DocChat: An Information Retrieval Approach for Chatbot Engines Using Unstructured Documents

Zhao Yan[†] [*], **Nan Duan**[‡] , **Junwei Bao**[+] , **Peng Chen**[§] , **Ming Zhou**[‡] ,
Zhoujun Li[†] , **Jianshe Zhou**[¶]

[†]State Key Laboratory of Software Development Environment, Beihang University
[‡]Microsoft Research [+]Harbin Institute of Technology
[§]Microsoft Search Technology Center [¶]BAICIT, Capital Normal University
[†]{yanzhao, lizj}@buaa.edu.cn [+]baojunwei001@gmail.com
[‡§]{nanduan, peche, mingzhou}@microsoft.com [¶]zhoujs@cnu.edu.cn

Abstract

Most current chatbot engines are designed to reply to user utterances based on existing utterance-response (or Q-R)[1] pairs. In this paper, we present DocChat, a novel information retrieval approach for chatbot engines that can leverage unstructured documents, instead of Q-R pairs, to respond to utterances. A learning to rank model with features designed at different levels of granularity is proposed to measure the relevance between utterances and responses directly. We evaluate our proposed approach in both English and Chinese: (i) For English, we evaluate DocChat on WikiQA and QASent, two answer sentence selection tasks, and compare it with state-of-the-art methods. Reasonable improvements and good adaptability are observed. (ii) For Chinese, we compare DocChat with XiaoIce[2], a famous chitchat engine in China, and side-by-side evaluation shows that DocChat is a perfect complement for chatbot engines using Q-R pairs as main source of responses.

1 Introduction

Building chatbot engines that can interact with humans with natural language is one of the most challenging problems in artificial intelligence. Along with the explosive growth of social media, like community question answering (CQA) websites (e.g., Yahoo Answers and WikiAnswers) and social media websites (e.g., Twitter and Weibo), the amount of utterance-response (or Q-R) pairs has experienced massive growth in recent years, and such a corpus greatly promotes the emergence of various data-driven chatbot approaches.

Instead of multiple rounds of conversation, we only consider a much simplified task, short text conversation (STC) in which the response $\mathcal{R}$ is a short text and only depends on the last user utterance $\mathcal{Q}$. Previous methods for the STC task mostly rely on Q-R pairs and fall into two categories: **Retrieval-based methods** (e.g., Ji et al., 2014). This type of methods first retrieve the most possible $\langle \hat{\mathcal{Q}}, \hat{\mathcal{R}} \rangle$ pair from a set of existing Q-R pairs, which best matches current utterance $\mathcal{Q}$ based on semantic matching models, then take $\hat{\mathcal{R}}$ as the response $\mathcal{R}$. One disadvantage of such a method is that, for many specific domains, collecting such Q-R pairs is intractable. **Generation based methods** (e.g., Shang et al., 2015). This type of methods usually uses an encoder-decoder framework which first encode $\mathcal{Q}$ as a vector representation, then feed this representation to decoder to generate response $\mathcal{R}$. Similar to retrieval-based methods, such approaches also depend on existing Q-R pairs as training data. Like other language generation tasks, such as machine translation and paraphrasing, the fluency and naturality of machine generated text is another drawback.

To overcome the issues mentioned above, we present a novel response retrieval approach, **DocChat**, to find responses based on unstructured documents. For each user utterance, instead of looking for the best Q-R pair or generating a word sequence based on language generation techniques, our method selects a sentence from given documents directly, by ranking all possible sentences based on features designed at different levels of granularity. On one hand, using documents rather than Q-R pairs greatly improve the adapt-

[*]Contribution during internship at Microsoft Research.

[1]For convenience sake, we denote all utterance-response pairs (either QA pairs or conversational exchanges from social media websites like Twitter) as Q-R pairs in this paper.

[2]http://www.msxiaoice.com

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 516–525,
Berlin, Germany, August 7-12, 2016.

ability of chatbot engines on different chatting topics. On the other hand, all responses come from existing documents, which guarantees their fluency and naturality. We also show promising results in experiments, on both QA and chatbot scenarios.

2 Task Description

Formally, given an utterance $\mathcal{Q}$ and a document set $\mathcal{D}$, the document-based chatbot engine retrieves response $\mathcal{R}$ based on the following three steps:

- *response retrieval*, which retrieves response candidates $\mathcal{C}$ from $\mathcal{D}$ based on $\mathcal{Q}$:

$$\mathcal{C} = Retrieve(\mathcal{Q}, \mathcal{D})$$

  Each $\mathcal{S} \in \mathcal{C}$ is a sentence existing in $\mathcal{D}$.

- *response ranking*, which ranks all response candidates in $\mathcal{C}$ and selects the most possible response candidate as $\hat{\mathcal{S}}$:

$$\hat{\mathcal{S}} = \arg\max_{\mathcal{S} \in \mathcal{C}} Rank(\mathcal{S}, \mathcal{Q})$$

- *response triggering*, which decides whether it is confident enough to response $\mathcal{Q}$ using $\hat{S}$:

$$\mathcal{I} = Trigger(\hat{\mathcal{S}}, \mathcal{Q})$$

  where $\mathcal{I}$ is a binary value. When $\mathcal{I}$ equals to $true$, let the response $\mathcal{R} = \hat{\mathcal{S}}$ and output $\mathcal{R}$; otherwise, output nothing.

In the following three sections, we will describe solutions of these three components one by one.

3 Response Retrieval

Given a user utterance $\mathcal{Q}$, the goal of response retrieval is to efficiently find a small number of sentences from $\mathcal{D}$, which have high possibility to contain suitable sentences as $\mathcal{Q}$'s response. Although it is not necessarily true that a good response always shares more words with a given utterance, this measurement is still helpful in finding possible response candidates (Ji et al., 2014).

