Generating natural language answers

Madhav Sharan
msharan@stanford.edu

Abstract

Question answering systems based on knowledge base takes a natural language query and generate facts like <subject,predicate,object> which can be used to answer the query. However answer presentation is usually left as a downstream task. In this work we try to generate natural language answers using a sequence to sequence network which can be trained as a single model. This task is very challenging due to multiple reasons like finding right entity, extracting correct facts, generating grammatically correct answer etc. What we found is that if we restrict the problem by providing the system with golden entities it becomes much easier but it still struggles with issues like using correct gender pronouns. What we also found is the lot of information needs to be copied from question to directly in answer like entity name. Main contributions of this effort is demonstrating that given a dataset of natural questions and answers we can make a end to end model which generates natural language answer. Work is mainly done on chinese but since there is no language specific component it should work on other language datasets as well.

1 Introduction

In a question answering task like SQUAD, system is presented with a text query and a paragraph which might have answer to the query. This system is expected to highlight answer to the query in provided passage or output a span which should answer the query. There are other variants of question answering task like OpenDomain question answering where system searches for answer in wikipedia paragraphs instead of taking a paragraph as input. KB (Knowledge Graphs) are data-stores that can store information in <S,P,O> (<Subject, Predicate, Object>) which can be used to represent a fact. Here subject can be an entity like country India, predicate can be an attribute of country like national motto and Object can be another entity or a value like Satyamev Jayate. A question answering system based on KB takes a query and generate <S,P,O> which can be used to answer the query. In all these approaches provided answer is either structured response or a wider output which should answer the question. This paper attempts to generate a natural language answer like "National motto of India is Satyamev Jayate" for a question "What is the national motto of India?" . This task is very challenging as answer needs to be fluent, it should take care of using entity specific language construct like gender specific pronoun etc.

2 Related work

There are some related work which generate text from a given content like [1] Puduppully et al which try to generate description of a data table. There are papers related to question generation like [2] Zero shot question generation like from Elsahar et al, which takes an answer and entity and generate questions. Elsahar et al did regex matching to collect answers like sentences from wikipedia text for a given subject and object. Then they feeded

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

that fact and a set of training questions to train a seq to seq network which takes a facts and answer and generate questions. This task was aimed to generate set of question dataset for any given fact. Task described in this paper aims at generating answers given question and set of related facts.

3 Approach

Given a question, answer pair and a set of facts about entity present in question and answer we attempt to learn mapping from questions and facts to answers. Figure 1 describe full model which we will describe in parts in below subsections. To summarize model takes a natural language question encodes it using a RNN and applies attention on encoded vector. Model also takes set of facts and encode each fact into a fixed size embedding and applies attention on set of facts. Decoder receives both encoded question and facts and decided whether to copy from question, or refer to fact or generate a altogether new token from what it has learned.

One assumption to be noted is that model expects that relevant facts are feeded with question. It does not attempts to find facts related to possible entity spans in question but instead assume that training and test set have annotated set of entities.

3.1 Encoder

Encoder takes input and learns a fixed size sequence which can be fed into subsequent network. In CoreQA[1] approach we use two encoders one for question and other for facts, both of them are explained in below subsections.

3.1.1 Question Encoder

A natural language question is split into tokens and then a bi directional RNN is used to transform them into a fixed sequence vector which is concatenation of hidden states of both forward and backward RNN. If forward and backward hidden layers are represented as $\{\vec{h_1}, ..., \vec{h_m}\}$ and $\{\overleftarrow{h_m}, ..., \overleftarrow{h_1}\}$ where m is length of tokens question Q then encoded representation of q can be written as $[\vec{h_m}, \overleftarrow{h_1}]$

3.1.2 Fact Encoder

Let's denote the subject, property and object of one fact f as s, p and o, then we can form e_s, e_p and e_o as individual embeddings of subject, property and object. One fact is then represented by concatenating e_s, e_p and e_o. List of all relevant facts become $\{f\} = \{f_1, ..., f_n\}$ where is n is total number of facts. This is refered as KB through out the paper and also in Figure 1.

In addition to learning distributed representation of question and candidate facts, we define the matching scores function between question and facts as matching function defined by a two-layer perceptron. This function takes question and one fact and is used while updating final output state.

3.2 Decoder

Decoder takes in Q and KB described above and tries to generate natural language tokens based on those, below approaches differentiates it with generic decoder.

