NaturalLanguageProcessing-Lecture05

Instructor (Christopher Manning):Okay. Hi, everyone. I'll get started again for CS 224n. Okay. So today's the final lecture on machine translations, and so today I plan to tell you everything else there is to know about machine translation. Well, I can't exactly do that, but I guess I sorta wanna concentrate on the sorta main additional ideas and do something of a survey that gets up to what's currently basically the state of the art, and what people do for statistical machine translation systems. And an interesting thing that we'll come to in a little bit is these IBM models that we're looking at. They actually are still used in state of the art machine translation systems, though they're not the whole system. They're actually used now as a component in how those systems are built. So first of all, I wanna quickly finish going through the hierarchy of IBM systems. And then I want to talk about evaluating MT systems, because that's, in general, important to know how you got a good one, and also comes up in the assignment. And then talk about some of the modern ideas in statistical phrase-based MT systems and syntax-based MT systems. Okay. And so, just on the other announcements, don't forget today's when the first assignment is due. And also today, we've got the second assignment coming out, which is then the statistical MT alignment models, which you can put together with your language model and have a little MT system. And then on Friday, the section's gonna be talking about the kind of EM algorithms that you need to make the assignment. Okay. So we spent a lot of time on IBM Model 1, and so at a high level, the important thing to think about is that, essentially, the only information that this uses as an alignment model to translate words is it's trying to learn these probabilities of how to translate an English word into a French word. That's all it's got. All other knowledge of language is completely disappeared.

So it doesn't have any ideas about knowing about phrase structure. It doesn't know that words at the start of the sentence will probably be translated near the start of the sentence. I mean, it literally knows nothing else. So in some sense, its character is very like a 1950s machine translation system, where the way you translate is you come to the next word, you look up in the dictionary the possible translations, and the only difference is we have a probability distribution over possible translation, rather than just the list in the dictionary. I mean, in particular, there's a question for these guys. So words – lots of words, as we discussed, have lots of meanings. The most famous word ambiguity in English that everyone uses as their first example is "bank," which can mean either the edge of a river or a financial institution. So when we want to translate that into French, those two meanings of bank get translated differently. So if we build a Model 1 – actually, I should be doing it the other way. I should be choosing a French word that has two meanings in English, actually. So if we're doing French to English machine translation, if we're taking a French word that has two possible translations in English, if we're building a Model 1 model, can these alignment probabilities help us choose which translation is appropriate in a certain context? Yeah?

Student: No.

Instructor (Christopher Manning):No. So these alignment probabilities are completely context-free. All they know is margin statistics. They know what the most common translation of that word is, and the second most common translation. Okay. But nevertheless, if we're building French to English machine translation system using the noisy channel model, have we got a chance of getting that right? And if so, how?

Student: Can you incorporate ideas from a language [inaudible]?

Instructor (Christopher Manning):Right. Yeah. So our language – yeah, so the answer is our language model. So at the moment, our whole system is built on this leap of faith that, essentially, our language model can save us. So the hope is that all the alignment model knows is more and less common with probabilities translations for an individual word. And so it'll just sorta suggest those as possible translations, and then we just pray mightily that when the language model comes along, it will save us, and it will propose appropriate translations in context. And you know, sometimes that's actually going to work. So that if you have something where there's a good context, like if you have a word – I mean, a lot of the time you can get context for this ambiguation by looking at what precedes. So that if you have some kind of complication like "river bank," the fact that river's the words before gives a lot of context as to what sense of the word follows. And so that context will go through the translation. And so the language model will prefer to put together words that have kinda good mutual collocational information that fit together well with each other. So quite a bit of the time you can actually choose appropriate translations by having the language model do all the work. On the other hand, if you start thinking to yourself, boy, you're really hoping for a lot there, that can't work all the time, you'd be absolutely right. It absolutely doesn't work all the time. Okay. So then if we head up the IBM models, I mean, essentially, the models get a ton more complex. But really, they only put in three very simple linguistic ideas that they try to capture. So it's just this minute bit of knowledge of how languages work that gets added as you go up the models. So the first bit that I talked about last time was when we went to Model 2, we have this idea of displacement, which is, by and large, the words at the beginning should appear at the beginning of the translation, and by and large, the words at the end should occur at the end of the translation. That's pretty crude, but it's a little bit true. And that gives us some value. Okay. And so then today I want to go on to the next three models, which add in a couple more ideas. I'll just mention at this point, in current practice, and you see this in the Jurafsky and Martin book, after Model 2, people normally intersperse what's called the HMM alignment model, and that's discussed in the Jurafsky and Martin chapter.

It wasn't an original idea model, and I'm not gonna present it here right now, mainly because since we haven't talked about HMMs at all, that'd be a big digression. But in practice, people have found it a useful model to stick into the hierarchy there. Okay. So if we go onto Model 3, what we're – the big idea that we wanna capture here is when we're doing this word alignment, that what we were doing is that we were just going through each position. So we were saying, okay, let's go through each position and let's have an alignment variable that gives a source for that position. And so there was nothing in our assigning alignment variables that stopped us aligning every word to one word. The alignment variables are independent of each other. And so that, in part, is a very good thing. Keeping the alignment variables independent of each other means that you can train Models 1 and 2 very efficiently, and it's precisely for that reason that we only do Models 1 and 2 in the assignment. Because if you want to start having complex dependences between things, things suddenly get a lot more complicated. In fact, you'll find in the text of the second assignment, when it suggests things that you could do for extra words, one of the suggestions – in fact, I think it's the first suggestion – is you could try implementing Model 3. As a little warning, no one who has attempted that has ever succeeded in getting it to work in the length of time you have to do assignment two. So unless you're really, really feeling gung-ho, you'd probably be better off ignoring that suggestion and looking at the other suggestions of ways in which you might be able to make things better. Okay. But at any rate, the idea that what we could do is just say that every word could be aligned to the same word, that's clearly linguistically ridiculous. In particular, there should be a kind of an idea that a lot of the time words do translate one-to-one, "proposal" to "proposition." That one-to-one is common. On the other hand, depending on the language pair, there are just regularly cases that'll be exceptions to that. So in French, "not," at least in written French, is standardly turned into this "ne... pas," which surrounds the verb. So you'd actually expect the word "not" to have the fertility of two when going from English to Spanish. And similarly here, "implemented" as being translated by this sorta put-in implementation little phrase. So "implemented" has a fertility of three, that it chooses three words in the translation.

