Okay, so welcome to the second lecture of advanced.  
  
Today we are planning to talk about text representation and modeling and yeah look at first models who try to use an eye technology to infer something out of text.  
  
As told last time, this will be a block of three lectures, two lectures and one practical session.  
  
And you have hopefully seen that we put in Iliers, a course, a survey about what your knowledge about A I, and especially if you have been listening to introduction to A I, thanks a lot for everybody who participated.  
  
It was very informative for us, so we assumed that most of you had or like visited the lecture of introduction to A I, but it seems not to be the case.  
  
This is something yeah, we just happen with your change, the, the, the.  
  
The course layout of of the programme we the the new introduction I have introduced but it's not yet mandatory for everything and like during the change of course ah sometimes ah things are not as as originally planned.  
  
We are trying to adapt our teaching a bit that we don't expect too much prior knowledge and try to explain.  
  
Everything. Of course, it shouldn't be too much overlaid because who has listened to is also allowed to take part in this course.  
  
And but we are, they are always open for feedback, so.  
  
Ah please, if you have the feeling, oh, there's too much, we we assume too much or too less, or there's too much over that. Please always feel free to contact me or any other lecturer or shaolin to to give you your feedback, and we try to yeah.  
  
Address that as good as possible.  
  
What we want to talk today about is this lower left corner here, so this was a picture. I showed you last time about how to build an AI system. What are core components that you will need?  
  
So we need a way of understanding our environment.  
  
And this can be done multi-modal.  
  
There is vision, audio and there is also higher level information and one important thing of course is text. A lot of information that you gather and everybody is gathering is about text and trying to get this knowledge from.  
  
And that is therefore what we are planning to look first in today.  
  
So how can we understand text? We had a bit of a view of the challenges last time already. We had these examples that natural language is ambigue and therefore it's not like straightforward. We cannot directly extract what something is meaning.  
  
And that's why people are researching that for many years and trying to build systems where we don't have to explicitly program all the knowledge in there but get all the information from the text which is out there. And if you think of the web of the internet, of course there is huge amount of data available.  
  
This is part of what is natural language processing. Natural language processing deals with communication with computers and the important thing is we try to communicate in a different way with a computer.  
  
So currently what we are mostly still doing is that we are clicking and programming so that the computer is defining how we communicate.  
  
We are looking into natural language processing. Can we communicate with a computer or an AI in a most natural way as if we are talking to somebody?  
  
And therefore we need to understand and process natural language.  
  
What we are looking today is we want to acquire knowledge from text. You saw that there is a part about generating text.  
  
If you think about tests like automatic summarization about machine translation, it's not only about understanding text but also about generating texts that will then be covered in a later lecture.  
  
And therefore there are these two Yeah picks up groups in natural language processing.  
  
One is like tasks that focus on natural language understanding. So given a text, we try to extract what are these information in there and there's the natural language generation part where it's more about how can we generate.  
  
Although these are now shown separate, there is not always a clear separation and for a lot of application a task might involve bows.  
  
Course. We are looked today, for example, about a sentiment analogous, so finding out what is a sentiment of a text, so is it a positive review. Is it a negative review?  
  
This is clearly natural language understanding. However, if you think about things like summarization you might need to understand the original text and then generate a new text. So in this case you have somehow combined both of them.  
  
So how can we do all of these tasks, and over the years there has been a development?  
  
People started with doing it like rule based systems, so they tried to do text as we do a programming language.  
  
And let's try to understand what they said by writing rules like detecting the verb. What is the word? What is the noun? What is the meaning of the word bank? In this case is the Finance Institute or is it?  
  
At the river bank you have to look around the surrounding woods and so on.  
  
So this was a lot about yet defining rules and then there were like experts, computational linguistics and linguistics who have a theory about how.  
  
Sentences are built, I guess you know, that a bit from school still a correct sentence has to be a noun or a verb, and then there might be some object.  
  
And then of course there are more complex structures, but there is a whole theory about how to do that, and now you can try to use this theory in order to process some type of.  
  
The challenge is it's not that clear. Follow these rules. They are always changing. New words are coming up.  
  
So the problem with like these rule based systems is normally they are fixed.  
  
It's a lot of work, a lot of expertise is needed.  
  
And therefore people looked into, can we do it differently?  
  
And that led to corpus based approaches where it's no longer that we have expert writing rules of doing it, but we are trying to use machine learning and a I to automatically learn how to do the task so.  
  
Instead of having experts, we're having input and output data. So we have some input. We have the reference input output for supervised learning. For unsupervised learning, maybe other things.  
  
And then we are giving this data to some machine learning model and this model.  
  
Traditionally that was done with statistical models where it's a lot about representing the text in the correct way in a way that it's easy for the task to do its work.  
  
So you try to learn a representation of text and then it might be very easy to solve the task.  
  
And that was what was done in statistical for a long time.  
  
Since around ten years then people are using neural models as we have shortly reviewed neural networks on Monday so where you use a neural network.  
  
You still have the challenge of how do you represent text in the new old network. We'll come to that on next Monday.  
  
And then you can use these neural networks to do like train and do this task, but.  
  
In the end, it's very similar where bows are referred to as corpus based, so it's mainly about using some type of A-I to do this task and predict.  
  
And what are these types of tasks today? We want to look first at one of these tasks, which is sentence classification.  
  
They're like, of course, they are very specific tests.  
  
Is like sentiment analyst, but you can group a lot of the tasks in NLP into three groups. There are some exceptions, but these are the most common one.  
  