In this paper, the BM25 term weighting formulas (Jones et al., 2000) is used to retrieve response candidates from documents. Given each document $\mathcal{D}_k \in \mathcal{D}$, we collect a set of sentence triples $\langle \mathcal{S}^{prev}, \mathcal{S}, \mathcal{S}^{next} \rangle$ from $\mathcal{D}_k$, where $\mathcal{S}$ denotes a sentence in $\mathcal{D}_k$, $\mathcal{S}^{prev}$ and $\mathcal{S}^{next}$ denote $\mathcal{S}$'s previous sentence and next sentence respectively. Two special tags, $\langle BOD \rangle$ and $\langle EOD \rangle$, are added at the beginning and end of each passage, to make sure that such sentence triples can be extracted for every sentence in the document. The reason for indexing each sentence together with its context sentences is intuitive: If a sentence within a document can respond to an utterance, then its context should be revelent to the utterance as well.

4 Response Ranking

Given a user utterance $\mathcal{Q}$ and a response candidate $\mathcal{S}$, the ranking function $Rank(\mathcal{S}, \mathcal{Q})$ is designed as an ensemble of individual matching features:

$$Rank(\mathcal{S}, \mathcal{Q}) = \sum_k \lambda_k \cdot h_k(\mathcal{S}, \mathcal{Q})$$

where $h_k(\cdot)$ denotes the k-th feature function, λ_k denotes $h_k(\cdot)$'s corresponding weight.

We design features at different levels of granularity to measure the relevance between S and Q, including *word-level*, *phrase-level*, *sentence-level*, *document-level*, *relation-level*, *type-level* and *topic-level*, which will be introduced below.

4.1 Word-level Feature

We define three word-level features in this work: (1) $h_{WM}(\mathcal{S}, \mathcal{Q})$ denotes a word matching feature that counts the number (weighted by the *IDF* value of each word in $\mathcal{S}$) of non-stopwords shared by $\mathcal{S}$ and $\mathcal{Q}$. (2) $h_{W2W}(\mathcal{S}, \mathcal{Q})$ denotes a word-to-word translation-based feature that calculates the IBM model 1 score (Brown et al., 1993) of $\mathcal{S}$ and $\mathcal{Q}$ based on word alignments trained on 'question-related question' pairs using GIZA++ (Och and Ney, 2003). (3) $h_{W2V}(\mathcal{S}, \mathcal{Q})$ denotes a word embedding-based feature that calculates the average cosine distance between word embeddings of all non-stopword pairs $\langle v_{\mathcal{S}_j}, v_{\mathcal{Q}_i} \rangle$. $v_{\mathcal{S}_j}$ represent the word vector of j^{th} word in $\mathcal{S}$ and $v_{\mathcal{Q}_j}$ represent the word vector of i^{th} word in $\mathcal{Q}$.

4.2 Phrase-level Feature

4.2.1 Paraphrase

We first describe how to extract phrase-level paraphrases from an existing SMT (statistical machine translation) phrase table.

$PT = \{\langle s_i, t_i, p(t_i|s_i), p(s_i|t_i) \rangle\}$[3] is a phrase table, which is extracted from a bilingual corpus, where s_i (or t_i) denotes a phrase, in source

[3]We omit lexical weights that are commonly used in phrase tables, as they are not useful in paraphrase extraction.

(or target) language, $p(t_i|s_i)$ (or $p(s_i|t_i)$) denotes the translation probability from s_i (or t_i) to t_i (or s_i). We follow Bannard and Callison-Burch (2005) to extract a paraphrase table $PP = \{\langle s_i, s_j, score(s_j; s_i)\rangle\}$. s_i and s_j denote two phrases in source language, $score(s_j; s_i)$ denotes a confidence score that s_i can be paraphrased to s_j, which is computed based on PT:

$$score(s_j; s_i) = \sum_t \{p(t|s_i) \cdot p(s_j|t)\}$$

The underlying idea of this approach is that, two source phrases that are aligned to the same target phrase trend to be paraphrased.

We then define a paraphrase-based feature as:

$$h_{PP}(\mathcal{S}, \mathcal{Q}) = \frac{\sum_{n=1}^{N} \frac{\sum_{j=0}^{|\mathcal{S}|-n} Count_{PP}(\mathcal{S}_j^{j+n-1}, \mathcal{Q})}{|\mathcal{S}|-n+1}}{N}$$

where $\mathcal{S}_j^{j+n-1}$ denotes the consecutive word sequence (or phrase) in $\mathcal{S}$, which starts from $\mathcal{S}_j$ and ends with $\mathcal{S}_{j+n-1}$, N denotes the maximum n-gram order (here is 3). $Count_{PP}(\mathcal{S}_j^{j+n-1}, \mathcal{Q})$ is computed based on the following rules:

- If $\mathcal{S}_j^{j+n-1} \in \mathcal{Q}$, then $Count_{PP}(\mathcal{S}_j^{j+n-1}, \mathcal{Q}) = 1$;
- Else, if $\langle \mathcal{S}_j^{j+n-1}, s, score(s; \mathcal{S}_j^{j+n-1})\rangle \in PP$ and $\mathcal{S}_j^{j+n-1}$'s paraphrase s occurs in $\mathcal{Q}$, then $Count_{PP}(\mathcal{S}_j^{j+n-1}, \mathcal{Q}) = score(s; \mathcal{S}_j^{j+n-1})$
- Else, $Count_{PP}(\mathcal{S}_j^{j+n-1}, \mathcal{Q}) = 0$.

