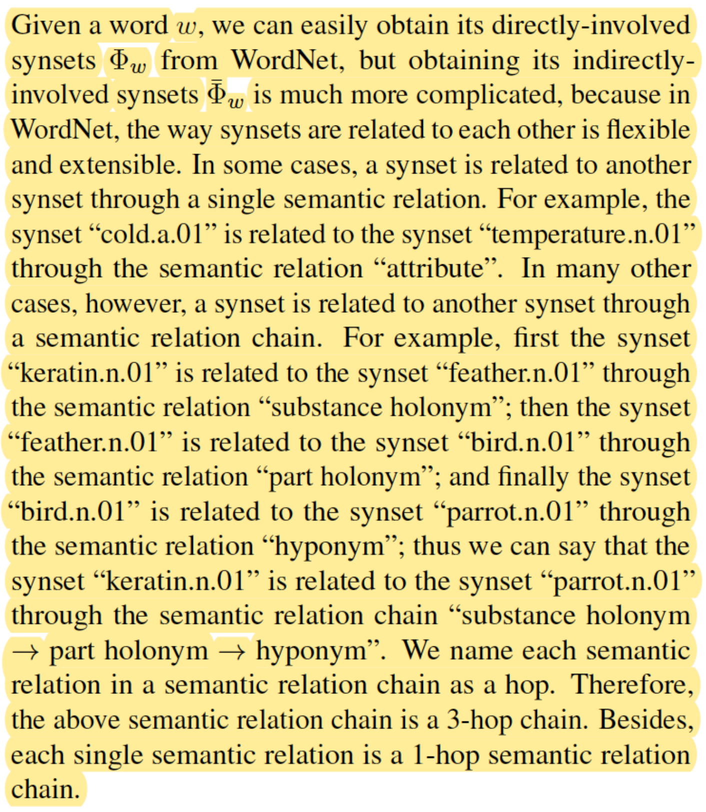
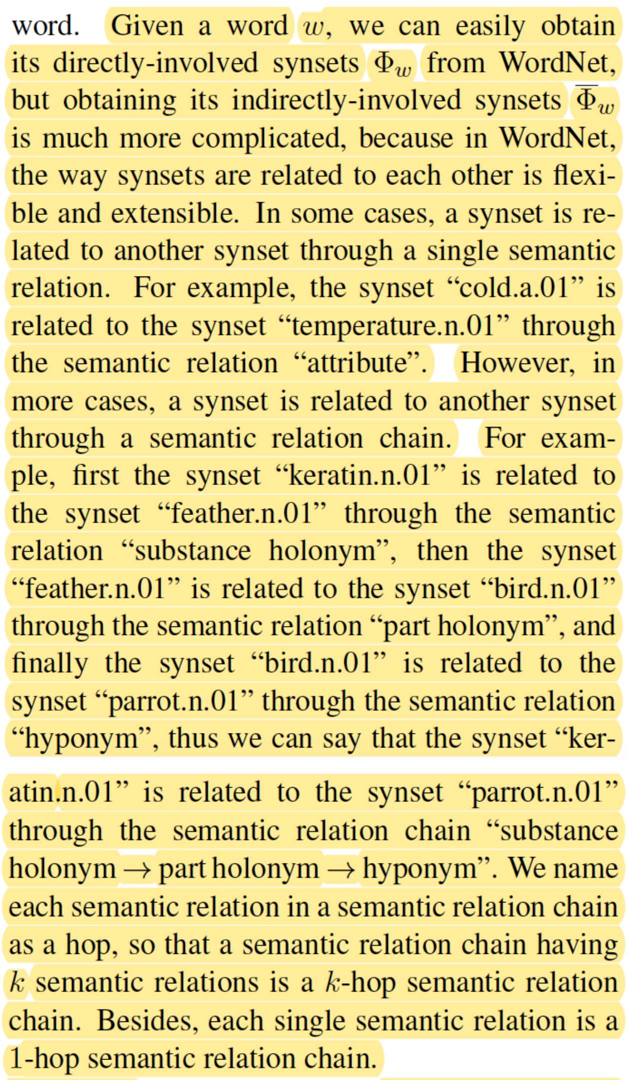
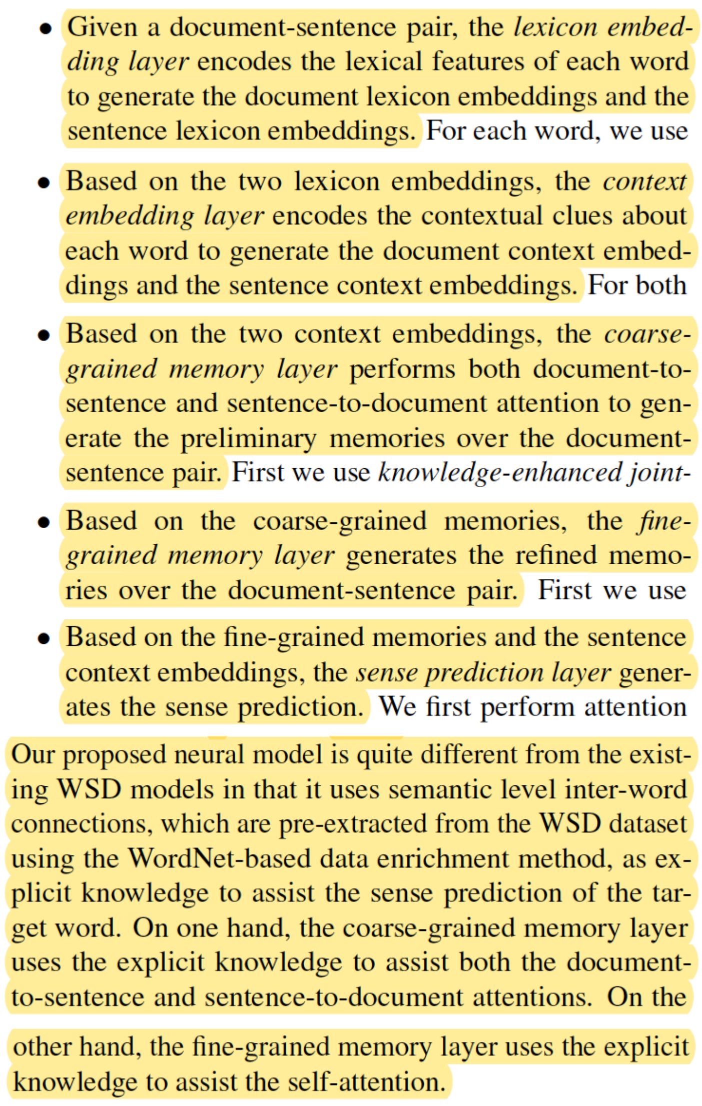
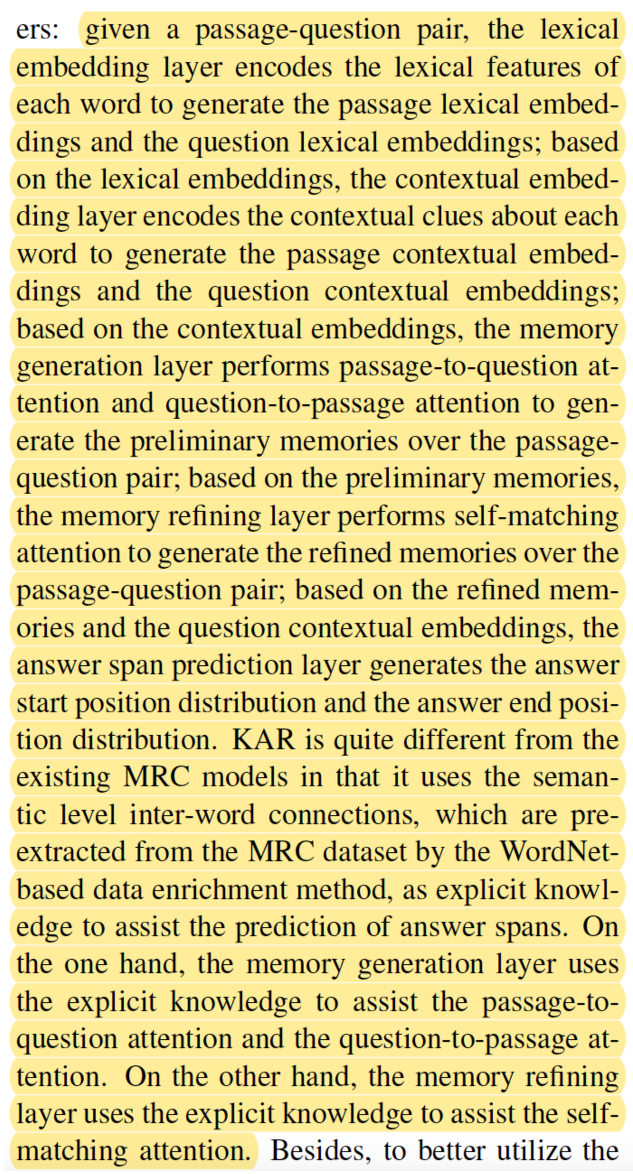
| Feng Wei's ICML 2020 paper<br>http://proceedings.mlr.press/v119/wei20a.html | The ACL 2019 paper<br>(EMNLP 2018 rejected version, submission ID 610) |
|---|---|
| 3.2.1. DATA ENRICHMENT WITH WORDNET<br><br>WordNet is a comprehensive lexical database for the English language (Miller, 1995), and is commonly used as the sense repository in WSD systems.<br><br>To provide our WSD model with explicit knowledge, we enrich the gloss information by extracting semantic level inter-word connections from each document-sentence pair in it; therefore we propose a WordNet-based data enrichment method.<br><br>Words in WordNet are organized into synsets, as shown in Figure 3, which in turn are related to each other through semantic relations, such as "hypernym" and "hyponym". In our data enrichment method, we use the semantic relations of WordNet to extract semantic level inter-word connections from each document-sentence pair in the WSD dataset. For each word $w$ in a document-sentence pair, we need to obtain a set $Z_w$, which contains the positions of the document words that $w$ is semantically connected to. Besides, when $w$ itself is a document word, we also need to ensure that its position is excluded from $Z_w$.<br><br>Given a word $w$, its directly-involved synsets $\Phi_w$ represents the synsets that $w$ belongs to, and its indirectly-involved synsets $\bar{\Phi}_w$ represents the synsets that are related to those in $\Phi_w$ through semantic relations. Based on the two concepts, we propose the following hypothesis: given a subject word $w_s$ and an object word $w_o$, $w_s$ is semantically connected to $w_o$ if and only if $(\Phi_{w_s} \bigcup \bar{\Phi}_{w_s}) \bigcap \Phi_{w_o} \neq \emptyset$. According to the hypothesis, Algorithm 1 describes the process of extracting semantic level inter-word connections from each document-sentence pair. | **3 WordNet-based Data Enrichment**<br><br>To provide our MRC model with explicit knowledge, we would like to enrich the content of the MRC dataset by extracting semantic level inter-word connections from each passage-question pair in it, therefore we propose a WordNet-based data enrichment method.<br><br>**3.1 What and how to extract from each passage-question pair**<br><br>WordNet is a lexical database for English. Words in WordNet are organized into synsets, which in turn are related to each other through semantic relations, such as "hypernym" and "hyponym". In our data enrichment method, we use the semantic relations of WordNet to extract semantic level inter-word connections from each passage-question pair in the MRC dataset. Considering the requirements of our MRC model, we need to represent the extraction results as positional information. Specifically, for each word $w$ in a passage-question pair, we need to obtain a set $Z_w$, which contains the positions of the passage words that $w$ is semantically connected to. Besides, when $w$ itself is a passage word, we also need to ensure that its position is excluded from $Z_w$.<br>The key problem to obtain the above extraction results is to determine if a subject word is semantically connected to an object word. To solve this problem, we introduce two concepts: the directly-involved synsets and indirectly-involved synsets of a word. Given a word $w$, its directly-involved synsets $\Phi_w$ represents the synsets that $w$ belongs to, and its indirectly-involved synsets $\overline{\Phi}_w$ represents the synsets that the synsets in $\Phi_w$ are related to through semantic relations. Based on the two concepts, we propose the following hypothesis: given a subject word $w_s$ and an object word $w_o$, $w_s$ is semantically connected to $w_o$ if and only if $(\Phi_{w_s} \cup \overline{\Phi}_{w_s}) \cap \Phi_{w_o} \neq \emptyset$. According to the hypothesis, Algorithm 1 describes the process of extracting semantic level inter-word connections from each passage-question pair. |

