Question Answering from Unstructured Text by Retrieval and Comprehension

Yusuke Watanabe[1,2], Bhuwan Dhingra[2], and Ruslan Salakhutdinov[2]

[1]Sony Corporation
[2]School of Computer Science, Carnegie Mellon University
{ywatanab, bdhingra, rsalakhu}@cs.cmu.edu

Abstract

Open domain Question Answering (QA) systems must interact with external knowledge sources, such as web pages, to find relevant information. Information sources like Wikipedia, however, are not well structured and difficult to utilize in comparison with Knowledge Bases (KBs). In this work we present a two-step approach to question answering from unstructured text, consisting of a *retrieval* step and a *comprehension* step. For comprehension, we present an RNN based attention model with a novel mixture mechanism for selecting answers from either retrieved articles or a fixed vocabulary. For retrieval we introduce a hand-crafted model and a neural model for ranking relevant articles. We achieve state-of-the-art performance on WIKIMOVIES dataset, reducing the error by 40%. Our experimental results further demonstrate the importance of each of the introduced components.

1 Introduction

Natural language based consumer products, such as Apple Siri and Amazon Alexa, have found wide spread use in the last few years. A key requirement for these conversational systems is the ability to answer factual questions from the users, such as those about movies, music, and artists.

Most of the current approaches for Question Answering (QA) are based on structured Knowledge Bases (KB) such as Freebase (Bollacker et al., 2008) and Wikidata (Vrandečić and Krötzsch, 2014). In this setting the question is converted to a logical form using semantic parsing, which is queried against the KB to obtain the answer (Fader et al., 2014; Berant et al., 2013).

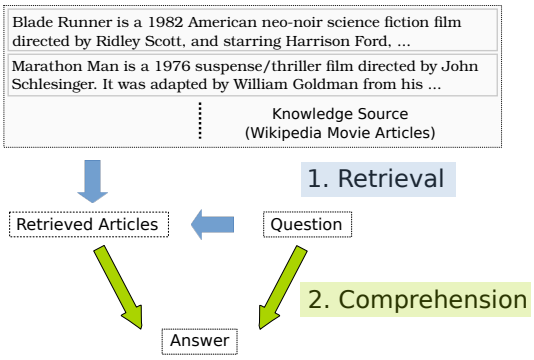

Figure 1: Overview of a retrieval + comprehension (r+c) QA system. First, movie articles relevant to a question are retrieved. Then, the retrieved articles along with the question are processed to obtain an answer.

However, recent studies have shown that even large curated KBs, such as Freebase, are incomplete (West et al., 2014). Further, KBs support only certain types of answer schemas, and constructing and maintaining them is expensive.

On the other hand, there is a vast amount of unstructured knowledge available in textual form from web pages such as Wikipedia, and hence an alternative is to directly answer questions from these documents. In this approach, shown in Figure 1, articles relevant to the question are first selected (*retrieval* step). Then, the retrieved articles and question are jointly processed to extract the answer (*comprehension* step). This retrieval based approach has a longer history than the KB based approach (Voorhees and Tice, 2000). It can potentially provide a much wider coverage over questions, and is not limited to specific answer schemas. However, there are still gaps in its performance compared to the KB-based approach (Miller et al., 2016). The comprehension step, which requires parsing information from natural language, is the main bottleneck, though suboptimal retrieval can also lead to lower performance.

Several large-scale datasets introduced recently (Rajpurkar et al., 2016; Hermann et al., 2015) have

A Funny Man is a 2011 Danish drama film directed by Martin Zandvliet about the Danish actor and comedian Dirch Passer.

Q. Martin Zandvliet directed which movies?
A. A Funny Man

Koi... Mil Gaya is a 2003 Bollywood science fiction film directed by Rakesh Roshan (who also has a cameo role), starring Hrithik Roshan, Rekha and Preity Zinta. The film's theme is largely inspired by the 1982 Hollywood hit "E.T. the Extra-Terrestrial."

Q. what language is Koi... Mil Gaya in?
A. Hindi, English

Figure 2: Example of comprehension step from WIKIMOVIES dataset. *Top*: answer is a span of text in article. *Bottom*: answer is not explicitly written in article.