4.2.2 Phrase-to-Phrase Translation

Similar to $h_{PP}(\mathcal{S}, \mathcal{Q})$, a phrase translation-based feature based on a phrase table PT is defined as:

$$h_{PT}(\mathcal{S}, \mathcal{Q}) = \frac{\sum_{n=1}^{N} \frac{\sum_{j=0}^{|\mathcal{S}|-n} Count_{PT}(\mathcal{S}_j^{j+n-1}, \mathcal{Q})}{|\mathcal{S}|-n+1}}{N}$$

where $Count_{PT}(\mathcal{S}_j^{j+n-1}, \mathcal{Q})$ is computed based on the following rules:

- If $\mathcal{S}_j^{j+n-1} \in \mathcal{Q}$, then $Count_{PT}(\mathcal{S}_j^{j+n-1}, \mathcal{Q}) = 1$;
- Else, if $\langle \mathcal{S}_j^{j+n-1}, s, p(\mathcal{S}_j^{j+n-1}|s), p(s|\mathcal{S}_j^{j+n-1})\rangle \in PT$ and $\mathcal{S}_j^{j+n-1}$'s translation $s \in \mathcal{Q}$, then $Count_{PT}(\mathcal{S}_j^{j+n-1}, \mathcal{Q}) = p(\mathcal{S}_j^{j+n-1}|s) \cdot p(s|\mathcal{S}_j^{j+n-1})$
- Else, $Count_{PT}(\mathcal{S}_j^{j+n-1}, \mathcal{Q}) = 0$

We train a phrase table based on 'question-answer' pairs crawled from community QA websites.

4.3 Sentence-level Feature

We first present an attention-based sentence embedding method based on a convolution neural network (CNN), whose input is a sentence pair and output is a sentence embedding pair. Two features will be introduced in Section 4.3.1 and 4.3.2, which are designed based on two sentence embedding models trained using different types of data.

In the input layer, given a sentence pair $\langle \mathcal{S}_X, \mathcal{S}_Y\rangle$, an attention matrix $\mathcal{A} \in R^{|\mathcal{S}_X|\times|\mathcal{S}_Y|}$ is generated based on pre-trained word embeddings of $\mathcal{S}_X$ and $\mathcal{S}_Y$, where each element $\mathcal{A}_{i,j} \in \mathcal{A}$ is computed as:

$$\mathcal{A}_{i,j} = cosine(v_i^{\mathcal{S}_X}, v_j^{\mathcal{S}_Y})$$

where $v_i^{\mathcal{S}_X}$ (or $v_j^{\mathcal{S}_Y}$) denotes the embedding vector of the i^{th} (or j^{th}) word in $\mathcal{S}_X$ (or $\mathcal{S}_Y$).

Then, column-wise and row-wise max-pooling are applied to $\mathcal{A}$ to generate two attention vectors $V^{\mathcal{S}_X} \in R^{|\mathcal{S}_X|}$ and $V^{\mathcal{S}_Y} \in R^{|\mathcal{S}_Y|}$, where the k^{th} elements of $V^{\mathcal{S}_X}$ and $V^{\mathcal{S}_Y}$ are computed as:

$$V_k^{\mathcal{S}_X} = \max_{1<l<|\mathcal{S}_Y|} \{\mathcal{A}_{k,l}\} \; and \; V_k^{\mathcal{S}_Y} = \max_{1<l<|\mathcal{S}_X|} \{\mathcal{A}_{l,k}\}$$

$V_k^{\mathcal{S}_X}$ (or $V_k^{\mathcal{S}_Y}$) can be interpreted as the *attention score* of the k^{th} word in $\mathcal{S}_X$ (or $\mathcal{S}_Y$) with regard to all words in $\mathcal{S}_Y$ (or $\mathcal{S}_X$).

Next, two *attention distributions* $D^{\mathcal{S}_X} \in R^{|\mathcal{S}_X|}$ and $D^{\mathcal{S}_Y} \in R^{|\mathcal{S}_Y|}$ are generated for $\mathcal{S}_X$ and $\mathcal{S}_Y$ based on $V^{\mathcal{S}_X}$ and $V^{\mathcal{S}_Y}$ respectively, where the k^{th} elements of $D^{\mathcal{S}_X}$ and $D^{\mathcal{S}_Y}$ are computed as:

$$D_k^{\mathcal{S}_X} = \frac{e^{V_k^{\mathcal{S}_X}}}{\sum_{l=1}^{|\mathcal{S}_X|} e^{V_l^{\mathcal{S}_X}}} \; and \; D_k^{\mathcal{S}_\mathcal{Y}} = \frac{e^{V_k^{\mathcal{S}_Y}}}{\sum_{l=1}^{|\mathcal{S}_Y|} e^{V_l^{\mathcal{S}_Y}}}$$

$D_k^{\mathcal{S}_\mathcal{X}}$ (or $D_k^{\mathcal{S}_\mathcal{Y}}$) can be interpreted as the *normalized attention score* of the k^{th} word in $\mathcal{S}_X$ (or $\mathcal{S}_Y$) with regard to all words in $\mathcal{S}_Y$ (or $\mathcal{S}_X$).

Last, we update each pre-trained word embedding $v_k^{\mathcal{S}_X}$ (or $v_k^{\mathcal{S}_Y}$) to $\hat{v}_k^{\mathcal{S}_X}$ (or $\hat{v}_k^{\mathcal{S}_Y}$), by multiplying every value in $v_k^{\mathcal{S}_X}$ (or $v_k^{\mathcal{S}_Y}$) with $D_k^{\mathcal{S}_X}$ (or $D_k^{\mathcal{S}_Y}$). The underlying intuition of updating pre-trained word embeddings is to *re-weight the importance of each word in $\mathcal{S}_X$ (or $\mathcal{S}_Y$) based on $\mathcal{S}_Y$ (or $\mathcal{S}_X$), instead of treating them in an equal manner.*

In the convolution layer, we first derive an input matrix $Z_{\mathcal{S}_X} = \{l_1, ..., l_{|\mathcal{S}_X|}\}$, where l_t is the concatenation of a sequence of $m = 2d-1$[4] updated word embeddings $[\hat{v}_{t-d}^{\mathcal{S}_X}, ..., \hat{v}_t^{\mathcal{S}_X}, ..., \hat{v}_{t+d}^{\mathcal{S}_X}]$, centralized in the t^{th} word in $\mathcal{S}_X$. Then, the convo-

[4] In this paper, m is set to 3.

lution layer performs sliding window-based feature extraction to project each vector representation $l_t \in Z_{\mathcal{S}_X}$ to a contextual feature vector $h_t^{\mathcal{S}_X}$:

$$h_t^{\mathcal{S}_X} = tanh(W_c \cdot l_t)$$

where W_c is the convolution matrix, $tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$ is the activation function. The same operation is performed to $\mathcal{S}_Y$ as well.

