The knowledge bot
NLP project

Abstract

In this project I present an end-to-end simple factoid question answering/question generating bot that exploits both supervised machine learning techniques like RNNs and shallow heuristics based on syntax.
The core parts learned models are able to reach pretty good accuracy level on the central knowledge base dataset as well as the combination of them in the full answer selection phase.

Introduction

Question answering techniques can be divided in two main paradigms: given an input question, *IR-based question answering* extract candidate answer strings from text documents available on the Web or in specialized knowledge collection, exploiting information retrieval techniques, whereas *knowledge-based question answering* build a semantic representation of the question (predicate calculus statement, RDF triple etc.) to query databases of facts.

The approach used in this project is focused mainly on the *knowledge-based* paradigm and on its basic ideas.

In the next sections I explain the dataset used and the methods and the learned models used in the core parts of answer selection, then there will be the non-machine learning part and finally the implementation section.

Dataset

The dataset at our disposal is the knowledge-base that we created from the third homework.
It contains all the necessary information that will be preprocessed to serve both as a training set and a database for answer selection.

At the moment of the retrieving, the dataset consisted of more than 1M records. This full dataset is queried during the answer selection phase to select a candidate answer and it is further enlarged thanks to the enriching phase and the online learning character of the project.

Technical considerations, analysis of data distributions and the huge presence of unavoidable noise have led to roughly 600K records to serve as the training set for all the learned models. In some cases it had been weighted to come up with class unbalancing.

Results from the analysis also showed the main idea behind the approach to use for the answer selection phase: increasingly narrow the possible candidates exploiting the answer type (binary vs. non-binary), the relation involved in the question, and finally the concepts. The real implementation just starts from this idea and then elaborates a bit on the consequences.

Methods and models

Let's define *QW* (*querying-workflow)* the question answering part that consist of the answer selection phase and *EW* (*enriching-worflow*) the question generating part.

The answer selection phase, actually, is better explained as a concept selection, indeed it just selects the (possible) right entry in the knowledge-base and then extracts the answer field which is (and should always be) a concept - excluding binary questions cases.
The answer-phrase generation is not part of the answer selection phase and does not exploit learned models as well as the whole EW part, therefore they will be treated in the next section.

From now on

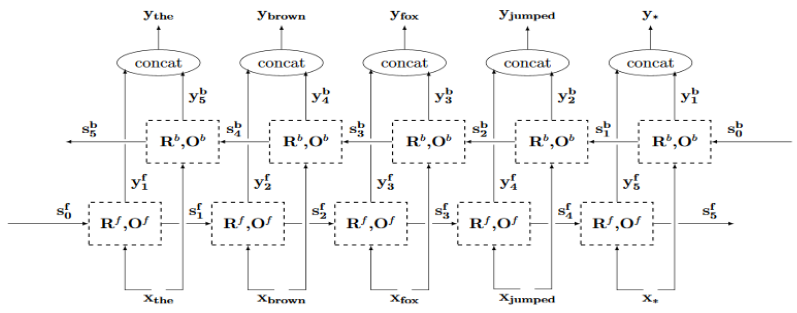
Q_C1C2 will identify the learned model that given and input question predicts the main (tokens of) the concepts involved.
Q_R will identify the learned model that given an input question predicts the relation between the concepts involved.
Q_A will identify the learned model that given an input question predicts the answer type: binary question or entity question.

Querying workflow

The answer selection phase exploits three RNNs models:

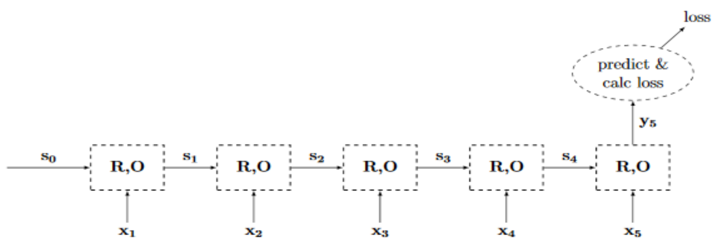
Q_C1C2

This learned model is used to predict the main tokens of the concept involved in the question. The learning task has been interpreted as a *sequence labeling* one in which given an input question (a sequence of tokens) it performs a 3-classification for each word: each class stands for "token in c1", "token in c2" and "others".
Generally, questions have both concepts in it (for binary ones) or may have just one of them which is tipically (if not only) the subject of the relation i.e.: c1.
The features involved in the training are the pre-trained words embeddings and the POS-tags fed into a BI-LSTM transducer recurrent model:


y_{the} y_{brown} y_{fox} y_{jumped} y_{*}
concat
y^b_5 y^b_4 y^b_3 y^b_2 y^b_1
s^b_5 s^b_4 s^b_3 s^b_2 s^b_1 s^b_0
R^b,O^b
y^f_1 y^f_2 y^f_3 y^f_4 y^f_5
s^f_0 s^f_1 s^f_2 s^f_3 s^f_4 s^f_5
R^f,O^f
x_{the} x_{brown} x_{fox} x_{jumped} x_{*}

Q_R

This learned model is used to predict the relation involved between the concepts which the question refers to. This time the learning task has been interpreted as a *sequence classification* one in which given an input question it performs a 16-classification among the possible existing relations.
The features involved are the words embeddings fed into an acceptor LSTM recurrent model:



Q_A

This learned model is used to predict the answer type i.e.: if the question has a binary answer (yes/no) or it has a concept answer (c). The learning task has been interepreted as a *sequence classification* in which given an input question it performs a binary classification among the possible answer type.
As before, the features are the words embeddings and the model is an acceptor LSTM.

Methods and heuristics

Enriching workflow

This workflow relates to the part in which the bot generates questions to query the user.
The question generation phase can be divided into two parts.
Given an input domain from the user,

1. selection among the concepts is performed according to a sampling strategy that favors lesser known-concepts-relation pairs in case of concept-domain existence, whereas, a random concept choice (holding domain and relation fixed) is performed in the case of non existence.
2. random selection is performed among the right question patterns at our disposal.

The received user answer preprocessing phase used to extract the concept from the answer is handled through shallow heuristics based on the syntactic structure of the answer itself.

Answer-phrase generation

The answer-phrase generation mimics the above steps, the main difference is that among the possible answer structures a shallow similarity measure between the question from the user and the answer-phrase structure is applied for ranking.

Bot implementation aspects

Online learning

To handle the online learning character of the whole project, at a fixed time the bot checks if there is enough knowledge in the online knowledge-base to justify a retraining.
If that is the case, data are downloaded, inserted into the local knowledge database and preprocessed for the retraining. The new training set will be constituted of this last downloaded knowledge plus the old one.
The bot alerts the start and the end of the retraining phase and does not answer any question in the meanwhile.

Workflows handling

QW and EW are randomly interchanged according to an adaptive, computed sampling probability distribution (based on simple counts) which favors the less encountered workflow at each turn.

Wikipedia style disambiguation and conversation turns

To allow for a more natural conversation and to come up with the problem of non-domained concepts the QW has been slightly modified: there is no need to give a domain as input during the first step, therefore the user can directly make questions which will be disambiguated in a Wikipedia style (if possible) and ranked according to senses predominance.
A panel with possible domains or a confirmation in case of only one possible domain, will then be visualized.
Priority is given to succesfully disambiguated and domained concept, therefore non disambiguated concept will be mapped into the prevalent disambiguated one, however, it is always possible to make the bot guess the right answer choosing disambiguated but undomained concepts.

Conclusions and future work

The learned models perform quite well both individually on the dataset and when combined for the answer selection. Roughly speaking, assuming independent predictions, it can be said that the overall combined test accuracy is the product of the tree accuracies, which indeed is roughly 95%.
Indeed it is not too far from the estimated combined test accuracy.
Both Q_R and Q_A are equally accurate (98.3% and 98.9%) nevertheless, among the tree the least performing model is Q_C1C2 which indeed constitutes the hardest part of the prediction; its test accuracy, at a sentence level, is roughly 97%.
It is quite understandable given that concepts are almost always made up of more than one token and very often include non-noun words.

Therefore a first aspect to improve is surely the concepts prediction phase itself (Q_C1C2) maybe using different sets of features or trying to interpret the learning task as another paradigm.
Heuristics also might be helpful to correct and align the prediction but, as usual, especially in this case, are hard to obtain from manually crafted pattern recognition.
A second aspect that must be improved and on which I will spend future time is the implementation of a sequence 2 sequence model for the answer phrase generation. The lack of a real dataset and technical limitations did not allowed me to focus on that direction.
Answer extraction techniques from Wikipedia documents also will be considered in the future.