Question	Answers
who directed the movie Blade Runner?	Ridley Scott
what movies can be described by mariah carey?	Precious, Glitter
what kind of film is The Hitman?	Action, Crime

Table 1: Example of questions and answers.

facilitated the development of powerful neural models for reading comprehension. These models fall into one of two categories: (1) those which extract answers as a span of text from the document (Dhingra et al., 2016; Kadlec et al., 2016; Xiong et al., 2016) (Figure 2 top); (2) those which select the answer from a fixed vocabulary (Chen et al., 2016; Miller et al., 2016) (Figure 2 bottom). Here we argue that depending on the type of question, either (1) or (2) may be more appropriate, and introduce a latent variable mixture model to combine the two in a single end-to-end framework.

We incorporate the above mixture model in a simple Recurrent Neural Network (RNN) architecture with an attention mechanism (Bahdanau et al., 2015) for comprehension. In the second part of the paper we focus on the retrieval step for the QA system, and introduce a neural network based ranking model to select the articles to feed the comprehension model. We evaluate our model on WIKIMOVIES dataset, which consists of 200K questions about movies, along with 18K Wikipedia articles for extracting the answers. Miller et al. (2016) applied Key-Value Memory Neural Networks (KV-MemNN) to the dataset, achieving 76.2% accuracy. Adding the mixture model for answer selection improves the performance to 85.4%. Further, the ranking model improves both precision and recall of the retrieved articles, and leads to an overall performance of 85.8%.

2 WIKIMOVIES Dataset

We focus on the WIKIMOVIES[1] dataset, proposed by (Miller et al., 2016). The dataset consists of pairs of questions and answers about movies. Some examples are shown in Table 1.

As a knowledge source approximately 18K articles from Wikipedia are also provided, where each article is about a movie. Since movie articles can be very long, we only use the first paragraph of the article, which typically provides a summary of the movie. Formally, the dataset consists of question-answer pairs $\{(q_j, A_j)\}_{j=1}^{J}$ and movie articles $\{d_k\}_{k=1}^{K}$. Additionally, the dataset includes a list of entities: movie titles, actor names, genres etc. Answers to all the questions are in the entity list. The questions are created by human annotators using SimpleQuestions (Bordes et al., 2015), an existing open-domain question answering dataset, and the annotated answers come from facts in two structured KBs: OMDb[2] and MovieLens[3].

There are two splits of the dataset. The "Full" dataset consists of 200K pairs of questions and answers. In this dataset, some questions are difficult to answer from Wikipedia articles alone. A second version of the dataset, "Wiki Entity" is constructed by removing those QA pairs where the entities in QAs are not found in corresponding Wikipedia articles. We call these splits WIKIMOVIES-FL and WIKIMOVIES-WE, respectively. The questions are divided into train, dev and test such that the same question template does not appear in different splits. Further, they can be categorized into 13 categories, including `movie_to_actors`, `director_to_movies`, etc.[4] The basic statistics of the dataset are summarized in Table 2.

We also note that more than 50% of the entities appear less than 5 times in the training set. This makes it very difficult to learn the global statistics of each entity, necessitating the need to use an external knowledge source.

3 Comprehension Model

Our QA system answers questions in two steps, as shown in Figure 1. The first step is *retrieval*,

[1] http://fb.ai/babi
[2] http://beforethecode.com/projects/omdb/download.aspx
[3] http://grouplens.org/datasets/movielens/
[4] Category labels are only available for dev/test dataset

# of questions (train/dev/test)	
FL train/dev/test	196453/10K/10K
WE train/dev/test	96185/10K/10K
Avg. # words in question	7.7
Avg. # of answers	1.9
# of movie articles	18127
Avg. # words in article	90.9
Vocabulary	61696
# of entities	71996

Table 2: Basic statistics of WIKIMOVIES dataset.

where articles relevant to the question are retrieved. The second step is *comprehension*, where the question and retrieved articles are processed to derive answers.

In this section we focus on the comprehension model, assuming that relevant articles have already been retrieved and merged into a *context* document. In the next section, we will discuss approaches for retrieving the articles.