So we should, actually, be able to do a better job of aligning words if we actually learned some information about the fertility of different words, because then we'd kinda know appropriate ways to do the alignments for them. And so that's precisely what's in Model 3. And so this is a picture of what Model 3 looks like. So Model 3 is, again, being done as a generative probabilistic model, and it's just a much more complex generative process. So what we're wanting to do is come up with an estimate of the probability of F given E, just like before. And the way we do that is if we start off with an English sentence, we first of all have a fertility for each word, where the fertility is a probability distribution over different fertilities. So if we think of it as a generative model, we roll our dice, and based on the fertility of each word, we work out that it's going to become some number of words in the other language. And I'm indicating that here just by replicating the word. So "did," when we rolled the dice it said its fertility is zero, so we just have it go to nothing. But "slap," we roll the dice, its fertility was three. And so I now produce three copies of the word, "slap." Okay. And we still need to have words in the output that have no source in the input, the spurious words. And so we have a second probability, which is a probability of between each word, sticking in one or more null words. So we stick in one null word here. And then at that point, we then have a translation model, which is like what we had in Model 1. So this is the equivalent of the probability of F given E in Model 1, where we're just doing a single word-for-word translation, which turns our English words in to here Spanish words.

And then finally, we still wanna keep what we inserted into Model 2, and so we also have the displacement model, and the displacement model is telling us the probabilities of these words being displaced relative to their position in English. And so that will reorder the sentence, and that will give us our output. So effectively, this is just a considerably more complex model where we now have a bunch of probability distributions. So we have fertilities for our words. And again, there are strong independence assumptions in this model. So fertility of the word just depends on what the word is. It doesn't depend on other fertilities or surrounding words. And then we have the translations of a word, which is still just like what I was complaining about. They're still just being done independently. So we're still relying on the language model to save us. And then we have these displacement probabilities, which are just like Model 2. So again, there are huge independence assumptions there. So displacement is just how likely words are to move how far around. It's not specific on what the words are. And so that's kinda what you're seeing here. So the translation probabilities, chances of putting in extra word for the translation, where that "are" is the Spanish "are," not the English "are." Fertilities of words, and then distortions. So these are the kind of distortions that you'll want to have in Model 2 for the assignment. So the distortion d(5¦2) is the probability that the English word in position two of the English sentence generates the French word in position five of the French translations, where formally, you want to have that conditioned on the length of the two sentences, where you've generated a length for the output sentence so you get a well-formed probability distribution. Yes?

Student:

[Inaudible].

Instructor (Christopher Manning):So, yeah, you can rely on the language model – you can hope dearly that the language model will do everything, and that's what Model 1 does. And to some extent, yes, since the language model will want to keep words in good order, to some extent that will capture what the distortion model does. So you can hope that even though your Spanish starts off with the order of article, noun, adjective, that when you translate into English, the English language model will know that adjectives don't come after the noun in English. And so it will just move it around and say, no, no, it will be much better if you produce article, adjective, noun. Yes, and that may work for you. But, I mean, there are crucially a lot of circumstances that it won't work. I mean, a kinda obvious one is if you take a sentence like, "A dog bit a man," or something like that, and you – well, I want to go with the opposite, the man biting the dog. So, "The man bit the dog," is the actual source sentence. Okay. If you, then, translate it without any knowledge of distortions, and you have a really good language model, the really good language model is just looking at the words that you're offering it, and it's gonna say, "Man biting dog," that's kinda really rare. "Dog biting man," that's pretty frequent. Let's move those around, and look, I get a much better language model score, so let's go with that one. And that would be wrong, right, because we've lost the semantics. And if, in that case, you had a distortion model, the distortion model, fingers crossed, might encourage you to get an output, which was still, "The man bit the dog." There's a lot of keeping your fingers crossed there, because you could imagine it going wrong in various ways, but at least it kinda indicates why distortions might be useful on top of the language model.

Okay. This is just a little fine point. I mean, on the distortions, you could kinda have the nulls inside the distortion probabilities for generating spurious words, but in practice, people don't normally do that. And the spurious words are just put in the positions that aren't claimed by the real words, because really, they don't have a position in the source sentence themselves. I mean, in particular, you kinda don't really want to say that the source for all the spurious words is at position zero at the left end of the sentence, because that would mean that most of the spurious words would have huge distortion penalties, where really they shouldn't have distortion penalties at all, because you're just inserting them in places in the sentence that are grammatically appropriate. That's another slide going through Model 3. But maybe I've had enough Model 3 at that point. And so, really, at the high level, the essential thing to have in your head is just there's this one new idea. We're modeling how many words an English word is typically translated by, and that will give us better translations, because we won't be able to get these funny alignments where multiple words align to one word. And that's something – you probably shouldn't build Model 3, but that's something that you should look out for when you're looking at the output of your Model 1 and Model 2. I mean, to digress for a moment on the day that the first assignment is due, in the assignments, it has this discussion of how we like people to do some error analysis and look at the output and think about it and say some pertinent things about it. And we really do. But what we'd like not really is just that your write-ups are as long as possible by dumping out lots of junk of probability tables or something, but that your assignments, rather, appear to have had some mental energy that went into the write-up. And the mental energy is really in looking at what it does, and being thoughtful about it, relative to kind of what you'd like to have a system do.

