nearly – the primary source of human generated information that we can get information out of is this unstructured text data, regardless of whether you're looking at Wikipedia, or you're looking at something in a science journal, that's where the information is and so we want to try and get out those forms of information.

So these slides I'm borrowing are all from William Cohen, Andrew McCallum and others, who were two of the people who worked for a while at this company called Wizbang, which had one successful product, which was Flipdog, but then kind of met its demise later on. But the idea of Flipdog was to do a large-scale information extraction task, and which was to help people find jobs and this product was actually then bought by Monster.com. So the idea is well, rather than having people have to go out and go to a site, to submit jobs for listing and pay money, as is the commercial model in some of

these things, why can't we build a system that runs around and goes and finds the jobs that people list on their websites. Stanford lists all job opening on their websites, and do information extraction to build a database, and that's the kind of thing that they set about to do, is to do that task in various domains. And so, the finding of job openings, that there's first of all, a kind of a directed crawling phase where what you want to do is you to be able to look at websites and perhaps not through all the entire web, but notice okay, on this website, here's the job listings link and so we should be able to pull down from the home page to directly where the jobs are listed and so they have an ice cream guru and a consumer food relations listing and then at that point we want to look in these job postings and effectively pull out these data base relational information. So the tile of the job, who is offering it, the employer, there's some text categorization, what category of job, there's a location for the job, contact phone number, all the obvious kind of relational information to get out.

Okay, and then at that point, you can do searches, you can ask for a search saying where you want the position, so you want a job in Alaska or something like that, the category of job, and you can get customized job listings. Okay, so if that your task, you can think of breaking it down into several subtasks, some of which we've looked at and some of which we haven't. So we want to fill a database with segments of text like this. So one way of thinking of it, and this kind of William Cohen's presentation of how to think of it, we can think of this as effectively a sequence of four tasks in a row. The first one is the segmentation task, where we're cutting these bits of text which are underlined in black here, which are things that we want to get out, and then the second one, is the classification task of saying, what category are you here; person, job title, organization, to give to them? There have been a few system that have done those two tasks separately, of doing a segmentation task first and a classification task second, but I mean, that's really a minority, that most of the time, as in the kind of models I've been presenting, those two are actually done jointly as a single piece of inference. But if you actually then want to do useful information extraction, there are two further tasks that you want to do. So the first one is associating information, so this is which bits of information should go together to be in the same database record. So if you just have this list of names, you could say that Richard Stoban is the vice-president of Microsoft, right. That's kind of a sequence of entities, but you don't want to be doing that one, you instead want to be grouping these ones together and well, obviously there are some features just like textual proximity, but in general, names of people and companies will get mixed up together in articles, so there's the association task and then the final job is then putting together which of these putative records are actually the same record.

So this record and this record are the same record, even though her the name is Gates, and here the name is Bill Gates and so that's then the task of clustering and so if you can do all of those tasks correctly, then you can produce this kind of database as before, in which here there are only three people named, mentioned, two other named Bill, which have their job titles and the companies that they work for. Okay, so if we just look at the general picture for information extraction, something to think about is just the parameters of different information extraction problems. I mean, implicitly what I've been basically talking about is, assuming that we have continuous running text, think newswire article,

and that we want to things like name identity recognition and information extraction over it and really, inside the NOP community, that's the problem that has been by far the most dealt with in research. But if you actually go out into the world and look at things, that lots of problems aren't really like that, so we've been thinking over here, but once you go into web pages, there's a lot of different structure and a lot of different things that you want to exploit. There's link structure, there's layout structure, there's formatting, some things in bold and large and things like this, there's the underlying HTML mark-up and all of those give a kind of a very different picture of what you want to do in information extraction. So a lot of the NOP work is here, with continuous text without formatting. In a way, you might think to yourself, boy, that's hard because I've got no clues at all apart from the words, maybe that's the hardest case.