| Feng Wei's ICML 2020 paper<br>http://proceedings.mlr.press/v119/wei20a.html | The ACL 2019 paper<br>(EMNLP 2018 rejected version, submission ID 610) |
|---|---|
|  Given a word $w$, we can easily obtain its directly-involved synsets $\Phi_w$ from WordNet, but obtaining its indirectly-involved synsets $\bar{\Phi}_w$ is much more complicated, because in WordNet, the way synsets are related to each other is flexible and extensible. In some cases, a synset is related to another synset through a single semantic relation. For example, the synset “cold.a.01” is related to the synset “temperature.n.01” through the semantic relation “attribute”. In many other cases, however, a synset is related to another synset through a semantic relation chain. For example, first the synset “keratin.n.01” is related to the synset “feather.n.01” through the semantic relation “substance holonym”; then the synset “feather.n.01” is related to the synset “bird.n.01” through the semantic relation “part holonym”; and finally the synset “bird.n.01” is related to the synset “parrot.n.01” through the semantic relation “hyponym”; thus we can say that the synset “keratin.n.01” is related to the synset “parrot.n.01” through the semantic relation chain “substance holonym $\rightarrow$ part holonym $\rightarrow$ hyponym”. We name each semantic relation in a semantic relation chain as a hop. Therefore, the above semantic relation chain is a 3-hop chain. Besides, each single semantic relation is a 1-hop semantic relation chain. |  word. Given a word $w$, we can easily obtain its directly-involved synsets $\Phi_w$ from WordNet, but obtaining its indirectly-involved synsets $\overline{\Phi}_w$ is much more complicated, because in WordNet, the way synsets are related to each other is flexible and extensible. In some cases, a synset is related to another synset through a single semantic relation. For example, the synset “cold.a.01” is related to the synset “temperature.n.01” through the semantic relation “attribute”. However, in more cases, a synset is related to another synset through a semantic relation chain. For example, first the synset “keratin.n.01” is related to the synset “feather.n.01” through the semantic relation “substance holonym”, then the synset “feather.n.01” is related to the synset “bird.n.01” through the semantic relation “part holonym”, and finally the synset “bird.n.01” is related to the synset “parrot.n.01” through the semantic relation “hyponym”, thus we can say that the synset “keratin.n.01” is related to the synset “parrot.n.01” through the semantic relation chain “substance holonym $\rightarrow$ part holonym $\rightarrow$ hyponym”. We name each semantic relation in a semantic relation chain as a hop, so that a semantic relation chain having $k$ semantic relations is a $k$-hop semantic relation chain. Besides, each single semantic relation is a 1-hop semantic relation chain. |

| Feng Wei's ICML 2020 paper<br>http://proceedings.mlr.press/v119/wei20a.html | The ACL 2019 paper<br>(EMNLP 2018 rejected version, submission ID 610) |
|---|---|
| Let us use $\Gamma := \{\gamma_1, \gamma_2, ...\}$ to represent the semantic relations of WordNet, and use $\Omega_\phi^{\gamma_i}$ to represent the synsets that a synset $\phi$ is related to through a single semantic relation $\gamma_i \in \Gamma$. Since $\Omega_\phi^{\gamma_i}$ is easy to obtain from WordNet, we can further obtain the synsets that $\phi$ is related to through 1-hop semantic relation chains: $\Psi_\phi^1 = \bigcup_{\gamma_i \in \Gamma} \Omega_\phi^{\gamma_i}$, the synsets that $\phi$ is related to through 2-hop semantic relation chains: $\Psi_\phi^2 = \bigcup_{\hat{\phi} \in \Psi_\phi^1} \bigcup_{\gamma_i \in \Gamma} \Omega_{\hat{\phi}}^{\gamma_i}$, and by induction, the synsets that $\phi$ is related to through $k$-hop semantic relation chains: $\Psi_\phi^k = \bigcup_{\hat{\phi} \in \Psi_\phi^{k-1}} \bigcup_{\gamma_i \in \Gamma} \Omega_{\hat{\phi}}^{\gamma_i}$. In theory, if we do not limit the hop counts of semantic relation chains, $\phi$ can be related to all other synsets in WordNet, which is meaningless in many cases. Therefore, we use a hyper-parameter $\tau \in \mathbb{N}$ to represent the maximum hop count of semantic relation chains, and only consider the semantic relation chains that have no more than $\tau$ hops. Based on the above descriptions, given a word $w$ and its directly-involved synsets $\Phi_w$, we can obtain its indirectly-involved synsets: $\bar{\Phi}_w = \bigcup_{\phi \in \Phi_w} \bigcup_{k=1}^{\tau} \Psi_\phi^k$ | Let us use $\Gamma := \{\gamma_1, \gamma_2, \ldots\}$ to represent the semantic relations of WordNet, and use $\Omega_\phi^{\gamma_i}$ to represent the synsets that a synset $\phi$ is related to through a single semantic relation $\gamma_i \in \Gamma$. Since $\Omega_\phi^{\gamma_i}$ is easy to obtain from WordNet, we can further obtain the synsets that $\phi$ is related to through 1-hop semantic relation chains: $\Psi_\phi^1 = \bigcup_{\gamma_i \in \Gamma} \Omega_\phi^{\gamma_i}$, the synsets that $\phi$ is related to through 2-hop semantic relation chains: $\Psi_\phi^2 = \bigcup_{\hat{\phi} \in \Psi_\phi^1} \bigcup_{\gamma_i \in \Gamma} \Omega_{\hat{\phi}}^{\gamma_i}$, and by induction, the synsets that $\phi$ is related to through $k$-hop semantic relation chains: $\Psi_\phi^k = \bigcup_{\hat{\phi} \in \Psi_\phi^{k-1}} \bigcup_{\gamma_i \in \Gamma} \Omega_{\hat{\phi}}^{\gamma_i}$. In theory, if we do not limit the hop counts of semantic relation chains, $\phi$ can be related to all other synsets in WordNet, which is meaningless in many cases. Therefore, we use a hyper-parameter $\chi \in \mathbb{N}$ to represent the maximum hop count of semantic relation chains, and only consider the semantic relation chains that have no more than $\chi$ hops. Based on the above descriptions, given a word $w$ and its directly-involved synsets $\Phi_w$, we can obtain its indirectly-involved synsets: $\bar{\Phi}_w = \bigcup_{\phi \in \Phi_w} \bigcup_{k=1}^{\chi} \Psi_\phi^k$. |

| Feng Wei's ICML 2020 paper<br>http://proceedings.mlr.press/v119/wei20a.html | The ACL 2019 paper<br>(EMNLP 2018 rejected version, submission ID 610) |
|---|---|
|  • Given a document-sentence pair, the *lexicon embedding layer* encodes the lexical features of each word to generate the document lexicon embeddings and the sentence lexicon embeddings. For each word, we use<br>• Based on the two lexicon embeddings, the *context embedding layer* encodes the contextual clues about each word to generate the document context embeddings and the sentence context embeddings. For both<br>• Based on the two context embeddings, the *coarse-grained memory layer* performs both document-to-sentence and sentence-to-document attention to generate the preliminary memories over the document-sentence pair. First we use *knowledge-enhanced joint-*<br>• Based on the coarse-grained memories, the *fine-grained memory layer* generates the refined memories over the document-sentence pair. First we use<br>• Based on the fine-grained memories and the sentence context embeddings, the *sense prediction layer* generates the sense prediction. We first perform attention<br>Our proposed neural model is quite different from the existing WSD models in that it uses semantic level inter-word connections, which are pre-extracted from the WSD dataset using the WordNet-based data enrichment method, as explicit knowledge to assist the sense prediction of the target word. On one hand, the coarse-grained memory layer uses the explicit knowledge to assist both the document-to-sentence and sentence-to-document attentions. On the other hand, the fine-grained memory layer uses the explicit knowledge to assist the self-attention. |  ers: given a passage-question pair, the lexical embedding layer encodes the lexical features of each word to generate the passage lexical embeddings and the question lexical embeddings; based on the lexical embeddings, the contextual embedding layer encodes the contextual clues about each word to generate the passage contextual embeddings and the question contextual embeddings; based on the contextual embeddings, the memory generation layer performs passage-to-question attention and question-to-passage attention to generate the preliminary memories over the passage-question pair; based on the preliminary memories, the memory refining layer performs self-matching attention to generate the refined memories over the passage-question pair; based on the refined memories and the question contextual embeddings, the answer span prediction layer generates the answer start position distribution and the answer end position distribution. KAR is quite different from the existing MRC models in that it uses the semantic level inter-word connections, which are pre-extracted from the MRC dataset by the WordNet-based data enrichment method, as explicit knowledge to assist the prediction of answer spans. On the one hand, the memory generation layer uses the explicit knowledge to assist the passage-to-question attention and the question-to-passage attention. On the other hand, the memory refining layer uses the explicit knowledge to assist the self-matching attention. Besides, to better utilize the |

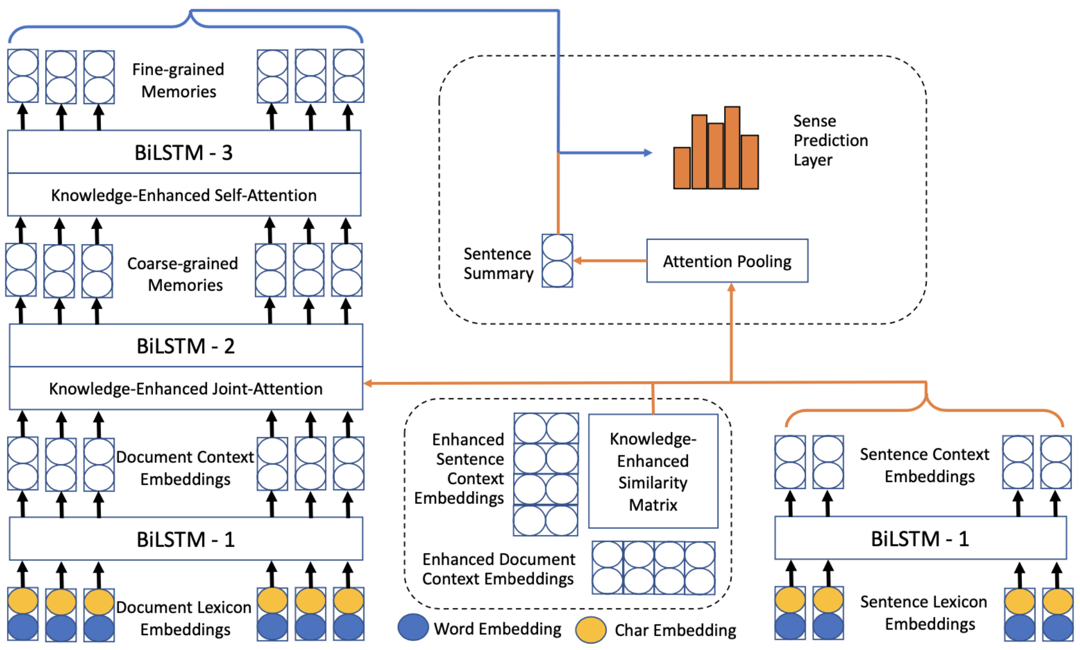
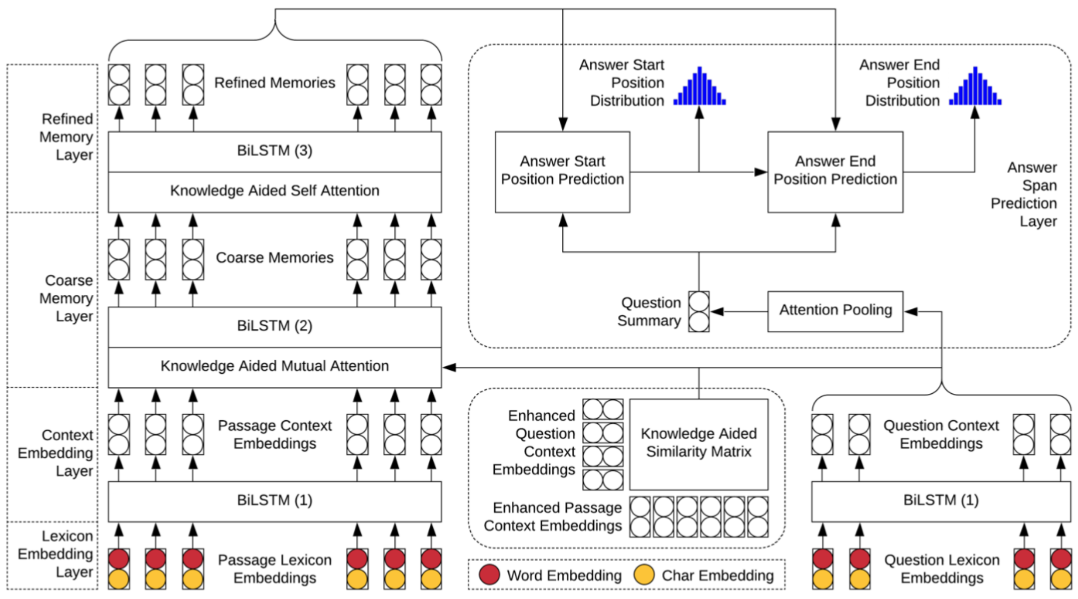
| Feng Wei’s ICML 2020 paper http://proceedings.mlr.press/v119/wei20a.html | The ACL 2019 paper https://www.aclweb.org/anthology/P19-1219 |
|---|---|
| into context embeddings. Moreover, we propose a data enrichment method, which uses WordNet to extract inter-word semantic connections as general knowledge from each given document. As shown in Example 1, such human semantic knowledge is essential to WSD tasks. In addition, we propose an end-to-end knowledge-based attentive neural WSD model, which explicitly uses the above extracted general knowledge to assist its attention mechanisms (Bahdanau | one hand, we propose a data enrichment method, which uses WordNet to extract inter-word semantic connections as general knowledge from each given passage-question pair. On the other hand, we propose an end-to-end MRC model named as Knowledge Aided Reader (KAR), which explicitly uses the above extracted general knowledge to assist its attention mechanisms. Based on the data |
| knowledge-based attentive model for WSD tasks. The key components of our model are the attention mechanisms, (i.e., knowledge-enhanced joint-attention and knowledge-enhanced self-attention). Knowledge-enhanced joint-attention aims to fuse the sentence representations into the document representations so as to obtain the sentence-aware document representations. Furthermore, Knowledge-enhanced self-attention aims to fuse the sentence-aware document representations into themselves so as to obtain the final document representations. More importantly, the most remarkable feature of our model is that it explicitly uses the general knowledge extracted by the data enrichment method to assist its attention mechanisms. | Knowledge Aided Reader (KAR). The key components of most existing MRC models are their attention mechanisms (Bahdanau et al., 2014), which are aimed at fusing the associated representations of each given passage-question pair. These attention mechanisms generally fall into two categories: the first one, which we name as mutual attention, is aimed at fusing the question representations into the passage representations so as to obtain the question-aware passage representations; the second one, which we name as self attention, is aimed at fusing the question-aware passage representations into themselves so as to obtain the final passage representations. Although KAR is equipped with both categories, its most remarkable feature is that it explicitly uses the general knowledge extracted by the data enrichment method to assist its attention mechanisms. There- |