So for this machine translation case, one thing that you could do is look at the alignments your system is getting, do see obvious places in which multiple three or four words are being aligned to a single word inappropriately, which is just the kind of thing that Model 3 is meant to fix. You don't actually have to write Model 3. You could, nevertheless, in your write-up, show a couple of examples of that, and sorta point out that this problem is, indeed, occurring with your model, too. And therefore, you kind of understand better now why Model 3, perhaps, is a useful thing to have around. Okay. How do you learn Model 3? I mean, the short version is in exactly the same way you run an EM. The long version is that, boy, is gets a lot more complicated. So we still start off with the same parallel sentences. And so we still, just like for the other EM algorithm, we just give some values to all of these parameters. And then what we're gonna do is, based on those parameters, find likely alignments. Then, based on those likely alignments, we're gonna re-estimate all the parameters. Because once we have likely alignments, we can just look at the fertilities. If a word has been aligned to three things, we say fertility three, and put that into our counts and run the EM algorithm. So at a very high level, it's exactly the same, and it's not really any more complex. At an implementation level, it becomes a ton, ton more complex. And the reason for that, I'm gonna kinda just gloss over here. But it's really an important thing that you should think about and understand. And for doing that, you should be looking at Kevin Knight's MT tutorial and in the Jurafsky and Martin Chapter 25. But the high-level point is if you're doing Model 3, precisely because you put fertility into the picture, that alignments are no longer independent. Because whether one alignment exists affects the probability of whether another alignment to the same word exists, because you've got a fertility factor in there. And so the fact that you've lost that independence assumption between alignments means that it's very hard to explore the space of alignments and work out the probabilities of different outputs. And in fact, in practice, people use various kinds of heuristic search procedures. Whereas in Models 1 and 2, precisely because alignments are independent, putting in one alignment makes no difference to the probability of another alignment. And it's precisely because of that that you can kinda ignore everything that goes around you and just kind of work out expectations for individual alignments, as I showed in that slide for Model 1 and said just implement this and it'll work.

But I really encourage you to actually sorta read through Kevin Knights tutorial and/or the Jurafsky and Martin, and actually kinda understand what assumptions you have to make to get that to work and why it breaks down for Model 3. Okay. So then we go on to Model 4. And I mean, Model 4 is, perhaps, best illustrated by this picture. So here we have a French and English sentence. So the English is, "On Tuesday Nov. 4, earthquakes rocked Japan once again." Now, what happens when we translate into French? Well, we have a word with a fertility of four here, because earthquakes is being translated by, "Des tremblements de terre," so, "the trembling of the earth." Okay, so that's kinda cool. "Have been newly rocking Japan." But the bit I want to focus on right now is right here at the beginning we have, "On Tuesday Nov. 4," and then the French translation appeared right down here, "Tuesday 4 November." Okay. So what happened was that this temporal modifier moved right from the beginning of the sentence to right at the end of the sentence. And to some extent, that's just a choice that sorta doesn't even mean a lot, because the English sentence could have been, "Earthquakes rocked Japan once again on Tuesday Nov. 4." But nevertheless, there's kind of slight preferences between languages, and in English people more commonly put those kind of temporal modifier at the beginning, and in French they're more commonly put late in the sentence. At any rate, this was the way the translation was done. Okay. Well, if you're wanting to learn this is the correct alignment, and you're using Model 2, you're kinda gonna be in big trouble, because if we look at these alignments, they're all gonna involve huge displacement penalties, because this word has a huge displacement penalty, this word has a huge displacement penalty, that word has a huge displacement penalty. And that kind of seems unfair, because really what we have is this one phrase that is expressing the date that's moving from the start of the sentence to the end of the sentence. And one appropriate way to start thinking about this is, gee, we can build models that actually have some phrase structure in them, so they know about the structures of phrases and languages, and we can model things that way. And that, indeed, has become an extremely popular topic in recent work in statistical machine translation that I'll mention in the end bit of the class.

But IBM Model 4 did something simpler, and what it did was, essentially, just to handle these cases by saying you should only be penalized badly once. So when we aligned "Tuesday" and "Mardi" here, you're gonna be penalized for there being a big displacement. But when you're then considering the alignment of four, the alignment of four's displacement can be calculated relative to the preceding word. And so, therefore, here you'd say, well, there's no displacement, because it's gone along with the previous word. And this one has no displacement – oh, actually, sorry, these two are swapped. So I guess there is a displacement of one. But at any rate, there's a small displacement here, and a small displacement there. And so you only get small displacement penalties, because you can base the size of the displacement on the previous word. And so that kinda simulates saying that it's roughly okay to move whole phrases, that you just get penalized once for moving the entire phrase. Okay. And then the difference between IBM Model 4 and Model 5 is purely technical and so I'm really not gonna dwell on it. So the way they did IBM Model 4, the model was deficient, which means that it didn't actually give a probability mass of one to possible alignments. Some of the probability mass was lost on illegal alignments. And so on Model 5, they did a bit more hard work and they managed to come up with a way to have a non-deficient probability model. But that's purely technical. Okay. So one final question that it's just good to have a sense of is, well, why do we have all of these models, 1 to 5? I mean, of course you could think, well, this is just an historical recapitulation. They invented Model 1 first and saw how that worked and decided it wasn't very good, so then they built Model 2, which was a bit better and kept on going. And in some sense, that's probably true as to what they did. But it's not really capturing why people still talk about all of these models. And the answer for that is that people actually still use most of these models. I mean, sometimes they don't use all of them, but they at least use most of them. So maybe in the current system, the kind of thing people might commonly do starting off building models is they'll run Model 1, they might skip Model 2 but use the HMM Model, and then they'll run Model 3, and then they'll run Model 4, something like that. They really run a bunch of these models. And the reason that they do it is, effectively, having simple models is a very effective way to bootstrap the more complex models and sort of the flavor of things.