In the pooling layer, we aggregate local features extracted by the convolution layer from $\mathcal{S}_X$, and form a sentence-level global feature vector with a fixed size independent of the length of the input sentence. Here, *max-pooling* is used to force the network to retain the most useful local features by $l_p^{\mathcal{S}_X} = [v_1^{\mathcal{S}_X}, ..., v_K^{\mathcal{S}_X}]$, where:

$$v_i^{\mathcal{S}_X} = \max_{t=1,...,|\mathcal{S}_X|} \{h_t^{\mathcal{S}_X}(i)\}$$

$h_t^{\mathcal{S}_X}(i)$ denotes the i^{th} value in the vector $h_t^{\mathcal{S}_X}$. The same operation are performed to $\mathcal{S}_Y$ as well.

In the output layer, one more non-linear transformation is applied to $l_p^{\mathcal{S}_X}$:

$$y(\mathcal{S}_X) = tanh(W_s \cdot l_p^{\mathcal{S}_X})$$

W_s is the semantic projection matrix, $y(\mathcal{S}_X)$ is the final sentence embedding of $\mathcal{S}_X$. The same operation is performed to $\mathcal{S}_Y$ to obtain $y(\mathcal{S}_Y)$.

We train model parameters W_c and W_s by minimizing the following ranking loss function:

$$L = \max\{0, M - cosine(y(\mathcal{S}_X), y(\mathcal{S}_Y)) + cosine(y(\mathcal{S}_X), y(\mathcal{S}_Y^-))\}$$

where M is a constant, $\mathcal{S}_Y^-$ is a negative instance.

4.3.1 Causality Relationship Modeling

We train the first attention-based sentence embedding model based on a set of 'question-answer' pairs as input sentence pairs, and then design a causality relationship-based feature as:

$$h_{SCR}(\mathcal{S}, \mathcal{Q}) = cosine(y_{SCR}(\mathcal{S}), y_{SCR}(\mathcal{Q}))$$

$y_{SCR}(\mathcal{S})$ and $y_{SCR}(\mathcal{Q})$ denote the sentence embeddings of $\mathcal{S}$ and $\mathcal{Q}$ respectively. We expect this feature captures the *causality relationship* between questions and their corresponding answers, and works on *question-like utterances*.

4.3.2 Discourse Relationship Modeling

We train the second attention-based sentence embedding model based on a set of 'sentence-next sentence' pairs as input sentence pairs, and then design a discourse relationship-based feature as:

$$h_{SDR}(\mathcal{S}, \mathcal{Q}) = cosine(y_{SDR}(\mathcal{S}), y_{SDR}(\mathcal{Q}))$$

$y_{SDR}(\mathcal{S})$ and $y_{SDR}(\mathcal{Q})$ denote the sentence embeddings of $\mathcal{S}$ and $\mathcal{Q}$ respectively. We expect this feature learns and captures the *discourse relationship* between sentences and their next sentences, and works on *statement-like utterances*. Here, a large number of 'sentence-next sentence' pairs can be easily obtained from documents.

4.4 Document-level Feature

We take document-level information into consideration to measure the semantic similarity between $\mathcal{Q}$ and $\mathcal{S}$, and define two context features as:

$$h_{DM}(\mathcal{S}^*, \mathcal{Q}) = cosine(y_{SCR}(\mathcal{S}^*), y_{SCR}(\mathcal{Q}))$$

where $\mathcal{S}^*$ can be $\mathcal{S}^{prev}$ and $\mathcal{S}^{next}$ that denote previous and next sentences of $\mathcal{S}$ in the original document. The sentence embedding model trained based on 'question-answer' pairs (in Section 4.3.1) is directly used to generate context embeddings for $h_{DM}(\mathcal{S}^{prev}, \mathcal{Q})$ and $h_{DM}(\mathcal{S}^{next}, \mathcal{Q})$. So no further training data is needed for this feature.

4.5 Relation-level Feature

Given a structured knowledge base, such as Freebase[5], a single relation question $\mathcal{Q}$ (in natural language) with its answer can be first parsed into a fact formatted as $\langle e_{sbj}, rel, e_{obj} \rangle$, where e_{sbj} denotes a subject entity detected from the question, rel denotes the relationship expressed by the question, e_{obj} denotes an object entity found from the knowledge base based on e_{sbj} and rel. Then we can get $\langle \mathcal{Q}, rel \rangle$ pairs. This rel can help for modeling semantic relationships between $\mathcal{Q}$ and $\mathcal{R}$. For example, the Q-A pair $\langle$*What does Jimmy Neutron do? – inventor*$\rangle$ can be parsed into $\langle$*Jimmy Neutron, fictional_character_occupation, inventor*$\rangle$ where the rel is *fictional_character_occupation*.

Similar to Yih et al. (2014), We use $\langle \mathcal{Q}, rel \rangle$ pairs as training data, and learn a *rel*-CNN model, which can encode each question $\mathcal{Q}$ (or each relation rel) into a relation embedding. For a given question $\mathcal{Q}$, the corresponding relation rel^+ is

[5]http://www.freebase.com/

treated as a positive example, and randomly selected other relations are used as negative examples rel^-. The posterior probability of rel^+ given $\mathcal{Q}$ is computed as:

$$P(rel^+|\mathcal{Q}) = \frac{e^{cosine(y(rel^+),y(\mathcal{Q}))}}{\sum_{rel^-} e^{cosine(y(rel^-),y(\mathcal{Q}))}}$$

$y(rel)$ and $y(\mathcal{Q})$ denote relation embeddings of rel and $\mathcal{Q}$ based on *rel*-CNN. *rel*-CNN is trained by maximizing the log-posterior.