There will be sentence classification where it's here where we have an input, a text. A text is a sequence of a variable length so it's not always same lengths and that's already challenging for a lot of machine learning.  
  
And then you want to output some type of label.  
  
So you want to do, for example, sentiment. You have this review about the movie and is that a positive review or a negative review.  
  
So that you can ultimately aggregate them and have things like in all these forums or so that you see how most people are positive or so. Of course they can do it by hand but can you do this?  
  
So sentiment analysis is one of these types of tasks where you have an input text of variable.  
  
Does anybody have ideas? What other interesting task of these types could be any ideas?  
  
A very good example of spam filters. Also you have a text like email and you want to determine whether.  
  
Similar task or topic where you want to determine the topic or classify it in news and domain. There is task on authorship so you want to know who has written that.  
  
So there's a lot of these types of town:  
  
The others we will look later on Monday where it's not one label but every word gets a label.  
  
And then we have the sequence generation where we want to generate a variable sequence. For example, machine translation that is automatic summarization and so on. So everywhere where you have an input sentence and.  
  
So how can we do this? We don't want to look into rule based systems, but today we'll look a bit into the traditional corpus based statistical model. The thing is the input, which is our text, and then we need to do what is referred to as feature extraction.  
  
So we need to extract some of our representation that we can do machine learning on top of that.  
  
And if you look at all the different types of classifiers, this would be a classifier.  
  
Can have a scale, then it's to, but that is.  
  
Typically a statistical classifier like that, he needs to have some input vector, so it should be a vector of fixed size.  
  
Where you describe your input as okay this describes the input and then you can use as we said neural networks. Last time you can use naive base classifiers you can use.  
  
A linear classifier's kernel method, so there's a lot of things you can do then on top of that. But what is important is that you get some of these types of representations.  
  
And of course this representation should be helpful. So if all articles get the same representation you cannot classify them somehow.  
  
So the important task is to get some of your representation representing the text as some type of vector so that it's hopefully.  
  
So we need to find this fixed type of representation because normally what we are using this neural networker.  
  
They get us input a fixed size vector like some vector like R to the power of N, and they output also a vector like one dimensional vector if it's only one class, but it can be also more depending on.  
  
So if we are doing that, of course, we are losing information typically because we need to do some approximation.  
  
And these things are difficult because we have an open vocabulary last time we talked about words like Brexit and so on. So you can never build a model which has the whole vocabulary which has ever been spoken in there.  
  
Because people always invent new words and we have to somehow deal with that,.  
  
Just think about things like a Deutschland ticket which was now used in Germany, but I guess nobody knew one year ago what it.  
  
Then these text here which we had is typically variable lengths. You don't want to be able to only get inputs of like twenty words but sometimes the sentence has ten words, sometimes a sentence has one hundred words and sometimes it has five words and you should be able to deal with all of them.  
  
So it should also be like able to deal with variable length input.  
  
And the order is also important, so you cannot know only which words are in there because we'll see that later there's a big difference between Mary and John.  
  
And so if we would just ignore that, that might not be that good.  
  
However, you will see that is an interesting thing and that's quite important. Sometimes we know we lose information but we can still do it and perform it. So maybe for some task word order might not be that.  
  
But you should still always keep in mind that you're losing information if you ignore the word.  
  
So how can we do that? That is like what we people are referring to as feature engineering.  
  
So people were like in for the last twenty-five years or so. A lot of research was done.  
  
What is the best way in order to represent text so what people were doing is they tested different text representations then on top of the trained classifiers and see if I changed the text representation.  
  
Do get a better representation? Do get a better result?  
  
Normally, of course, it's very difficult to directly evaluate how good your representation is by using it in a classifier.  
  
What will come then and what is a big advantage of deep learning approaches is that often we don't have to do this manual work of creating features.  
  
So we are having very simple ways of inputting the text into the neural network and let the neural network.  
  
But yeah, this type of feature engineering was very, very important and it gives us still a good understanding and it's important to like understand a bit of how we can represent things.  
  
So if you do it by hand you have to do a lot of work and it's often very task dependent. So it can be that one type of representation is very helpful for spam detection but might not be good for sentiment.  
  
Or like just think about words like great and brilliant are maybe important for doing sentiment analysis, but might not be useful if you do spam detection.  
  
So what could be a way of representing the text? Anybody has any initial ideas of how to do.  
  
The first the most simple way, so we always start with the most simple way.  
  
That is exactly what is a standard thing. It has even a name. It's called a bag of words representation and we can use accounts and even ignore the counts. So then it's more about which words do occur and which words don't occur.  
  
So if we're talking about back of words representation, we have something like a text here, and then we're counting how often these words are occurring.  
  
And that is some of our representation.  
  
We have then somehow a fixed vocabulary, which says okay. We have maybe words in that.  
  
So ten thousand different words, not ten thousand tokens, because the word occurs several times. It's just a different count and that's our fixed representation. So we are representing our text and this type of vector where we are just saying each word how often does that occur.  
  
This is, of course, absolutely not correct, and Mary kills John and John Cales. Mary is exactly represented the same way here we are ignoring any word order.  
  
However, it might be ended. It was shown that in some for many applications,.  
  
It's working fine and then it's a good representation because it's easy. It's fast to do like.  
  
Where all traditional search algorithms typically work like that, they represent the document in a vector space, compare it, and.  
  
That is, a very traditional search in some other standards basedly.  
  
Though when we are doing this bag of representation you have to use a fixed vocabulary.  
  
That's even a further thing where we do approximation. So if we then have new or unknown words in training, we will just throw them away and ignore them. So they're not helping us for doing the classification.  
  
