Decoding Strategies

Decoding Hyperparameters & Practical considerations

Learning goals

- Learn how to use the `generate()` function of the `Transformers` library
- Learn how to choose from the different decoding strategies with the choice of hyperparameters
- See how hyperparameters affect the output of the `generate()` function

GENERATING TEXT WITH LLMs

In order to generate text with a generative language model you have to use their built in `generate()` method:

- You need a tokenized input prompt
- This will be the input to the model
- Then you specify the hyperparameters of the models' `generate()` method to control the output length and to choose the desired decoding strategy
- You can find all the code in this notebook: ▸ decoding_examples.ipynb

GREEDY SEARCH

The default decoding strategy is **greedy search**, if all the default hyperparameters are used

Prompt: "Once upon a time"

- `model.generate(tokenized_prompt)`
- **Output:** Once upon a time, the world was a place of great beauty and great danger. The world was a place of great danger, and the world was a place of great danger. The world was a place of great danger, and the world was a place of great danger.

BEAM SEARCH

In order to use **beam search** you simply have to add the `num_beams` argument to the `generate()` method and set it to a value > 1

Prompt: "Once upon a time"

- `model.generate(tokenized_prompt, num_beams=5)`
- Once upon a time, it was said, there would be a time when the world would be a better place. It was a time when the world would be a better place. It was a time when the world would be a better place. It was a time when the world would be a better place. It was a time when the world would be a better place.

SAMPLING WITH TEMPERATURE (1)

In order to activate **sampling** you have to set `do_sample=True`. To additionally use **temperature** you have to set the `temperature`. Also `top_k` has to be set to 0, as 50 is the default value

Prompt: "Once upon a time"

- `model.generate(tokenized_prompt, do_sample=True, temperature=0.7, top_k=0)`
- **Output:** Once upon a time, we could have seen ourselves as a modern day, modern version of ourselves. Ken, who was assembled with great courage and bravery to fight for the cause of women's reproductive rights, has been doing so for more than 15 years, and his career is full of inspiring stories. He is a poet, a civil rights leader, a hard-nosed activist and a true American hero.

SAMPLING WITH TEMPERATURE (2)

If we set `temperature` to a super high value, the output distribution will approximate a uniform distribution. For `temperature` $\rightarrow 0$ the output distribution will have all the probability mass in the most probable token and `generate()` will be equivalent to **greedy search**

Prompt: "Once upon a time"

- `model.generate(tokenized_prompt, do_sample=True, temperature=10000000.0, top_k=0)`
- **Output:** Once upon a time indicators Gran slew extrater rockedIA moaningJohn Chin knocking transgender Button nin forgiving submarine traveller efforts Pascal despairaways Hem Melvin Toryunn Amb velvet

This output is as expected since with such a high `temperature` every token in the vocabulary will have the same probability

TOP-K SAMPLING

In order to activate **top-k sampling** you have to set `do_sample=True` and set `top_k` to a integer > 0. If you use `top_k=1` it will again be equivalent to **greedy search** as only the most probable token is chosen

Prompt: "Once upon a time"

- `model.generate(tokenized_prompt, do_sample=True, top_k=10)`
- **Output:** Once upon a time, he did not think he had a chance and turned to her and said:
  "'My dear, what do you mean by that? I know your father, but he is dead.'
  "So she said:
  'I knew it. I know your father, but I do not know his whereabouts. You know my mother.'

TOP-P SAMPLING

Set `top_p` to some float and `do_sample=True` to use **top-p sampling**. Using a very small value is equal to using **greedy search**. Using `top_p=1.0` will consider all tokens

Prompt: "Once upon a time"

- `model.generate(tokenized_prompt, do_sample=True, top_p=0.7, top_k=0)`
- **Output:** Once upon a time they arrived at the well of water, they saw that their creature was burning in flames. After they had brought their clothes back to the dry land, they were found that they had died in the burning body.

CONTRASTIVE SEARCH

For **contrastive search** you have to set `penalty_alpha` to a float and set `top_k` as previously mentioned. Remember we are **not** sampling here. For `penalty_alpha=0` we only maximize the model confidence term and it becomes **greedy search** again. Using `penalty_alpha=1` we only care about the degeneration penalty term

Prompt: "Once upon a time"

- `model.generate(tokenized_prompt, do_sample=True, penalty_alpha=0.6, top_k=20)`
- **Output:** Once upon a time, the man who had been the object of the most intense and most intense hatred, the most intense and most intense hatred, was the man who had been the object of the most intense and most intense hatred, and the man who had been the object of the most intense and most intense hatred.

CONCLUSION

- In practice you wouldn't use one decoding strategy in isolation
- You would have to play around with the different hyperparameters and decoding strategies until you get a good result
- But how do we determine if a generated text is good or not?
- For that there are several evaluation metrics $\rightarrow$ next chapter