We then define a relation-based feature as:

$$h_{RE}(\mathcal{S}, \mathcal{Q}) = cosine(y_{RE}(\mathcal{Q}), y_{RE}(\mathcal{S}))$$

$y_{RE}(\mathcal{S})$ and $y_{RE}(\mathcal{Q})$ denote relation embeddings of $\mathcal{S}$ and $\mathcal{Q}$ respectively, coming from *rel*-CNN.

4.6 Type-level Feature

We extend each $\langle \mathcal{Q}, e_{sbj}, rel, e_{obj} \rangle$ in the SimpleQuestions data set to $\langle \mathcal{Q}, e_{sbj}, rel, e_{obj}, type \rangle$, where $type$ denotes the type name of e_{obj} based on Freebase. Thus, we obtain $\langle \mathcal{Q}, type \rangle$ pairs.

Similar to *rel*-CNN, we use $\langle \mathcal{Q}, type \rangle$ pairs to train another CNN model, denoted as *type*-CNN. Based on which, we define a type-based feature as:

$$h_{TE}(\mathcal{S}, \mathcal{Q}) = cosine(y_{TE}(\mathcal{Q}), y_{TE}(\mathcal{S}))$$

$y_{TE}(\mathcal{S})$ and $y_{TE}(\mathcal{Q})$ denote type embeddings of $\mathcal{S}$ and $\mathcal{Q}$ respectively, coming from *type*-CNN.

4.7 Topic-level Feature

4.7.1 Unsupervised Topic Model

As the assumption that Q-R pair should share similar topic distribution, We define an unsupervised topic model-based feature h_{UTM} as the average cosine distance between topic vectors of all non-stopword pairs $\langle v_{\mathcal{S}_j}, v_{\mathcal{Q}_i} \rangle$, where $v_w = [p(t_1|w), ..., p(t_N|w)]^T$ denotes the topic vector of a given word w. Given a corpus, various topic modeling methods, such as pLSI (probabilistic latent semantic indexing) and LDA (latent Dirichlet allocation), can be used to estimate $p(t_i|w)$, which denotes the probability that w belongs to a topic t_i.

4.7.2 Supervised Topic Model

One shortcoming of the unsupervised topic model is that, the topic size is pre-defined, which might not reflect the truth on a specific corpus. In this paper, we explore a supervised topic model approach as well, based on 'sentence-topic' pairs.

We crawl a large number of $\langle \mathcal{S}, topic \rangle$ pairs from Wikipedia documents, where $\mathcal{S}$ denotes a sentence, $topic$ denotes the content name of the section that $\mathcal{S}$ extracted from. Such content names are labeled by Wikipedia article editors, and can be found in the *Contents* fields.

Similar to *rel*-CNN and *type*-CNN, we use the $\langle \mathcal{S}, topic \rangle$ pairs to train another CNN model, denoted as *topic*-CNN. Based on which, we define a supervised topic model-based feature as:

$$h_{STM}(\mathcal{S}, \mathcal{Q}) = cosine(y_{STM}(\mathcal{S}), y_{STM}(\mathcal{Q}))$$

$y_{STM}(\mathcal{S})$ and $y_{STM}(\mathcal{Q})$ denote topic embeddings of $\mathcal{S}$ and $\mathcal{Q}$ respectively, coming from *topic*-CNN.

4.8 Learning to Ranking Model

We employ a regression-based learning to rank method (Nallapati, 2004) to train response ranking model, based on a set of labeled $\langle \mathcal{Q}, \mathcal{C} \rangle$ pairs, Feature weights in the ranking model are trained by SGD based on the training data that consists of a set of $\langle \mathcal{Q}, \mathcal{C} \rangle$ pairs, where $\mathcal{Q}$ denotes a user utterance and $\mathcal{C}$ denotes a set of response candidates. Each candidate $\mathcal{S}$ in $\mathcal{C}$ is labeled by $+$ or $-$, which indicates whether $\mathcal{S}$ is a suitable response of $\mathcal{Q}$ $(+)$, or not $(-)$.

As manually labeled data, such as WikiQA (Yang et al., 2015), needs expensive human annotation effort, we propose an automatic way to collect training data. First, 'question-answer' (or Q-A) pairs $\{Q_i, A_i\}_{i=1}^M$ are crawled from community QA websites. Q_i denotes a question. A_i denotes Q_i's answer, which includes one or more sentences $A_i = \{s_1, ..., s_K\}$. Then, we index answer sentences of all questions. Next, for each question Q_i, we run response retrieval to obtain answer sentence candidates $\mathcal{C}_i = \{s_1^{'}, ..., s_N^{'}\}$. Last, if we know the correct answer sentences of each question Q_i, we can then label each candidate in $\mathcal{C}_i$ as $+$ or $-$. In experiments, manually labeled data (WikiQA) is used in open domain question answering scenario, and automatically generated data is used in chatbot scenario.

5 Response Triggering

There are two types of utterances, *chit-chat utterances* and *informative utterances*. The former should be handled by chit-chat engines, and the latter is more suitable to our work, as documents usually contain formal and informative contents. Thus, we have to respond to informative utterances only. Response retrieval cannot always guarantee to return a candidate set that contains

at least one suitable response, but response ranking will output the best possible candidate all the time. So, we have to decide which responses are confident enough to be output, and which are not.

In this paper, we define *response triggering* as a function that decides whether a response candidate $\mathcal{S}$ has enough confidence to be output:

$$\begin{aligned}\mathcal{I} &= Trigger(\mathcal{S}, \mathcal{Q}) \\ &= \mathcal{I}_U(\mathcal{Q}) \wedge \mathcal{I}_{Rank}(\mathcal{S}, \mathcal{Q}) \wedge \mathcal{I}_R(\mathcal{S})\end{aligned}$$

where $Trigger(\mathcal{Q}, \mathcal{S})$ returns $true$, if and only if all its three sub-functions return $true$.