The flavor of things is something like kinda doing some kind of a kneeling process, so that the idea is for the high-order models, they have a ton of parameters. And if you just directly try and train those models, you get a minor bad effect, which is it's really slow to train the models. And you get a major bad effect, which is since you're trying to do EM optimization in this highly non-convex space, that you very easily get stuck in bad local maxima, and you come out with a very bad model. So instead of that, what you can do is, first of all, train Model 1. So Model 1 you can train really fast, because there's no search over alignments. And you can get out of it sort of – you know, Model 1 isn't a bad space to search over, so you can find decent models to the extent you can relative to Model 1. And so, you can precisely use those models to initialize Model 2. And so that will mean the Model 2 parameters will already sort of start off in a sensible part of the probability space. And then you repeat that over. So you train Model 2, and then you can use the Model 2 parameters to initialize Model 3. And so that means in the very first round of EM estimation of Model 3, rather than it being like when we did Model 1, where all you were doing was counting sentence co-currents and getting out who knows what for things like fertility probabilities, that your first estimates of your fertility probabilities in Model 3 are based on the alignments that you found with Model 2. So they're hopefully already halfway reasonable and you kinda hope to kinda ride a wave in the good part of the optimization space and find a good model. Yeah?

Student: By the time you iterate up to Model 5, do you just use Model 5, or do you use some sort of interpolation between the five models?

Instructor (Christopher Manning):Normally people just use the final model. I mean, that's certainly an idea that you could think of trying, but in practice, people don't do that. I guess I'm not actually – I guess I actually feel I don't know for sure that there's no value in that. I can't think of anyone who actually tried that and reported that it worked either better or worse. So I don't actually know an answer for that. But clearly, that kind of interpolating simple models is something that's been used to good effect in other contexts, so you can imagine it actually working to do that. I know. That's a good idea. Okay. But then on the final note on alignment models, it is important, in some sense, they end up with these big, complex probabilistic models. I mean, at the time that IBM first proposed these models, which was in the early '90s, the models were so complex that, essentially, only someone with the resources of IBM could actually build models like Model 4 and Model 5. I mean, fortunately, in the intervening 15 years, computers have become way, way, way cheaper and faster, and these days, anybody who has a dozen Pentiums can train these models up in a fairly short amount of time. So that's good for the field. But it's important not to be lost in the complexities, to not realize that there's still just very little knowledge of language in these models. So here's this simple fact about French or Spanish, that in French or Spanish, adjectives normally come after the noun, and in English they basically always come before the noun. You kinda like to know that. You'd kinda like your machine translation model to know facts like that, that you need to reorder adjectives and nouns when you translate between the languages. And so it's crucial to realize that there's just nothing in these models still that can actually capture facts at that level of generality.

All we can learn is that displacements of one in either direction seem to turn up kind of commonly between these language pairs. But displacements are global parameters. They don't know about the part of speech of words or anything like that. That's obviously a suggestion of something that you could do to try and improve these models. If you knew the parts of speech of the import, and you learned part of the speech specific distortion parameters, would that help? I think it would. I mean, that's the kind of thing that you could try. So really here, though, we're only relying on the language model to try and fix things up. Okay. So now I skip onto the next lot of slides. Okay, so the next thing I should say a little bit about is evaluation, then I'll go onto more advanced work in machine translation. So here are some illustrative translation results to have something in your head. "La politique de la haine." The reference translation was, "the politics of hate." And the IBM said, "The policy of the hatred." And you kind of see that the machine translation is sort of more literal, like it's translated the "la" as "the," and the "de la" as "of the." So "la" is "the" both times, where the human translator has decided that putting in "la" as "the" in the English translation isn't really appropriate in English. The IBM translation also sorta goofed by changing "politique" to "policy," rather than "politics." Okay. So we get some translations out, and our question is, well, how are we actually gonna look at these translation and work out whether they're good or bad translations? And up until this day, the gold standard answer to that is, basically, we get human beings to look at them. And the problem is that model's kind of expensive. In particular, it's expensive if you're like most people in the machine learning LLP business, spending your days trying to change and improve your models to make them better.

So if you have lots of money on hand – I know for the Microsoft machine translation group, they literally, every week, take the output of their latest research system and ship it off to an evaluation team, who do human evaluation. And they just get back weekly deltas on how the systems is improving or getting worse based on human evaluation. Well that's kinda good if you're Microsoft, but that doesn't work quite so well if you're sitting doing research at Stanford University, notwithstanding the fact that everyone else thinks that Stanford University is very rich. So that's sorta led to an enormous interest in other ways to do machine translation evaluation. And I'll say a bit about that. But before I do, how do people do manual evaluation? I mean, there are various ways in which it's done. One way is just doing sort of subjective scores of how good is this sentence, seven-point scale. The common idea that's often used is this division between adequacy and fluency. So the idea there is you're trying to differentiate the adequacy of the translation, which sort of means it does translate all the content that's in the source sentence, but I don't care if the translation sounds kinda stilted or, perhaps, it even put in the termina where it would have been better – an article like "the," where it would have been better to leave it out. Don't care about that. Is all of the sense of the original recoverable? That's adequacy. And then fluency is is what you're producing expressing the ideas the way that the target language speaker would, correctly using grammar function words and all of that kind of stuff. Sometimes people just do a simple binary good/bad on translations. I guess what that's doing is sort of, by giving a very easy evaluation task, you make it much easier for the humans to quickly evaluate sentences. And the second way of doing a machine translation evaluation – this kinda parallels what we did with language models and perplexity versus word area rate – is to do it in the context of a task.