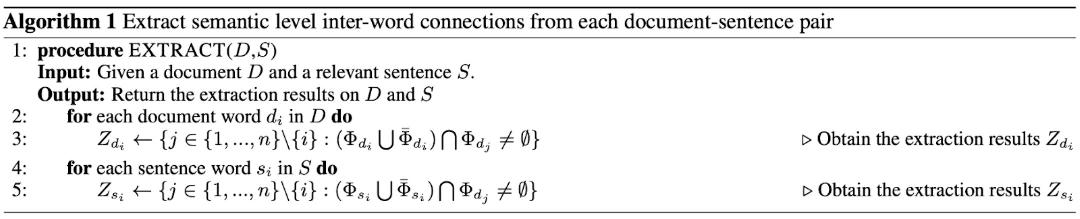
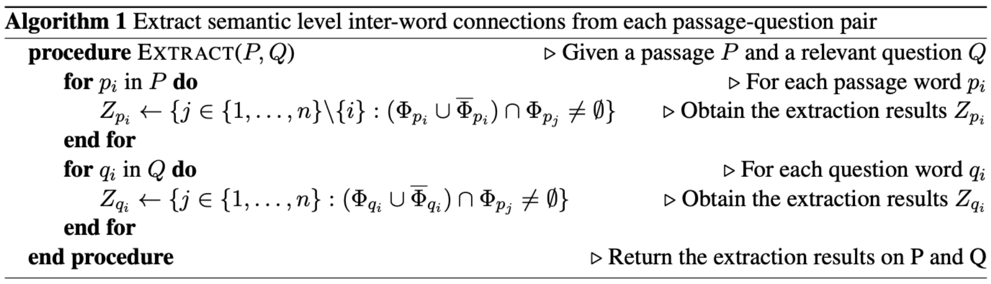
| Feng Wei's ICML 2020 paper<br>http://proceedings.mlr.press/v119/wei20a.html | The ACL 2019 paper<br>https://www.aclweb.org/anthology/P19-1219 |
|---|---|
| Given a document $D = \{d_1, ..., d_n\}$ and a relevant sentence $S = \{s_1, ..., s_m\}$, the task is to predict a sense among a list of $K$ candidates $\mathscr{C} = \{\mathscr{c}_1, ..., \mathscr{c}_k\}$. As depicted in Figure 4, our proposed end-to-end supervised WSD model consists of five layers:<br><br>• Given a document-sentence pair, the *lexicon embedding layer* encodes the lexical features of each word to generate the document lexicon embeddings and the sentence lexicon embeddings. For each word, we use our pre-trained context embeddings based on position-wise encoding approach and orthogonal framework described in 3.1, and obtain its character embedding with a Convolutional Neural Network (CNN) (Kim, 2014). For both the document and the sentence, we pass the concatenation of the word embeddings and the character embeddings through a shared dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the document lexicon embeddings $L_D \in \mathbb{R}^{d \times n}$ and the sentence lexicon embeddings $L_S \in \mathbb{R}^{d \times m}$.<br><br>• Based on the two lexicon embeddings, the *context embedding layer* encodes the contextual clues about each word to generate the document context embeddings and the sentence context embeddings. For both the document and the sentence, we process the lexicon embeddings (i.e., $L_D$ for the document and $L_S$ for the sentence) with a shared bidirectional LSTM (BiLSTM) (Hochreiter & Schmidhuber, 1997), whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the document context embeddings $C_D \in \mathbb{R}^{d \times n}$ and the sentence context embeddings $C_S \in \mathbb{R}^{d \times m}$.<br><br>• Based on the two context embeddings, the *coarse-grained memory layer* performs both document-to-sentence and sentence-to-document attention to generate the preliminary memories over the document-sentence pair. First we use *knowledge-enhanced joint-attention* (discussed in section 3.2.3) to fuse $C_D$ into $C_S$, the outputs of which are represented as $\tilde{G} \in \mathbb{R}^{d \times n}$. Then we process $\tilde{G}$ with a BiLSTM, whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the coarse-grained memories $G \in \mathbb{R}^{d \times n}$, which are the sentence-aware document representations.<br><br>• Based on the coarse-grained memories, the *fine-grained memory layer* generates the refined memories over the document-sentence pair. First we use *knowledge-enhanced self-attention* (discussed in section 3.2.4) to fuse $G$ into themselves, the outputs of which are represented as $\tilde{H} \in \mathbb{R}^{d \times n}$. Then we process $\tilde{H}$ with a BiLSTM, whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the fine-grained memories $H \in \mathbb{R}^{d \times n}$, which are the final document representations. | Given a passage $P = \{p_1, \ldots, p_n\}$ and a relevant question $Q = \{q_1, \ldots, q_m\}$, the task is to predict an answer span $[a_s, a_e]$, where $1 \leq a_s \leq a_e \leq n$, so that the resulting subsequence $\{p_{a_s}, \ldots, p_{a_e}\}$ from $P$ is an answer to $Q$.<br><br>**3.2 Overall Architecture**<br><br>As shown in Figure 1, KAR is an end-to-end MRC model consisting of five layers:<br>**Lexicon Embedding Layer.** This layer maps the words to the lexicon embeddings. The lexicon embedding of each word is composed of its word embedding and character embedding. For each word, we use the pre-trained GloVe (Pennington et al., 2014) word vector as its word embedding, and obtain its character embedding with a Convolutional Neural Network (CNN) (Kim, 2014). For both the passage and the question, we pass the concatenation of the word embeddings and the character embeddings through a shared dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the passage lexicon embeddings $L_P \in \mathbb{R}^{d \times n}$ and the question lexicon embeddings $L_Q \in \mathbb{R}^{d \times m}$.<br>**Context Embedding Layer.** This layer maps the lexicon embeddings to the context embeddings.<br><br>For both the passage and the question, we process the lexicon embeddings (i.e. $L_P$ for the passage and $L_Q$ for the question) with a shared bidirectional LSTM (BiLSTM) (Hochreiter and Schmidhuber, 1997), whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the passage context embeddings $C_P \in \mathbb{R}^{d \times n}$ and the question context embeddings $C_Q \in \mathbb{R}^{d \times m}$.<br><br>**Coarse Memory Layer.** This layer maps the context embeddings to the coarse memories. First we use knowledge aided mutual attention (introduced later) to fuse $C_Q$ into $C_P$, the outputs of which are represented as $\tilde{G} \in \mathbb{R}^{d \times n}$. Then we process $\tilde{G}$ with a BiLSTM, whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the coarse memories $G \in \mathbb{R}^{d \times n}$, which are the question-aware passage representations.<br>**Refined Memory Layer.** This layer maps the coarse memories to the refined memories. First we use knowledge aided self attention (introduced later) to fuse $G$ into themselves, the outputs of which are represented as $\tilde{H} \in \mathbb{R}^{d \times n}$. Then we process $\tilde{H}$ with a BiLSTM, whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the refined memories $H \in \mathbb{R}^{d \times n}$, which are the final passage representations. |

| Feng Wei's ICML 2020 paper<br>http://proceedings.mlr.press/v119/wei20a.html | The ACL 2019 paper<br>https://www.aclweb.org/anthology/P19-1219 |
|---|---|
| **3.2.3. KNOWLEDGE-ENHANCED JOINT-ATTENTION**<br><br>As a part of the coarse-grained memory layer, knowledge-enhanced joint-attention is aimed at fusing the sentence context embeddings $C_S$ into the document context embeddings $C_D$, where the key problem is to calculate the similarity between each document context embedding $c_{d_i}$ (i.e., the $i$-th column in $C_D$ ) and each sentence context embedding $c_{s_j}$ (i.e., the $j$-th column in $C_S$). To solve this problem, we follow (Wang & Jiang, 2019) to incorporate a similarity function:<br><br>$f(c_{d_i}, c_{s_j}) = v_f^\top [c_{d_i}; c_{s_j}; c_{d_i} \odot c_{s_j}] \in \mathbb{R})$ (5)<br><br>where $v_f$ is a trainable parameter; $\odot$ represents element-wise multiplication. Since context embeddings contain high-level information, we believe that introducing the pre-extracted general knowledge into the calculation of such similarities will enhance the ability of the model to identify the boundaries of word senses and is helpful for the final disambiguation. Therefore, we use the pre-extracted general knowledge to construct the enhanced context embeddings. Specifically, for each word $w$, whose context embedding is $c_w$, to construct its enhanced context embedding $\tilde{c}_w$, first recall that we have extracted a set $E_w$, which includes the positions of the document words that $w$ is semantically connected to, thus by gathering the columns in $C_D$ whose indexes are given by $E_w$, we obtain the matching context embeddings $Z \in \mathbb{R}^{d \times \lvert E_w \rvert}$. Then by constructing a $c_w$-attentive summary of $Z$, we obtain the matching vector $c_w^+$ (if $E_w = \emptyset$, which makes $Z = \{\}$, we will set $c_w^+ = 0$):<br><br>$t_i = v_c^\top \tanh(W_c z_i + U_c c_w) \in \mathbb{R}$ (6)<br><br>$c_w^+ = Z \,\text{softmax}(\{t_1, ..., t_{\lvert E_w \rvert}\}) \in \mathbb{R}^d$ (7)<br><br>where $v_c$, $W_c$, and $U_c$ are trainable parameters; $z_i$ represents the $i$-th column in $Z$. Finally we pass the concatenation of $c_w$ and $c_w^+$ through a dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the enhanced context embedding $\tilde{c}_w \in \mathbb{R}^d$.<br><br>Based on this context embeddings, to perform knowledge-enhanced joint-attention, first we construct a knowledge-enhanced similarity matrix $A \in \mathbb{R}^{n \times m}$, where each element $A_{i,j} = f(\tilde{c}_{d_i}, \tilde{c}_{s_j})$. Then we construct the document-attentive sentence summaries $R_S$ and the sentence-attentive document summaries $R_D$ :<br><br>$R_S = C_S \,\text{softmax}_r^\top(A) \in \mathbb{R}^{d \times n}$ (8)<br><br>$R_D = C_D \,\text{softmax}_c(A) \,\text{softmax}_r^\top(A) \in \mathbb{R}^{d \times n}$ (9)<br><br>where $\text{softmax}_r$ represents softmax along the row dimension and $\text{softmax}_c$ along the column dimension. Finally we pass the concatenation of $C_D$, $R_S$, $C_D \odot R_S$, and $R_D \odot R_S$ through a dense layer with ReLU activation, whose output dimensionality is $d$. Thus, we obtain the outputs $\tilde{G} \in \mathbb{R}^{d \times n}$. | **3.3 Knowledge Aided Mutual Attention**<br><br>As a part of the coarse memory layer, knowledge aided mutual attention is aimed at fusing the question context embeddings $C_Q$ into the passage context embeddings $C_P$, where the key problem is to calculate the similarity between each passage context embedding $c_{p_i}$ (i.e. the $i$-th column in $C_P$) and each question context embedding $c_{q_j}$ (i.e. the $j$-th column in $C_Q$). To solve this problem, Seo et al. (2016) proposed a similarity function:<br><br>$f(c_{p_i}, c_{q_j}) = v_f^\top [c_{p_i}; c_{q_j}; c_{p_i} \odot c_{q_j}] \in \mathbb{R}$<br><br>where $v_f$ is a trainable parameter; $\odot$ represents element-wise multiplication. This similarity function has also been adopted by several other works (Clark and Gardner, 2017; Yu et al., 2018). However, since context embeddings contain high-level information, we believe that introducing the pre-extracted general knowledge into the calculation of such similarities will make the results more reasonable. Therefore we modify the above similarity function to the following form:<br><br>$f^*(c_{p_i}, c_{q_j}) = v_f^\top [c^*_{p_i}; c^*_{q_j}; c^*_{p_i} \odot c^*_{q_j}] \in \mathbb{R}$<br><br>where $c^*_x$ represents the enhanced context embedding of a word $x$. We use the pre-extracted general knowledge to construct the enhanced context embeddings. Specifically, for each word $w$, whose context embedding is $c_w$, to construct its enhanced context embedding $c^*_w$, first recall that we have extracted a set $E_w$, which includes the positions of the passage words that $w$ is semantically connected to, thus by gathering the columns in $C_P$ whose indexes are given by $E_w$, we obtain the matching context embeddings $Z \in \mathbb{R}^{d \times \lvert E_w \rvert}$. Then by constructing a $c_w$-attended summary of $Z$, we obtain the matching vector $c_w^+$ (if $E_w = \emptyset$, which makes $Z = \{\}$, we will set $c_w^+ = 0$):<br><br>$t_i = v_c^\top \tanh(W_c z_i + U_c c_w) \in \mathbb{R}$<br><br>$c_w^+ = Z \,\text{softmax}(\{t_1, \ldots, t_{\lvert E_w \rvert}\}) \in \mathbb{R}^d$<br><br>where $v_c$, $W_c$, and $U_c$ are trainable parameters; $z_i$ represents the $i$-th column in $Z$. Finally we pass the concatenation of $c_w$ and $c_w^+$ through a dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the enhanced context embedding $c^*_w \in \mathbb{R}^d$.<br>Based on the modified similarity function and the enhanced context embeddings, to perform knowledge aided mutual attention, first we construct a knowledge aided similarity matrix $A \in \mathbb{R}^{n \times m}$, where each element $A_{i,j} = f^*(c_{p_i}, c_{q_j})$. Then following Yu et al. (2018), we construct the passage-attended question summaries $R_Q$ and the question-attended passage summaries $R_P$:<br><br>$R_Q = C_Q \,\text{softmax}_r^\top(A) \in \mathbb{R}^{d \times n}$<br><br>$R_P = C_P \,\text{softmax}_c(A) \,\text{softmax}_r^\top(A) \in \mathbb{R}^{d \times n}$<br><br>where $\text{softmax}_r$ represents softmax along the row dimension and $\text{softmax}_c$ along the column dimension. Finally following Clark and Gardner (2017), we pass the concatenation of $C_P$, $R_Q$, $C_P \odot R_Q$, and $R_P \odot R_Q$ through a dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the outputs $\tilde{G} \in \mathbb{R}^{d \times n}$. |