$\mathcal{I}_U(\mathcal{Q})$ returns $true$, if $\mathcal{Q}$ is an informative query. We collect and label chit-chat queries based on conversational exchanges from social media websites to train the classifier.

$\mathcal{I}_{Rank}(\mathcal{S}, \mathcal{Q})$ returns $true$, if the score $s(\mathcal{S}, \mathcal{Q})$ exceeds an empirical threshold τ:

$$s(\mathcal{S}, \mathcal{Q}) = \frac{1}{1 + e^{-\alpha \cdot Rank(\mathcal{S}, \mathcal{Q})}}$$

where α is the scaling factor that controls the distribution of $s(\cdot)$ smooth or sharp. Both α and τ are selected based on a separated development set.

$\mathcal{I}_R(\mathcal{S})$ returns $true$, if (i) the length of $\mathcal{S}$ is less than a pre-defined threshold, and (ii) $\mathcal{S}$ does not start with a phrase that expresses a progressive relation, such as *but also*, *besides*, *moreover* and etc., as the contents of sentences starting with such phrases usually depend on their context sentences, and they are not suitable for responses.

6 Related Work

For modeling dialogue. Previous works mainly focused on rule-based or learning-based approaches (Litman et al., 2000; Schatzmann et al., 2006; Williams and Young, 2007). These methods require efforts on designing rules or labeling data for training, which suffer the coverage issue.

For short text conversation. With the fast development of social media, such as microblog and CQA services, large scale conversation data and data-driven approaches become possible. Ritter et al. (2011) proposed an SMT based method, which treats response generation as a machine translation task. Shang et al. (2015) presented an RNN based method, which is trained based on a large number of single round conversation data. Grammatical and fluency problems are the biggest issue for such generation-based approaches. Retrieval-based methods selects the most suitable response to the current utterance from the large number of Q-R pairs. Ji et al. (2014) built a conversation system using learning to rank and semantic matching techniques. However, collecting enough Q-R pairs to build chatbots is often intractable for many domains. Compared to previous methods, DocChat learns internal relationships between utterances and responses based on statistical models at different levels of granularity, and relax the dependency on Q-R pairs as response sources. These make DocChat as a general response generation solution to chatbots, with high adaptation capability.

For answer sentence selection. Prior work in measuring the relevance between question and answer is mainly in word-level and syntactic-level (Wang and Manning, 2010; Heilman and Smith, 2010; Yih et al., 2013). Learning representation by neural network architecture (Yu et al., 2014; Wang and Nyberg, 2015; Severyn and Moschitti, 2015) has become a hot research topic to go beyond word-level or phrase-level methods. Compared to previous works we find that, (i) Large scale existing resources with noise have more advantages as training data. (ii) Knowledge-based semantic models can play important roles.

7 Experiments

7.1 Evaluation on QA (English)

Take into account response ranking task and answer selection task are similar, we first evaluate DocChat in a QA scenario as a simulation. Here, response ranking is treated as the *answer selection* task, and response triggering is treated as the *answer triggering* task.

7.1.1 Experiment Setup

We select *WikiQA*[6] as the evaluation data, as it is precisely constructed based on natural language questions and Wikipedia documents, which contains 2,118 'question-document' pairs in the training set, 296 'question-document' pairs in development set, and 633 'question-document' pairs in testing set. Each sentence in the document of a given question is labeled as 1 or 0, where 1 denotes the current sentence is a correct answer sentence, and 0 denotes the opposite meaning. Given a question, the task of WikiQA is to select answer sentences from all sentences in a question's corresponding document. The training data settings of response ranking features are described below.

[6] http://aka.ms/WikiQA

F_w denotes 3 word-level features, h_{WM}, h_{W2W} and h_{W2V}. For h_{W2W}, GIZA++ is used to train word alignments on 11.6M 'question-related question' pairs (Fader et al., 2013) crawled from WikiAnswers.[7]. For h_{W2V}, Word2Vec (Mikolov et al., 2013) is used to train word embedding on sentences from Wikipedia in English.

F_p denotes 2 phrase-level features, h_{PP} and h_{PT}. For h_{PP}, bilingual data[8] is used to extract a phrase-based translation table (Koehn et al., 2003), from which paraphrases are extracted (Section 4.2.1). For h_{PT}, GIZA++ trains word alignments on 4M 'question-answer' pairs[9] crawled from Yahoo Answers[10], and then a phrase table is extracted from word alignments using the *intersect-diag-grow* refinement.

F_s denotes 2 sentence-level features, h_{SCR} and h_{SDR}. For h_{SCR}, 4M 'question-answer' pairs (the same to h_{PT}) is used to train the CNN model. For h_{SDR}, we randomly select 0.5M 'sentence-next sentence' pairs from English Wikipedia.

F_d denotes document-level feature h_{DM}. Here, we didn't train a new model. Instead, we just reuse the CNN model used in h_{SCR}.

F_r and F_{ty} denote relation-level feature h_{RE} and type-level feature h_{TE}. Bordes et al. (2015) released the *SimpleQuestions* data set[11], which consists of 108,442 English questions. Each question (e.g., *What does Jimmy Neutron do?*) is written by human annotators based on a triple in Freebase which formatted as $\langle e_{sbj}, rel, e_{obj} \rangle$ (e.g., $\langle$*Jimmy Neutron, fictional_character_occupation, inventor*$\rangle$) Here, as described in Section 4.5 and 4.6, 'question-relation' pairs and 'question-type' pairs based upon *SimpleQuestions* data set are used to train h_{RE} and h_{TE}.

