NLP 2016/2017 - Homework 3

Marco Favorito 1609890

September 30, 2017

1 Introduction

In the Section "Overview" are described the main characteristics of the project; In the section "Models" every used machine learning model is briefly described and evaluated. In section "Conclusions" there are some conclusions about the project.

Technical details can be found in the README files and in the comments across the code.

2 Overview

2.1 Building blocks

The knowledge chatbot is built upon the following machine learning models:

- Relation Classifier: used in the Querying mode for predict the relation of the user query;
- Concept Recognizer: used in the Querying mode for determine the concepts involved in the query of the user;
- Answer Generator: used for answering to the query in the Querying mode.
- Answer Concept Recognizer: used in the Enriching mode for determine the concept in the user answer;

The machine learning algorithms employed are mainly neural-based. In the next sections there will be provided clear descriptions and quantitative evaluations for each of them.

2.2 Architecture summary and technical choices

The project code has been written in Python, version 3. Among the most important libraries/frameworks used there are SQLAlchemy for the interaction with the local mirror of the KB, Tensorflow as backend for Keras and PyTorch (the latter used only in the Answer Generator model), spaCy for some NLP utils (dependency parsing, tokenizing etc.) and Telepot for manage with the Telegram API.

The KB is mirrored with paging, checking if there are new items to add in the local database from the central server. The mirror database engine is SQLite. The interaction is made with the ORM SQLAlchemy. The messages are processed synchronously. No special input checking is made on the input messages in order to validate the workflow.[1].

2.3 Workflow summary

At the beginning, the bot asks to the user the domain of conversation, which are retrieved dynamycally from the database. Then it asks if the user want to ask a question (Querying mode) or to let be the bot to ask something (Enriching mode).

2.3.1 Querying mode

Upon "Yes" (Querying mode), tho bot wait for a query. The Querying mode is addressed in the following way:

- Predict class probabilities for the query with the Relation Classifier and get the class which maximizes the probability according to the domain of conversation;
- Predict concepts strings in the query with the Concept Recognizer, with the known relation;
- Given the relation and the concepts mentions, look for an answer in the Knowledge Base;
  - if the query is binary check occurrences in the knowledge base of the left and the right concept mention on the same triple with the predicted arelation, and answer to the user accordingly;
  - if the query is "open", check only for items which have the left concept "like" (with SQL expression semantic) the queried concept. There is a variant for "inverse relation pairs" (i.e. "SPECIALIZATION" and "GENERALIZATION"), in which the concepts are switched and the search is retried.
- In case of insuccess, reply to the user with a "failure" message; otherwise, use the Answer Generator to generate a new answer, after some preprocessing of the query in input.
- Build a new question pattern with the query and the concepts and load it in the database.

2.3.2 Enriching mode

Upon "No", the bot has to query the user. The Enriching mode is addressed in the following way:

[1]I chosed to focus to specifc-NLP aspects of the project (i.e. how to tackle the proposed tasks), which results in a piece of software not still ready for a deploy in a production environment. I hope you will agree with this approach.

- Retrieve the concept belonging to the chosen domain, thanks to the mapping concept-domain you provided and the "domains" field in the "items" relation (i.e. the triples);
- Get the relation according to the occurrences of that concept into the KB triples, chosing the one with less occurrences;
- Generate the query picking a "unary" question pattern (not boolean question battern) randomly among the ones belonging to the chosen relation, and replace the placeholder "X" with the chosen concept mention. Then send the query to the user and wait;
- Wait the user answer; then detect right concept with the Answer Concept Recognizer and store the new triple into the KB.

At the end of every procedure (either the former or the latter mode), the bot asks again to the user the domain of conversation.

3 Models

3.1 Relation Classifier

The input of the model is the plain query and the classes are the relations. The neural network is formed by and embedding layer followed by two layers of bidirectional LSTM and a fully-connected layer. The query in input is represented as a one-hot encoding which is then processed by the embedding layer.