| Feng Wei's ICML 2020 paper<br>http://proceedings.mlr.press/v119/wei20a.html | The ACL 2019 paper<br>https://www.aclweb.org/anthology/P19-1219 |
|---|---|
| 3.2.4. KNOWLEDGE-ENHANCED SELF-ATTENTION | **3.4 Knowledge Aided Self Attention** |
| As a part of the fine-grained memory layer, knowledge-enhanced self-attention is aimed at fusing the coarse-grained memories $G$ into themselves. We use the pre-extracted general knowledge to guarantee that the fusion of coarse-grained memories for each document word will only involve a precise subset of the other document words. Specifically, for each document word $d_i$, whose coarse-grained memory is $g_{d_i}$ (i.e., the $i$-th column in $G$), to perform the fusion of coarse-grained memories, first recall that we have extracted a set $E_{d_i}$, which includes the positions of the other document words that $d_i$ is semantically connected to, thus by gathering the columns in $G$ whose indexes are given by $E_{d_i}$, we obtain the matching coarse-grained memories $Z \in \mathbb{R}^{d\times\|E_{d_i}\|}$. Then by constructing a $g_{d_i}$-attentive summary of $Z$, we obtain the matching vector $g^+_{d_i}$ (if $E_{d_i} = \emptyset$, which makes $Z = \{\}$, we will set $g^+_{d_i} = 0$): | As a part of the refined memory layer, knowledge aided self attention is aimed at fusing the coarse memories $G$ into themselves. If we simply follow the self attentions of other works (Wang et al., 2017; Huang et al., 2017; Liu et al., 2017b; Clark and Gardner, 2017), then for each passage word $p_i$, we should fuse its coarse memory $g_{p_i}$ (i.e. the $i$-th column in $G$) with the coarse memories of all the other passage words. However, we believe that this is both unnecessary and distracting, since each passage word has nothing to do with many of the other passage words. Thus we use the pre-extracted general knowledge to guarantee that the fusion of coarse memories for each passage word will only involve a precise subset of the other passage words. Specifically, for each passage word $p_i$, whose coarse memory is $g_{p_i}$, to perform the fusion of coarse memories, first recall that we have extracted a set $E_{p_i}$, which includes the positions of the other passage words that $p_i$ is semantically connected to, thus by gathering the columns in $G$ whose indexes are given by $E_{p_i}$, we obtain the matching coarse memories $Z \in \mathbb{R}^{d\times\|E_{p_i}\|}$. Then by constructing a $g_{p_i}$-attended summary of $Z$, we obtain the matching vector $g^+_{p_i}$ (if $E_{p_i} = \emptyset$, which makes $Z = \{\}$, we will set $g^+_{p_i} = 0$): |
| $t_i = v_g^\top \tanh(W_g z_i + U_g g_{d_i}) \in \mathbb{R}$ (10)<br>$g^+_{d_i} = Z\, \mathrm{softmax}(\{t_1, ..., t_{\|E_{d_i}\|}\}) \in \mathbb{R}^d$ (11) | $t_i = v_g^\top \tanh(W_g z_i + U_g g_{p_i}) \in \mathbb{R}$<br>$g^+_{p_i} = Z\, \mathrm{softmax}(\{t_1, \ldots, t_{\|E_{p_i}\|}\}) \in \mathbb{R}^d$ |
| where $v_g$, $W_g$, and $U_g$ are trainable parameters. Finally we pass the concatenation of $g_{d_i}$ and $g^+_{d_i}$ through a dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the fusion result $\tilde{h}_{d_i} \in \mathbb{R}^d$, and further the outputs $\tilde{H} = \{\tilde{h}_{d_1}, ..., \tilde{h}_{d_n}\} \in \mathbb{R}^{d\times n}$. | where $v_g$, $W_g$, and $U_g$ are trainable parameters. Finally we pass the concatenation of $g_{p_i}$ and $g^+_{p_i}$ through a dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the fusion result $\tilde{h}_{p_i} \in \mathbb{R}^d$, and further the outputs $\tilde{H} = \{\tilde{h}_{p_1}, \ldots, \tilde{h}_{p_n}\} \in \mathbb{R}^{d\times n}$. |

| Feng Wei's ICML 2020 paper<br>http://proceedings.mlr.press/v119/wei20a.html | The ACL 2019 paper<br>https://www.aclweb.org/anthology/P19-1219 |
|---|---|
| First, some inter-word semantic connections are distracting for the disambiguation of word sense. For example, the inter-word semantic connection between "ballpoint" and "pen" makes no sense given the context "Little John was looking for his toy box. Finally, he found it. The box was in the pen." It is the knowledge-enhanced attention mechanisms that enable PoKED to ignore such distracting inter-word semantic connections so that only the important ones are used.<br><br>Second, PoKED is designed to utilize the pre-extracted inter-word semantic connections extracted by the data enrichment method. Several approaches have been presented in recent years to make use of ontologies or types (such as (Dasigi et al., 2017)) to represent word tokens based on knowledge bases (e.g., WordNet). Nevertheless, the importance of inter-word semantic connections from document-sentence pairs was overlooked. Therefore, in this paper, we explore an explicit (i.e., understandable and controllable) way to utilize general knowledge. Some inter-word semantic connections, especially those obtained through multi-hop semantic relation chains, enhance the ability of the model to identify the boundaries of word senses and is helpful for the final disambiguation.<br><br>Finally, an inter-word semantic connection extracted from a document-sentence pair usually also appears in many other document-sentence pairs; therefore it is very likely that the inter-word semantic connections extracted from a small number of training examples cover a larger amount of training examples. That is, we are using more training examples for model optimization than the available ones. | • KAR is designed to utilize the pre-extracted inter-word semantic connections from the data enrichment method. Some inter-word semantic connections, especially those obtained through multi-hop semantic relation chains, are very helpful for the prediction of answer spans, but they will be too covert to capture if we simply leverage recurrent neural networks (e.g. BiLSTM) and pre-trained word vectors (e.g. GloVe).<br><br>• An inter-word semantic connection extracted from a passage-question pair usually also appears in many other passage-question pairs, therefore it is very likely that the inter-word semantic connections extracted from a small amount of training examples actually cover a much larger amount of training examples. That is to say, we are actually using much more training examples for model optimization than the available ones.<br><br>• Some inter-word semantic connections are distracting for the prediction of answer spans. For example, the inter-word semantic connection between "bank" and "waterside" makes no sense given the context "the bank manager is walking along the waterside". It is the knowledge aided attention mechanisms that enable KAR to ignore such distracting inter-word semantic connections so that only the important ones are used. |

| | |
|---|---|
| Feng Wei's ICML 2020 paper http://proceedings.mlr.press/v119/wei20a.html |   Fine-grained Memories<br>BiLSTM - 3<br>Knowledge-Enhanced Self-Attention<br>Coarse-grained Memories<br>BiLSTM - 2<br>Knowledge-Enhanced Joint-Attention<br>Document Context Embeddings<br>BiLSTM - 1<br>Document Lexicon Embeddings<br>Sense Prediction Layer<br>Sentence Summary<br>Attention Pooling<br>Enhanced Sentence Context Embeddings<br>Knowledge-Enhanced Similarity Matrix<br>Enhanced Document Context Embeddings<br>Word Embedding<br>Char Embedding<br>Sentence Context Embeddings<br>BiLSTM - 1<br>Sentence Lexicon Embeddings *Figure 4.* Our proposed end-to-end supervised WSD model (KED). |
| The ACL 2019 paper https://www.aclweb.org/anthology/P19-1219 |   Refined Memory Layer<br>Refined Memories<br>BiLSTM (3)<br>Knowledge Aided Self Attention<br>Coarse Memory Layer<br>Coarse Memories<br>BiLSTM (2)<br>Knowledge Aided Mutual Attention<br>Context Embedding Layer<br>Passage Context Embeddings<br>BiLSTM (1)<br>Lexicon Embedding Layer<br>Passage Lexicon Embeddings<br>Answer Start Position Distribution<br>Answer End Position Distribution<br>Answer Start Position Prediction<br>Answer End Position Prediction<br>Answer Span Prediction Layer<br>Question Summary<br>Attention Pooling<br>Enhanced Question Context Embeddings<br>Knowledge Aided Similarity Matrix<br>Enhanced Passage Context Embeddings<br>Word Embedding<br>Char Embedding<br>Question Context Embeddings<br>BiLSTM (1)<br>Question Lexicon Embeddings Figure 1: An end-to-end MRC model: Knowledge Aided Reader (KAR) |