However, although it's a very rough estimation, it's giving us one thing we can work on and based on.  
  
We can have binary features or count features. The nice thing is now we no longer have a variable.  
  
So this vector has always the same length, but in most typical classifiers some type of classification.  
  
And this is done so. We are going from variable lengths to fixed lengths by just ignoring all the word.  
  
And yet this is the first basic idea of how we're doing that. We'll go now a bit more into detail how to do it.  
  
The question now is how do we define the vocabulary we are doing? Typically we might not even use all words, but.  
  
And on the other hand, of course, depends on the task that we're doing. As I said before, for sentiment, another is great. It might be a very helpful thing, but it might not be helpful for spam detection. There's other phrases which might be their helpful.  
  
But in generally there are some rules which are typically done. If we are doing back of words representation, stop words.  
  
So stuffed words are the words which occur the most often, so it's like he, she, it or so.  
  
That's normally not really helpful because they anyway occur in all of your documents and if the feature is always on, it's not helping you to distinguish. So if this feature is just always one, it's not helped you and you can just leave it out.  
  
That's why typically if you do this type of representations you might throw away all your stop words because yeah you're not getting any gain from that. It's just telling you this word is in there but you anyway know that this word is in all of your sentences so it doesn't help you to disambigate different.  
  
Another thing is unique words. Words that only occur once are also not really helpful because it's more random if they're in it or not.  
  
So if the word culture is mentioned one and there was a positive review then you will learn oh if culture is in there I should always make it positive but this is not really helpful because it was just random that the person was happier about culture and not another person.  
  
So in order to prevent overfitting, one technique is to ignore words which occur less than a minimum account of occurrences. So at least the unique words may be also words which occur less than two or three.  
  
This is again motivated machine learning. We are good at machine learning. If we have seen things very often then we can learn something and that of course is here very challenging because a lot of words occur only very rarely.  
  
In total, we can based on that have like two types of back up words representations:  
  
We have the binary unigram features where it's like you have one feature, one feature, one entry in your space for each of the words.  
  
Then, of course, it's always important to know the features in machine learning.  
  
You need more data if you have more features because you also have more parameters. Typically if the number of features is large this will get more difficult and here then the number of features is exactly the number of your.  
  
So if represent the text with words, then your feature space is words as a input.  
  
If your thing goes, I'll use a vocabulary of words. You have features which occur less often.  
  
And the values in your vector in the binary case are only zero and one.  
  
While you can have count features where you really count how often each word occurs,.  
  
And then it's value n, and of course it might be more helpful in some cases because if the word occurs great occurs once it might be an indication, but if it occurs like ten times it might be more informative. This seems to be really a great movie if he's always talking about great.  
  
Importantly he will not say great great great but it's positional independence so you'll say the movie was great and the cast was great and the music was great and so on and then he hopefully has a lot of different grades in there.  
  
One problem we have now is that we ignore the the position of of the words, so we are just counting how often there is great, and it's not about. We had the example beforehand with the with the film.  
  
And there it was a great plot, but it was also. The book was great, and so on.  
  
So maybe if he's talking about a great book, but it's a movie that might not be that helpful because the original movie was great.  
  
So it might be helpful to not only know single positions but to get a bit of word ordering in there again.  
  
And you can do one trick by not only having features for this word occurs, but this pair of.  
  
So if there's other thing about like maybe it's not as informative that there's New and York in there, but if you know New York is in there you have a lot more information because then you know the city.  
  
Or if you want to do a person in there, it might be helped to have both names together and each of them.  
  
In this case it might be an advantage of also introducing features like this. This biogram features though people call it biogram because it's like a gram is always a sequence of words. We come to that later and buy us because we have two words here.  
  
So we have then the number of sequences somehow in there by having this type of feature.  
  
Though we are considered the advantage, we have a fixed size representation and we still consider somehow the word order. Mary kills John and John kills Mary because in the one we have Mary kills and in the other we have John.  
  
Of course you can now construct a lot of sentences. It's not perfect you can construct sentences which have a complete different meaning but which have exactly the same representation because you only have this window of two words where you're always looking at and you cannot get that.  
  
However, of course, it always comes with a disadvantage. The disadvantage in this case is we are increasing the number.  
  
But we no longer have a number of features in here, but a number of bigram lengths. So if you have a quadratic.  
  
This explodes fast so you can't do that too much. It's a way of introducing the ability to represent sequences of words.  
  
Okay, before we are now going to the challenges of why that might still have been,.  
  
It's a very common approach and if you have a standard classifier and you have this text classification problem that's often a good way of starting to see how difficult the challenge is, if this model already gives a good performance it might not be worse to put more effort in it.  
  
However, there is also challenges before we come to this. Do you have any more questions?  
  
Why is this not always perfect? Here are some examples. Just imagine you only have the feature great, so you only have one feature, one vocabulary you only doing sentiment on. Does the word great occur or does the word great not occur?  
  
It's not a perfect way but to get a bit of an understanding and there is this Stanford Sentimentary Bank where there is a large data set where you have different reviews and then have the sentiment.  
  
The one problem, of course, is this type of ambiguity which we saw before. If you count only the sentences where great occurs, you would get an accuracy of:  
  
So if you look at all the sentences with great, of them are positive and are negative.  
  
So it's great is not always referring to positive review. It might be that there are negative reviews as we saw in the beginning, which still the word great occurs because it's used for anything.  
  
And so these type of representations are in big because there is not clear if great really always.  
  
The second thing is the variability. You can express a positive segment in a lot of ways:  
  
So there were like nearly examples in this review, but only of them had even the word great.  
  