In Table 3.1 is shown the evaluation on training data retrieved randomly from the database and filtered with some heuristic

3.2 Concept Recognizer

The inputs of the model are questions and relations, whereas the outputs are the tags for every token of the question. The possible tags are:

- `l`, i.e. "left concept tag";
- `r`, i.e. "right concept tag";
- `n`, i.e. "no concept tag";

For instance, the following question "Is the Colosseum in Rome?" should be tagged as:

```
Is the Colosseum in Rome ?
n  n   l         n  r    n
```

Then some postprocessing is required for merging the words that belong to the same concept.

Relation	Test size	Correct	Acc.
COLOR	43	40	93.02%
SIZE	663	662	99.84%
PLACE	638	638	100%
TIME	635	618	97.32%
PURPOSE	674	673	99.85%
GENERALIZATION	678	668	98.52%
MATERIAL	651	650	99.84%
SOUND	656	654	99.69%
TASTE	368	368	100%
ACTIVITY	174	172	98.85%
HOW_TO_USE	130	130	100%
PART	677	676	99.85%
SPECIALIZATION	657	649	98.78%
SHAPE	210	207	98.57%
SIMILARITY	444	442	99.54%
SMELL	124	124	100%
Total	**7422**	**7371**	**99.31%**

Table 1: Statistics for the Relation Classifier. The train data size is two times the test size, and the split ratio for validation is one third.

The neural network has two inputs: one from the sequence of words of the question and the other from the relation vector, both one-hot encoded. The former input pass through an embedding layer, while the latter is added to the former through a fully-connected layer. The following layers are two layer of bidirectional LSTM and one fully-connected layer.

In Table 3.2 is shown the evaluation on training data retrieved randomly from the database and filtered with some heuristic (e.g. only "open" questions, left concept mention present into question string etc.)

3.3 Answer Generator

The inputs are more complex compared to the previous examples.

- Firstly, the items in the KB are filtered according to some rules, such as the presence of the concept mentions both in the question and the context.
- Then, the context is transformed, replacing concept mentions with placeholder different for the left concept and the right one (e.g. I used `xxx` and `yyy`.
- The transformed context is then dependency parsed, and the shortest path between the left and the right concept is retrieved.

Relation	Support	Correct	Acc.
SIZE	663	656	98.94%
TIME	647	600	92.73%
GENERALIZATION	662	653	98.64%
PLACE	696	683	98.13%
HOW_TO_USE	128	125	97.65%
SIMILARITY	438	421	96.11%
PART	641	626	97.65%
SOUND	665	659	99.09%
SHAPE	213	194	91.07%
MATERIAL	639	629	98.43%
COLOR	51	48	94.11%
ACTIVITY	178	162	91.01%
PURPOSE	650	639	98.30%
SMELL	125	125	100%
SPECIALIZATION	666	656	98.49%
TASTE	360	359	99.72%
Total	**7422**	**7235**	**97.48%**
Tags	**45166**	**44877**	**99.36%**

Table 2: Statistics for the Concept Recognizer. The train data size is two times the test size, and the split ratio for validation is one third.

- some additional words in the "nearest" of the tags in the path are added to the final string.

The purpose of this procedure is to get a complex answer from the context, instead of the too simple answers in the KB. For example:

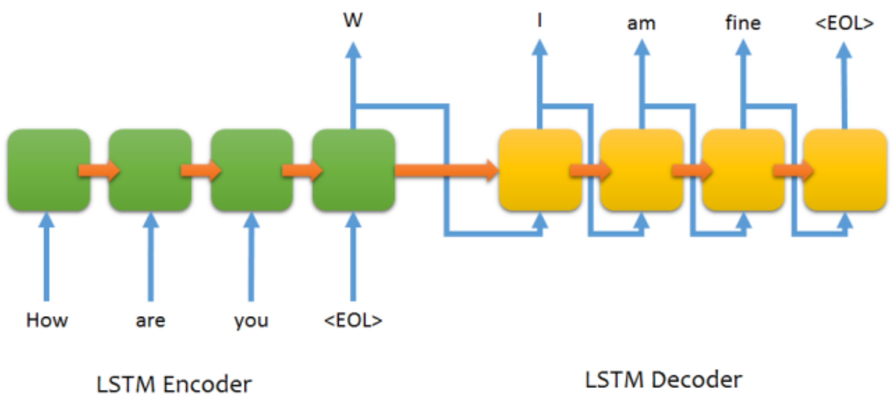
```
CONTEXT: The Colosseum, also known as the Flavian Amphitheatre, is located in Rome.
MODIFIED CONTEXT: The xxx, also known as the Flavian Amphitheatre, is located in yyy.
SHORTEST PATH: xxx located in yyy
ENRICHED PATH: The xxx is located in yyy
```

Beside the context, also in the question the concept mentions are replaced with the placeholders of above.

The neural network model simply applies the Sequence-to-Sequence model [**Sutskever**] (see Figure 1). As recurrent neural network layer I used GRU (better for short sequences and computationally easier) and implemented an Attention mechanism for the decoder.

In Table 3.3 is shown the evaluation on the "biased" corpus (items are filtered according to some heuristic described above). The evaluation consists on a Macro-averaged F1 score, which measures the average overlap between the prediction and ground truth answer. I considered the prediction and ground truth as bags of tokens, and then I computed their F1.

Figure 1: The Sequence-to-Sequence model



3.4 Answer Concept Recognizer

The Answer Concept Recognizer works almost as the Concept Recognizer, but it is trained on the answers instead of on the questions. In table 3.4 there is the evaluation of the model.

4 Conclusions

The main problem coped with this project is the data provided by the KB: they are quite noisy, both in their intrinsic nature and misformatting of the string (e.g. the concept string). Most of the triples are not usable since they do not have mention concepts, which as described above are necessary for the training of the model.

However, even with a small quantity of triples for the rarest relations, the trained models seem to perform quite good. An overall evaluation is hard, because each step in the workflow can fail in some particular case not treated or not present in the training data.

An important limitation of the system is that the check in the Knowledge Base is made without any spell-checking, which increases the false negatives percentage; moreover, for the concept retrieval is used the SQL "like" operator on the concept mention, and since it is case-insensitive it may lead to false positives.

Also the Answer Generator is not perfect, due to the noisy training data provided (often the proposed refactor is not perfect). The Answer Concept Recognizer "prefers" short and coincise answers, which are the features of the majority of samples seen during the training phase.

Relation	Avg F1 score
TASTE	79.88%
TIME	86.11%
PLACE	77.26%
GENERALIZATION	96.88%
SPECIALIZATION	93.40%
SIMILARITY	75.82%
HOW_TO_USE	79.69%
SHAPE	72.24%
MATERIAL	79.59%
SMELL	49.41%
PART	89.50%
COLOR	77.44%
SIZE	73.74%
SOUND	91.04%
ACTIVITY	70.84%
PURPOSE	84.46%
Total	**83.73%**

Table 3: Statistics for the Answer Generator. The train size is 6166, while the test size is 3037.

Relation	Support	Correct	Acc.
PART	288	288	100%
SPECIALIZATION	360	360	100%
MATERIAL	451	450	99.77%
SIZE	251	251	100%
ACTIVITY	92	91	98.91%
TASTE	152	152	100%
GENERALIZATION	219	219	100%
TIME	355	355	100%
PURPOSE	316	315	99.68%
SIMILARITY	197	189	95.93%
SOUND	408	404	99.01%
PLACE	388	388	100%
COLOR	28	28	100%
SMELL	24	24	100%
SHAPE	122	118	96.72%
HOW_TO_USE	48	46	95.83%
Total	**3699**	**3678**	**99.43%**
Tags	**6124**	**6091**	**99.46%**

Table 4: Statistics for the Answer Concept Recognizer. The train data size is two times the test size, and the split ratio for validation is one third.