I think in practice though, it's actually not the hardest case, that although there are no overt clues of structure, it turns out that having just a single linear sequence of words that do fit together into sentences actually gives a lot of constraint to the problem and in practice it can often be harder to things in other contexts. You can then move to this kind of middle context which you often see in the web, where you've still basically got a paragraph of text that's just like this, but on the other hand, you are getting some additional structure, so there's this little bar down here that's staying some stuff, there's bold over here and stuff like that. But then commonly you'll move to more structured cases again. So here we have a table, so there's still some bits of text that it does running text in the box, but lots of it is being laid out kind of table like with entries, and here's another kind of complex table. So one of the most important things in doing web information extraction in practice, is often being able to do a good job and understanding all of these different kinds of tables because they have helpful information. Like here's a header row, saying something about the topic of each of these talks, but often there's a lot of complexity in the table, right, because a lot of tables just aren't simple, here are five columns and there's always the same thing in each column, because you kind of have this four column structure here, but then interspersed between it, there's these things which has a kind of a vertical structure where the first line gives their rank and the second paragraph, if there is one, is then talking about their research interests. Okay, another parameter of thinking about the landscape of IE tasks, is how controlled versus how broad is the problem that you want to be dealing with. At the one hand, you have huge providers of information such as Amazon, where the information they provide is their product catalog, now I mean, actually these days with Amazon, you can get their product catalog and structured data, but if we just imagined for a site like that, wanting to kind of grab web pages and do information extraction. I mean, actually they've got a piece of software that produces these web pages, that they're not done by people tinkering and an editor. So effectively what you're doing is reverse engineering their bit of software that lays out the web pages and the way the web pages are laid out has some parameterization, but it's essentially completely regular, and so that you can write one very precise kind of wrapper which may be largely rolled out and which can essentially decode Amazon web page and get stuff out of them.

Okay, in the middle here, we have things in which there isn't one way of doing things, but there's quite a lot of commonality, so I mean, people – it's not that there's some piece

of software that in general lays out people's resumes and general people just type it in Microsoft Word or whatever they use and lay it out however they want. So it kind of can be arbitrary, but in practice though, in these strong conventions, you usually have these centered titles, and that's usually followed by addresses and then there might be stuff like objectives, but then often you might have education, or sometimes you have jobs first. But there's kind of a lot of semi-fixed, genre specific structure which you could hope to exploit in an information extraction system. On the other hand, you can then into these very wide and non-specific cases, so if you take the task, which some people are trying to do commercially now for things like people search is to okay, what I want to do is just scour the web and find information about people, I want to find their name, what company they work for, what age they are, some bunch of facts about people. Well then, you're just gonna be scouring all kinds of stuff because sometimes like the academics you might find that kind of information on conference listings, sometimes you can find it in their résumé, sometimes you can find it on their Facebook pages, right. There are just a million and one very different kinds of formats that you want to have and in that kind of space, it would be hopeless to be writing rules for a particular format in the scrape Amazon kind of scenario, and you clearly want some kind of accurate but flexible machine learning kind of framework that can identify person names and get out related attributes of them in context.

Okay, so a third way of thinking about different information extraction tasks, sort of in particular for name identities, is what kind of thing do you want to extract? So the people who sell named entity recognizers in their advertising, that they always tell you that they extract a really large number of different kinds of entities, so that they extract 176 entity kinds, or 231 entity kinds, but if you then start digging down and finding out what those entity kinds are, it turns out a lot of them are always made up of things like U.S. states, Canadian provinces, Mexican prefectures, are they, states? Whatever they have in Mexico, and so these things, they're small closed sets, right. Like, it's not that there's not a teeny bit of ambiguity, there's a teeny bit of ambiguity, because something like Washington, that's ambiguous as to whether it's a state name, but effectively, you've got this closed set of words, and they're the possible candidates and the only thing you have to do is have a classifier in some cases to rule out false positives, for Wyoming you kind of don't really need anything, it's just you find the word Wyoming, you are done, but for Washington, you want to rule out some false positives. Okay, so this easy, you make a list. Then there are ones like U.S. phone numbers. Well, there's some interesting variation here, people write in a different way, they write it like this, they write it with brackets sometimes, they write it with periods, but this is just the kind of thing that if you like to hack on your pillar of regular expressions, you can sit around for half an hour and write a really good regular expression that does a really good job at recognizing phone numbers. Sot here's kind of that space, and some parts you can do really well like that with regular expression and you might wonder why do anything more?

Then there are things that are still, they're kind of structurey and might make you think of things like finite [inaudible] regular expression methods that are kind of getting more complex, so something like U.S. postal addresses. Well, they're fairly regularized, you get the number and the street and the town, state, but there's starting to be more

complexity here because well, the states might be a closed class, but really you have to be treating the streets as an open class because people keep building more of them, right, and you're not gonna know all the possible street names. And then you go into something like person names and person names is perhaps the classic case of you can't possibly have lists that cover everything. You can't possibly have a regular expression that finds – and there's kinds of, lots of cases they're ambiguous. Like Charles Schwab, whether it's the person or the business or things like that. So you really need to be having something that yes, in knows about name patterns, it knows that first names are normally capitalized, but it's using clever features in the context, and lots of sources evidenced to see whether this is a reasonable person name. And I mean, of course, one of the possible sources of evidence that people often use is name lists. You can kind of download the 200 most common baby names and there's no reason not to use that as a feature, but clearly you want to recognize lots of things that are outside that class.