Miller et al. (2016), who introduced WIKIMOVIES dataset, used an improved variant of Memory Networks called Key-Value Memory Networks. Instead, we use RNN based network, which has been successfully used in many reading comprehension tasks (Kadlec et al., 2016; Dhingra et al., 2016; Chen et al., 2016).

WIKIMOVIES dataset has two notable differences from many of the existing comprehension datasets, such as CNN and SQuAD (Kadlec et al., 2016; Dhingra et al., 2016; Chen et al., 2016). First, with imperfect retrieval, the answer may not be present in the context. We handle this case by using the proposed mixture model. Second, there may be multiple answers to a question, such as a list of actors. We handle this by optimizing a sum of the cross-entropy loss over all possible answers.

We also use attention sum architecture proposed by Kadlec et al. (2016), which has been shown to give high performance for comprehension tasks. In this approach, attention scores over the context entities are used as the output. We term this the attention distribution p_{att}, defined over the entities in the context. The mixture model combines this distribution with another output probability distribution p_{vocab} over all the entities in the vocabulary. The intuition behind this is that named entities (such as actors and directors) can be better handled by the attention part, since there are few global statistics available for these, and other entities (such as languages and genres) can be captured by vocabulary part, for which global statistics can be leveraged.

3.1 Comprehension model detail

Let $\mathcal{V}$ be the vocabulary consisting of all tokens in the corpus, and $\mathcal{E}$ be the set of entities in the corpus The question is converted to a sequence of lower cased word ids, $(w_i) \in \mathcal{V}$ and a sequence of 0-1 flags for word capitalization, $(c_i) \in \{0, 1\}$. For each word position i, we also associate an entity id if the i-th word is part of an entity, $e_i \in \mathcal{E}$ (see Figure 3). Then, the combined embedding of the i-th position is given by

$$x_i = W_w(w_i) + W_c(c_i) \| W_e(e_i), \quad (i = 1, \ldots, L_q), \tag{1}$$

where $\|$ is the concatenation of two vectors, L_q is the number of words in a question q, and W_w, W_c and W_e are embedding matrices. Note that if there are no entities at i-th position, $W_e(e_i)$ is set to zero. The context is composed of up to M movie articles concatenated with a special separation symbol. The contexts are embedded in exactly the same way as questions, sharing the embedding matrices.

To avoid overfitting, we use another technique called *anonymization*. We limit the number of columns of W_e to a relatively small number, n_e, and entity ids are mapped to one of n_e columns randomly (without collision). The map is common for each question/context pair but randomized across pairs. The method is similar to the anonymization method used in CNN / Daily Mail datasets (Hermann et al., 2015). Wang et al. (2016) showed that such a procedure actually helps readers since it adds coreference information to the system.

Next, the question embedding sequence (x_i) is fed into a bidirectional GRU (BiGRU) (Cho et al., 2014) to obtain a fixed length vector v

$$v = \overrightarrow{h}_q(L_q) \| \overleftarrow{h}_q(0), \tag{2}$$

where $\overrightarrow{h}_q$ and $\overleftarrow{h}_q$ are the final hidden states of forward and backward GRUs respectively. 上下文拼接的数目

The context embedding sequence is fed into another BiGRU, to produce the output $H_c = [h_{c,1}, h_{c,2}, \ldots h_{c,L_c}]$, where L_c is the length of the context. An attention score for each word position i is given by

$$s_i \propto \exp(v^T h_{c,i}). \tag{3}$$

The probability over the entities in the context is then given by

$$p_{att}(e) \propto \sum_{i \in I(e,c)} s_i, \tag{4}$$

问题经过BiGRU编码后与上下文BiGRU编码输出的相内积, 得到attention score 的概率分布

词汇表(小写)的id,是否大写的标记,实体表的id

	who	directed	the	movie	Blade	Runner
$W_w(w_i)$	W_w(who)	W_w(directed)	W_w(the)	W_w(movie)	W_w(blade)	W_w(runner)
$W_c(c_i)$	$W_c(0)$	$W_c(0)$	$W_c(0)$	$W_c(0)$	$W_c(1)$	$W_c(1)$
$W_e(e_i)$	0	0	0	0	W_e(Blade Runner)	W_e(BladeRunner)

Figure 3: Example of embedded vectors for a question "who directed the movie Blade Runner?"