3.2.1 Answer words prediction

Final output token is decided in a probabilistic way from below three mode, ***predict mode***, ***copy mode*** and ***retrieve mode***. In ***predict mode*** output is chosen from target vocabulary, in ***copy mode*** token is chosen from question and in ***retrieve mode*** it's chosen from KB. It's done this way as generated word could be in question or facts or could be generated from vocab to make linguistic sense. At each step final generated word is decided by adding

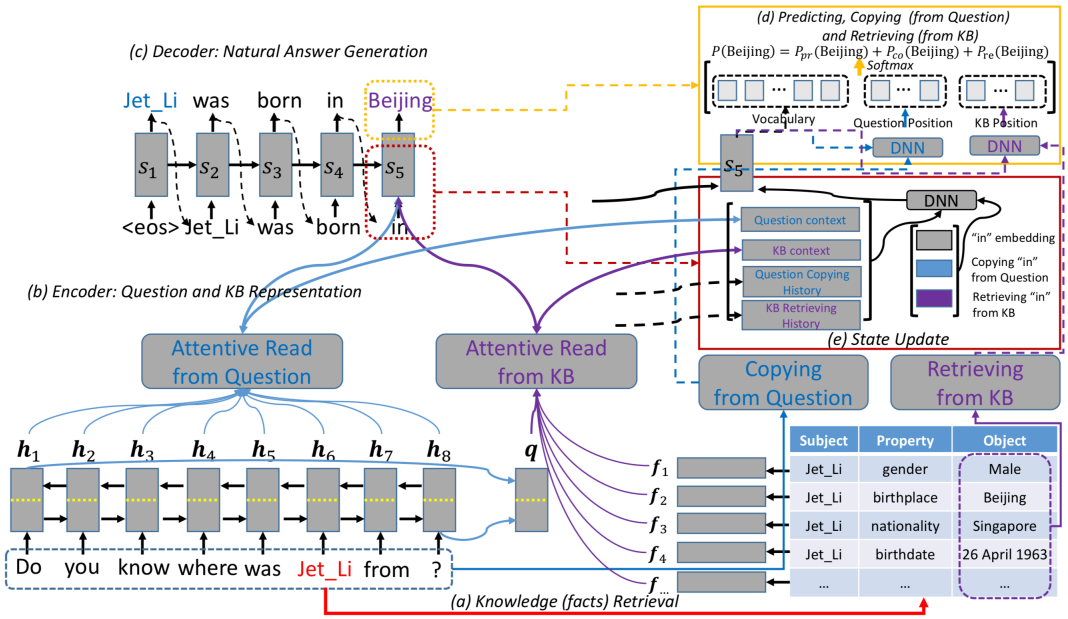

Figure 1: Model description

$\sum_{mode->copy,predict,retrieve} p_{mode}(y_t|s_t, y_{t-1}, c_t).p_m(mode|s_t, y_{t-1})$

where p_m is the probability function to compute probability mode being copy, predict or retrieve.

3.2.2 State update

Like usual decoder predicted word at step t − 1 is used to update state t, but model does not uses just word embedding but also attention outputs of Q and KB. This is done because final word will not be always from vocabulary. More specifically, y_{t-1} will be represented as concatenated vector of $[e(y_{t-1}), r_{q_{t-1}}, r_{kb_{t-1}}]$, where $e(y_{t-1})$ is the word embedding associated with y_{t-1}, $r_{q_{t-1}}$ and $r_{kb_{t-1}}$ are the weighted sum of hidden states in Q and KB corresponding to y_{t-1} respectively.

3.2.3 Reading short-Memory Q and KB

Q and KB are fed in both ways the encoded form to represent meaning and also attention scores which tell the positional information of tokens in question and <s,p,o> in fact.

4 Experiments

For first milestone I have understood the training data and evaluation metrics. I have first few layers ready and I am working on next few in coming weeks. In my analysis I tried to objectively look at training data and understand variation. Authors were kind to open source training and evaluating data which is in chinese and I'll be describing that in below sections. I have already written code for -

- Tokenizing chinese sentences into sub words using jieba tool[2]. There are some options available like doing word tokenization or sub word.
- Converting data in multiple formats to a single format.
- Making embedding layers from question and facts.
- Evalauting synthetic dataset.

I also spent some time to look for data in english, closest I could find is WikiAnswers [3] in which each row has *a list of question* a *answer* and it's accompanied by another dataset having knowledge base.

