Chapter 2

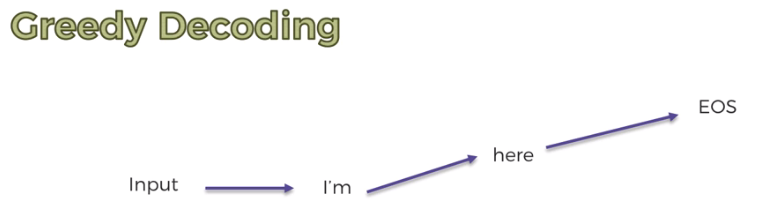
**Beam Search &**

**Attention Mechanism**

**2.1 Beam Search Decoding**

Here we discuss about *Beam search decoding*. Actually we're going to talk about two things: *Greedy decoding* vs *Beam search decoding*.

* Greedy decoding: We feed the input into the network and then it produces the first word. Eg: for 20000 words we select "**I'm**" and it is one of the highest.



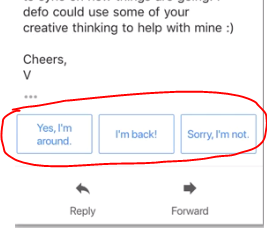
* Next feed "**I'm**" back into the network and then 2nd word comes up with highest probability from 20000 words. Say it is "**here**".
* And then again we feed in that words "**here**" and "**I'm**" into the network and we see that the network produces "**EOS**" the next step.
* So the response for "Are you back in Australia" will be "I'm here". That's the whole process. The reason it's called *Greeley's* because every time we just pick the word with the highest probability, like we're very greedy we only look at the one with the highest.
* Beam search decoding: On the other hand we have Beam search decoding which is a bit more sophisticated. We feed the input into the network but then *instead* of looking at just the *one word* with the *highest probability* we'll look at the very top three with highest probability for example "**Yes**", "**I'm**", "**Thanks**".
* It doesn't have to be 3 words it could be 5, 10 any number. It's up to you to decide how many beam's you want. If you look at 3 then you'll have three Beams and this time we have "**Yes**", "**I'm**", "**Thanks**".
* "**I'm**" has perhaps a higher score and "**Yes**" or "**Thanks**" have lower scores but nevertheless we're going to look at all three.
* Now it splits into *three possible answers* and we have three versions of our Seq2Seq from here they're like developing independent.
* For this *first Seq2Seq model* where the first word is "**Yes**" we now look at the second word we feed the word "**Yes**" back to the model. Then we get again top 3 options for the 2nd word "**sure**", "**I'm**", "**EOS**".
* So it grows like a tree. Each time we feed the words to the model we get three new branches.



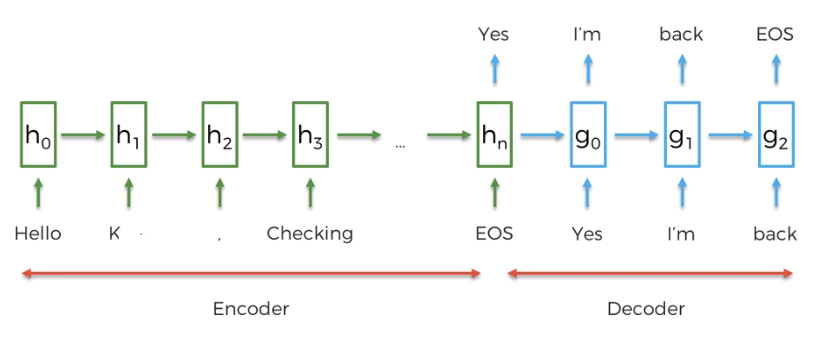
* How to pick wiener: We look at the highest ***Joint Probability*** across the *Whole Beam* to pick the winner. So in this case as you can see it might be a different result. It might be "yes I'm back EOS" but previously we had "I'm here EOS" for Greedy Decoding.