Okay, and so then the final dimension of complexity that I mentioned here is how complex your information extraction task is, and at that level, the simplest task is when you just want to do something like name identity recognition, where this is kind of like a one column table in data base terms. You just want to get out all the person names that are mentioned in this document, and you don't need to do anything relational, so that's the easiest case. Then the middle case here, and one on which there's been a ton of work and practice is all you're doing is learning binary relation. So you've got some relation of person and employer say, and you want to get person names, company names, and employment relation. But then, in general you can go to arbitrarily complex relational structures, so you might have [inaudible] relations and you might even be hoping to fill out a list of tables or something lie that. Okay, right, so up until now, if you then think about these four classes, that everything that we've done has been segmented and classified. That's been what we spent really two-and-a-half classes on or whatever it was for the NER, and I'm not gonna leave it immediately, I'm gonna show two more slides of segmented classify, but then I wanted to say a little bit that talks about the associating cluster that gets into the other classes. Yeah, so just so we've got a real life example, this is the kind of pattern people write when they are writing regular expression style, information extraction systems. Have people written patterns like this, done that kind of information extractor? I certainly have, sometimes if you just want to get something working it's just the quickest and fastest way to go about it. So what this one is meant to be recognizing is academic conference names, so there's a pattern up here for ordinal numbers, written out as words, first, second, third, fourth, fifth, etcetera, and if this is big enough that you can read it, I mean, it's really easy to poke holes in this if you just look at it as a pattern.

And I think that's kind of typical for these handwritten patterns, that they sort of work, they always have lots of holes in them. So the way this one is written, ordinal words are written out in a disjunction of the regular expression of the fifteenth, and it seems to be like the conviction of this person, that once you get past 15, it's gonna be changed into a number, so if you have the sixteenth conference on discreet algorithms and you write out sixteenth as a word, this pattern fails immediately, but then it's got a pattern for numbers – ordinals written as numbers where it matches 21st, 22nd, 23rd and ones like that and

this is all of these kind of pull tricks question mark, colon and all that stuff. I won't go into it. But if you look at that pattern carefully, it actually only matches two digit ordinal numbers, so again, if you're – maybe this was written by computer scientists for computer scientists and none of – no computer science conference has gone long enough for it to be up to three digits, but as soon as you're up to the 101st conference on discreet algorithms, this patter is also gonna fail to match it correctly and then it kind of keeps on going and you write these kinds of things. So all I really want you to take away from that is, if you're actually writing these regular expressions and getting them so that they work even reasonably well, and this one has lots of holes, you end up writing kind of a lot of complex, hard to read and hard to maintain stuff. And so then the contrast with that was with the kind of machine learning methods that we've talked about and you know, they're good and they're especially good at more flexibly matching things that they haven't seen before.

I mean, of course they're not a panacea, because you can only the kind of machine learning based entity classification we've talked about if you've got training, if you've got training data, and so then in any kind of domain specific task, the question is where is the training data and there's kind of an investment needed to produce that. And I'm not really gonna talk about it today. I mean, it's something that there's been a lot of interest in recent years is, are there much more effective ways in which you can do things in semi-supervised remotely supervised ways rather than building the classic kinds of training data and so for at least for some things such as person, location, name density recognition. A great source of information these days is Wikipedia, that there are several people who have shown that you can do very successful name identity recognition by effectively having fairly simple patterns that pick out candidate names, and then you run off to Wikipedia and see if you can find that name in Wikipedia and then see if it's a person name, an organization name or something like that. If someone wanted to try that for a final project, that could be a fun thing to do, provided you don't hammer Wikipedia so hard that people get upset with us. Okay, so now let me say a little bit for the end of the class, and talking about these final two tasks, which are the associated and cluster tasks. Okay, so at this point we then want to go on beyond just having the – recognizing the entities in the text and actually sort of put them together as a triple, okay. So this is a disease, this is an instance of the disease outbreak relation where on this date; this disease broke out in this place. And so this is the kind of task that an information extraction system traditionally did, was to fill these kinds of relations out. Okay, another very common case of this in the biomedical domain is during protein interactions, so that you're finding the names of lots of proteins and then you're wanting to get out these interaction facts. So this guy and this guy interact, that one is pretty obvious, but then later on this one does not interact with that, but it associates with a complex and so you want to get out this association fact. And the crudest method of doing this is just to sort of use textural colorants to say, it's a putative interaction, but that obviously leads to false positives and errors of various sorts and you want to be able to do better.