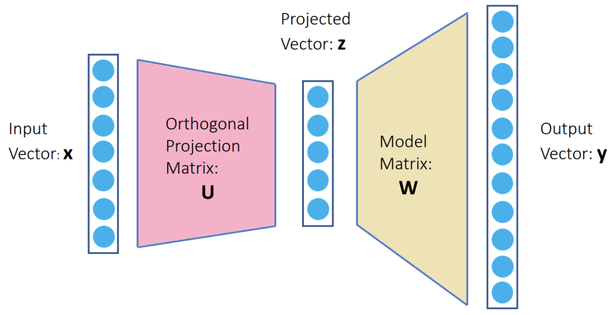
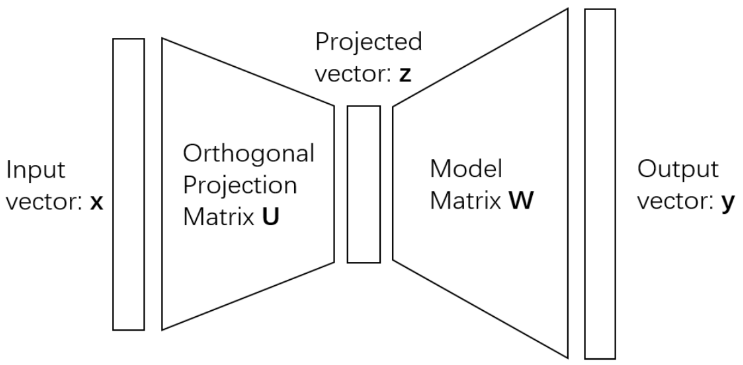
| | |
|---|---|
| Feng Wei's ICML 2020 paper http://proceedings.mlr.press/v119/wei20a.html |  |
| The ACL 2019 paper (EMNLP 2018 rejected version, submission ID 610) |  |

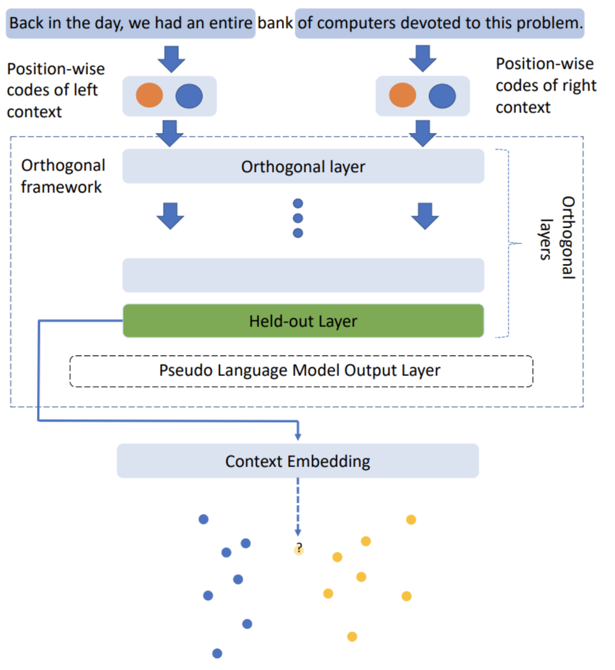
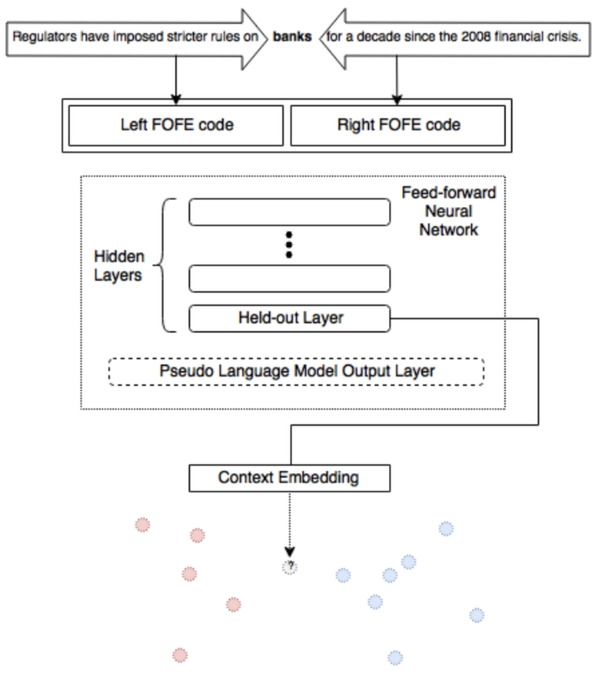
**Algorithm 1** Extract semantic level inter-word connections from each document-sentence pair

```
1: procedure EXTRACT($D$,$S$)
   Input: Given a document $D$ and a relevant sentence $S$.
   Output: Return the extraction results on $D$ and $S$
2:     for each document word $d_i$ in $D$ do
3:         $Z_{d_i} \leftarrow \{j \in \{1,...,n\} \backslash \{i\} : (\Phi_{d_i} \bigcup \bar{\Phi}_{d_i}) \bigcap \Phi_{d_j} \neq \emptyset\}$        ▷ Obtain the extraction results $Z_{d_i}$
4:     for each sentence word $s_i$ in $S$ do
5:         $Z_{s_i} \leftarrow \{j \in \{1,...,n\} \backslash \{i\} : (\Phi_{s_i} \bigcup \bar{\Phi}_{s_i}) \bigcap \Phi_{d_j} \neq \emptyset\}$        ▷ Obtain the extraction results $Z_{s_i}$
```

**Algorithm 1** Extract semantic level inter-word connections from each passage-question pair

```
procedure EXTRACT($P, Q$)        ▷ Given a passage $P$ and a relevant question $Q$
    for $p_i$ in $P$ do        ▷ For each passage word $p_i$
        $Z_{p_i} \leftarrow \{j \in \{1, \ldots, n\} \backslash \{i\} : (\Phi_{p_i} \cup \overline{\Phi}_{p_i}) \cap \Phi_{p_j} \neq \emptyset\}$        ▷ Obtain the extraction results $Z_{p_i}$
    end for
    for $q_i$ in $Q$ do        ▷ For each question word $q_i$
        $Z_{q_i} \leftarrow \{j \in \{1, \ldots, n\} : (\Phi_{q_i} \cup \overline{\Phi}_{q_i}) \cap \Phi_{p_j} \neq \emptyset\}$        ▷ Obtain the extraction results $Z_{q_i}$
    end for
end procedure        ▷ Return the extraction results on P and Q
```

| Feng Wei's ICML 2020 paper<br>http://proceedings.mlr.press/v119/wei20a.html | The ACML 2017 paper<br>http://proceedings.mlr.press/v77/pan17a.html |
|---|---|
| An example of an orthogonal layer in deep feedforward networks is shown in Figure 2. For one hidden layer with input vector $x$ ($x \in \mathbb{R}^D$) and output vector $y$ ($y \in \mathbb{R}^G$), it is first split into two layers:<br><br>• The first layer is a linear orthogonal projection layer, which is used to project $x$ to a feature vector $z$ ($z \in \mathbb{R}^M, M < D$) and remove the noise signals by using an orthogonal projection matrix $U : z = Ux$.<br><br>• The second layer is a non-linear model layer, which converts $z$ to the output vector $y$ following the selected model $f_k()$ and a nonlinear log-likelihood pruning operation.<br><br> Projected Vector: **z**<br>Input Vector: **x**<br>Orthogonal Projection Matrix: **U**<br>Model Matrix: **W**<br>Output Vector: **y**<br><br>*Figure 2.* The orthogonal framework is viewed as a hidden layer in deep feedforward networks. | be learned in either supervised or unsupervised ways. For one hidden layer with input vector $\mathbf{x}$ ($\mathbf{x} \in R^D$) and output vector $\mathbf{y}$ ($\mathbf{y} \in R^G$), it is first split into two layers: i) The first layer is a linear orthogonal projection layer, which is used to project $\mathbf{x}$ to a feature vector $\mathbf{z}$ ($\mathbf{z} \in R^M, M < D$) and remove the noise signals by using an orthogonal projection matrix $\mathbf{U}$:<br><br>$$\mathbf{z} = \mathbf{U}\mathbf{x}. \quad (3)$$<br><br>ii) The second layer is a non-linear model layer, which convert $\mathbf{z}$ to the output vector $\mathbf{y}$ following the selected model $f_k()$ and a nonlinear log-likelihood pruning operation (in supervised learning the model can be learned automatically from the training dataset). An example of a HOPE layer in DNNs is shown in Figure 1.<br><br> Projected vector: **z**<br>Input vector: **x**<br>Orthogonal Projection Matrix **U**<br>Model Matrix **W**<br>Output vector: **y**<br><br>Figure 1: The HOPE model is viewed as a hidden layer in DNNs. |

| Feng Wei’s ICML 2020 paper [http://proceedings.mlr.press/v119/wei20a.html](http://proceedings.mlr.press/v119/wei20a.html) | The arXiv paper [https://arxiv.org/abs/1902.10246](https://arxiv.org/abs/1902.10246) |
|---|---|
|   Back in the day, we had an entire bank of computers devoted to this problem.<br>Position-wise codes of left context<br>Position-wise codes of right context<br>Orthogonal framework<br>Orthogonal layer<br>Orthogonal layers<br>Held-out Layer<br>Pseudo Language Model Output Layer<br>Context Embedding<br>? *Figure 1.* The diagram of the unsupervised position-wise orthogonal (PoNet) language module. |   Regulators have imposed stricter rules on<br>banks<br>for a decade since the 2008 financial crisis.<br>Left FOFE code<br>Right FOFE code<br>Feed-forward Neural Network<br>Hidden Layers<br>Held-out Layer<br>Pseudo Language Model Output Layer<br>Context Embedding<br>? Figure 1: Context abstraction through FOFE-based pseudo language model and WSD classification over context embeddings |

# PoKED: A Semi-Supervised System for Word Sense Disambiguation

**Feng Wei** [1]

## Abstract

In this paper, we propose a semi-supervised neural system, named Position-wise Orthogonal Knowledge-Enhanced Disambiguator (PoKED), which effectively supports attention-driven, long-range dependency modeling for word sense disambiguation (WSD) tasks. The proposed PoKED system incorporates position-wise encoding into an orthogonal framework and applies a knowledge-based attentive neural model to solve the WSD problem. Our proposed unsupervised language model is trained over unlabeled corpora. Then the pre-trained language model is used to abstract the surrounding context of polysemous words in labeled corpora into context embeddings. We further use the semantic relations in the WordNet, by extracting semantic level inter-word connections from each document-sentence pair in the WSD dataset. Our experimental results from standard benchmarks show that our proposed system, PoKED, can achieve competitive performance compared with state-of-the-art knowledge-based WSD systems.

## 1. Introduction

Word Sense Disambiguation (WSD) is the task of mapping an ambiguous word in a given context to its correct meaning. For example, the word `bank` can either mean a `financial institution` or `sloping land`, based on different contexts. WSD is an important problem in natural language processing (NLP), both in its own right and as a stepping stone to more advanced tasks such as machine translation, information extraction and retrieval, and question answering. WSD, being "AI-complete" (Navigli, 2009), is still an open problem after over two decades of research. We can roughly distinguish between supervised and unsupervised approaches. Supervised methods require sense-annotated training data and are suitable for lexical sample WSD tasks when systems are required to disambiguate a restricted set of target words. The performance of supervised systems, however, is limited in WSD tasks as labeled data for the full lexicon is sparse and difficult to obtain. As WSD tasks are challenging and have practical applications, there has been interest in developing unsupervised systems. These systems only require an external knowledge source (e.g., WordNet) but no labeled training data.

**Example 1: An example of the importance of human semantic knowledge to the word sense disambiguation. Our experiment shows without incorporating human semantic knowledge, the model would wrongly predict a sense of the target word "document". We believe the reason is due to the difficulty for the model to find a direct relationship through the given word and plain document. In the example, we can find the answer because we know "document" is a hypernym of "information", or "information" is a hyponym of "document". The example is selected from SemEval-15.**

**Document:** *This document is a summary of the European Public Assessment Report (EPAR). It explains how the Committee for Medicinal Products for Human Use (CHMP) assessed the studies performed, to reach their recommendations on how to use the medicine. If you need more information about your medical condition or your treatment, read the Package Leaflet (also part of the EPAR) or contact your doctor or pharmacist. If you want more information on the basis of the CHMP recommendation, read the Scientific Discussion (also part of the EPAR).* . . .

**Sentence:** *This document is a summary of the European Public Assessment Report (EPAR).*

**Answer:** *<noun.communication>[10] S: (n) document.01 (document%1:10:00::), written document.01 (written_document%1:10:00::), papers.01 (papers%1:10:00::) (writing that provides information (especially information of an official nature))*

In this paper, we propose a semi-supervised neural system for WSD tasks, which utilizes the whole document as the context for a word, rather than just the encompassing sentence used by most WSD systems. In order to model the whole document for WSD, we propose to incorporate recent position-wise encoding (Watcharawittayakul et al., 2018; Wei et al., 2019) into an orthogonal framework (Zhang et al., 2016; Wei et al., 2020). Our proposed model is used to train a pseudo-language model over unlabeled corpora. The pre-trained language model is then used to abstract the surrounding context of polysemous words in labeled corpora into context embeddings. Moreover, we propose a data enrichment method, which uses WordNet to extract inter-word

[1]Department of Electrical Engineering and Computer Science, York University, 4700 Keele St, Toronto, ON M3J 1P3, Canada. Correspondence to: Feng Wei <fwei@cse.yorku.ca>.