So if you would concentrate on this type of feature you would only be able to classify:  
  
So there's a lot of ways how you can express that you're thinking positive of something.  
  
So you really need to include a lot of features in that this again says you. Maybe it's not good to do that by hand, but to really do machine learning.  
  
Of course, usually um it.  
  
Great indicates that it's positive, but there are several ways when this does not work. So there are several possibilities where you can express that something is great.  
  
And it's not really great, so we've seen some of them. There are some more.  
  
So, for example, it can be of course negation. Negation in text is always a big challenge.  
  
It's often ignored by standard models because it doesn't occur too often, but it's difficult.  
  
So it's not a great monster movie, then of course if you have a bag of word representation.  
  
On the other hand, it's not so easy to change everything with a knot in there because it's also not clear what is changed by the knot.  
  
The knot can be in another sentence that can be referring to something else:  
  
So dealing with negations is typically a very challenging situation in text, and negation can be done others than not.  
  
There's different senses of great like a great deal.  
  
And you can have multiple sentiments as we are in the initial thing where we say first which things are all great but then that the all work together is not great. So if you don't have one single sentiment but very different sentiments in there that is of course also.  
  
Very challenging and will not make it possible to do this type of thing.  
  
Yeah, in generally when we are talking about this type of feature engineering,.  
  
You always have to find the good way. On the one hand you shouldn't have too few features. What happens is you cannot disambiguate and your performance.  
  
Because if you only use one feature grade in cases out of your cases,.  
  
The representation all looks the same, it all says the word grade does not occur, and so you cannot disambigate of your data or even more.  
  
So if you have too few features you're more.  
  
It doesn't really matter what powerful machine learning model you have. If all the examples are represented the same way, the model cannot do anything about it because if the input is the same, the output will also be the same.  
  
On the other hand, you have to be careful because this might lead to overfitting if you have features which occur only once.  
  
They seem to be very good because if you trust them you can perfectly classify them.  
  
Now we only look into which features to use. The second question is how much importance to trust more.  
  
And there is different ways of how you can model the feature of importance.  
  
The one thing is you can do it in a supervised way. If you think about the neural networks, we take our input features. We multiply them with the weights in green and the sum based on what we are doing the classification.  
  
And of course we can learn these weights in a supervised way to do the best classification.  
  
And that is something which is, of course, always done. However, sometimes it might also be helpful to have also more expressive features. So not only saying this word occurs or this word doesn't occur, but giving some more information how important is this word for the current input.  
  
That can be, for example, done by count based methods. So if a word doesn't occur once but occurs ten times, of course it seems to be more important for the text. If the word occurs one hundred times it might be even more important so you can do count based features.  
  
And there is another term which is especially used:  
  
If you use this type of feature representation in more unsupervised cases like search,.  
  
Where you just compare and find the most similar way, and that is called the term frequency inverse document frequency representation.  
  
And that representation we want to look now as a bit of an extension. So if a word occurs many times in the document, it should be more important.  
  
And therefore it might make sense to not only count it as one, but how often does a word occur in the.  
  
There is a variation of that saying that yeah, it's more important than more often occurs, but the difference between a word that occurs once and ten times is bigger than the difference between a word.  
  
So the larger your absolute number gets, the less important the difference between.  
  
And that's why you can then use the logarism in there, so you're saying okay for large numbers. The difference isn't as important anymore, and instead of taking the direct count, we're taking the logarism of it.  
  
Why Do We Do the Plus One Year?  
  
Yes, so otherwise for words that don't occur this would get us into problems. In this case everything.  
  
And that is the one part of this type, and then as you can see in the title there seems to be another.  
  
And that of course depends a bit on your task.  
  
But the general idea, especially if you think about information retrieval, so searching things words that occur in less documents are typically more useful to do some type of clustering topic clustering. So it's again the thing with the stopword.  
  
If the word he and she and it occurs that doesn't give you that much information,.  
  
But if you have a word which occurs only very rarely, but it's occurring often in this one, it might be a very helpful thing to disambigrade something.  
  
And therefore you have this inverse document frequency where you check how many documents or how many examples did this word occur.  
  
How are you doing it? You're taking the number of documents and dividing it by how often this word occurs.  
  
Now what is the example if you have the most probable?  
  
Will be logarithm out of divided by and that is zero, so this is not very helpful for a lot of tasks that.  
  
If you have an A I for language technology to our institute that will not occur in that many documents maybe it occurs only in one and so you have here a logarithm of N which will be then a relatively large value.  
  
So this is a possibility of introducing into your representation a bias that some of these words are more important and other words are less important.  
  
And as I said this type of text representations where you represent a text as a vector, it's not only used for classification, it can also be used for information retrieval. So think about you want to search the most similar document giving a quarry.  
  
You take your query, you take all the other documents, represent each one as a vector, and then.  
  
Yes, so and this is about one way of doing text modeling.  
  
That is like representing a text using the bag of words representation where you can then represent.  
  
With an open vocabulary text with a fixed sized vector, which hopefully contains still a lot of the meaning and that you can use them for different things, you can use it for similarity measures. You can use it for classification or or any other machine learning problems.  
  
The big advantage that it is very simple calculation doesn't take along so you can generate this type of representation.  
  
The disadvantage are like some of them are that, of course, it's still ignoring word order.  
  
So a lot of sentences or meaning is like important. The phrases have a complete different meaning if you have.  
  
The other problem is that typically this leads to many, many features, especially a lot of features.  
  
