Practical No. 5

Adriana Bukała
ab394064@students.mimuw.pl

May 17, 2022

1 Part 1

2 Part 2

e)

Performance of the vanilla model measured on the last (10th) epoch:

GREEDY
Valid BLUES SCORE 0.2307998435
Valid Corpus Matches : 83.84%
Valid Corpus Success : 54.55%
Corpus BLUES SCORE 0.2267021986
Corpus Matches : 84.85%
Corpus Success : 45.45%

BEAM
Valid BLUES SCORE 0.2267650959
Valid Corpus Matches : 84.85%
Valid Corpus Success : 57.58%
Corpus BLUES SCORE 0.2122434568
Corpus Matches : 86.87%
Corpus Success : 49.49%

f)

Performance of the softmax policy model measured on the last (10th) epoch:

GREEDY
Valid BLUES SCORE 0.2425197277
Valid Corpus Matches : 94.95%
Valid Corpus Success : 77.78%

Corpus BLUES SCORE 0.2207359974
Corpus Matches : 90.91%
Corpus Success : 60.61%

BEAM
Valid BLUES SCORE 0.2456288883
Valid Corpus Matches : 92.93%
Valid Corpus Success : 65.66%
Corpus BLUES SCORE 0.2273391975
Corpus Matches : 90.91%
Corpus Success : 55.56%

Corpus matches and corpus success ratios are higher in the second scenario, i.e. for the model utilizing softmax policy. This difference is especially visible, when comparing performances of greedy search policy. BLEU scores are higher as well, although differences are not that large compared to the success ratio ones.

3 Part 3

a)

question:
i would like an [value_pricerange] restaurant in the [value_area] part of town
ground truth:
[restaurant_name] is a [value_food] restaurant located in the [value_area] part of town and the price range is [value_pricerange] . would you like their address ?
model's output:
there are [value_count] restaurant -s in the [value_area] part of town . would you like to try something else ?
possible fix:
Model misunderstood the intent of a given question and possibly even left out information about the price range, which is obviously as crucial as the restaurant's location. To fix this, one could increase training time of the model and enrich training data in more examples.

question:
1. i would like a [value_pricerange] -ly priced restaurant that serves [value_food] food
2. how about [value_food] food ?

ground truth:
1. there are no restaurant -s [value_pricerange] -ly priced that serves [value_food] food .
2. [restaurant_name] is a [value_pricerange] -ly priced restaurant that serves [value_food] food in the [value_area] part of town .

model's output:
1. [restaurant_name] is a [value_food] restaurant in the [value_area] part of town .
2. [restaurant_name] is a [value_food] restaurant in the [value_area] part of town .

possible fix:

Once again, model did not consider information about price range and even additionally gave information about restaurant's location. This could be caused by two things: 1) model is precise, i.e. learnt to give as much information as possible, which could be beneficial for the user, 2) training data is more biased towards restaurant's location than its price range, in which case more training examples should be added. Moreover, model repeated the answer, which could be fixed by adding some kind of penalty.

question:

[value_pricerange] restaurant , please

ground truth:

what type of cuisine would you prefer ?

model's output:

[restaurant_name] is a [value_food] restaurant in the [value_area] part of town .

possible fix:

Once again model skipped information about price range. Moreover, it generated very precise answer based on very imprecise question. This behaviour could be improved by adding more imprecise examples to the train set, so that model would learn to ask additional questions more often.

Additionally, it would be beneficial to use a pre-trained model, regardless of the type of errors mentioned above.

b)

BLEU has many advantages such as determinism or computational speed, however it requires both the question and the answer to be very precise. That does not apply to natural language in general, with a few exceptions such as TV show "Jeden z dziesieciu"[1]. Due to openness of dialogue conversations and inexactness of humankind, I would not recommend anyone to rely on BLEU score for evaluation of dialogue systems.

c)

Model utilizing softmax policy consists of over 100k parameters (108450 to be exact). Model is trained using 478 dialogues for 1935 turns per epoch. By the rule of thumb, there should be at least one order of magnitude more examples than trainable parameters for neural networks (source: Google Developers Blog). That naturally leads to overfitting to the training set. As a result of that, model learns some structurally fixed answers. Moreover, it repeats them over and over again, even in the same dialogue, what has been shown in previous sections.

[1] TV quiz consisting of rather precise, general knowledge questions, for example "In which year did Napoleon die?"

d)

First of all, task-oriented conversations are very specific to the given task, meaning that there is a high probability that there is no open-sourced data set for a required task in a required language. Because of that, developers often create a dialogue system for the task for the sole purpose of gathering the data. That creates vicious loop, i.e. there is no data to train neural network, so additional system has to be developed, but there is no data for that either, so developers have to create such conversations manually or use Wizard-of-Oz studies [1]. That is extremely time consuming, especially since the process has to be repeated every time the task slightly changes or the new functionality is added. Moreover, this process is prone to bias the model, since there is a limited number of people involved and a limited number of generated conversations.

e)

One possibility is merging different task-oriented dialogue data and utilizing them for the creation of the pre-trained model. This idea has been tested by the authors of TOD-BERT [2]. With the usage of 9 human-human and multi-turn task oriented dialogue datasets, differentation between user and system tokens, and contrastive learning, authors of TOD-BERT were able to come up with a model outperforming standard baselines (such as BERT) on 4 downstream task- oriented dialogue applications. NLP models would also benefit from such pre-training regarding the out of vocabulary words. Moreover, since neural networks in general benefit from large amount of data, one could pre-trained model in a semi-supervised or unsupervised manner, which could yield discriminative features (source: OpenAI Blog). A variety of data sources could be utilized in that way; from Wikipedia articles, through paper abstracts (which could be a good source of domain-specific words), textbooks or questions from TV quizzes.

References

[1] Walter Lasecki, Ece Kamar, and Dan Bohus. Conversations in the crowd: Collecting data for task-oriented dialog learning

[2] Chien-Sheng Wu, Steven C.H. Hoi, Richard Socher, and Caiming Xiong. 2020. TOD-BERT: Pre-trained Natural Language Understanding for Task-Oriented Dialogue. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 917–929, Online. Association for Computational Linguistics