*Proceedings of the 37th International Conference on Machine Learning*, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

semantic connections as general knowledge from each given document. As shown in Example 1, such human semantic knowledge is essential to WSD tasks. In addition, we propose an end-to-end knowledge-based attentive neural WSD model, which explicitly uses the above extracted general knowledge to assist its attention mechanisms (Bahdanau et al., 2015). Our semi-supervised WSD system (PoKED), comprised of one unsupervised position-wise orthogonal network (PoNet) and one supervised knowledge-enhanced word sense disambiguator (KED) based on attentive networks. We evaluate our system, PoKED, on standard benchmarks (Raganato et al., 2017) and show that the proposed model, utilizing the whole document as the context for a word to be disambiguated, achieves better performance than the previous state-of-the-art knowledge-based models.

## 2. Related Work

Word sense disambiguation (WSD) approaches can be divided into two main categories: supervised, which require human intervention in the creation of sense-annotated datasets, and the so-called knowledge-based approach (Navigli, 2009), which requires the construction of a task-independent lexical-semantic knowledge resource. Once that work is available, it uses models that are completely autonomous.

**Supervised.** A popular system, *It makes sense* (Zhong & Ng, 2010), takes advantage of standard WSD features such as POS-tags, word co-occurrences, and collocations and creates individual support vector machine classifiers for each ambiguous word. Newer supervised models use deep neural networks and especially long short-term memory (LSTM) networks, a type of recurrent neural network particularly suitable for handling arbitrary-length sequences. (Yuan et al., 2016) proposed a deep neural model trained with large amounts of data obtained in a semi-supervised fashion. This model was re-implemented by (Le et al., 2018), reaching comparable results with a smaller training corpus. (Raganato et al., 2017) introduced two approaches for neural WSD using models developed for machine translation and substituting translated words with sense-annotated ones. (Luo et al., 2018) proposed combining labeled data and knowledge-based information in recent work. (Uslu et al., 2018) proposed *fastSense*, a model inspired by *fastText* (Joulin et al., 2017) which, rather than predicting context words, predicts word senses. More recently, (Huang et al., 2019) constructed context-gloss pairs and propose three BERT-based models for WSD. (Hadiwinoto et al., 2019) explored different strategies of integrating pre-trained contextualized word representations.

**Knowledge-based.** Knowledge-based models, instead, use the structural properties of a lexical-semantic knowledge base, and typically use the relational information between concepts in the semantic graph together with the lexical information contained therein (Navigli & Lapata, 2009). A popular algorithm used to select the sense of each word in this graph is PageRank (Page et al., 1999) that performs random walks over the network to identify the important nodes (Mihalcea et al., 2004). Another knowledge-based approach is Babelfy (Moro et al., 2014), which defines a semantic signature for a given context and compares it with all the candidate senses in order to perform the disambiguation task. (Chaplot & Salakhutdinov, 2018) proposed a method that uses the whole document as the context for the words to be disambiguated. It models word senses using a variant of the Latent Dirichlet Allocation framework (Blei et al., 2003), in which the topic distributions of the words are replaced with sense distributions modeled by a logistic normal distribution according to the frequencies obtained from WordNet. More recently, (Maru et al., 2019) introduced *SyntagNet*, a novel resource consisting of manually disambiguated lexical-semantic combinations. (Tripodi & Navigli, 2019) presented *WSDG*, a flexible game-theoretic model for WSD.

## 3. Position-wise Orthogonal Knowledge-Enhanced Disambiguator (PoKED)

In this section, we describe in detail the proposed semi-supervised neural system (PoKED) for WSD tasks. We aim to explore how position-wise embedding (unsupervised) could help the downstream WSD task, and how information from descriptive linguistic knowledge graphs (WordNet) can be incorporated into neural network architectures to improve the linguistic WSD task.

### 3.1. Unsupervised Language Model (PoNet)

First, we elaborate on the proposed unsupervised language model named PoNet.

The linguistic distribution hypothesis states that words that occur in close proximity should have a similar meaning. It implies that the particular sense of a polysemous word is highly related to its surrounding context. Moreover, humans decide the sense of a polysemous word by firstly understanding its occurring context (Harris, 1954). Following this theory, our proposed model has two stages: training a position-wise orthogonal network (PoNet) that abstracts context as embeddings as shown in Figure 1, and performing knowledge-based attentive WSD classification over pre-trained context embeddings as shown in Figure 4.

#### 3.1.1. Position-wise Encoding

Recently an alternative method (Watcharawittayakul et al., 2018; Wei et al., 2019) of commonly used sequence embed-

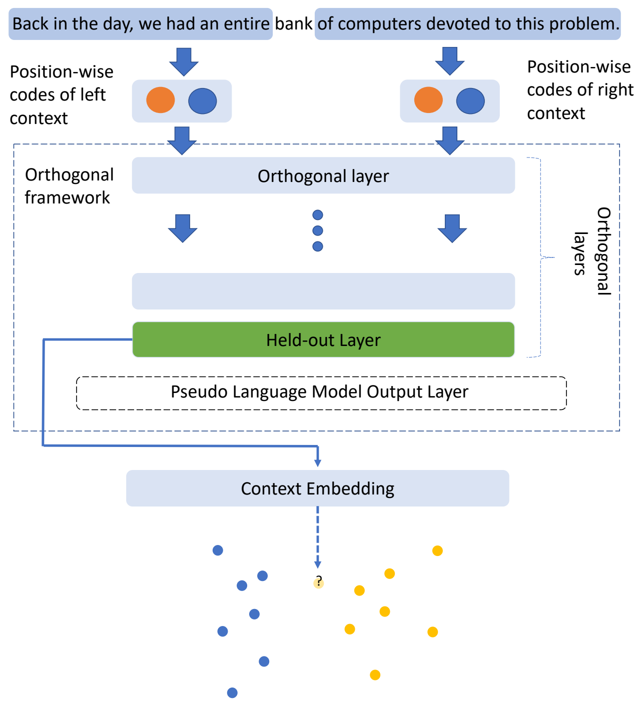

Back in the day, we had an entire bank of computers devoted to this problem.
Position-wise codes of left context
Position-wise codes of right context
Orthogonal framework
Orthogonal layer
Orthogonal layers
Held-out Layer
Pseudo Language Model Output Layer
Context Embedding
?

*Figure 1.* The diagram of the unsupervised position-wise orthogonal (PoNet) language module.

ding representations has been proposed, and it achieved competitive results in language modeling. The authors proved a nice theoretical property that guarantees that extracted codes can almost uniquely encode any variable-length sequence of words into a fixed-size representation without losing any information.

Given a vocabulary $V$, where each word can be represented by a 1-of-$|V|$ one-hot vector. Let $S = \{w_1, \cdots, w_N\}$ denote a sequence of $N$ words from $V$, and $e_n$ denote the one-hot vector of the $n$-th word in $S$, where $1 \leq n \leq N$. Assuming $z_0 = 0$, the extracted code $z_n$ of the sequence from word $w_1$ to $w_n$ is as follows:

$$z_n = \alpha \cdot z_{n-1} + e_n \tag{1}$$

where $\alpha$ is a constant forgetting factor. Thus, $z_n$ can be viewed as a fixed-size representation of the subsequence $\{w_1, \cdots, w_n\}$. We can see that, according to the theoretical properties presented in (Zhang et al., 2015), any sequence of variable length can be uniquely and losslessly encoded into a fixed-size representation.

The main idea of position-wise encoding is to generate augmented encoding codes by concatenating two codes using two different forgetting factors. Each of these codes is still computed in the same way as the mathematical formulation shown in Equation (1). By using two different forgetting factors in the two codes, we can represent both short-term and long-term dependencies. Hence, our position-wise encoding can maintain the sensitivity to both nearby and faraway context.

### 3.1.2. Orthogonal Framework

More recently, a novel orthogonal framework (Zhang et al., 2016; Wei et al., 2020) has been proposed to learn neural networks in either supervised or unsupervised way. This framework introduces a linear orthogonal projection to reduce the dimensionality of the raw high-dimension data and then uses a finite mixture distribution to model the extracted features. By splitting the feature extraction and data modeling into two separate stages, it can derive a good feature extraction model that can generate better low-dimension features for the further learning process. More importantly, based on the analysis in (Zhang et al., 2016), the orthogonal framework has a tight relationship with neural networks since each hidden layer can also be viewed as an orthogonal model being composed of the feature extraction stage and data modeling stage. Therefore, the maximum likelihood-based unsupervised learning as well as the minimum cross-entropy error based supervised learning algorithms can be used to learn neural networks under the orthogonal framework for deep learning. In this case, the standard back-propagation method can be used to optimize the objective function to learn the models except that the orthogonal constraints are imposed for all projection layers during the training procedure.

Simply put, in practice in terms of the orthogonal formulation, (Zhang et al., 2016) proposed modeling $z$, which is heavily de-correlated but may still exist in a rather high dimension feature space, with a finite mixture model:

$$p(z) = \sum_{k=1}^{K} \pi_k \cdot f_k(z|\theta_k) \tag{2}$$

where $K$ is the number of mixture components, $\pi_k$ is the mixture weight of the $k$-th component ($\sum_{k=1}^{K} \pi_k = 1$), $f_k()$ denotes a selected distribution from the exponential family, and $\theta_k$ denotes all model parameters of $f_k()$.

An example of an orthogonal layer in deep feedforward networks is shown in Figure 2. For one hidden layer with input vector $x$ ($x \in \mathbb{R}^D$) and output vector $y$ ($y \in \mathbb{R}^G$), it is first split into two layers:

- The first layer is a linear orthogonal projection layer, which is used to project $x$ to a feature vector $z$ ($z \in \mathbb{R}^M, M < D$) and remove the noise signals by using an orthogonal projection matrix $U : z = Ux$.
- The second layer is a non-linear model layer, which converts $z$ to the output vector $y$ following the selected model $f_k()$ and a nonlinear log-likelihood pruning operation.

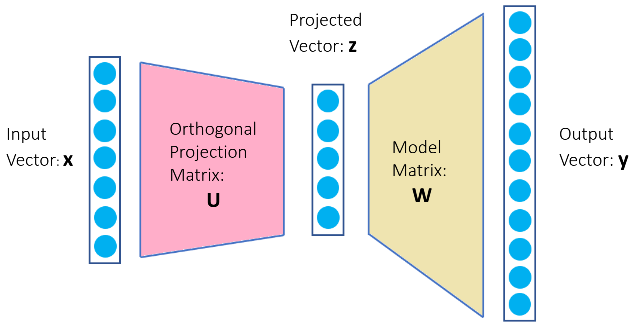


*Figure 2.* The orthogonal framework is viewed as a hidden layer in deep feedforward networks.

As the pseudo-language model (PoNet) is trained to predict the target word, the output layer is irrelevant to WSD tasks. The remaining layers, however, have learned the ability to generalize features from word to context during the training process. The held-out layer (the second last layer) are retained as context embeddings, which provides an effective representation of the surrounding context of a given target word.

### 3.2. Supervised Knowledge-based Attentive Model (KED)

In this section, we describe in detail the proposed supervised knowledge-enhanced attentive networks named KED for WSD tasks, along with the data enrichment method with WordNet.

#### 3.2.1. Data Enrichment with WordNet

WordNet is a comprehensive lexical database for the English language (Miller, 1995), and is commonly used as the sense repository in WSD systems.

To provide our WSD model with explicit knowledge, we enrich the gloss information by extracting semantic level inter-word connections from each document-sentence pair in it; therefore we propose a WordNet-based data enrichment method.

Words in WordNet are organized into synsets, as shown in Figure 3, which in turn are related to each other through semantic relations, such as "hypernym" and "hyponym". In our data enrichment method, we use the semantic relations of WordNet to extract semantic level inter-word connections from each document-sentence pair in the WSD dataset. For each word $w$ in a document-sentence pair, we need to obtain a set $Z_w$, which contains the positions of the document words that $w$ is semantically connected to. Besides, when $w$ itself is a document word, we also need to ensure that its position is excluded from $Z_w$.

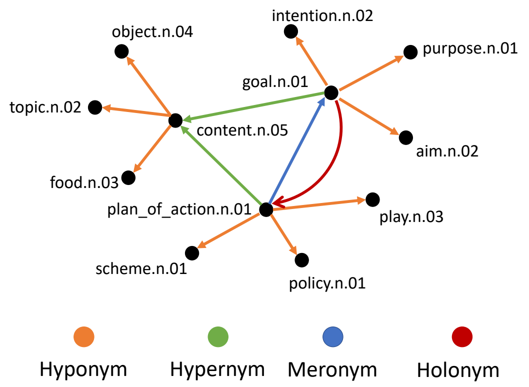


*Figure 3.* WordNet example showing several synsets and the relations between them.