4.1 Data description

4.1.1 Synthetic data

We have two datasets available one of them is synthetic and created by hand crafted set of approximately 100 question answer pattern. Example from paper is *When is %e birthday? -> She was born in %m %dth* where the variables %e, %y, %m, %d and %g (deciding she or he) indicates the person' s name, birth year, birth month, birth day and gender, respectively. KB is randomly generated which has only birth year, birth month, birth day and gender facts for 80,000 entities. Below is some data which is finally provided in original text, here *<ent_1>* denotes entity

Question and answer pair -

<ent_1>是多会出生的　　<ent_1>是1963年10月20日出生的
<ent_1>的生日是几号　　<ent_1>的生日是10月20号
<ent_1>出生于几月　　<ent_1>在10月出生

KB facts about *<ent_1>* -

<ent_1> 出生年　1963
<ent_1> 出生月　10
<ent_1> 出生日　20
<ent_1> 性别　男

Finally we have 320,000 KB facts and 239,922 synthetic question and answer pairs.

4.1.2 Open Domain dataset

Authors of [1] found GenQA dataset [4] which is based on a community QA website. KB is constructed by crawling web pages and a set of fixed question answer pairs. This data is very noisy authors quote However, the original Q-A pairs only matched with just one single fact. In fact, we found that a lot of questions need more than one fact (about 20% based on sampling inspection). We also found by spot checking that some Q-A pairs and facts were unrelated and many questions were not factoid.

Some statistics about dataset are

- Total rows - 505,021
- Maximum character in question 13
- Maximum character in answer 10
- Total unique **subject** 24,216
- Total unique **predicate** 3,271
- Total unique **object** 23,921

#of QA pair with 1 fact	418,615
#of QA pair with 2 facts	681,69
#of QA pair with 3 facts	13,629
#of QA pair with 4 facts	3104
#of QA pair with 5 facts	921
#of QA pair with 6 facts	283
#of QA pair with 7 facts	273
#of QA pair with 8 facts	15
#of QA pair with 9 facts	5
#of QA pair with 10 facts	6

4.2 Baseline model

As a baseline we used similar model like assignment 4 and feeded question as source and answer as target. This baseline will show how well a sequence to sequence model will do without a KB and with no operation of predict, retrieve and copy.

4.3 Evaluation

We will rely on automatic evaluation for our problem. As paper suggests in automatic evaluations we check if gender is predicted correctly, if answer is able to retrieved from KB correctly as in object from KB is present in generated answer.

This evalaution is borrowed from paper [1] apart from this we also want to get some sense of fluency through automatoc evaluation. For this we will use dataset BLEU score for which we already have code from assignment 4.

5 Results

Due to limited time we could only train on synthetic datset. Author hope to work on this post course completion. Here $P(gender)$ denotes how many time model used correct *gender* terms, this was annotated in dataset so we end up calculating true positives / total positives. Same applies for *year*.

Automatic eval

		P(gender)	P(year)
Baseline	–	70.1	10
CoreQA impl	–	86.3	69.1

This worked below expectation as paper clearly showed approximately 15% better results. This could due to bug in our implementation or less carefully tuned network.

6 Analysis

This was very complex to implement model as it just not uses a sequence to sequence but passes on outputs of intermediate layer to many other layers. Given data was available in chinese we learned how to go language independent and trusted more on math that print examples. We realized that hyperparameter tuning is very difficult, For sake of keeping experiments simple we kept embedding size of question and fact constant and did not use multilayer RNN. We kept changing learning rate, batch size and tried to use less number of facts and see how it affects performance. We looked at scraped data and found it very noisy like many examples had unrelated facts, many questions were not factoids and many answers were statemnets which were not related.

7 Conclusion

It's very possible to use end to end model to generate question and answer but it's hard to train. Obtaining good training data is very difficult and scarped data has lot of noise.

8 References

[1] Data-to-Text Generation with Content Selection and Planning Ratish Puduppully et al

[2] Zero-Shot Question Generation from Knowledge Graphs for Unseen Predicates and Entity Types Hady Elsahar et al

[3] Generating Natural Answers by Incorporating Copying and Retrieving Mechanisms in Sequence-to-Sequence Learning. Shizhu He1, Cao Liu1,2, Kang Liu1 and Ju 1, http://www.nlpr.ia.ac.cn/cip/shizhuhe/articles/acl2017-coreqa.pdf

[4] jieba for tokenizing chinese https://github.com/fxsjy/jieba

[5] https://github.com/afader/oqa#wikianswers-corpus

[6] https://github.com/jxfeb/Generative QA