So if you want to do question answering of foreign language materials, you could kind of have the internal MT component. But again, the problem is it's very hard to know how the internal components results are kinda being reflected through the rest of the system. And so that's led people to want to try and come up with automatic evaluation methods for MT, where you can just sort of evaluate your new system every night and decide whether it's getting better or worse in an automatic fashion. And so the problem of coming up with automatic metrics for machine translation is it's kind of difficult. So why is it hard to automatically evaluate machine translation?

Student:[Inaudible].

Instructor (Christopher Manning):Yes. So fact one is, normally there are just lots of good ways to translate something, so it's not that you can say – it's not one of these tasks of here's the unique right answer; did they get it right or wrong? There are lots of possible translations. And human translators could quibble a little about whether one's a little bit prettier or more fluent than another one, but to a large extent, there are just lots and lots of translations. And in particular, as a sort of an aspect of that lots and lots of translations, normally there are lots of ways that you can order things that are okay. So, "yesterday I went swimming," or "I went swimming yesterday." Both of them are okay. So you kind of can't go through in any position-by-position basis. So doing word error rate for speech recognition just doesn't work doing a kind of edit distance with insertions and deletions, because there are these big reorderings that can be perfectly okay. Occasionally, people have used an idea of a position independent word error rate, where you're allowed to have any alignment between the words, and then you're trying to account substitutions, insertions, deletions. But if you do that, then it's paying no attention at all as to whether you've got the right words modifying the right words, and that seems so lax.

So people have tried to come up with various other ways of doing machine translation evaluation, and there are a whole bunch of them, to be honest. It's sorta become almost a sub-industry of coming up with automatic evaluations for machine translation. And there must be something like a dozen now that different people have proposed. But the original one was BLEU, which was proposed by IBM, and that's the one that I'm gonna present. And the fact of the matter is, this BLEU evaluation metric is still kinda just the reference point for automatic evaluation metrics. What I mean by that is, all of the other eleven, the way you write your paper on evaluation metrics is that you write your paper introducing an evaluation metric and proving that it works better than BLEU. But in the real world, the majority of people the majority of the time still compare BLEU scores, the BLEU's near enough to okay within the space of the evaluation measures that exist. So the idea of BLEU is what it's gonna do is evaluate N-gram precision. So you're gonna have one or more reference translations, and then you're gonna have a machine translations, and you're gonna say that the machine translation is good to the extent that you can find N-gram sequences, which are also present in the source. And so, a little bit this just seems like position independent error rate, that you're looking for stuff here that's also there. And in some sense, you are. But the crucial extra stuff that you're putting in is that you're getting a lot more points if you match subsequences. And so the idea is, if you match subsequences of words with the original, that means that you're kinda capturing whole phrases correctly.

Okay. Then there's the stop people cheating clause, which is the final part of the BLEU measure. So this part of the BLEU measure only measures precision. So it's saying, of the stuff here, which words and word sequences can I find in the reference translation? That there's nothing in the measure that's looking at recall, saying, in the original it says here Osama Bin Laden. Does the translation mention Osama Bin Laden? And that could worry you. And some of the other MT measures do pay attention to recall. But the way it was implemented in the BLEU measure was to simply say, let's put in a brevity penalty. Because if you just had it up to here, you could totally cheap. You just take the input, whatever it is, and say the translation of that is "the." And then, almost certainly, the word "the" would appear in the input and in the reference translation, and you do really well. So the brevity penalty is saying, well, you're gonna be penalized unless your translation is the same length as the reference translation, or you'll be penalized to the extent that it's shorter. So that sorta forces you to put words in, and so therefore, you'll do better if you're putting in words that are found in the reference translation. Okay. And then I give this sorta mixed evaluation here. The initial claims from IBM and some of the initial results were that once you put in the brevity penalty, that the BLEU measure was hard to gain, that is, that increasing – if you do things that improve your BLEU score, that those genuinely are improving the performance of your machine translation system, as evaluated by humans. And if I sorta skip ahead and show that there were initial results, such as these ones, where here a human judgments of translation, which were separately done for adequacy and fluency. And here are BLEU scores. And look, you're getting a beautiful linear relationship right there. So that was the early claim.

At this point I think I have to have a big caveat to that, because in the last several years, people have started getting cleverer and cleverer at using high tech discriminative machine learning methods of various kinds, whose only job in life is to optimize BLEU scores. And what people have been increasingly finding is that the correlation between BLEU score increases and human evaluation of translation quality has been breaking down to the extent that for the kind of increases that people are now getting in BLEU scores, that they're not really correlated at all with improvements in translation fluency. So it sorta seems to have really reached the point where people have to be coming up with something that is a better automatic evaluation metric than the BLEU score. But nevertheless, the BLEU Metric, simply because it did give this automated metric, was very effective at driving forward machine translation for a number of years. Okay. So what you're actually doing for the BLEU score is that you're calculating a weighted geometric mean of your N-gram precision up to a certain M. So the most common measure that's used is the BLEU 4 measure, which means you're considering up to four grams in the measure, and you're putting a weighting here, where you've got a kind of a lower weighting on four grams and higher weighting on unigrams. And there's one very weird thing about this metric that you want to be aware of, which is that in a geometric mean, if you're translation has no 4-grams that match the reference translation, you're gonna have a zero term there, and when you multiply the things together, you get zero. So the BLEU Metric falls apart, unless you have a non-zero score for each of the N-gram measures.