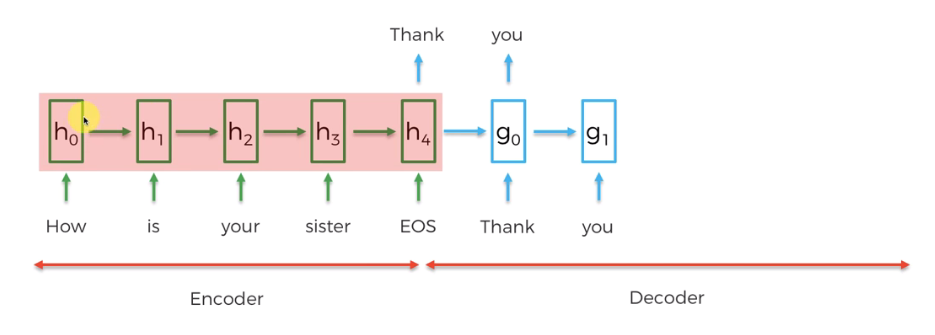
* In case of "***Greedy Decoding***" only "**I'm**" has *highest probability*, but in case of "***Beam Search Decoding***" "yes I'm back EOS" has *highest* Joint Probability than "I'm here EOS".
* Meaning that "yes I'm back EOS" a better fit for our answer.
* How to Limit Growth of the Beams: You can see the beams they grow very quickly (geometrically) 3, 9, 27 etc. The technique is called Truncating The Beams. The model calculates the ***Joint Probabilities*** *of the* ***Growing Branches*** *during the growth*.
* The algorithm will look at the beams that have already a very low joint probability.
* Say "Thanks please send …" might not be the best response so it has *very low joint probability*. Therefore the algorithm would just throw it away it won't it won't continue looking at this one.
* So algorithm *truncates the beams* that looks like they *won't be the winners* in the end. So there not gonna be too many Beams at the end. As the *beam propagates*, *number of beam grows* also and *algorithm truncates the beams having already Low Joint Probabilty*.
* So the Beam search decoding is the reason why we have "Reply Suggestions in Gmail":



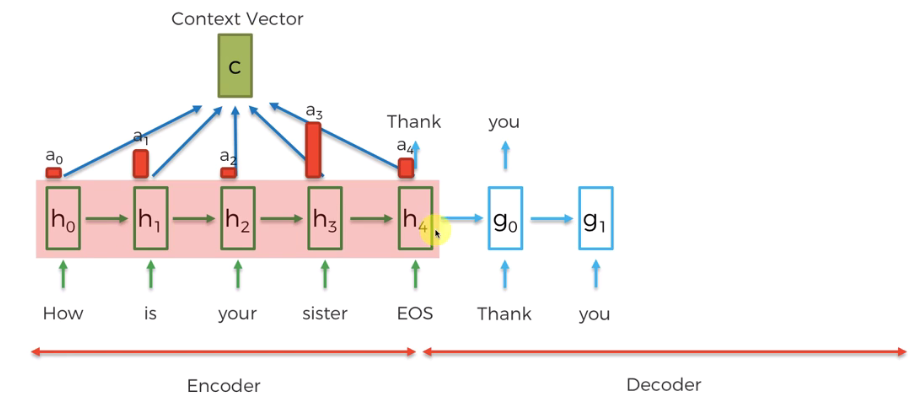
**2.2 Attention Mechanism**



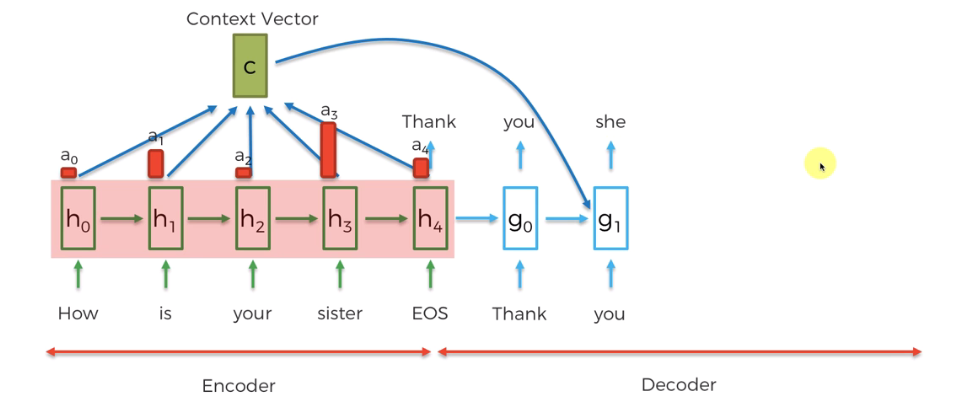
* Vector of meaning: When we done encoding, our whole input is encoded in the - layer which is called the *vector of meaning*. So the whole meaning of our sentence is contained to and then we use that to decode our whole NN over Decoder part.
* The *weak point* of this whole architecture is exactly in . Because this is a *fixed dimensional component* of our NN. Remember we have *variable length inputs* (no of words in a sentence varies), but all those are stored in this *fixed-dimensional* .
* That's not ideal.
* There's a lot of information to store in and moreover it doesn't really change depending on what type of sentence is the input.
* Other thing is that the meaning contained in has to be carried through the whole of the *Decoder-part* maintaining all of the information. Which is used to generate a response.
* However this approach is OK for short sentences & short responses and questions. But it's not good for but as they grow longer sentences (hard for the NN to be able to provide proper responses and maintain that meaning).
* Translation Analogy: We're going to discuss what happens even in translation. A Seq2Seq networks can be used for translation. Suppose you are a translator, you cannot just hear a whole big-sentence and translate it all at once.
* You *translate* it a *bit by* bit then you look to *certain point* and then you look at the *sentence* again and you *keep going*. So you have the opportunity of looking back at what you have.
* Context vector: So now we have to make our model to *"Look back to the input meaning"* during the *generation of output*. Let's say we have a input "How is your sister" and the generated response was "Thank you she is fine".