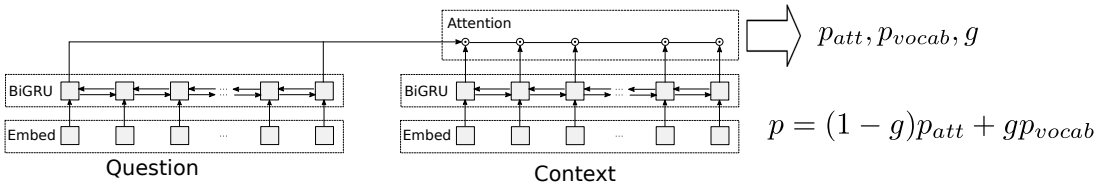

Figure 4: Visualization of our model. A question is encoded to a vector by a BiGRU. With this vector, attention is computed over another BiGRU. Output probabilities p_{att}, p_{vocab} and the mixture coefficient g are computed from those attentions and BiGRU states.

where $I(e, c)$ is the set of word positions in the entity e within the context c.

We next define the probability p_{vocab} to be the probability over the complete set of entities in the corpus, given by

词汇表中实体作为答案的概率

$$p_{vocab}(e) = \text{Softmax}(Vu), \quad (5)$$

where the vector u is given by $u = \sum_i s_i h_{c,i}$. Each row of the matrix V is the coefficient vector for an entity in the vocabulary. It is computed similar to Eq. (1).

$$V(e) = \sum_{w \in e} W_w(w) + \sum_{c \in e} W_c(c) \| W_e(e). \quad (6)$$

The embedding matrices are shared between question and context.

The final probability that an entity e answers the question is given by the mixture $p(e) = (1-g)p_{att}(e) + gp_{vocab}(e)$, with the mixture coefficient g defined as

$$g = \sigma(W_g g_0), \quad g_0 = v^T u \| \max Vu. \quad (7)$$

The two components of g_0 correspond to the attention part and vocabulary part respectively. Depending on the strength of each, the value of g may be high or low.

Since there may be multiple answers for a question, we optimize the sum of the probabilities:

$$\text{loss} = -\log \Big(\sum_{a \in A_j} p(a|q_j, c_j) \Big) \quad (8)$$

Our overall model is displayed in Figure 4.

We note that KV-MemNN (Miller et al., 2016) employs "Title encoding" technique, which uses the prior knowledge that movie titles are often in answers. Miller et al. (2016) showed that this technique substantially improves model performance by over 7% for WIKIMOVIES-WE dataset. In our work, on the other hand, we do not use any data specific feature engineering.

4 Retrieval Model

Our QA system answers questions by two steps as in Figure 1. Accurate retrieval of relevant articles is essential for good performance of the comprehension model, and in this section we discuss three approaches for it. We use up to M articles as context. A baseline approach for retrieval is to select articles which contain at least one entity also present in the question. We identify maximal intervals of words that match entities in questions and articles. Capitalization of words is ignored in this step because some words in the questions are not properly capitalized. Out of these (say N) articles we can randomly select M. We call this approach (r0). For some movie titles, however, this method retrieves too many articles that are actually not related to questions. For example, there is a movie titled "Love Story" which accidentally picks up the words "love story". This degrades the performance of the comprehension step. Hence, we describe two more retrieval models – (1) a dataset specific hand-crafted approach, and (2) a general learning based approach.

4.1 Hand-Crafted Model (r1)

In this approach, the N articles retrieved using entity matching are assigned scores based on certain heuristics. If the movie title matches an entity in the question, the article is given a high score, since it is very likely to be relevant. A similar heuristic was also employed in (Miller et al., 2016). In addi-