That is why this is sometimes called a sparse representation because you have this vector of.  
  
Dimensions like one for each vocabulary, and if you think of a text, one sentence or one document maybe or fifty different words occurs.  
  
Which is then a challenge for some machine learning problems. Because there are a lot of features you only see in one or two examples or five examples and yeah you have to address that when training a model in order to not overfit on these types of.  
  
Therefore, in the next part we then want to look at another way of modeling of text. We try to model the text and we want to know how probable this text.  
  
What is the probability of this text which is used in a lot of different scenarios?  
  
But before we come to this are the first questions about this vector-based representation:  
  
Hey hope, that means that it was clear and not where done with this.  
  
So one way or the most common way of representing the probability of the text and a very important concept in all these types of text processing is our language model.  
  
Language models traditionally always are a function which gets input a sentence, and they try to model the probability that this sentence was produced by a native speaker of that language.  
  
So, for example, if you have the house is small that should be a quite high probability that is a correct sentence.  
  
And this does model different things. On the one hand, the structure should be correct, so.  
  
Know how a sentence can be built. We have that it should start with a subject and then a verb and so on, but it's not only about syntax, it also does some type of semantic. So if you look at a language model and what is the probability of the airplane flies,.  
  
It should be a higher probability than the airplane drives.  
  
Or the airplane walks because it has seen and we'll see how we can model. The one is a typical sentence as a native speaker, but I hope few people will say the airplane.  
  
And so these language models are a way of modeling this type of probability of accent.  
  
You might now be a bit surprised if you heard about new large language models, but not today, but will come that they are basically all trained on this type of basic idea. At the end then they first tried to.  
  
Yeah model, what is text, what is the probability of the text, and if you look at new large language models like g p t four or something like that.  
  
They are still modeling and trained in order to model this, but by doing that we can develop very powerful tools.  
  
So the first crash is how can we model this probability? So how can you estimate the probability of the house?  
  
Do you have any ideas from prior knowledge? How would you build a model where you can input a sentence?  
  
Any Ideas of How to Do That.  
  
Very simple statistics. If you have a dies that you calculate, that you get a, or would you calculate?  
  
Yes, that would be the first approach if you want to know if it dies.  
  
The first approach might be exactly the same.  
  
You look at a large database. You look at, for example, good morning. It occurs times you have there around.  
  
Are very easy. We don't need any complex models to estimate. We just look at our large corpus and calculate.  
  
That is first a very good view. We don't have to build any large deep neural networks or so, so it seems to be very easy. However, there might be some problems with this.  
  
Has anybody of you also an idea why people might not do it this way?  
  
Yes, every sentence has a low probability, but that is even so with language models. Each of them has a low probability.  
  
So that is something which is true, but we can deal with that if you compare low probabilities you can still compare.  
  
Yes, that is also true. I mean, we don't know what it really means. If the similarity is that we don't directly measure similarity or so, so this will only give us first. It's like directly the probability how probable the sentence is.  
  
But that's still yeah, it's mean that's only a first step maybe here.  
  
It's going exactly in the right direction. The problem is, you not only need a lot of data, will you ever have enough data to express that there is not enough data, there will always be new sentences.  
  
So for a lot of sentences you will never have seen them.  
  
All of you can say a sentence which all the others have never heard. You can just make up some crazy sentence.  
  
And I'm quite sure nobody else here in the room will ever have heard it before, but it's still a valid sentence. It's still structurally correct, but it's a sentence that nobody ever has said.  
  
And you're still able to deal with that.  
  
If you are doing this and not taking a sentence as your basic unit, you cannot estimate the probability.  
  
Somehow we cannot work on these large blocks.  
  
The problem is we have these sparse data that we only have seen a finite number of sentences. However, there is like nearly unlimited number of sentences that you can produce and.  
  
Therefore yeah we might.  
  
Then of course give a zero score to all of these sentences. However, that is of course not correct because these sentences are perfect English sentences that make sense that are correct. But yeah, we have never seen them so there would not be a good model because it yeah wouldn't display is this a sentence that you can say or not.  
  
So what can we do instead? If we can't do it directly for the whole sentence, maybe it makes sense to split the sentence.  
  
And therefore we can use basic probability and chain rule so we can split up the sentence into its.  
  
So hopefully you can somehow remember that the conditional probability P of A and B given P for A, and this can be rewritten that the joint probability of A and B is a probability of A times the probability of B given.  
  
And this is an easy case where you have two random variables that you should hopefully all know from some basic probability theory.  
  
So if you have now four variables, the probability of A, B, C and D is the probability of A times the probability of P given A and B times the probability of D given.  
  
And so we can do this chain rule in general. So the probability of our sequence of the words x one to x n is the purability of the first word times the purability of the second word given the first word, and so on till the purability of the last word given all the previous.  
  
Is more the mathematical formulation. You can, of course, all to do that really apply now to.  
  
A real sentence. Then we have the probability of its water is so transparent as the probability of its.  
  
So now we are always estimating words like probability of single words. The words are seeing more often than sentences.  
  
So let's at the beginning now assume that we have seen each word at least once will come to the problem of words which we haven't seen later. So first assume, I think we can agree that it's more probable that we've seen most of the words than that we have seen most of.  
  
But did we now solve our problem? Can we now estimate probabilities on our corpus as we did before and then use the model?  
  
Is that so did we now address a challenge that a lot of these sentences have not seen, and therefore we cannot estimate the probability directive? So if you think about again now you can estimate the probabilities again. P phone it is like how often it's occurred through how many words occur.  
  
How often does it occur divided by? How often does it occur so you can estimate?  
  
