Bazinga-Bot

NLP - Final Project

Francesco Fabbri
Sapienza, University of Rome
francesco_fabbri@outlook.com

ABSTRACT

We present Bazinga-Bot: a simple but effective Knowledge Chatbot developed as final project of the Natural Language Process course. Exploiting the data stored in the Knowledge Base Server (KBS), our dialog system is able to answer questions and even to learn by asking questions itself. For the **Querying** step the system exploits a Deep Learnig framework where, it first predicts the type of question through a simple LSTM architecture and then, depending on the type, activates its specific model. Meanwhile in the **Enriching** phase our Bazinga-Bot generates the questions looking at the missing information in the KBS data, exploiting the Babelfy framework for extracting the concepts from the generated Q/A pairs. In our Deep Learning setting, for each type of questions it has been implemented a specific NN architecture: this choice is given by the structure of the data, which basically presents an unbalanced distribution of the questions types. There are basically three detected type of questions: the generic open-ended, where for every question there is at most one answer; the multiple choice open-ended ones where it's possible to assign many answers to the same question; finally, the closed-ended, i.e. the Y/N questions, which represents the majority of Q/A pairs in our data. For the first 2 types we provide a simple but effective Seq2Seq model and, in the closed-ended we provide an accurate LSTM architecture which deals with a binary classification task, i.e. predicting Y/N.
Our Knowledge-Bot, even suffering due to the presence of a lot of meaningless sentences in the dataset, is likely to improve using an annotated corpus. Moreover, exploiting the long-term usage, i.e. increasing the interaction with many users, is expected to improve the quality of the data and then the quality of the responses.

1 INTRODUCTION

The interaction with the machine via natural language is actually one of the most vibrant topic of Deep Learning. The dialogue systems, or chatbots needs ideally to provide the user with an informative answer and the Bazinga-Bot can be definied as goal-oriented, since, after specified the topic, it has to answer questions related to that one.
Main Features. During the Querying step, given a specific pair (domain, question) our Bazinga-Bot is capable to: (1) predict the relation, (2) recognize the type of question, (3) exploiting all the information included in the dataset. In the Enriching the chatbot: (1) generates a question which the aim is to improve his knowledge (2) parse the answer exploiting the Babelfy method (3) stores the gathered information in order to improve the old dataset.
User check. In both cases in order to improve the quality of his knowledge and learning process it always interacts with the user, which is asked to check the sense and reasonableness of the Q/A pair just obtained after the conversation. This feature allows the user to solve misspelled and no-sense Q/A pairs.
Dirty Data. We have to point out that the KBS data has been generated extracting informations and concepts from Wikipedia pages, so they are basically meta-data. Usually, the data exploited for building chatbots are annoted corpus so, taking into account this setting, which looks as a limitation, we have to clean as much as possible the Q/A pairs before training our NN architectures: we have to discard all the "bad" couples extracted, which are either meaningless or useless for our purpose.

2 DATA

In this section we describe the processed dataset: building a simple but meaningful dataset is the priority through all the steps. In this process we have to deal with a crucial trade-off: cleaning as much as possible the dataset, discarding all the meaningless sentences, but without loosing too much information. Since we couldn't analyze the structure of each single sentence, we looked at the main patterns raised by the statistical analysis applied on the data.

2.1 Cleaning KBS entries

After downloaded more than 1 milion of entries from the KBS, we keep the ones which contain the Babelnet IDs associated to the concepts involved in the observation. Looking at the distribution of the number of concepts involved in each question, we discard the couples with more than 2 concepts involved, which represents approx. the 12% of the dataset. Then, we connect the concepts to the domains through the edges provided by the application of BabelDomains, a method which enrich lexical items with domain information [1]. In this section we loose approx.ly the 28% of the dataset, which after all these processes counts 658946 entries.