I mean, one way of dealing that for just comparing your own systems is to say, okay, if I can't do BLEU 4 I can just back off to BLEU 3, or something like that. But that sort of seems a weird to bad feature of the metric. But that's how it is. In particular, a problem that that raises for the metric is you kind of can't use BLEU scores to evaluate individual sentences, because at the individual sentence level, you might have a 4-gram overlap, and you'll do really well, or you'll commonly not have a 4-gram overlap, and your BLEU score will be zero. And it's sorta not really assessing the quality of the individual sentence, it's just kind of random stuff. So BLEU scores are really only sensible when done over the entire corpus of the translation, and so therefore, the whole test set. And therefore, you can't really sensibility say which are my best and worst sentences based on BLEU scores. Yes?

Student:

[Inaudible].

Instructor (Christopher Manning):I think the intuition is the following: you do want to give people credit for matching long N-grams, but at the end of the day, there's a lot of luck in whether you match long N-grams, because this one matched "airport" and "it's", even though actually it messed up and it put "the" there, which is ungrammatical. Now, the system could equally translated it as "airport" and "the it's office," and to a human being, that seems kind of equally as bad, because in both cases it's put in a "the" that shouldn't be there, given that the "it's" is there and it's only a question of which side of the "it's" it puts it on. But if it had put the "the" on this side, it would have only gotten a bigram match, whereas because it put the "the" on this side, it got lucky and got a trigram match. So, although you want to pay attention to higher-order N-grams, it's kind of – it's quite noisy and there's quite a bit of luck in whether you get them right. So, I mean, it seems like for content words that should be translated, that the word, "Guam" should appear here, and if the word, "Guam" does appear here, that's bad. But that seems a more fundamental fact, and that encourages you to weight the unigrams higher.

Student:[Inaudible].

Instructor (Christopher Manning):Yes, you are right. Yeah, so it's kind of a compromise. I guess the bigrams still have a decent weight.

Student:[Inaudible].

Instructor (Christopher Manning):I mean, it doesn't have to be huge, but typically what's used is the order of 500, 1,000 sentences. Enough that you could expect that – enough that you won't be getting a zero 4-gram overlap count.

Student:[Inaudible] displacement penalty, because once you have [inaudible] there could be a matching word from the last sentence that really matches –

Instructor (Christopher Manning):Oh, sorry, sorry. I should explain that. Yeah, it's not that the overlap – no, sorry. That was something I appreciate having to explain. The overlaps – the counts of the overlaps are calculated her sentence. So you're taking here's a sentence and here's how it was translated. And you're just looking within that sentence for N-gram overlaps, that you can't get scores for ones that belong to different sentences of translations. Yeah, sorry. Okay. So you can run BLEU just like that by having one reference translation and one machine translation and counting up the N-gram overlaps, and people do. And so this is just one more slide that sort of shows possible outputs and sort of shows where there are N-gram matches of different sizes and gives some idea as to sort of how you do or don't get lucky, in terms of bigger N-grams matching. But the idea after that is, in the original version of BLEU, and still quite commonly in evaluations, people don't just use one reference translation. They use multiple reference translations. Having four is actually quite common. And the idea here is, effectively, if you have four reference translations, you've sort of sampled the space of possible translations, so that if a system translated something in a reasonable way that a human being might translate it, you've kind of got four chances for it to show up in reference translations, and then you're more likely to get higher-order N-gram matches. And so people have used that idea of having multiple reference translations. On the other hand, if you think about it statistically, you could kind of think to yourself, well, that shouldn't matter. I could just have one reference translation, and it's effectively sampling.

It'll still be the case that if you translate things well, more often you'll match the reference translation. That seems to be true, too. So sometimes people use multiple reference translations; sometimes they just use one. And you kind of really only have problems with using one reference translation if somehow there's a correlation between the style of a system and the style of one particular translator, which wouldn't be found with other translators. At this point, here's also one of the defects of the BLEU measure, that BLEU scores aren't absolute scores that you can just interpret like an accuracy score, because how high a BLEU score is crucially depends on how many reference translations there are. So that if you have four translations, you can quite commonly see BLEU scores in the 30s, 40s, 50s. Those are sort of pseudo-percentages, 0.3, 0.4, 0.5. Whereas, if you only have a single reference translation, BLEU scores will commonly be around 0.1. They might even be below 0.1 or 0.15, or something like that. And that's not meaning the system's three times worse. It just means you've only got one reference translation. Okay. So moving on to translation systems. So the huge part that I haven't really said anything about is, boy, how do we actually make a machine translation system? And so, in principle, this is straightforward. We have built an alignment model, probability of F given E. We've built a language model, which gives probability of E. So once we're given a French sentence, all we need to do is find the English sentence, E, that maximizes that product. But all we need to do is actually a very difficult problem.

So doing machine translation decoding, even for the simplest models, is NP hard. And the reason for that is that, simply, because you're able to do these different orderings of words, that simply because you can translate words in any order, that even apart from the choice of how to translate a word, that's already giving you an exponential factor right there, so you end up with an NP hard problem. So what do people do about that? People have explored some different search strategies. So you can actually do a reduction from machine translation to traveling salesman problem, because effectively, the reordering of the sentence is the order in which you visit your cities in the traveling salesman problem. The problem is that you can't actually use that to translate sentences longer than about five words, because it becomes impossibly slow and complex. By and large, the technique that people use, which I'll briefly mention, is using beam search. So this is the standard way that people do search. So you have a sentence that you want to translate. So you have this French sentence and you want to produce an English sentence. So you start over here at the beginning and you start off with an empty hypothesis for an English sentence. So what you're gonna do is want to insert the first English word in your sentence, which will just be the first word of your translation. And you have two sources of knowledge with which you could do that. One source of knowledge is your language model. The language model will know that English sentences commonly start with the word, "the." But that's not very good evidence. And the other source of knowledge you have is what words were in the French sentence. Now of course, the first word in the translation might not be the first word in the French sentence, so what you're gonna be doing is exploring all words in the French sentence, and considering possible translations for them, and then they could be the first word of the English translation. And so that gives you a big number of possible translation hypotheses.