Okay, so how the people go about this, so these days basically the de rigueur way of doing it is a two step or long multi-step relation where first of all, you are doing name identity recognition and then you're building higher level relations on top of the output of

name identity recognition, and at that point, for doing the higher level stuff, you've effectively got exactly the same choice before. Well, you could do the high-level stuff in the hand specified writing rules for things or you can do that as machine learning based as well and I'll say a little bit about both approaches. So New York University is kind of a traditional place, it's had a long history of information extraction, and they have more recent, more machine learning based work, but a lot of their work has been around this Pruteus system, which is again, a hand based rule – hand made, rule based information extraction system and as you can see, it's old enough that it's written in lisp. But we've got these kinds of rules, which are kind of looking at bits of text and getting out things. And again, the rules that you're building are quite long and detailed to extract various things. There's then been a lot of other work. There's been quite a lot of work in machine learning relational patterns, so that you're learning relational patterns which relate entities and then are looking for words or texts that occur between entities and therefore seeing them as instances of particular relations or not.

Okay, another way of thinking of that is just as a binary classification task. So if you want to get out instances of a person and a role, well, one way you could – whereas here there's a kind of a conference law, one way you could do that is have named entity recognizers, which recognize the person, role pair and then say okay, within some window which might be a paragraph, if I see a person and a role in the same paragraph, I will expect them as a pair and then what I'll do is I'll have a second phase classifier and the second phase classifier's job is just to say, binary, is this an instance of the relation or is it not an instance of the relation? And so that's a technique that's been applied by several people and this classifier can then of course make use of the structure of the sentence in arbitrary ways, so it can look at what words occur between them, it can pass up the sentence and use the sentence structure, etcetera, etcetera. Okay, another common technique has been to build finite state machines that then go over the text and so that you're got the kind of NER recognized elements like proteins and locations and then you're writing finite state patterns which can then connect them together in ways that are valid ways of recognizing relations. A more general and clever machine learning framework is this work that was done by Dan Roth at Banner Champagne and so in this work, what it's doing is setting up one big joint probabilistic inference model so that what you have is entities that you can extract from the text and they're going to be given classes and then between each pair of entities, there can be a relation, so that you have these relation nodes.

Okay, so the entity random variables are then doing the class of the entity, personal location and then the relation random variables are talking about possible relations. And so then the little models that are on the side here, are then kind of local context models that let you use local context aside, how to classify things. So this is kind of like the kind of features we've done for NER, of looking at the context of segmented entity to say, is this a person name or location name and similarly, this is looking at the sentence context aside possible relations. But then the final clever part of this is you're then doing joint inference over all of the nodes to work out what's the best assignment of variables and so he's doing that with a loopy belief propagation style inference. Okay, how well does it – various kinds of information extraction work? To a fair extent, I guess I kind of feel like I

could sort of, given the slide ten years ago and the numbers that I would have put up really aren't any different than what I'd put up today. For recognizing entities, you can get very high accuracies. It turned out that people could get very high accuracies even using fairly crude methods when they worked at tuning them very hard for a long time. Maybe we have better, more automated machine learning tools, but either way, for entities you can get high accuracy. So for things like person, location, names, normally accuracies are quoted as above 90 percent. But what happens when you try and do more difficult tasks? So what happens when you are trying to get out these kinds of multi-place relations or attributes so that you want to have something like what's the title of a person, going to sort of facts, these multi-slot relations. Events here are kind of being defined as these slightly bigger things, like an event in the news of somebody kidnapped somebody and they belong to some organization and what they want for the person's release is this. So you're doing these kinds of bigger stat style information extraction tasks. The picture is, and this has sort of been true for a decade, that although people do try and do clever things with joint inference and things like that, that basically the errors cascade doing the largest scale tasks of working out four relations correctly.

It's just much harder, and so the accuracies really go down, so the kind of figure that a lot people have in their head is that if you're wanting to extract sort of relations which are something like three, four spot relations, that a typical accuracy to be getting out for [inaudible] correct records is only sort of in the range of 60 to 70 percent; which is not completely useless, but on the other hand, it's not wonderfully impressive either. And so it's still really an active area of research, as to well, how could you do this more accurately or in a more middle level as to what kind of technologies you can use, such as just exploiting the redundancy of information and finding information multiple times to improve on these accuracies, and I think myself one of the kind of interesting ways going forward is to more try and turn this into a closed loop feedback task so that most of the time, information extraction systems just run through text, they extract whatever they do from their fixed model and they go on to their next bit of text and they do the same, and the information extraction system when doing the next bit of text is taking no advantage of what they've extracted from previous bits of text; and it's not involved in any kind of closed loop learning system and doing something more along those lines seems to me a hopeful direction to be heading. And then this note at the bottom says, sometimes even these numbers are very optimistic, because these numbers are effectively when you've got a system that is well tuned to a particular task. I think everyone has had the experience of you take your name identity recognizer, which was trained on news wire text and recognizes person's names with 94 percent accuracy and if you start then running that name identity recognizer on something like foreign posts, you will find that you'll be lucky if the percent accuracy is 70, 80 percent. You get this huge mismatch between the training and test data that greatly lowers accuracy. Yeah?

