IN4080 exercises (autumn 2023)

Week 9: Dialogue systems and chatbots

Linguistic analysis

Analyse the following dialogue (audio, transcript) and answer the following questions:

1. How are the dialogue turns structured, based on the observed linguistic cues?
2. What kind of speech acts are used through the dialogue, according to Searle's taxonomy?
3. What are the grounding signals and strategies used through the dialogue?
4. Can you find some examples of conversational implicatures?
5. List a few (2-3) deictic markers occurring in the dialogue.

Cross-encoders

We reviewed during the lecture two types of retrieval-based chatbots:

- Systems that computes the semantic similarities between the user utterance (and its context) and all utterances in a corpus, and then use what follows from the most similar utterance as response.
- Dual encoders (also called *bi-encoders*) that rely on two fine-tuned encoders, one for the user utterance (and its context) and one for a candidate response. The relevance of the candidate response for the user utterance is then simply the dot product of their two vectors.

There is, however, a third type of approach that can be used, based on a *cross-encoder*. The idea is to use a single encoder which takes as input the *concatenation* of the utterance (and its context) and a potential response, with a special separation symbol between the two, like this:

> This is the user utterance <sep> And here a possible response.

The vector obtained by encoding this concatenated string is employed for a binary classification task, namely predicting whether the candidate is a good response for the user input. The cross-encoder is then fined-tuned in the same way as the dual encoder, using positive and negative examples.

Question: What would be the advantages and disadvantages of cross-encoders relative to bi-encoders? Explain your answer. You can of course look on the web to learn more on bi-encoders and cross-encoders.

Pointwise, pairwise and listwise ranking

Retrieving good responses to a given input is, at its core, a *ranking problem*: we have a long list of candidate responses, and our goal is to find the best possible one. This is done by creating a ML model that outputs a "score" for a given response, and then sorting the response to find the one with the best score. But the most important isn't really to produce highly accurate scores for each candidate. Rather, the most important is that the relative *ranking* of those responses is correct, especially at the top (since we will select the top-ranked candidate as output).

There are several ways to design ML models that are specifically optimized to rank potential candidates, using special loss functions. There are three families of approaches: pointwise ranking, pairwise ranking and listwise ranking.

Question: give a brief summary of each of those three approaches and explain how they would be applied in practice to create a retrieval-based chatbot model. Look on the web to learn more about those ranking approaches.

Efficiency considerations

Retrieving the best response might be computationally (and memory-wise!) difficult if one has a very large set of possible candidates. There are several ways to improve the efficiency of such approaches:

- You should normalize the vectors of all candidate responses (to reduce the cosine similarity to a simple dot product) and store those normalized vectors in memory (to avoid having to recompute them every time).

- You typically don't need to sort the full list of candidate responses according to their score, since you are only interested in the top-scoring responses. To this end, you can use functions such as topk in Pytorch.

- Finally, instead of using exact search through the list of candidate responses, you can rely on various approximate search techniques, often based on Approximate Nearest Neighbour (ANN) techniques, which works by partitioning the vectors into smaller fractions that contain similar embeddings. Popular libraries for approximate search are Annoy (from Spotify) and FAISS (from Facebook Research).

Question: Have a look at the code here which relies on the FAISS library for approximate semantic search and write a short step-by-step explanation of what the code does. How could this method be adapted for a retrieval-based chatbot?