F_{to} denotes 2 topic-level features, h_{UTM} and h_{STM}. For h_{UTM}, we run LightLDA (Yuan et al., 2015) on sentences from English Wikipedia, where the topic is set to 1,000. For h_{STM}, 4M 'sentence-topic' pairs are extracted from English Wikipedia (Section 4.7.2), where the most frequent 25,000 content names are used as topics.

[7]http://wiki.answers.com

[8]We use 0.5M Chinese-English bilingual sentences in phrase table extraction, i.e., LDC2003E07, LDC2003E14, LDC2005T06, LDC2005T10, LDC2005E83, LDC2006E26, LDC2006E34, LDC2006E85 and LDC2006E92.

[9]For each question, we only select the first sentence in its answer to construct a 'question-answer' pair, as it contains more causality information than sentences in other positions.

[10]https://answers.yahoo.com

[11]https://research.facebook.com/research/-babi/

Features	MAP	MRR
F_w	60.25%	61.70%
F_p	61.31%	62.61%
F_s	61.99%	64.32%
F_d	59.15%	61.17%
F_r	46.95%	45.89%
F_{ty}	45.67%	43.37%
F_{to}	58.34%	59.96%

Table 1: Impacts of features at different levels.

#	Methods	MAP	MRR
(1)	Yih et al. (2013)	59.93%	60.68%
(2)	Yang et al. (2015)	65.20%	66.52%
(3)	Miao et al. (2015)	68.86%	70.69%
(4)	Yin et al. (2015)	69.21%	71.08%
(5)	DocChat	68.25%	70.73%
(6)	DocChat+(2)	**70.08%**	**72.22%**

Table 2: Evaluation of AS task on WikiQA.

7.1.2 Results on Answer Selection (AS)

The performance of answer selection is evaluated by *Mean Average Precision (MAP)* and *Mean Reciprocal Rank (MRR)*. Among all 'question-document' pairs in WikiQA, only one-third of documents contain answer sentences to their corresponding questions. Similar to previous work, questions without correct answers in the candidate sentences are not taken into account.

We first evaluate the impact of features at each level, and show results in Table 1. F_w, F_p, and F_s perform best among all features, which makes sense, as they can capture lexical features. F_r and F_{ty} perform not very good, but make sense, as the training data (i.e. SimpleQuestions) are based on Freebase instead of Wikipedia. Interestingly, we find that F_{to} and F_d can achieve comparable results as well. We think the reason is that, their training data come from Wikipedia, which fit the WikiQA task very well.

We evaluate the quality of DocChat on WikiQA, and show results in Table 2. The first four rows in Table 2 represent four baseline methods, including: (1) Yih et al. (2013), which makes use of rich lexical semantic features; (2) Yang et al. (2015), which uses a bi-gram CNN model with average pooling; (3) Miao et al. (2015), which uses an enriched LSTM with a latent stochastic attention mechanism to model similarity between Q-R pairs; and (4) Yin et al. (2015), which adds the attention mechanism to the CNN architecture.

Table 2 shows that, without using WikiQA's training set (only development set for ranking weights), DocChat can achieve comparable per-

Methods	MAP	MRR
CNN$_{WikiQA}$	0.6575	0.7534
CNN$_{QASent}$	**0.6951**	0.7633
DocChat	0.6896	**0.7688**

Table 3: Evaluation of AS on QASent.

formance with state-of-the-art baselines. Furthermore, by combining the CNN model proposed by Yang et al. (2015) and trained on WikiQA training set, we achieve the best result on both metrics.

Compared to previous methods, we think DocChat has the following two advantages: First, our feature models depending on existing resources are readily available (such as Q-Q pairs, Q-A pairs, 'sentence-next sentence' pairs, and etc.), instead of requiring manually annotated data (such as WikiQA and QASent). Training of the response ranking model does need labeled data, but the size demanded is acceptable. Second, as the training data used in our approach come from open domain resources, we can expect a high adaptation capability and comparable results on other WikiQA-like tasks, as our models are task-independent.

To verify the second advantage, we evaluate DocChat on another answer selection data set, QASent (Wang et al., 2007), and list results in Table 3. CNN_{WikiQA} and CNN_{QASent} refer to the results of Yang et al. (2015)'s method, where the CNN models are trained on WikiQA's training set and QASent's training set respectively. All these three methods train feature weights using QASent's development set. Table 3 tells, DocChat outperforms CNN_{WikiQA} in terms of MAP and MRR, and achieves comparable results compared to CNN_{QASent}. The comparisons results show a good adaptation capability of DocChat.

Table 4 evaluates the contributions of features at different levels of granularity. To highlight the differences, we report the percent deviation by removing different features at the same level from DocChat. From Table 4 we can see that, 1) Each feature group is indispensable to DocChat; 2) Features at sentence-level are most important than other feature groups; 3) Compared to results in Table 1, combining all features can significantly promote the performance.

7.1.3 Evaluation of Answer Triggering (AT)

In both QA and chatbot, response triggering is important. Similar to Yang et al. (2015), we also evaluate answer triggering using Precision, Recall, and F1 score as metrics. We use the WikiQA development set to tune the scaling factor α and trigger threshold τ that are described in Section 5, where α is set to 0.9 and τ is set to 0.5.

Models	MAP	Change	MRR	Change
DocChat	68.25%		70.73%	
DocChat - F_w	66.06%	-2.19	67.99%	-2.74
DocChat - F_p	66.80%	-1.45	68.66%	-2.07
DocChat - F_s	65.49%	**-2.76**	67.27%	**-3.46**
DocChat - F_d	68.02%	-0.23	69.79%	-0.94
DocChat - F_r	67.00%	-1.25	69.07%	-1.66
DocChat - F_{ty}	67.09%	-1.16	69.28%	-1.45
DocChat - F_{to}	66.85%	-1.40	68.96%	-1.77

Table 4: Impacts of different feature groups.