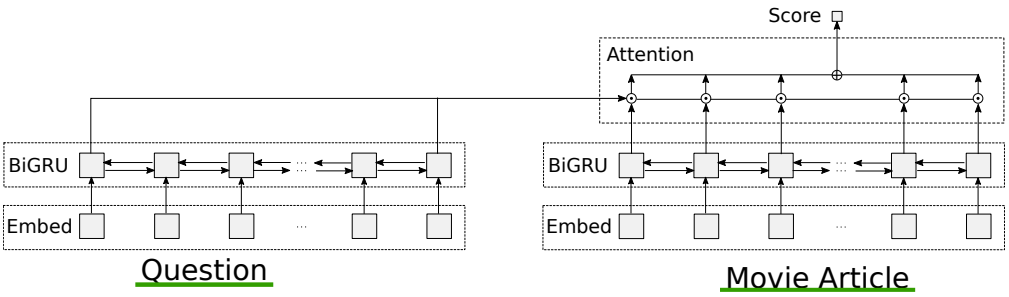

Figure 5: Overview of retrieval model. Similar to the comprehension model, a question is encoded to a fixed length vector. Attention is computed over the words of the movie article.

tion, the number of matching entities is also used to score each article. The top M articles based on these scores are selected for comprehension. This hand-crafted approach already gives strong performance for the WIKIMOVIES dataset, however the heuristic for matching article titles may not be appropriate for other QA tasks. Hence we also study a general learning based approach for retrieval.

4.2 Learning Model (R2)

The learning model for retrieval is trained by an oracle constructed using distant supervision. Using the answer labels in the training set, we can find appropriate articles that include the information requested in the question. For example, for `x_to_movie` question type, the answer movie articles are the correct articles to be retrieved. On the other hand, for questions in `movie_to_x` type, the movie in the question should be retrieved. Having collected the labels, we train a retrieval model for classifying a question and article pair as relevant or not relevant.

Figure 5 gives an overview of the model, which uses a Word Level Attention (WLA) mechanism. First, the question and article are embedded into vector sequences, using the same method as the comprehension model. We do not use anonymization here, to retain simplicity. Otherwise, the anonymization procedure would have to be repeated several times for a potentially large collection of documents. These vector sequences are next fed to a Bi-GRU, to produce the outputs v (for the question) and H_c (for the document) similar to the previous section.

To classify the article as relevant or not, we introduce a novel attention mechanism to compute the score.

$$s = \sum_i ((w\tilde{v} + b)^T \tilde{h}_{c,i})^4 \quad (9)$$

Each term in the sum above corresponds to the match between the query representation and a token in the context. This is passed through a 4-th order non-linearity so that relevant tokens are emphasized more[5]. Next, we compute the probability that the article is relevant using a sigmoid:

$$o = \sigma(w's + b') \quad (10)$$

In the above, $\tilde{x}$ is the normalized version (by L2-norm) of vector x, w, b, w', b' are scalar learnable parameters to control scales.

5 Experiments

We evaluate the comprehension model on both WIKIMOVIES-FL and WIKIMOVIES-WE datasets. The performance is evaluated using the accuracy of the top hit (single answer) over all possible answers (all entities). This is called hits@1 metric.

For the comprehension model, we use embedding dimension 100, and GRU dimension 128. We use up to $M = 10$ retrieved articles as context. The order of the articles are randomly shuffled for each training instance to prevent over-fitting. The size of the anonymized entity set n_e is 600, since in most of the cases, number of entities in a question and context pair is less than 600.

For training the comprehension model, the Adam (Kingma and Ba, 2015) optimization rule is used with batch size 32. We stop the optimization based on dev-set performance, and training takes around 10 epochs. For WIKIMOVIES-FL (resp. WIKIMOVIES-WE) dataset, each epoch took approximately 4 (resp. 2) hours on an Nvidia GTX1080 GPU.

For training the retrieval model R2, we use a binary cross entropy objective. Since most articles are not relevant to a question, the ration of positive and negative samples is tuned to $1 : 10$. Each

[5] We use exponent $d = 4$ here. Higher d tend to have better performance. Empirically, this approach works better than exponential and softmax non-linearities.