Given a word $w$, its directly-involved synsets $\Phi_w$ represents the synsets that $w$ belongs to, and its indirectly-involved synsets $\bar{\Phi}_w$ represents the synsets that are related to those in $\Phi_w$ through semantic relations. Based on the two concepts, we propose the following hypothesis: given a subject word $w_s$ and an object word $w_o$, $w_s$ is semantically connected to $w_o$ if and only if $(\Phi_{w_s} \bigcup \bar{\Phi}_{w_s}) \bigcap \Phi_{w_o} \neq \emptyset$. According to the hypothesis, Algorithm 1 describes the process of extracting semantic level inter-word connections from each document-sentence pair.

Given a word $w$, we can easily obtain its directly-involved synsets $\Phi_w$ from WordNet, but obtaining its indirectly-involved synsets $\bar{\Phi}_w$ is much more complicated, because in WordNet, the way synsets are related to each other is flexible and extensible. In some cases, a synset is related to another synset through a single semantic relation. For example, the synset "cold.a.01" is related to the synset "temperature.n.01" through the semantic relation "attribute". In many other cases, however, a synset is related to another synset through a semantic relation chain. For example, first the synset "keratin.n.01" is related to the synset "feather.n.01" through the semantic relation "substance holonym"; then the synset "feather.n.01" is related to the synset "bird.n.01" through the semantic relation "part holonym"; and finally the synset "bird.n.01" is related to the synset "parrot.n.01" through the semantic relation "hyponym"; thus we can say that the synset "keratin.n.01" is related to the synset "parrot.n.01" through the semantic relation chain "substance holonym $\rightarrow$ part holonym $\rightarrow$ hyponym". We name each semantic relation in a semantic relation chain as a hop. Therefore, the above semantic relation chain is a 3-hop chain. Besides, each single semantic relation is a 1-hop semantic relation chain.

Let us use $\Gamma := \{\gamma_1, \gamma_2, ...\}$ to represent the semantic relations of WordNet, and use $\Omega_{\phi}^{\gamma_i}$ to represent the synsets that a synset $\phi$ is related to through a single semantic relation

**Algorithm 1** Extract semantic level inter-word connections from each document-sentence pair

```
1: procedure EXTRACT($D$,$S$)
   Input: Given a document $D$ and a relevant sentence $S$.
   Output: Return the extraction results on $D$ and $S$
2:     for each document word $d_i$ in $D$ do
3:         $Z_{d_i} \leftarrow \{j \in \{1, ..., n\} \backslash \{i\} : (\Phi_{d_i} \bigcup \bar{\Phi}_{d_i}) \bigcap \Phi_{d_j} \neq \emptyset\}$    ▷ Obtain the extraction results $Z_{d_i}$
4:     for each sentence word $s_i$ in $S$ do
5:         $Z_{s_i} \leftarrow \{j \in \{1, ..., n\} \backslash \{i\} : (\Phi_{s_i} \bigcup \bar{\Phi}_{s_i}) \bigcap \Phi_{d_j} \neq \emptyset\}$    ▷ Obtain the extraction results $Z_{s_i}$
```

$\gamma_i \in \Gamma$. Since $\Omega_\phi^{\gamma_i}$ is easy to obtain from WordNet, we can further obtain the synsets that $\phi$ is related to through 1-hop semantic relation chains: $\Psi_\phi^1 = \bigcup_{\gamma_i \in \Gamma} \Omega_\phi^{\gamma_i}$, the synsets that $\phi$ is related to through 2-hop semantic relation chains: $\Psi_\phi^2 = \bigcup_{\hat{\phi} \in \Psi_\phi^1} \bigcup_{\gamma_i \in \Gamma} \Omega_{\hat{\phi}}^{\gamma_i}$, and by induction, the synsets that $\phi$ is related to through $k$-hop semantic relation chains: $\Psi_\phi^k = \bigcup_{\hat{\phi} \in \Psi_\phi^{k-1}} \bigcup_{\gamma_i \in \Gamma} \Omega_{\hat{\phi}}^{\gamma_i}$. In theory, if we do not limit the hop counts of semantic relation chains, $\phi$ can be related to all other synsets in WordNet, which is meaningless in many cases. Therefore, we use a hyper-parameter $\tau \in \mathbb{N}$ to represent the maximum hop count of semantic relation chains, and only consider the semantic relation chains that have no more than $\tau$ hops. Based on the above descriptions, given a word $w$ and its directly-involved synsets $\Phi_w$, we can obtain its indirectly-involved synsets: $\bar{\Phi}_w = \bigcup_{\phi \in \Phi_w} \bigcup_{k=1}^{\tau} \Psi_\phi^k$

#### 3.2.2. Knowledge-based Attentive Neural Model

In this section, we describe in detail the proposed knowledge-based attentive model for WSD tasks. The key components of our model are the attention mechanisms, (i.e., knowledge-enhanced joint-attention and knowledge-enhanced self-attention). Knowledge-enhanced joint-attention aims to fuse the sentence representations into the document representations so as to obtain the sentence-aware document representations. Furthermore, Knowledge-enhanced self-attention aims to fuse the sentence-aware document representations into themselves so as to obtain the final document representations. More importantly, the most remarkable feature of our model is that it explicitly uses the general knowledge extracted by the data enrichment method to assist its attention mechanisms.

Given a document $D = \{d_1, ..., d_n\}$ and a relevant sentence $S = \{s_1, ..., s_m\}$, the task is to predict a sense among a list of $K$ candidates $\mathscr{C} = \{c_1, ..., c_k\}$. As depicted in Figure 4, our proposed end-to-end supervised WSD model consists of five layers:

- Given a document-sentence pair, the *lexicon embedding layer* encodes the lexical features of each word to generate the document lexicon embeddings and the sentence lexicon embeddings. For each word, we use our pre-trained context embeddings based on position-wise encoding approach and orthogonal framework described in 3.1, and obtain its character embedding with a Convolutional Neural Network (CNN) (Kim, 2014). For both the document and the sentence, we pass the concatenation of the word embeddings and the character embeddings through a shared dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the document lexicon embeddings $L_D \in \mathbb{R}^{d\times n}$ and the sentence lexicon embeddings $L_S \in \mathbb{R}^{d\times m}$.

- Based on the two lexicon embeddings, the *context embedding layer* encodes the contextual clues about each word to generate the document context embeddings and the sentence context embeddings. For both the document and the sentence, we process the lexicon embeddings (i.e., $L_D$ for the document and $L_S$ for the sentence) with a shared bidirectional LSTM (BiLSTM) (Hochreiter & Schmidhuber, 1997), whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the document context embeddings $C_D \in \mathbb{R}^{d\times n}$ and the sentence context embeddings $C_S \in \mathbb{R}^{d\times m}$.

- Based on the two context embeddings, the *coarse-grained memory layer* performs both document-to-sentence and sentence-to-document attention to generate the preliminary memories over the document-sentence pair. First we use *knowledge-enhanced joint-attention* (discussed in section 3.2.3) to fuse $C_D$ into $C_S$, the outputs of which are represented as $\tilde{G} \in \mathbb{R}^{d\times n}$. Then we process $\tilde{G}$ with a BiLSTM, whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the coarse-grained memories $G \in \mathbb{R}^{d\times n}$, which are the sentence-aware document representations.

- Based on the coarse-grained memories, the *fine-grained memory layer* generates the refined memories over the document-sentence pair. First we use *knowledge-enhanced self-attention* (discussed in section 3.2.4) to fuse $G$ into themselves, the outputs of which are represented as $\tilde{H} \in \mathbb{R}^{d\times n}$. Then we process $\tilde{H}$ with a BiLSTM, whose hidden state dimensionality

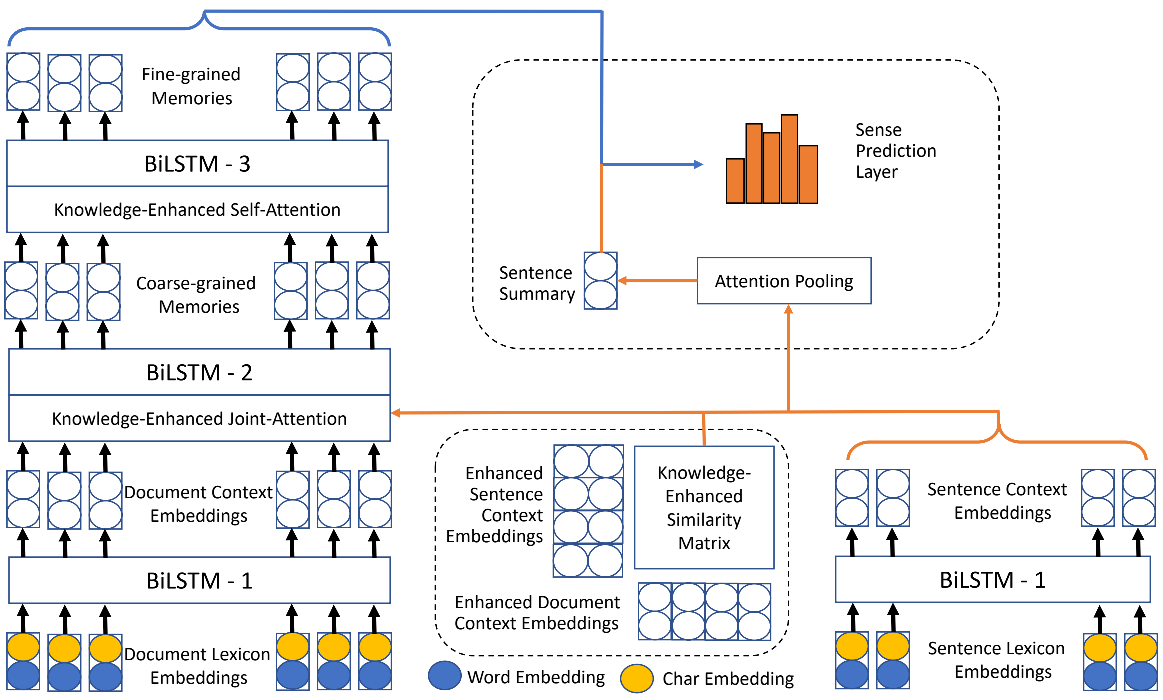

Fine-grained Memories
BiLSTM - 3
Knowledge-Enhanced Self-Attention
Coarse-grained Memories
BiLSTM - 2
Knowledge-Enhanced Joint-Attention
Document Context Embeddings
BiLSTM - 1
Document Lexicon Embeddings
Sense Prediction Layer
Sentence Summary
Attention Pooling
Enhanced Sentence Context Embeddings
Knowledge-Enhanced Similarity Matrix
Enhanced Document Context Embeddings
Word Embedding
Char Embedding
Sentence Context Embeddings
BiLSTM - 1
Sentence Lexicon Embeddings

*Figure 4.* Our proposed end-to-end supervised WSD model (KED).

is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the fine-grained memories $H \in \mathbb{R}^{d\times n}$, which are the final document representations.

- Based on the fine-grained memories and the sentence context embeddings, the *sense prediction layer* generates the sense prediction. We first perform attention pooling on $C_S$ to obtain a summary of the sentence:

$$\epsilon = C_S \text{softmax}(\tanh(W_\epsilon C_S)^\top v_\epsilon) \in \mathbb{R}^{2d} \quad (3)$$

where $W_\epsilon$ and $v_\epsilon$ are trainable parameters. Then, with $\epsilon$ as a query, we compute a posterior distribution of all sense candidates in the list:

$$o = \text{softmax}(\tanh(W_\eta \epsilon H))^\top v_\eta) \in \mathbb{R}^n \quad (4)$$

where $W_\eta$ and $v_\eta$ are trainable parameters. Thus, for the training, we minimize $-\log(o, t)$ on each training sample whose labeled sense is $t$. For the inference, we take the index of the maximum element as $o$.