Table 1: Summary of Cleaning Process

Data	N. Obs	Discarded
KBS Data	1128884	
BabelNet ID	949185	179699
2 concepts	843074	106111
Edge (C1,C2) - Domain	658946	184128
Without symbols	641445	17501
len(Q) $\leq$ 10 + len(A) $\leq$ 16	627841	13549
Meaningful	616656	13549

2.2 Bulding Dataset

Now we detect all the possible useless and no-sense patterns present in the Q/A couples.
Special characters. First, we remove the pairs which contain some

special characters: these sentences are mainly useless and removing them we just loose a small portion of the dataset (2.6 percent approx).

Analysis of the distributions. Trying to specify the nature of the data, we split them separating the closed-ended questions (**Y/N**) from the others, **open-ended** ones. Focusing on the latter type, we decide to look at the distribution of the lenght of each Q/A, i.e. the number of words in each sentence, in order to check whether it's possible to reduce the expensive of the Seq2Seq pre-processing step given by the One-hot Encoding phase during the training of the NN architectures. As shown in the Figure 1, the majority of the Q/A couples are associated to the initial values of the distributions: this means we can filter the pairs without loosing too much information. We choose to keep the pairs with at most 10 words for the questions and 16 for the answers, loosing approx.ly the 3% of the entries.

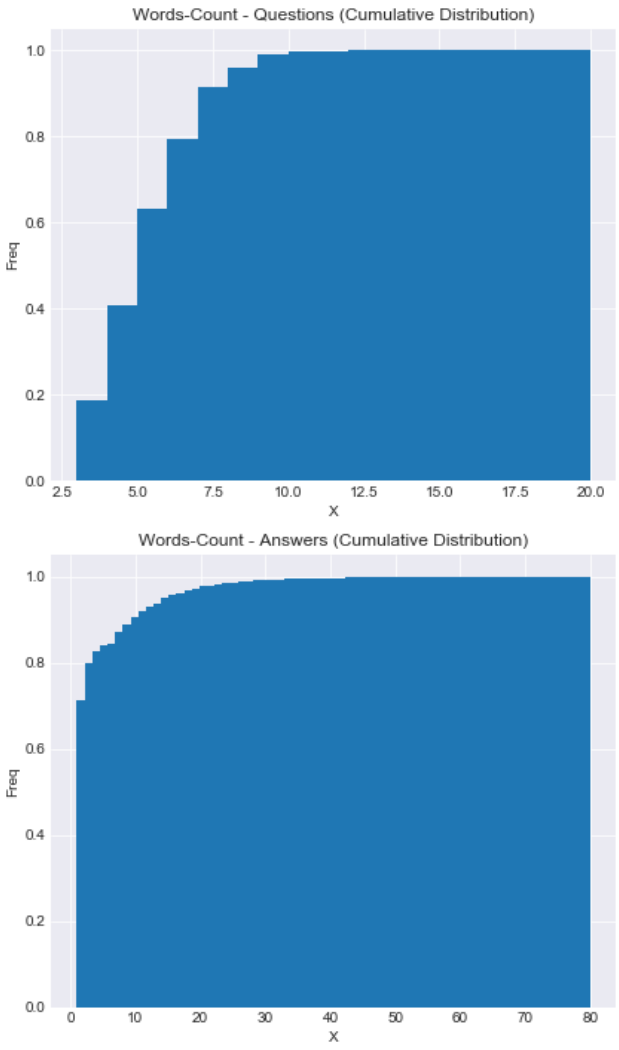

Figure 1: Cumulative Distributions Q/A

Redundancy and Stopwords. At this point there are still some bad patterns which characterize meaningless answers, as the followings:

- How can I use IAST? - It can be
- What is Hardt Railway? - line in
- Where is located Dachau concentration camp? - state of
- What is a mass? - property of
- Where is City of Armagh High School? - city of
- Where is located Enonkoski? - province of

We apply a simple heuristic to remove these kind of pairs: analyzing the POS-TAG [2] of the answers, we discard the ones which as last word have either a coordinating conjunction (CC) or a preposition or a subordinating conjunction (IN) or a modal verb (MD). Moreover, we discard also the ones which have as last word the article 'the' and the verb 'to be'. Moreover, we remove all the answers which dont't provide any information to the user applying another simple heuristic rule: we discrard all the answers which are completly inside the questions. In this way, we remove pairs as the followings:

- What is an example of a television broadcasting? - television
- Where is Ossipee Lake located? - Ossipee
- What is Religious clothing? - clothing
- Where is Linden Railway Station placed? - Linden
- Where can Perimeter Center be found? - Perimeter

Answer selection. Looking at the open-ended questions, we have to deal with the following setting: considering 2 or more entries which have associated the same tuples (question, domain-relation) but different answers, there are two main scenarios that would occur. In the first, it would be possible that all the answers are correct and no-one is better than the others. From the other side, it would even happen that only one, among the possible choices, is correct. In other words, we cannot know which answer is more coherent or meaningul than the others and, taking into account the training step of our model on the data, we have to solve this issue in order to avoid disambiguation. Indeed, the model is effective if and only if for a specific question we can associate at most only one answer, so for many tuples we may have the same answer but for different answers we don't want to have the same tuple.

2.3 1st Scenario

Focusing on the patterns within the questions we compute the n-gram for n equal to 2, 3 and 4. Looking at the most frequent patterns in these 3 different settings it's possible to classify a specific type of question which is included in the 1st scenario mentioned earlier: the multiple choice open-ended questions, characterized by the following patterns: (1) example (2) part of (3) instance of (4) can be. The answers associated to this kind of questions are all equivalently right, then for each tuple (question, domain-relation) the model used in this case has to predict a subset of answers.

2.4 2nd Scenario

Open-ended questions. The remaining tuples (question, domain-relation) of the open-ended questions, which belong to the 2nd Scenario,i.e. the Generic ones, can be classified looking at the distribution of the related answers:

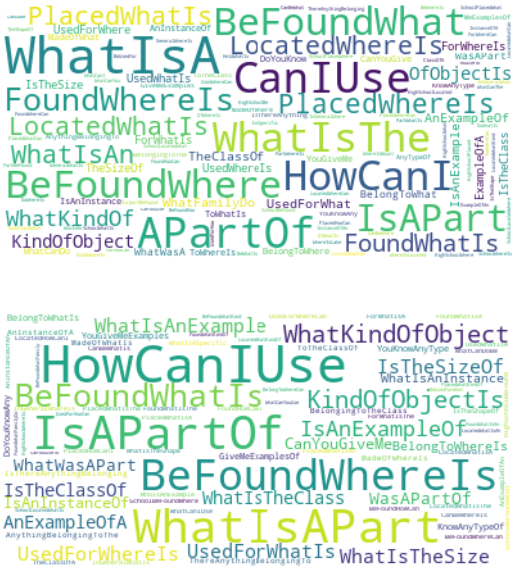

Figure 2: Wordclouds of 3-gram and 4-gram

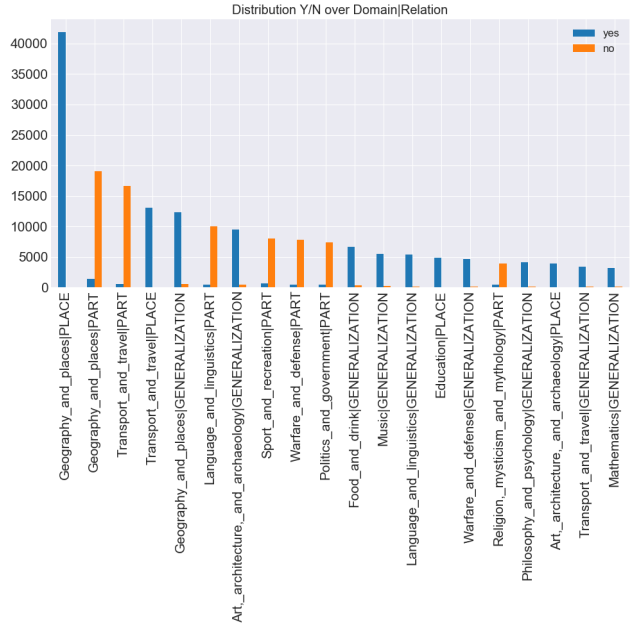

Figure 3: Y/N related to Domain-Relation pairs

(1) **Unique.** All the tuples which have associated only one answer. Assuming that, at this point, we have already discarded all the meaningless sentences, this whole set can be used in our predicting model.
(2) **Most frequent.** In this set are included all the tuple which are associated to more than one answer but, one of them is more frequent than the others. Assuming that the most recurrent answer is the correct one, we keep for each tuple only the answer with the higher frequency.
(3) **Annoying.** This tuples are quite tricky. For each one there are associated more than one answer but no one is most frequent than the others. This means that we cannot make any assumptions on the data: anyway, this data could be used in the Enriching part as shown below in the section 3.

Closed-ended questions. Analyzing the Y/N observations, we apply the same classification approach, keeping only the pairs in the Unique set ("Most frequent" results to be empty) and looking just at distribution of the total number of Yes or No, the affermative answers are twice the negative ones. Being more specific, the main concern about the quality of the data regards the answers' distribution over the possible domain-relation pairs. Given a tuple (question, domain-relation) stored in the training set, the prediction could be affected by the distribution of the Y/N answers defined by the same domain-relation pair. So, looking at the distributions over all the possible doman-relation couples, we point out that the affermative answer of the association Geography and places - PLACE is overrepresented. Since we already have over than 330000 distinct observations for the closed-ended questions, we just dropout the half of these affermative answers in order to balance the dataset.

3 WORKFLOW

Here we explain the main worflow and how the user can interacts with the bot. After starting the conversation, the user can select a domain from 5 randomly chosen topic, then he can ask a question related to that topic or reply to a question given by the bot. In both cases, either when the bot answers the question or the user reply to a question, our dialogue system asks to the user to check the general sense of the conversation just had. In this way we test the quality of the all conversations generated, tracking the possible no-sense conversations generated by the system. Being more specific about the enriching part, we generate the questions extracing them from all the pairs which we labeled before as Annoying. Since in these cases, we are not able to assign a specific unique answer, we try to improve the data, exploiting the knowledge of the users, asking them a correct answer for the query. Moreover, to identify the entity or concept produced by the answers of the users, we exploit the approach and results produced by Babelfy, a graph-based method to address the Entity Linking and Word Sense Disambiguation [5]. In this manner, we extract the concepts from the answer in order to add this observation to the KBS.

4 MODEL

Double prediction. Our bot, given a couple of topic and question, is able to predict the relation along with type of the question (open-ended multiple choice, open-ended generic and closed-ended) and after that, predicts a specific answer. The prediction is given by a simple recurrent neural network, an LSTM (Long Short Term Memory), a type of architecture which usually performs very well in sequence classification. At this point the system, based on the predicted type of question, is able to predict the answer. Indeed, for each type of question it has been trained a different neural

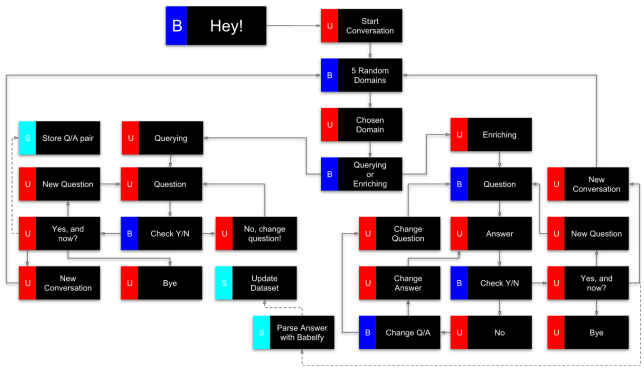

Figure 4: Chatbot - Workflow

network: for the closed-questions we used the model of the first prediction with only the Y/N pairs, while the models of the other 2 are both a Sequence-to-Sequence (Seq2Seq) model but, each one with a different approach.