Methods	Precision	Recall	F1
Yang et al. (2015)	28.34	35.80	31.64
DocChat	**28.95**	**44.44**	**35.06**

Table 5: Evaluation of AT on WikiQA.

Table 5 shows the evaluation results compare to Yang et al. (2015). We think the improvements come from the fact that our response ranking model are more discriminative, as more semantic-level features are leveraged.

7.2 Evaluation on Chatbot (Chinese)

XiaoIce is a famous Chinese chatbot engine, which can be found in many platforms including WeChat official accounts (like business pages on Facebook Messenger). The documents that each official account maintains and post to their followers can be easily obtained from the Web. Meanwhile, a WeChat official account can choose to authorize XiaoIce to respond to its followers' utterances. We design an interesting evaluation below to compare DocChat with XiaoIce, based on the publicly available documents.

7.2.1 Experiment Setup

For ranking features, 17M 'question-related questions' pairs crawled from Baidu Zhidao are used to train word alignments for h_{W2W}; sentences from Chinese Wikipedia are used to train word embeddings for h_{W2V} and a topic model for h_{UTM}; the same bilingual phrase table described in last experiment is also used to extract a Chinese paraphrase table for h_{PP} which use Chinese as the source language; 5M 'question-answer' pairs crawled from Baidu Zhidao are used for h_{PT}, h_{SCR} and h_{DM}; 0.5M 'sentence-next sentence' pairs from Chinese Wikipedia are used for h_{SDR}; 1.3M 'sentence-topic pairs' crawled from Chinese Wikipedia are used to train $topic-$CNN for

Utterance	Response
你知道北京的历史么？ (Do you know the history of Beijing?)	[XiaoIce Response]: 我的历史课学的不太好。 (I am not good at history class)
	[DocChat Response]:北京历史悠久，可以追溯到3000年前。 (Beijing is a historical city that can be traced back to 3,000 years ago.)

Table 6: XiaoIce response is more colloquial, as it comes from Q-R pairs; while DocChat response is more formal, as it comes from documents.

h_{STM}. As there is no knowledge base based labeled data for Chinese, we ignore relation-level feature h_{RE} and type-level feature h_{TE}.

For ranking weights, we generate 90,321 $\langle \mathcal{Q}, \mathcal{C} \rangle$ pairs based on Baidu Zhidao Q-A pairs by the automatic method described in Section 4.8. This data set is used to train the learning to rank model feature weights $\{\lambda_k\}$ by SGD.

For documents, we randomly select 10 WeChat official accounts, and index their documents separately. The average number of documents is 600.

Human annotators are asked to freely issue 100 queries to each official account to get XiaoIce response. Thus, we obtain 100 ⟨query, XiaoIce response⟩ pairs for each official account. We also send the same 100 queries of each official account to DocChat based on official account's corresponding document index, and obtain another 100 ⟨query, DocChat response⟩ pairs. Given these 1,000 ⟨query, XiaoIce response, DocChat response⟩ triples, we let human annotators do a side-by-side evaluation, by asking them which response is better for each query. Note that, the source of each response is masked during evaluation procedure. Table 6 gives an example.

7.2.2 DocChat v.s. XiaoIce

Table 7 shows the results. **Better** (or **Worse**) denotes a DocChat response is better (or worse) than a XiaoIce response, **Tie** denotes a DocChat response and a XiaoIce response are equally good or bad. From Table 7 we observe that: (1) 156 DocChat responses (58+47+51) out of 1,000 queries are triggered. The trigger rate of DocChat is 15.6%. We check un-triggered queries, and find most of them are chitchat, such as "hi", "hello", "who are you?". (2) Better cases are more than worse cases. Most queries in better cases are non-chitchat ones, and their contents are highly related to the domain of their corresponding WeChat official accounts. (3) Our proposed method is a perfect complement for chitchat engines on informative utterances. The reasons for bad cases are two-fold: First, a DocChat response overlaps with a query, but cannot actually response it. For this issue, we need to refine the capability of our response ranking model on measuring causality relationships. Second, we wrongly send a chitchat query to DocChat, as currently, we only use a white list of chitchat queries for chitchat/non-chitchat classification (Section 5).

	Better	Worse	Tie
Compare to XiaoIce	58	47	51

Table 7: Chatbot side-by-side evaluation.

8 Conclusion

This paper presents a response retrieval method for chatbot engines based on unstructured documents. We evaluate our method on both question answering and chatbot scenarios, and obtain promising results. We leave better triggering component and multiple rounds of conversation handling to be addressed in our future work.

Acknowledgments

This paper is supported by Beijing Advanced Innovation Center for Imaging Technology (No.BAICIT-2016001), the National Natural Science Foundation of China (Grant Nos. 61170189, 61370126), National High Technology Research and Development Program of China under grant (No.2015AA016004), the Fund of the State Key Laboratory of Software Development Environment (No.SKLSDE-2015ZX-16).

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 597–604.

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

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1011–1019.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

K Sparck Jones, Steve Walker, and Stephen E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information Processing & Management*, 36(6):809–840.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 1:48–54.

Diane Litman, Satinder Singh, Michael Kearns, and Marilyn Walker. 2000. Njfun: a reinforcement learning spoken dialogue system. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, pages 17–20.

Yishu Miao, Lei Yu, and Phil Blunsom. 2015. Neural variational inference for text processing. *arXiv preprint arXiv:1511.06038*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NIPS)*, pages 3111–3119.

Ramesh Nallapati. 2004. Discriminative models for information retrieval. In *Proceedings of the international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 583–593.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(02):97–126.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1577–1586.

Mengqiu Wang and Christopher D Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 1164–1172.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 707–712.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 7, pages 22–32.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2013–2018.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1744–1753.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 643–648.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *NIPS Deep Learning and Representation Learning Workshop*.

Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, and Wei-Ying Ma. 2015. Lightlda: Big topic models on modest computer clusters. In *Proceedings of the Annual International Conference on World Wide Web (WWW)*, pages 1351–1361.