* What does attention do: Before we have an output come in from a Decoder-Layer (say or ), all of our input layers i.e. , , … (highlighted read) are taken and then NN will look back at them at that output stage.
* The decoder has complete liberty to access any of the information that is available before not just **-** *vector of meaning*but also , , , . Also the **-** *vector of meaning* passed to the, , **…** but they can also access, , , .
* How does this access work: Through training the decoder will know which of these elements , , , , to pay the most attention.
* It has learned to give *weights of importance* to these components. And create a context vector using this vector of weigts by multiplication .

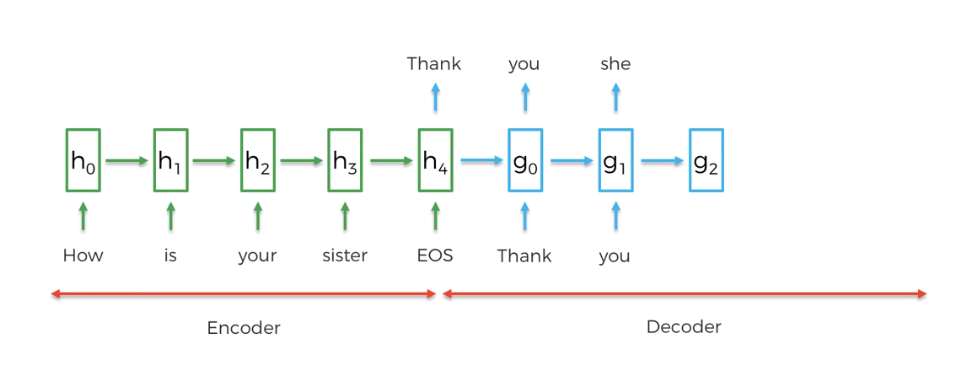


* There are *five weights*. The *bigger* the *weight* the more *significant* is that *component* and these weights created through training.
* Now in our example, at the stage of next output needs to pay-attention to most weight on this component i.e and .
* Here ***Soft-Max function*** (which makes ) will be used to make these operation easier.
* So this context vector is a *weighted sum* of all of these *weight vector* and or all of layers in our RNN.
* We feed this context vector into this new layer of our decoder as an *additional input*. And then we get the output "**she**".