4.1 1st Prediction and Closed-ended Questions

Predicting the relation-type. LSTM networks are a special kind of RNN, capable of learning long-term dependencies. In our case, the core of the model consists of an LSTM cell that processes one sentence at a time and computes probabilities of the possible pair (type of question, relation) related to the input. To represent the sentences in our NN architecture we apply the word embeddings, which provides a dense representation of words and their relative meanings, mapping the words of each sentence: each word is represented as a 100-dimension vector. The possible output is composed by all the possible combinations given by the three types of questions described before and all the possible relations. Moreover, to enable the bot to recognize relation and type of question without knowning the concepts in the question, we use as dictionary of the trainset only the Top-50% of the words distribution over the data. The model trained only after 5 epochs reachs an accuracy of more than 98%.

Closed-questions model. The model is the same used before, but with the closed-ended questions and the accuracy reached after only few epochs is more than 96%.

4.2 Seq2Seq

The Seq2Seq model, widely used in Neural Machine Translation (NMT) frameworks, is characterized by combining two recurrent neural networks (RNN), which in our case are two LSTM. The first one, called encoder, encodes a sequence of characters or words into a fixed length vector representation, and the other, the decoder, decodes the representation into another sequence of characters or words. The encoder along with the decoder are jointly trained to maximize the conditional probability of a target sequence given a source sequence: applying this NMT approach to our model, we aim to 'translate' a sentence given by a specific (question, domain-relation) tuple into a target answer. We choose to use the LSTM because of his ability to store sequential information over extended time interval (Fig 5).

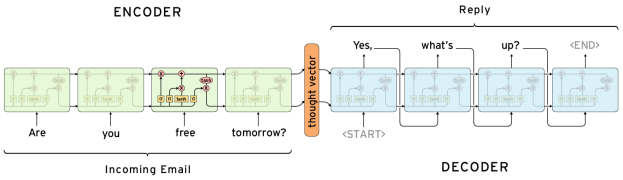

Figure 5: LSTM applied in a Seq2Seq Model

Multiple choice Model. In this model there are 3 main differences with respect to the one applied for the generic open-ended questions: (1) the sentences are mapped as sequence of words (2) the encoder LSTM reads each input sequence in reverse in order to helps the learning algorithm to establish a connection between two sequences [3] (3) for each tuple (question, domain-relation) there is associated a set of answers which are all equivalently true, this means that we are predicting not a single answer, but an ID associated to a set of answers specific to this tuple. In other words, given a tuple of (topic, domain, question) the Seq2Seq Model is used to predict a single word which stands for the ID of a specific set. The final answer is randomly selected from the predicted set and, the accuracy reached from the model after 30 epochs is more than 99%.

Generic Questions Model. Since it would be computationally expensive applying the embedding layer for all the possible words in the open-ended generic questions, our Seq2Seq Model is trained at character-level [4]. After more than 200 epochs, we evaluate the model on a sample of the train-data(10%), where the accuracy reached is over the 70%. Taking into account the presence of nonsense Q&A the model can even perform better, if trained more time with a well-defined corpus.

5 CONCLUSIONS

We have developed a knowledge-bot, a dialogue sysyem able to ask questions and predict answers on many topics, where both the tasks, querying and enreaching, have been thought exploiting as much as possible the structure of the metadata stored in the KBS. Unlike the conventional bot, our dialog system needs to improve the quality of his dataset: for this reason, a widespread use of it, through the enriching part, would improve the dataset as well as the quality of the prediction.

REFERENCES

[1] J Camacho-Collados, R Navigli, *BabelDomains: Large-Scale Domain Labeling of Lexical Resources*
[2] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
[3] I Sutskever, O Vinyals, QV Le, *Sequence to Sequence Learning with Neural Networks*
[4] blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html
[5] A Moro, A Raganato, R Navigli, *Entity Linking meets Word Sense Disambiguation: a Unified Approach*