Exactly that is a problem. If we want to estimate this last probability its water was so transparent, we would have seen the whole sentence.  
  
We haven't really gained anything because we can estimate the first probabilities here, but we cannot estimate the last probability.  
  
Because if we haven't seen the whole sentences, we cannot estimate the last probability. Because yeah, we wouldn't know how probable is the word transparent given that previously there was its water is so.  
  
That's maybe a bit disappointing at the beginning. However, there is a way of doing it. So the problem is yeah, this probability cannot be really estimated for long history.  
  
As said, they have to be on the one. Most of them will not be there. On the other hand, there will be too many.  
  
So if you think of a vocabulary of sixty-four thousand words and a length of twenty-five words, you have more than ten to the power of possible combinations of N grams, and you cannot even store them.  
  
So this doesn't really work.  
  
And then the idea is, but there's luckily a way of dealing with this, and the main idea is if we haven't seen all these histories, so we'll refer to the right side of the condition of probability as history.  
  
So we have here the word given its history and its history is to minus one.  
  
And maybe if we can't estimate that well what we can do is somehow cluster all its histories, so we are taking all these histories and not taking all of the information again, but looking which histories are maybe similar, and then just treating all these histories as the same if they are similar.  
  
And then we would be able to estimate that because we are no longer estimating it for each history, but.  
  
Does anybody have an idea how we could classify or classify how we could cluster these histories?  
  
It will come to that. That's exactly what people are doing.  
  
You are right. It might not be the smartest idea. On the other hand, it's an idea which worked for many, many, many years and surprisingly good. We'll see both. Of course, it's not correct again as we had before. It's the same approximation as we are doing if we're ignoring the word position.  
  
If we are only taking the last three words, it's not correct. It might be that the purability is changed by some other part of the way. However, it's a very easy solution and it's working very well, so you're ignoring all the initial history.  
  
The motivation is that the last words or the last words is the most important.  
  
And therefore you can do this type of cluster.  
  
That is how we are doing it in statistics. We also come to neural language models as you know about large language.  
  
There is a difference that this type of clustering is done by the neural network and it's learning an internal representation which then works even better. But this type of clustering just based on the history was very successful and they are referred to.  
  
You remember this biogram, so they refer to it as N gram.  
  
And what they mainly use is the so-called mic of assumption. The decision is only based on the.  
  
So you can have the extreme one, the first one which is referred to as a bigram language model.  
  
May be surprisingly while looking at one word called biogram, but this is called biogram because you look at.  
  
Or you can have a trigram and so on, so what you're clustering is you cluster all the words to and to be represented by and.  
  
So the whole sequence is only represented by the last two words.  
  
And now you have a big advantage. Now you don't look at the whole sentence anymore, but you always look only at pairs of two words and model that. And there is a chance that you have seen them somewhere is a lot higher than the probability that you have really seen the whole.  
  
So Um.  
  
Yeah, we are doing that then, of course, in each component. So the probability of the whole sequence is no longer this initial probability, but we are in each of the factors. We are replacing it by only looking at the last n minus one words in order to:  
  
We can, of course, do the most extreme one? All history is ignored, so we are just directly modeling the probability of the.  
  
So we are saying the probability of the sequence is just the product of the probability of each of each words. So there you're ignoring the full history that is referred to as a unique RAM model, of course the most extreme case. But it still might be helpful, so in this way we don't have any positional information.  
  
In order to understand what is happening, it might be interesting. What does sampling mean?  
  
So we are taking now randomly a first word based on the probability distribution. So if a word has a probability of two we'll take it of zero point two. We'll take it with zero point two. If it has a probability of zero point five we're taking it with.  
  
And that we are doing repetitively,.  
  
And this is a text with a generator with a unigram model.  
  
And you're seeing directly the probalty of a unigram model, so the text is not really readable. It's really happening what you expect, words which also occur frequently here.  
  
And all these words occur very often in this year, so it's really just outputting their unigram frequency.  
  
And of course there is no sequence in there because it's ignoring, and the probability of the second word is independent of what is standing before there.  
  
Then we can do that and now go for a A bigram model. So now yeah yeah.  
  
That's the difference between selecting always the most probable word and sampling.  
  
So in sampling what you're doing is just assume you have only two possible words, and the one per word has the probability of and the other has the probability of.  
  
Then doing in sampling is you draw a random variable. If the value is lower than it will be, if it's higher than it will be.  
  
And thereby most of the time you will have here, but every 10's word in average will also be.  
  
And here it's done similarly. You're not always taking the most probable one because then you'll just have and it's not helpful but you're sampling choosing but not with a uniform distribution.  
  
So the next thing is that you can do a biogram model where the probability of the word depends on the previous word.  
  
And here you already see more like text. It's still not readable, but suddenly some makes sense.  
  
So you have in this that somehow is now something where there is some type of text in.  
  
Very interesting also here in the second paragraph you have new car after car that the probability of parking is very high makes some sense then the interesting thing is after parking that there comes lots.  
  
Is also not surprisingly because the probability of a parking lot might be very high.  
  
And now something has, now you're only looking at lots.  
  
And of course, the more common use of lot is not the parking lot but the lot of something.  
  
So he's then saying the next thing is lot off. Of course parking lot off doesn't make sense, but it doesn't have this long of a context.  
  
Only does the probability of lot given parking, which might be a very high probability, and then the probability after lot will mostly be off because this phrase lot of is a very common phrase in.  
  
So here you see it's really only having this very short span of attention and only looking at the previous.  
  