And you prune somewhere as you usually do. And each translation hypothesis will cover some number of words in the French source sentence. And so, the examples here kind of all look like they cover one word, but in the general case, you could also generate an English word and the output that didn't cover any French word, because it's just an extra word that's turning up in the English output, which had fertility zero in the alignment model. So you get a bunch of candidates and the idea of beam search is simply you're gonna keep some good ones, and throw away the rest of them. And so you keep some good ones, and then you expand them by saying, okay, let me put the second English word in the sentence and, using the same kind of sources of information – except your language model is now much better. Because you know what the first word in your English sentence is, you can common do a much better estimate of what would be a good second word, according to the language model. So you generate candidates for a two-word English translation. And for each candidate, again, you record which French words you're covering in the input. And you kind of keep on going along, and you stop when you've covered all French words in the sentence, because every word either has to end up being aligned to some English word, or it's being classed as a spurious word. So you go along until you've completed coverage. And so that's the general idea of the search. And then when you then kind of go backwards through your search tree and come out with the best translation. And so this is a kind of classic beam search algorithm. So the main question, then, is how do you trim these beams to only keep candidates around? Yes?

Student:[Inaudible].

Instructor (Christopher Manning):You generate the English translation – so this is your starting off with the French sentence and you're translating to English. You generate the English translation sequentially as the sentence from left to right. So any candidate is some initial prefix of an English translation, that it can cover an arbitrary subset of the words in the French original. Okay. So if you're doing a beam search, you have to throw away low-ranked candidates. And the question is, how to throw them away. In a lot of algorithms of this sort, you just have some global score for candidates and you throw away the ones that have a low score. And we score by our product of probability of English according to the language model times the probability of French given English, according to the alignment model. But there's a problem in doing that here to get it to work effectively is that any long candidate is worse than a short candidate, because the probabilities keep on going down. And while that might suggest to you other ideas as to, well, I could do some kind of A-star search like idea, where I could consider the stuff that I've yet to do – and ideas like that are possible – but the most common method that's used in practice is that you actually keep separate beams of candidates that cover a different number of French words. So all candidates that cover two French words or all candidates that cover three French words are compared against each other, but candidates aren't compared that cover a different number of words in the source. And that kind of gives you a way of keeping things roughly comparable. Okay. So now I wanna say a bit about sort of some of the more recent stuff that's been done in MT. So this is what we have so far. If you go back to the original IBM paper, Brown et al., they call is the “Fundamental Equation of Machine Translation.” I think you’ll always want to be suspicious in MLP when people call something “the fundamental equation of something.”

But as we know, it’s just Bayes Rule, right. So we’ve got that. Is this really the best thing to use? It turns out that no, you can easily do better than that. It turns out that you can always do better by introducing yourself a hyperparameter, in which you can scale the two halves, simply because, since the different component models – both of the component models have bad independence assumptions in them. But depending on whether one half model has better or even worse independence assumptions, that you’re kind – independence assumptions always make your probability values more extreme than they should be. And so you can effectively attenuate over extreme probability values by putting an exponent in one of the component models. So since the probability of English might be 0.001, when I raise it to the 2.4, that becomes a much smaller number, 0.0000 – whatever. So if I raise the language model to the 2.4th power, does that mean I’m paying more or less attention to the language model in my machine translation? Am I paying more attention to it or less attention to it, in terms of what dominates the score?

Student:

More?

Instructor (Christopher Manning):I got a more. So the answer is more. And the way that you have to think about this is by looking at likelihood ratios between two candidates. So that when I raise a number less than one to a positive power, a power greater than one, the number is getting smaller. But if you think of the ratios, the ratios are actually getting bigger, so I’m paying more attention to it. So I did this before I came to class, so I had the numbers. There are with big probabilities. So if I had probabilities of 0.5 and 0.75, the ratio between them is 1.5. If I raise those two to the 2.4th power, I get 0.19 and 0.5. And so the ratio between them is now 2.6. And so that means I’m now actually paying more attention to differences in language model score. So that’s a little parameter you can put in and improve the quality of your translation output. If you start thinking along these lines, well, maybe I can put in other factors that’ll improve the quality of my translation output. And here’s one that’s commonly used. You put in a factor that is just the length of English translation. And the reason that this seems like it’d be useful is all else being equal, the language model actually biases you against long translations, because all else being equal, the language model score is just higher the shorter a sentence is. Short sentences like, “Hello,” they get good language model scores. Long sentences, like, “In light of current economic indicators, the Federal Reserve Board is going to meet next week and blah, blah, blah,” really low language model scores. Now to some extent, that will be catered to by the alignment model, which will want to have translations for words, but kind of, it turns out, insufficiently. So you kind of end up, people have found, with a bias toward too short sentences. So you can counteract that by putting in just a factor that’s the length of the English sentence, which again, you can weight with a hyperparameter to prefer longer sentences.

And you can do that, and it'll improve the quality of your translation output. So once you've started doing that, you start thinking, oh by, probably there are all sorts of things I can tack on the end and I'll be able to improve the quality of my translation model output. And that's exactly what people have done in recent machine translation work, so that people still have language model scores, people still have alignment model scores, but what people have built is effectively feature-based models of machine translation quality, where you can think of anything else that's useful. Like you can just have features. A kind of a useful feature for evaluating English translations is just to look whether the English translation has a verb in it. If there's no verb in the output, it's probably bad and you're missing something. So I can simply have a "does the sentence have a verb" feature. And I can put a weight on that. And that will help my translation quality a bit. And so you kind of effectively have a feature-based scoring, where you can have an arbitrary set of features, and then people do things like build log-linear models over these features. I'm not gonna get into the details of that later. We'll come back to some of those kind of models in other contexts later. Okay. The second major idea that people have then gone to is getting away from this word-based MT. So all of the IBM models, where at the end of the day it was just probability of French word given English word, context independently – and that was clearly just really bad and didn't seem like it could work very well. And I mean, in particular, if you look at real language, it's just full of all of these what linguists often call collocations, these little sequences of words that go together. So things like "real estate," "interested in." Note that many of these aren't conventional phrases in a linguistic sense, so "interested in" isn't a conventional phrase in the linguistic sense, because "interested in soccer," you'd say that "in soccer" is a prepositional phrase.