Model Type	R@1	R@10	R@30	R@100	P@1	P@10	P@30	P@100
Entity Matching Baseline (r0)	0.733	0.937	0.963	0.985	0.642	0.917	0.958	0.983
Entity Matching + Rule (r1)	0.942	0.994	0.998	0.999	0.827	0.979	0.996	0.999
Entity Matching + WLA (R2)	0.957	0.997	0.999	0.999	0.835	0.986	0.999	0.999

Table 3: Performance of retrieval methods. (WikiMovies-WE)

Model Type	R@1	R@10	R@30	R@100	P@1	P@10	P@30	P@100
Word Level Attention	0.800	0.986	0.990	0.993	0.684	0.968	0.984	0.988
Sum of hidden state	0.530	0.817	0.860	0.900	0.467	0.786	0.825	0.865
Query Free Attention	0.628	0.833	0.873	0.909	0.556	0.798	0.835	0.873

Table 4: Performance of scoring models

	WIKIMOVIES-WE			WIKIMOVIES-FL		
	r0	r1	R2	r0	r1	R2
KV-MemNN	76.2			-		
Vocab Model (V)	77.5	81.0	81.9	54.2	55.8	57.5
Attention Model (A)	78.1	82.6	82.9	42.8	45.2	45.1
Attention+Vocab Model (AV)	79.4	83.4	85.1	58.2	60.4	60.9
Attention+SubVocab Model (AsV)	81.0	85.4	85.8	59.9	61.9	62.2

Table 5: Performance (hits@1) comparison over different models and datasets.

epoch for training the retrieval model takes about 40 minutes on an Nvidia GTX1080 GPU.

5.1 Performance of Retrieval Models

We evaluate the retrieval models based on precision and recall of the oracle articles. The evaluation is done on the test set. R@k is the ratio of cases where the highest ranked oracle article is in the top k retrieved articles. P@k is the ratio of oracle articles which are in the top k retrieved results. These numbers are summarized in Table 3. We can see that both (r1) and (R2) significantly outperform (r0), with (R2) doing slightly better. We emphasize that (R2) uses no domain specific knowledge, and can be readily applied to other datasets where articles may not be about specific types of entities.

We have also tested simpler models based on inner product of question and article vectors. In these models, a question q_j and article d_k are converted to vectors $\Phi(q_j), \Psi(d_k)$, and the relevance score is given by their inner product:

$$\text{score}(j,k) = \Phi(q_j)^T \Psi(d_k). \qquad (11)$$

In the view of computation, those models are attractive because we can compute the article vectors offline, and do not need to compute the attention over words in the article. Maximum Inner Product Search algorithms may also be utilized here (Chandar et al., 2016; Auvolat et al., 2015). However, as shown in upper block of Table 4, those models perform much worse in terms of scoring. The "Sum of Hidden State" and "Query Free Attention" models are similar to WLA model, using BiGRUs for question and article. In both of those models, $\Phi(q)$ is defined the same way as WLA model, Eq (2). For the "Sum of Hidden States" model, $\Psi(d)$ is given by the sum of BiGRU hidden states. This is the same as the proposed model by replacing the fourth order of WLA to one. For the "Query Free Attention" model, $\Psi(d)$ is given by the sum of BiGRU hidden states.

We compare our model and several ablations with the KV-MemNN model. Table 5 shows the average performance across three evaluations. The (V) "Vocabulary Model" and (A) "Attention Model" are simplified versions of the full (AV) "Attention and Vocabulary Model", using only p_{vocab} and p_{att}, respectively. Using a mixture of p_{att} and p_{vocab} gives the best performance.

Interestingly, for WE dataset the Attention model works better. For FL dataset, on the other hand, it is often impossible to select answer from the context, and hence the Vocab model works better.

The number of entities in the full vocabulary is 71K, and some of these are rare. Our intuition to use the Vocab model was to only use it for common entities, and hence we next constructed a smaller vocabulary consisting of all entities which appear at least 10 times in the corpus. This results in a subset vocabulary $\mathcal{V}_S$ of 2400 entities. Using

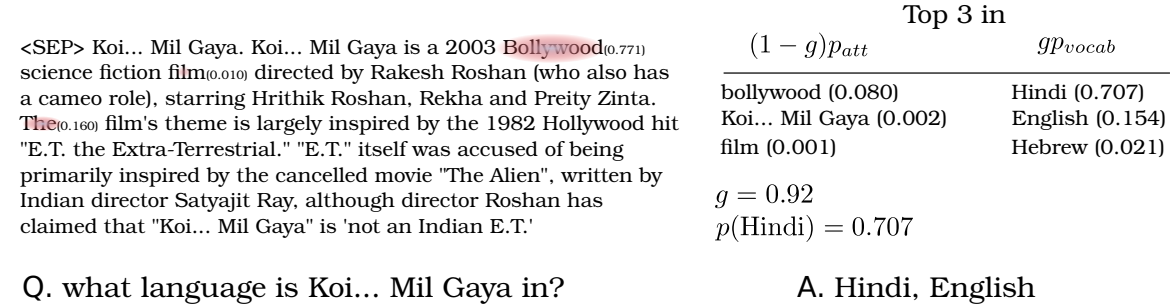

Figure 6: The model uses the p_{vocab} output to answer the question. The word "Bollywood" is attended. The word implies the "Hindi" language.