And then of course you can extend it to more and more and more formula if you have a small example.  
  
And then if you would take the probability of would like to swim, it would be the probability of.  
  
Or give him sentence beginning sentence beginning of wood, give an eye of lie, give an eye words of two, give him wood like of swim, wouldn't like two, and sometimes you have an extra phase at the end to model the sentence, and then you could have something of sentence end given to swim.  
  
So this is then the formal definition of an N gram language model where the N says like how long context do you consider, so this would be a trigram language model. We have seen Bigram and Unigram. Of course you can extend that to more and more.  
  
Typically that is extended up to a large corporal, maybe four or five grams, and then somehow it's getting difficult.  
  
So if you're going on more than grams, this is typically not really done anymore because just too many of your end grams.  
  
Of course, this is somehow not enough to really model long distance dependency.  
  
If you have something like the computer, which I just put into the machine room on the 5th floor crashed, it cannot model the dependencies between computer and crashed, and you can build these sentences:  
  
Especially in German you can very nicely build that where if you have the verb split into two parts you harbor mesh on the conference.  
  
Then this upgraded is like depending on the previous thing, so this long context of course you cannot really model, but the good solution is:  
  
Or the good situation is luckily most information is always narrow, so mostly in the directing neighboring.  
  
So these are also the most important information. This is modeled well, but yea all these long-range dependencies.  
  
And therefore we can often get away with this type of engram language models and model of the language.  
  
But this is now how we yeah, how we use it. Of course, one question is how can we train it?  
  
And the training is really what they refer to as maximum likelihood estimation. That means we want to train our parameters in a way that they maximize the probability of the training.  
  
That's a general approach. Good model is one which gives a higher probability to your training data because:  
  
And the nice thing is in machine for engram language model. That's a quite convenient definition. You're doing that exactly when you take the probability as how often do the words w i minus one and w i curricer.  
  
So how often do they occur together divided by? How often does the previous work? How often does the history?  
  
So if you do this type of estimation, most of you would have done it anyway. If you're doing it this way, this is the standard way of doing that.  
  
We won't talk about that if you really want to use language models.  
  
The main issue is dealing. Still, there's a lot of cases where engrams didn't occur and there's therefore a lot of work in how you can deal with the situation and how you can apply smoothing so that you can also estimate probabilities for.  
  
Events for N-grams that didn't occur, and that makes this calculation more complicated, but this is a default one.  
  
So if you have a training data like AM, SAM and don't like green eggs and ham,.  
  
Then you can train your language model and you will get here the probability so the probability of I give a sentence start will be two thirds because there's twice that a sentence starts with an I. There's three times a sentence start so it's two thirds.  
  
Are if you have Sam given sentence start with one third because one sentence starts with Sam?  
  
If you have the probability of am give an eye, you look how often this eye occur. It does three times after that, two times the word am occurred, but again two serds, and so on. If you have sam give an.  
  
You have that in the first sense there's them or the other thing it's not, and so you can directly calculate.  
  
Which can be then used for many tasks and will see more to do.  
  
And the interesting thing is you can look at the probabilities and they seem to be really telling.  
  
So here is one large corpus where a lot of people work with is the proceedings of the European Parliament. So every speech in the European Parliament is transcribed and even translated. So you have there a big corpus of what people have said in the European Parliament since I think the' 80s or the' 90s.  
  
And you can take this corpus and calculate the counts and see what are the probabilities.  
  
It's done for the green, the red and the blue. The most probable word afterwards is the green paper.  
  
Seems that they have this concept of guess some type of paper.  
  
Then not surprisingly group is a political text, then of course the green group has quite hyperability because there is a green group in the European Parliament and that is a similar party. Green light is more general.  
  
Maybe if you have a more general text, the green light would have a higher probability than the green.  
  
So you see, of course, your probabilities really depend on what you're training you're representing.  
  
Here not since you can't really have all text ever spoken in English, so with your probability you're not modeling the probability that this is said by a native English speaker.  
  
Modeling the probability that this is in your text corpora. So that's a very important issue, so you're therefore like looking at your training data.  
  
And you see that it can distinguish green and red for very high probability because that is a typical.  
  
You also see that in some situations it is easier to predict the next word. There is not that much information in other situations.  
  
So for blue they are all probabilities. The highest is and all the others are lower, while for the red it's very common that in half of the time it's always the red cross.  
  
So then you can easily predict what is the next word, while in other scenarios that might be.  
  
In now the last thing, so in generally what is challenging is how do we deal with unknown?  
  
Because this normally destroys everything, if we have something unknown, we always get the probability zero, and that is, of course, not what we want to do. So just imagine your vocabularies. Only I go to KIT, and now you get the sentence I went to.  
  
Problem is if you have a trigram language model here, the trigram then like three out of your five probabilities are all zero. Any engram where Wendt is in is all zero probability because we haven't seen the word Wendt, so we have not seen all of these engrams.  
  
That means this calculation gets destroyed if we have a word and we have to deal with.  
  
And the larger your context is, so if you have a trigram you see that three probabilities are zero.  
  
Bigram probability you're lucky and only two of your probabilities get destroyed. On the one hand, of course, you want to have a larger ngram.  
  
On the other hand, with his traditional models, it's a challenge because if you have a larger angle and have one word, it really affects.  
  
And therefore, the challenging might not be the best. The problem here is if we are thinking about words, that's how we're dealing.  
  
So if you're seeing now more as a human, how could you? I mean, you will also see sometimes words that you haven't seen, so what could be other ways of dealing with words that we haven't seen instead of just ignoring them any ideas?  
  