This is just the very followed by a preposition. But these are exactly the kind of collocational units that are very useful and commonplace in language. And also, when you get into technical terminology, that that's mainly collocations, so things like "disc drive," and "network router" and all of those things, these kind of standard collocations, and so you'd like to be able to learn those. So that was a big gap in the IBM models. And the second gap, that I already mentioned with the adjective-noun word ordering, is we'd like to know something about syntax so we can do syntax-level transformations. And so for the last few minutes of today, I'll do the phrasal translation part of that. Okay. So the idea of phrase-based statistical MT is that what you're going to do with the input is you're gonna chunk it into phrases, whereby phrases people hear just mean any subsequences of the input. No notion of linguistics whatsoever, just any subsequences of the input. And then what we're going to do is have a probability distribution over phrase translations instead of individual word translations. And so you'd have the probability that this phrase is being translated by "to the conference." And so then there's still gonna be reordering, which will have displacement factors in that, but the reordering will be done in terms of this phrase chunk. So effectively, the model is, in some sense, kind of like before, apart from rather than doing it for each individual word, you're working on these multi-word units, which are called phrases. And that's really the only new idea. But up until the moment, essentially doing these statistically phrase-based models is the state of the art of what people do for machine translation. There's starting to be some evidence for some languages that models that try to do more with real syntax in them are better, but this is still pretty much the state of the art. As it turns out, you can just capture a lot with these kind of multi-word phrases.

Student:[Inaudible].

Instructor (Christopher Manning):I will do that on the next slide, or maybe the one after. So I'm almost there. So once we have these multi-word phrases, note that we actually are not dependent on the language model totally anymore. Because, effectively, these multi-word collocational chunks will include appropriate translations for a word inside their multi-word chunks. So we are getting kind of context-based choice of how to translate individual words. And the phrases that we use aren't of just one size. They're arbitrary size. So what you really do find is if they were just common utterances that keep on occurring, like your – there's actually sort of formulaic prose you see in the newspaper almost every day, so you'd kind of have, "At the close of New York trading, blah, blah, blah." And it'll just learn big phrases like, "at the close of New York trading," and how to translate them. How do you learn the phrases? Here is the answer. And this is why the stuff that we've learned already isn't useless. What you do is you assume that you start with word alignments. How do you get those word alignments? You run Models 1, 2, HMM, 4, etc. Once you've got those word alignments, what you do is you consider possible phrases. So a possible phrase like this one is defined as a place where you get this kind of block structure. So there's nothing down here, nothing across there, nothing there, and nothing there. So that's a possible phrase that could be taken as a chunk. There are also smaller phrases. So this is a possible phrase. It has the same properties. But this single word is no longer a possible phrase, because there are other things outside of it over here. And there are bigger phrases, so I could take all of this up to here, that's a big phrase.

So the possible phrases are things that are defined by this property of having nothing out here, which means that you're kind of capture – for the words, "did not slap," you're capturing all of the words that are aligned with them. And similarly, in reverse, for these Spanish words, you're capturing all of the words that align by them. If things satisfy that property, you have a statistical phrase. Okay. So conventionally, how that is done is you actually run the IBM models twice. This is really the state of the art and basically what everyone does. If you look at it as sort of a funny process and you think how this couldn't possibly be optimal. And there has started to be a bit of research on better ways of doing things. But it really is in terms of the people who built the state of the art MT systems, of which, I guess, really the biggest players in statistical MT systems these days are Google, Microsoft, and then Southern California ISI and their commercial offices Language Weaver, this is really still what people do. You run your IBM models in one direction, English to French, and then you recall that there was this crucial limitation of the IBM models, which is that they allowed n-to-1 alignments in one direction, but not in the other direction. That seems crummy. So what you do is you run the IBM models in the opposite direction.

So that allows you to get 1-to-n in the opposite direction. And so now you have two possible sentence alignments, and you want to merge them into a consensus alignment.

And at this point, people use various heuristics. The obvious first thing is to think of union and intersection. But it turns out that intersections are too sparse and unions are too dense to work well, so by and large, people are using heuristic methods with funny names like rodiag final and things like that, which are doing what seems to be a sensible job of combining alignments from the two. And then this is giving you your final alignment matrix. And then, effectively, you extract off all statistical phrases up to a certain maximum size, and they become your phrase lexicon. So when we go back to here, when I just glibly said foreign input segmented into phrases, the foreign input isn't uniquely segmented into phrases. It could be segmented into phrases lots of ways. But the possible ways that it could be segmented into phrases is determined by what bilingual phrases exist in your statistical phrase lexicon. So you will explore any way of cutting things up that fits inside the statistical phrase lexicon. And effectively, you're doing that in a decoder, just like before, because when you translate, you're starting off with an empty candidate English translation. You're saying, okay, I want to lay down my first statistical phrase. And so the first statistical phrase will be something in your phrase lexicon, which is some number of words in English corresponding to some number of words of the foreign language. And so you're not actually segmenting the foreign language before you begin. You're kind of segmenting the foreign language in the process of running the decoder. Okay. Well, that's where I can get to today, so I'll stop there and be back there, and maybe say a few more minutes on that and then we'll be going on to the next topic of information extraction.

[End of Audio]

Duration: 76 minutes