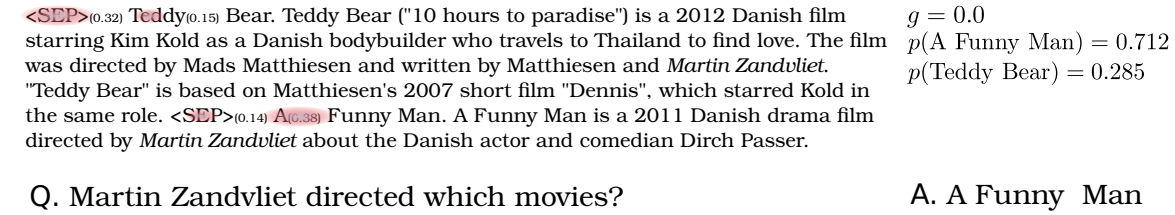

Figure 7: Model behavior of a question "Martin Zandvliet directed which movies?" Martin Zandvliet is a writer of Teddy Bear, not a director.

	WE	FL
r1+AsV	85.4	61.9
no shuffling	83.7	61.0
no anonymization	84.5	61.0

Table 6: Shuffling and anonymization lead to higher performance.

this vocabulary in the mixture model (AsV) further improves the performance.

Table 5 also shows a comparison between (r0), (r1), and (R2) in terms of the overall task performance. We can see that improving the quality of retrieved articles benefits the downstream comprehension performance. In line with the results of the previous section, (r1) and (R2) significantly outperform (r0). Among (r1) and (R2), (R2) performs slightly better.

5.2 Benefit of training methods

Table 6 shows the impact of anonymization of entities and shuffling of training articles before the comprehension step, described in Section 3.

Shuffling the context article before concatenating them, works as a data augmentation technique. Entity anonymization helps because without it each entity has one embedding. Since most of the entities appear only a few times in the articles, these embeddings may not be properly trained. Instead, the anonymous embedding vectors are trained to distinguish different entities. This technique is motivated by a similar procedure used in the construction of CNN / Daily Mail (Hermann et al., 2015), and discussed in detail in (Wang et al., 2016).

5.3 Visualization

Figure 6 shows a test example from the WIKIMOVIES-FL test data. In this case, even though the answers "Hindi" and "English" are not in the context, they are correctly estimated from p_{vocab}. Note the high value of g in this case. Figure 7 shows another example of how the mixture model works. Here the the answer is successfully selected from the document instead of the vocabulary. Note the low value of g in this case.

5.4 Performance in each category

Table 7 shows the comparison for each category of questions between our model and KV-MemNN for the WIKIMOVIES-WE dataset [6]. We can see that performance improvements in the `movie_to_x` category is relatively large. The KV-MemNN model has a dataset specific "Title encoding" feature which helps the model `x_to_movie` question types. However without this feature performance in other categories is poor.

5.5 Analysis of the mixture gate

The benefit of the mixture model comes from the fact that $p_{pointer}$ works well for some question

[6] Categories "Movie to IMDb Votes" and "Movie to IMDb Rating" are omitted from this table because there are only 0.5% test data for these categories and most of the answers are "famous" or "good".

Question Type	KV	r1+AsV
Movie to Year	83	94
Movie to Writer	64	90
Movie to Tags	48	57
Movie to Language	84	89
Movie to Genre	86	90
Movie to Director	79	91
Movie to Actors	64	84
Writer to Movie	91	93
Tag to Movie	49	45
Director to Movie	91	93
Actor to Movie	83	85
Total	**76**	**85.4**

Table 7: Hits@1 scores for each question type. Our model gets $> 80\%$ in all cases but two.