Yes, that is done, that is, the smoothing technologies techniques I could give one complete lecture only about how to smooth things, so that is one way of doing it that is right.  
  
But is there another way if you think about words that you haven't seen yet?  
  
What we can do is deal with words that you haven't seen. Can we also enable that for the system?  
  
So the first one houses. I know you all know what houses are. But even if you wouldn't know what houses are, you could like interpret it. Oh, the house I know, and then there is an S at the end. So if there is an S to a noun, that means it's a plural form so you can deal.  
  
You have no idea what the second word here means. Anybody knows or wants to guess what it.  
  
At least guess that's very good exactly, so there seems to be a word exactly for that.  
  
But if you haven't seen the I guess, there is at least an amount of people here who have never seen that word before, and they might not know it. But you were able to estimate, oh, this seems to be a fear of something because it's end on phobia.  
  
And the question is if humans can do that so they can split words into parts and then work better. Maybe we can also do that with computer.  
  
And this can be done. We can go for the extreme case that we will not touch, which was partly done.  
  
But the most successful, which is currently used also, that is also used in neural networks because there we'll see we have exactly the same problem. If we have never seen the word, it won't work. Is we can split words into parts so we'll split words into some?  
  
Parts, and a successful part is not even using any linguistic based knowledge, but it's called the bite pairing coding, which is originally a compression method for text.  
  
And this method was then adapted in order to split work.  
  
The idea is you start with directors and then you're recursively replacing the most frequent pair of bites by an unused one.  
  
And that's why it's a compression algorithm. If it's an unused bite, it wasn't used, and now you're replacing bites by.  
  
And of course it's a greedy idea to first replace the pair of bites which occur the most often.  
  
So how is it applied here so?  
  
We are looking at the sequence of chair actors, and then we are replacing the two chair actors, which occur most commonly by a new symbol, which is just a symbol of the joint of these two chair actors, and we'll do that until we reach the fixed vocabulary.  
  
So the nice thing here is that we can then represent any text with any number of words with a fixed vocabulary.  
  
And that is achieved by if it's a common uncommon word which we haven't seen before, we're just replacing it by a sequence of subwords.  
  
So what we would do in the original example is here, so you have the sequence A, B, C, A, B, C. For example, then you would count the most frequent or most frequent sequence is A.  
  
So you're replacing by and then your sequence now looks like. The most frequent sequence is and so.  
  
And then, of course, you have to store some all year your vocabulary, your dictionary.  
  
But how does it now work for words?  
  
Are starting with representing strings as characters, and then recursively we are joining characters which often co occur into a subword.  
  
So let's look at the very small corpus. Of course, there are some things not really realistic if you do it by hand size. But if you have the corpus I go, he goes and she goes. Just assume that is our corpus, and now we are trying to run a bite pairing coding on this type of.  
  
What we then would have is we have our initial vocabulary that is mainly done only for computation.  
  
So the first step we are doing is we are splitting each word into its individual character.  
  
We have to add a special token at the end of the word, otherwise we don't know anymore where a new word.  
  
Because if you have this type of character representation, there's a space between each character and you don't no longer know is it now a space between a character or a word. So you're introducing this dot, which is a special symbol saying okay, hear the word end.  
  
So you still have ego, he goes, and she goes.  
  
And then this is your representation of the text of your text, now with Charact.  
  
And now we calculate the diagram statistics and hope they are correctly here.  
  
And then you are now looking, which is a diagram which occurs the most often, and in this time.  
  
So your first rule is that and together build the world. Go, it can be somewhere inside any.  
  
So what you're doing is you're replacing the tokens by your new token. Go if you're doing that.  
  
You're having the new corpus which has a little bit less tokens because is always replaced by gold and that is now your representation.  
  
So you also have to update your statistics and you're again looking what is the most frequent diagram.  
  
In realistic scenarios, that will be less often the case because the difference will be larger.  
  
So you can then, for example, extract the rule that and together build heat.  
  
And then you do that, and importantly you replace both occurrences of he.  
  
And you can then continue until you have the number of rules you will have. For example, you can go like this:  
  
If this is your targeted vocabulary, so that I is represented by I and sentenced length, so it has two. Go is also two. He has only one token, Goes is one token, she has two tokens, and Goes is one.  
  
So now you have your vocabulary and you have replaced this text by this type of representation.  
  
Here you can by hand predefined how big your vocabulary is because you said how many merging rules you apply your merging.  
  
That's the nice thing, and the second nice thing is that you still can represent any text.  
  
Because if you now, for example, have a new text, so what we had here is we had all our rules, so the rule was G and O together go to go, he and H and E go to he, and so he, and the word end goes to he, and word end, and so.  
  
So if we apply that now to a new text we have a sentence. They go. We first split it into all its part. Now I missed one space between word.  
  
So you see that doesn't always make grammatically correct because the he is very different from the he somewhere.  
  
But we have the ability to express any text now with a fixed vocabulary, and thereby can ensure that we have seen all our words.  
  
And this will be especially important if we are doing that as pre-processing to use text later in.  
  
And of course we have there modeling some type of morphology. You can now imagine often you'll have more rules, so typically you'll have maybe twenty thousand rules.  
  
And then you see that for example the word endings get joined together and then you have shewn and then you can add different endings.  
  
So to summarize today's lectures, these are the three most important things you should hopefully remember on the one hand: the bag of words representation.  
  
You should remember how a language model is modeled and you can use subword units to represent auto vocabulary entries in a text.  
  
Are there any questions then let's thank for joining you and hope see you all on Monday in the.