Student:[Inaudible].

Instructor (Christopher Manning):So, I mean, I think I'm being bad here since I just loosely labeled that accuracy at the top where I really should have labeled it not as accuracy, but F measure; because really, these are F measure things. But for a roughly

balanced F measure, this is kind of nevertheless, sort of the percent of things, roughly that – yeah, so I mean, you're getting errors here both from precision errors and recall errors from things that you missed, yeah.

Okay, and so then after that, the final task once we've got some multi-place relations is then this task of clustering, of when the two extracted strings refer to the same object. Okay, so here are my two Kennedys. Okay, and so this is a huge problem that is thought to be something that there's been a very large amount of work on. So typically you're finding all of these names in various places and the task that you want to do is this task of saying, okay, some of these names are talking about one person and some of these names are referring about, to another person, and so this is this problem of there's some – there are lots of names that it's given, and it's been attempted in lots of fields, so it's been attempted in fields that range from the database end of the world where it gets names like record merging and Dejuping and names like that.

It's referred to as entity resolution quite a bit in the machine learning world, in that in LP world is most often referred to as co-reference resolution of working out which names co-refer to the same person. And it's a problem that plagues a lot of real life applications so most of you have probably – know Stuart Russell who is co-author on the AI textbook for 221. He worked on this with a student for a few years and he describes that the reason that he started working on this problem was he was frustrated by how many entries there were for artificial intelligence and modern approach inside Sightseer, which were really all entries that were about the same book; but because of various kinds of variants in how the name was written in different references and citations, which is all this stuff about do you put down someone's full names or just their initials? And all these various different factors will cause records not to collapse together in Sightseer and so he had something like 50 different variant versions of his book appearing in Sightseer and so the goal is to how you can then put these instances together, and some of the features are obviously textural overlap features that both of these have Kennedy in them, but when you go to other domains like things like conference names, there are a lot of other factors because you want to recognize also strings that have no words in common with each other for things like acronyms, so ICML International Conference in Machine Learning, you want to recognize are the same. And some conferences it turns out even have even kind of weirder pairs of names so in natural language processing there's a conference that's regularly abbreviated as COLING, but its full name is International Conference on Computational Linguistics and you can sort of make a mesh together of computational linguistics into COLING, but it's not any kind of conventional acronym. And so people are building various kinds of, again, probabilistic relational models that are doing inference about connecting these kinds of names together.

Okay, and that inference of putting things together occurs not only at the level of just individual names but also then occurs at a higher level and in general becomes more difficult at a higher level. So when I showed those figures for the accuracy of doing events where it was around the sort of 60 percent mark, one of the reasons why that task becomes so difficult and the accuracies become kind of low is that if you're reading multiple pieces of text and they're talking about okay, that somebody had a collision and

somebody ran into somebody and somebody did this. You have to work out which of these different events are talking about the same car crash and which ones are talking about different car crashes and unless you do the co-reference of events correctly, well then you'll be starting to have false extra events and you'll be hit on precision and if you merge too many things you'll both have errors and incorrectness and failures in recall. And so this event co-reference is a different task and so here's the kind of picture that you typically get if you've tried to – if you've looked on the web in various course listings and you've got two things out like this. I mean, to a human being, it's kind of obvious that these are the same things, right; but if you look at them basically nothing's the same, all right, so 101, CS101, Intro to Comp Size, being abbreviated there. The name, here it has a title and a dieresis mark on the U, here the title is gone and the dieresis is gone, but you've got some initials. This one lists the start time, this one lists the time, but the start times actually aren't the same because here it's 9:10 and here it's 9:00. So it's all kind of a little bit different, but it's obviously the same and so that somehow you want to match, that there's enough sameness there so you can do event co-reference so that you can make this only be one record in your database. Okay, so basically that's what I had for today. So I guess we can end a few minutes early for today and then next – don't forget to come pick-up the assignments and then next time I'll start into doing parsing.

[End of Audio]

Duration: 66 minutes