Question Type	ratio	r1+A	r1+V
Movie to Year	0.00	**93**	92
Movie to Writer	0.00	**90**	86
Movie to Tags	0.01	**57**	50
Movie to Language	0.32	81	**87**
Movie to Genre	0.72	76	**90**
Movie to Director	0.00	**91**	90
Movie to Actors	0.00	**82**	74
Writer to Movie	0.00	**92**	89
Tag to Movie	0.03	**46**	41
Director to Movie	0.00	**91**	85
Actor to Movie	0.00	**81**	80
Total		82.6	81.0

Table 8: Ratio of the gate being open. ($g > 0.5$) If the answer is named entity, the model need to select answer from text. Therefore, $g = 0$. Bold font indicates winning model. Vocabulary Only model wins when g is high.

types, while p_{vocab} works well for others. Table 8 shows how often for each category p_{vocab} is used ($g > 0.5$) in AsV model. For question types "Movie to Language" and "Movie to Genre" (the so called "choice questions") the number of possible answers is small. For this case, even if the answer can be found in the context, it is easier for the model to select answer from an external vocabulary which encodes global statistics about the entities. For other "free questions", depending on the question type, one approach is better than the other. Our model is able to successfully estimate the latent category and switch the model type by controlling the coefficient g.

6 Related Work

Choi et al. (2016) solve the QA problem by selecting a sentence in the document. They show that joint training of selection and comprehension slightly improves the performance. In our case, joint training is much harder because of the large number of movie articles. Hence we introduce a two-step retrieval and comprehension approach.

Recently Zoph and Le (2016) proposed a framework to use the performance on a downstream task (e.g. comprehension) as a signal to guide the learning of neural network which determines the input to the downstream task (e.g. retrieval). This motivates us to introduce neural network based approach for both retrieval and comprehension, since in this case the retrieval step can be directly trained to maximize the downstream performance.

In the context of language modeling, the idea of combining of two output probabilities is given in (Merity et al., 2016), however, our equation to compute the mixture coefficient is slightly different. More recently, Ahn et al. (2016) used a mixture model to predict the next word from either the entire vocabulary, or a set of Knowledge Base facts associated with the text. In this work, we present the first application of such a mixture model to reading comprehension.

7 Conclusion and Future Work

We have developed QA system using a two-step retrieval and comprehension approach. The comprehension step uses a mixture model to achieve state of the art performance on WIKIMOVIES dataset, improving previous work by a significant margin.

We would like to emphasize that our approach has minimal heuristics and does not use dataset specific feature engineering. Efficient retrieval while maintaining representation variation is a challenging problem. While there has been a lot of research on comprehension, little focus has been given to designing neural network based retrieval models. We present a simple such model, and emphasize the importance of this direction of research.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318* .

Alex Auvolat, Sarath Chandar, Pascal Vincent, Hugo Larochelle, and Yoshua Bengio. 2015. Clustering is efficient for approximate maximum inner product searrch. *arXiv:1507.05910* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1533–1544.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD international conference on Management of data*. ACM, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv:1506.02075* .

Sarath Chandar, Sungjin Ahn, Hugo Larochelle, Pascal Vincent, Gerald Tesauro, and Yoshua Bengio. 2016. Hierarchical memory networks. *arXiv:1605.07427* .

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*. pages 1724–1734.

Eunsol Choi, Daniel Hewlett, Alexandre Lacoste, Illia Polosukhin, Jakob Uszkoreit, and Jonathan Berant. 2016. Hierarchical question answering for long documents. *arXiv: 1611.01839* .

Bhuwan Dhingra, Hanxiao Liu, William Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv: 1606.01549* .

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 1156–1165.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 1693–1701.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 908–918.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Stephen Merity, Xiong Caiming, Bradbury James, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv: 1609.07843* .

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1400–1409.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 200–207.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM* 57(10):78–85.

Hai Wang, Takeshi Onishi, Kevin Gimpel, and David McAllester. 2016. Emergent logical structure in vector representations of neural readers. *arXiv: 1611.07954* .

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*. ACM, pages 515–526.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .

Barret Zoph and Quoc V. Le. 2016. Neural architecture search with reinforcement learning. *arXiv: 1611.01578* .