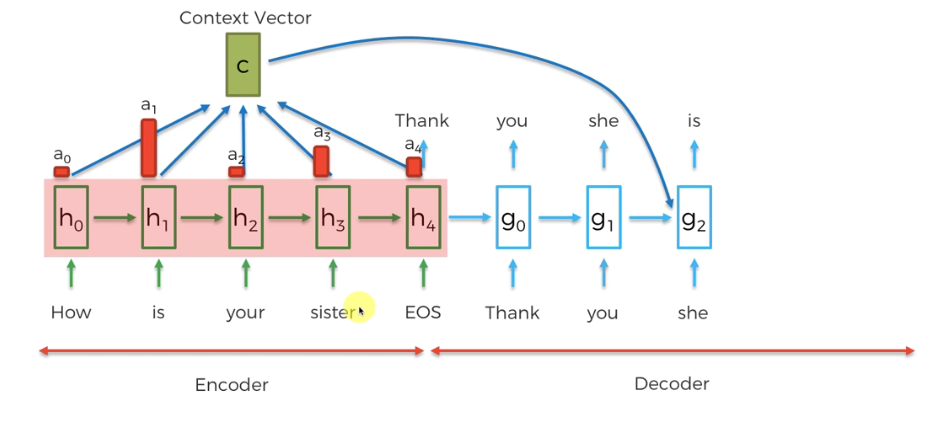


|  |
| --- |
| * What just happened: In this stage why the weights are exactly like this? * In English the *pronoun* will depend on the *gender*. If it's a *female* person it's a "*she*". If it's a *male* person it's going to be "*he*". * The meaning is the same. "He is fine" or "She is fine". That's the Decoder is doing using the context vector. Decoding the meaning of what we want to answer. * We go to a stage where *we need to* decide what *pronoun* to use. * It all based on *prior training* (we are now application in mode). Through the *training* the model learned to find *pronouns* in the *question* (using the weights). * The model doesn't care, it doesn't even know the actual meaning, but it all happens through the training. When there's a decision to say ***he*** or ***she*** it needs to go back and find something like this. |

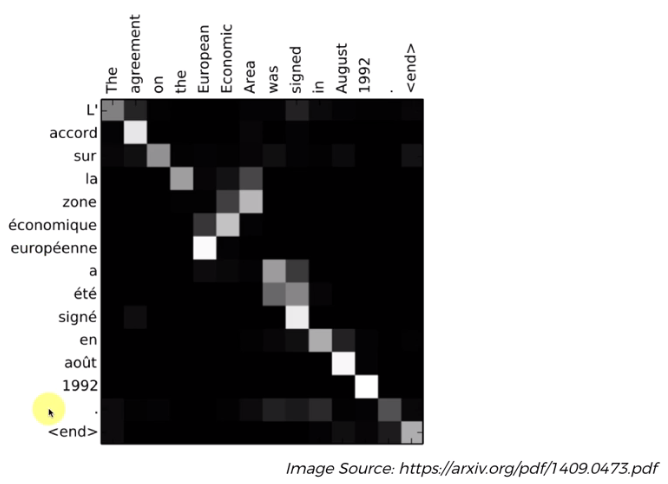
* Basically that's where this *high-weight* of comes from.
* Notice "***is***" also has impact on "***she***". This word is it's tallying us, *how many persons* are there.
* So "is your Sister" indeed is singular. That’s why "***is***" or "***are***" matters in this case, and that’s why it has high weight.
* Also notice "EOS" has a significant weight too, telling us the *end of sentence*.
* Let's consider another step. Notice the ***weight-vectors*** are ***different*** now. We build the Context Vector we feed it as an extra input to the RNN layer that generates the current output.



* It is going to output "**is**". In this case in our *encoder* part "**is**" has the *largest weight* and "*sister*" has *second large weight*. We are kind of deciding singular/plural "*these people are*" or "*that person/subject is*".



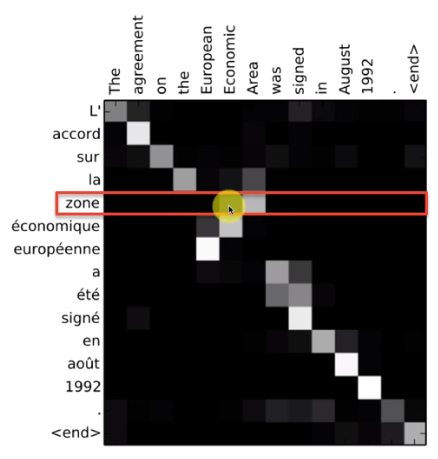
* Example: Let's have some look at something fun. This is from a research paper where ***Yoshua Benjio*** was one of the co-authors and this is a translation example from English to French. We got the following sentence is translated into French
* *The agreement on the European Economic area was signed in August 1992*, and then this whole translated into French:



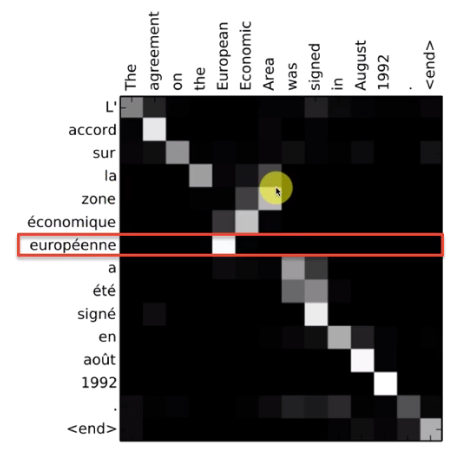
* These white dots is where the NN was paying attention to as it was translating. Those white dots *signify* the *weights* on every *single word* as it was *translated*.
* We gave the NN the ability to observe the input every single time as we just discussed.

|  |  |
| --- | --- |
| * You can see that here, it's pretty straightforward. When it done with first three words " *The* ", " *agreement* ", " *on* ", then when it reaches to 4th word "*the*", it looks at the word "*area*" also. * It's because *in French there are three different kind* of "the": "La", "Le", "L`" for Male, Female and Plural. * The machine/algorithm knows this and when it looks at "***the***" it needed more information, so it's paying attention to the word "***area***". There are also little bit attentions to "*Economic*" word. |  |

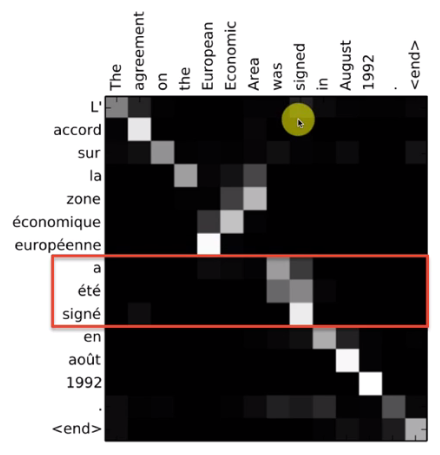
* Next French word is "zone", here you can see it's paying attention to "area" again and a little bit on "*Economic*" word.



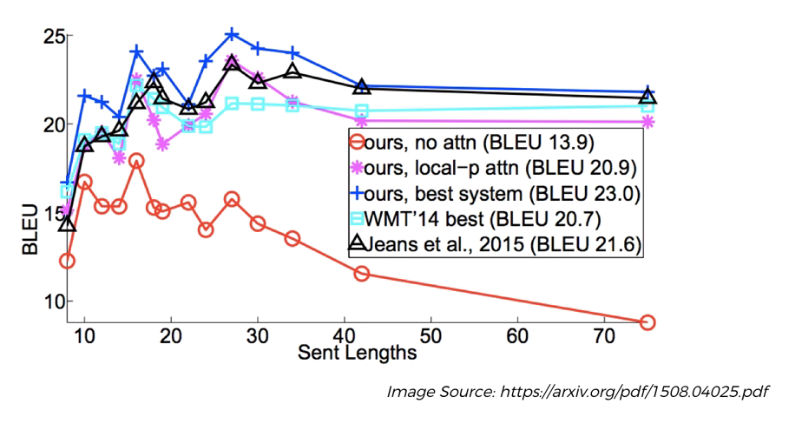
* Also notice when in English we proceed forward as " *European* ", " *Economic* ", " *area* " the NN goes back ward as "zone", "e'conomique", "europe'ene", it's because in French language its natural.



* Also notice in French following highlighted words are iner-dependent to each other, so we can see NN paying attention all of them together.



* Example: And then finally we have this other paper where **Christopher Manning** is one of the authors from Stanford.



* Those lines are representing different models "*Attention Mechanism*" in *translation*. The *blue line* represent the "*Best Attention*" where the *red line* represents "*Poor Attention*". In case of *red line*, it gives attention in the beginning but gradually it lose its attention. In case of blue, magenta, black, cyan lines they have "*Steady attention*" .
* BLEU represent the rating/score of attention. Most translation models BELU score are below 40.
* The point is when sentences get so long your attention drop off. That’s happening to the Red-Line.
* Global & Local Attention: When you have long sentence, you give your attention to 3-4 words, or just a couple words and that is more similar to what humans do.
* Like if you have like a 20 word sentence and you're translating it or trying to respond to it. You don't look at the whole sentence every time, you just look at parts of it. And that's more close to what humans do.
* Additional Read: If you'd like to learn more about attention then read the paper called Effective Approaches To Attention Based Neural Machine Translation by Minh-Thang Loung.