### 3.2.3. Knowledge-Enhanced Joint-Attention

As a part of the coarse-grained memory layer, knowledge-enhanced joint-attention is aimed at fusing the sentence context embeddings $C_S$ into the document context embeddings $C_D$, where the key problem is to calculate the similarity between each document context embedding $c_{d_i}$ (i.e., the $i$-th column in $C_D$ ) and each sentence context embedding $c_{s_j}$ (i.e., the $j$-th column in $C_S$). To solve this problem, we follow (Wang & Jiang, 2019) to incorporate a similarity function:

$$f(c_{d_i}, c_{s_j}) = v_f^\top [c_{d_i}; c_{s_j}; c_{d_i} \odot c_{s_j}] \in \mathbb{R}) \quad (5)$$

where $v_f$ is a trainable parameter; $\odot$ represents element-wise multiplication. Since context embeddings contain high-level information, we believe that introducing the pre-extracted general knowledge into the calculation of such similarities will enhance the ability of the model to identify the boundaries of word senses and is helpful for the final disambiguation. Therefore, we use the pre-extracted general knowledge to construct the enhanced context embeddings. Specifically, for each word $w$, whose context embedding is $c_w$, to construct its enhanced context embedding $\tilde{c}_w$, first recall that we have extracted a set $E_w$, which includes the positions of the document words that $w$ is semantically connected to, thus by gathering the columns in $C_D$ whose indexes are given by $E_w$, we obtain the matching context embeddings $Z \in \mathbb{R}^{d\times|E_w|}$. Then by constructing a $c_w$-attentive summary of $Z$, we obtain the matching vector $c_w^+$ (if $E_w = \emptyset$, which makes $Z = \{\}$, we will set $c_w^+ = 0$):

$$t_i = v_c^\top \tanh(W_c z_i + U_c c_w) \in \mathbb{R} \quad (6)$$

$$c_w^+ = Z \text{ softmax}(\{t_1, ..., t_{|E_w|}\}) \in \mathbb{R}^d \quad (7)$$

where $v_c$, $W_c$, and $U_c$ are trainable parameters; $z_i$ represents the $i$-th column in $Z$. Finally we pass the concatenation of

$c_w$ and $c_w^+$ through a dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the enhanced context embedding $\tilde{c}_w \in \mathbb{R}^d$.

Based on this context embeddings, to perform knowledge-enhanced joint-attention, first we construct a knowledge-enhanced similarity matrix $A \in \mathbb{R}^{n\times m}$, where each element $A_{i,j} = f(\tilde{c}_{d_i}, \tilde{c}_{s_j})$. Then we construct the document-attentive sentence summaries $R_S$ and the sentence-attentive document summaries $R_D$ :

$$R_S = C_S \text{ softmax}_r^\top(A) \in \mathbb{R}^{d\times n} \tag{8}$$

$$R_D = C_D \text{ softmax}_c(A) \text{ softmax}_r^\top(A) \in \mathbb{R}^{d\times n} \tag{9}$$

where softmax$_r$ represents softmax along the row dimension and softmax$_c$ along the column dimension. Finally we pass the concatenation of $C_D$, $R_S$, $C_D \odot R_S$, and $R_D \odot R_S$ through a dense layer with ReLU activation, whose output dimensionality is $d$. Thus, we obtain the outputs $\tilde{G} \in \mathbb{R}^{d\times n}$.

### 3.2.4. Knowledge-Enhanced Self-Attention

As a part of the fine-grained memory layer, knowledge-enhanced self-attention is aimed at fusing the coarse-grained memories $G$ into themselves. We use the pre-extracted general knowledge to guarantee that the fusion of coarse-grained memories for each document word will only involve a precise subset of the other document words. Specifically, for each document word $d_i$, whose coarse-grained memory is $g_{d_i}$ (i.e., the $i$-th column in $G$), to perform the fusion of coarse-grained memories, first recall that we have extracted a set $E_{d_i}$, which includes the positions of the other document words that $d_i$ is semantically connected to, thus by gathering the columns in $G$ whose indexes are given by $E_{d_i}$, we obtain the matching coarse-grained memories $Z \in \mathbb{R}^{d\times|E_{d_i}|}$. Then by constructing a $g_{d_i}$-attentive summary of $Z$, we obtain the matching vector $g_{d_i}^+$ (if $E_{d_i} = \emptyset$, which makes $Z = \{\}$, we will set $g_{d_i}^+ = 0$):

$$t_i = v_g^\top \tanh(W_g z_i + U_g g_{d_i}) \in \mathbb{R} \tag{10}$$

$$g_{d_i}^+ = Z \text{ softmax}(\{t_1, ..., t_{|E_{d_i}|}\}) \in \mathbb{R}^d \tag{11}$$

where $v_g$, $W_g$, and $U_g$ are trainable parameters. Finally we pass the concatenation of $g_{d_i}$ and $g_{d_i}^+$ through a dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the fusion result $\tilde{h}_{d_i} \in \mathbb{R}^d$, and further the outputs $\tilde{H} = \{\tilde{h}_{d_1}, ..., \tilde{h}_{d_n}\} \in \mathbb{R}^{d\times n}$.

Our proposed neural model is quite different from the existing WSD models in that it uses semantic level inter-word connections, which are pre-extracted from the WSD dataset using the WordNet-based data enrichment method, as explicit knowledge to assist the sense prediction of the target word. On one hand, the coarse-grained memory layer uses the explicit knowledge to assist both the document-to-sentence and sentence-to-document attentions. On the other hand, the fine-grained memory layer uses the explicit knowledge to assist the self-attention.

## 4. Experiments and Analysis

We conducted experiments on standard benchmark datasets introduced by (Raganato et al., 2017) (i.e., Senseval-2 (S2), Senseval-3 (S3), SemEval-2007 (SE07), SemEval-2013 (SE13) and SemEval-2015 (SE15)), in order to evaluate the performance of our proposed semi-supervised neural system (PoKED).

**Implementation.** To train the proposed unsupervised position-wise orthogonal (PoNet) pseudo-language model, we used BooksCorpus (Zhu et al., 2015) and English Wikipedia as part of our pretraining data. In addition, we included Giga5 (Parker et al., 2011), ClueWeb 2012-B (extended from (Callan et al., 2009)), and Common Crawl (Crawl) for pretraining. The vocabulary consists of the most frequent 1M words without lemmatization or case normalization. The dimension of word embedding was set to 128. The position-wise codes led to a dimension of 1024 for the input layer of the orthogonal framework. Then we appended three hidden layers of dimension 2048. Additionally, we chose constant forgetting factors $\alpha = (0.5, 0.9)$ for the position-wise codes.

To implement our proposed supervised neural model, we exploited the Stanford CoreNLP (Manning et al., 2014) to pre-process datasets. We used the WordNet interface provided by NLTK (BIRD & LOPER, 2004) to perform the WordNet-based data enrichment method. Additionally, we implemented a knowledge-based attentive neural model using Tensorflow (Abadi et al., 2016) and train it on SemCor (Miller et al., 1994) corpus. For each BiLSTM, we set its hidden state size to 256. For the training, we used ADAM (Kingma & Ba, 2014) as our optimizer, set the learning rate to 0.001, and set the mini-batch size to 32. To avoid overfitting, we applied Dropout (Srivastava et al., 2014) to dense layers and BiLSTMs with a value of 0.35. Besides, we applied an exponential moving average with a decay rate of 0.999.

Following (Raganato et al., 2017), (Luo et al., 2018) and (Hadiwinoto et al., 2019), we chose SE07, the smallest among these test sets, as the development set. When fine-tuning, we used the development set to find the optimal settings for our experiments. The reported WSD task results in F1-score are averaged over three runs.

**Experimental results.** We performed a comparison with three configurations of our model, namely: PoKED$_\alpha$, obtained using our pre-trained position-wise orthogonal (PoNet) context embeddings with general knowledge; PoKED$_\beta$, obtained using Roberta (Liu et al., 2019) and general knowledge; PoKED$_\gamma$, obtained using XLNet (Yang et al., 2019) and general knowledge.

As comparison systems we included four semi-supervised approaches mentioned above, namely: Babelfy (Moro et al., 2014), ppr$_{w2w}$, the best configuration of UKB (Agirre et al., 2018), WSD-TM (Chaplot & Salakhutdinov, 2018), and WSDG (Tripodi & Navigli, 2019). In addition, we also report the performances of relevant supervised models, namely: IMS (Zhong & Ng, 2010), IMS$_{w2v}$ (Iacobacci et al., 2016), Yuan$_{LSTM}$ (Yuan et al., 2016), Raganato$_{BLSTM}$ (Raganato et al., 2017), GAS (Luo et al., 2018), fastSense (Uslu et al., 2018), GLU-LW (Hadiwinoto et al., 2019) and GlossBERT (Huang et al., 2019).

The results of our evaluation are shown in Table 1. As we can see our models achieve state-of-the-art performances on four datasets, (i.e., S2, S3, SE07, SE15). It is worth noting that, on SE13 and SE15 datasets, ours perform better than most supervised systems. In general, the gap between supervised and semi-supervised systems is narrowed. This encourages new research in this direction. Our models perform particularly well on the disambiguation of *nouns* and *adjectives*.

**Effect of general knowledge extraction.** We obtain seven enriched WSD datasets by setting $\tau$ from zero to six separately, and train a different PoKED$_\alpha$ system on each enriched WSD dataset. As shown in Table 2, by increasing $\tau$ from zero to six in the data enrichment method, the amount of general knowledge rises monotonically, but the F1-score of our proposed PoKED$_\alpha$ first rises until $\tau$ reaches three for SE07 and SE15 datasets, and reaches four for S2, S3 and SE13 datasets, respectively, and then drops. We can conclude that the explicit knowledge provided by the WordNet-based data enrichment method plays an effective role in the training of our proposed PoKED$_\alpha$ system.

**Effect of knowledge-enhancement.** We conducted an ablation study and obtained a version without knowledge-enhancement based on PoKED$_\alpha$. Specifically, we replace knowledge-enhanced joint-attention and knowledge-enhanced self-attention with full bipartite attention and standard dot-scale attention, respectively. As shown in Table 3, we observe that the performance of the model without knowledge-enhancement drops dramatically compared with that of PoKED$_\alpha$, especially with the SE15 dataset. In this case, F1-score declined by 5.4%, 70.5 vs. 65.1. It indicates that human semantic knowledge is helping with WSD tasks remarkably, and plays an important role in our semi-supervised neural WSD system.

**Quantitative analysis of the hunger for data.** Specifically, instead of using all the training examples, we produce several training subsets (i.e., subsets of the training examples) so as to study the relationship between the proportion of the available training samples and the performance. We produce each training subset by sampling a specific number of sentences from all the sentences relevant to each document. By separately sampling 1, 2, 3, and 4 sentences on each document, we obtain four training subsets, which separately contain 20%, 40%, 60%, and 80% of the training samples. As shown in Figure 5, with PoKED$_\alpha$ trained on these training subsets, we evaluate its performance on the SE15, S3 and ALL, and find that PoKED$_\alpha$ performs much better than MFS baseline even when only 80% of the training samples are used. That is, when only a subset of the training examples is available, PoKED$_\alpha$ is still comparable to the state-of-the-art WSD models.

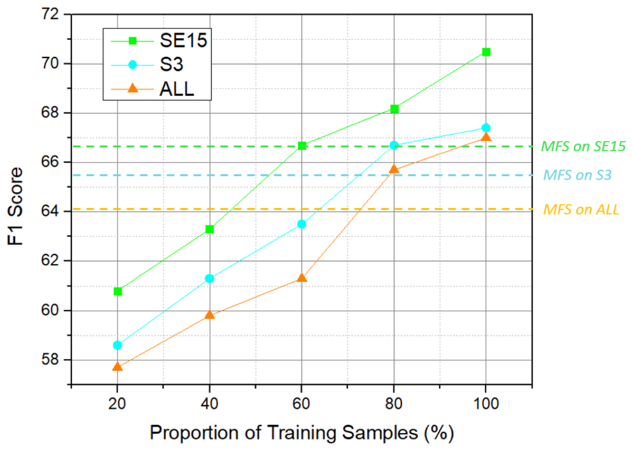


*Figure 5.* Quantitative Analysis of the Hunger for Data. With PoKED$_\alpha$ trained on the training subsets, we evaluate its performance on the SE15, S3 and ALL.

**Discussion.** According to the experimental results, PoKED not merely achieves state-of-the-art performances on most datasets, but performs better than most supervised systems. The reasons for these achievements, we believe, are as follows.

First, some inter-word semantic connections are distracting for the disambiguation of word sense. For example, the inter-word semantic connection between "ballpoint" and "pen" makes no sense given the context "Little John was looking for his toy box. Finally, he found it. The box was in the pen." It is the knowledge-enhanced attention mechanisms that enable PoKED to ignore such distracting inter-word semantic connections so that only the important ones are used.

Second, PoKED is designed to utilize the pre-extracted inter-word semantic connections extracted by the data enrichment method. Several approaches have been presented in recent years to make use of ontologies or types (such as (Dasigi et al., 2017)) to represent word tokens based on knowledge bases (e.g., WordNet). Nevertheless, the importance of inter-word semantic connections from document-sentence pairs

*Table 1.* Comparison with state-of-the-art algorithms: semi-supervised or knowledge-based (*semi-sup.*), and supervised (*sup.*). MFS refers to the Most Frequent Sense heuristic computed on SemCor (Miller et al., 1994) on each dataset. The results are provided as F1-score. **Bold** font indicates best systems. Results in the first and last blocks come from (Tripodi & Navigli, 2019; Hadiwinoto et al., 2019; Huang et al., 2019).

| | Model | S2 | S3 | SE07 | SE13 | SE15 | ALL | N | V | A | R |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **semi-sup.** | *MFS* | 64.7 | 65.4 | 53.9 | 62.9 | 66.6 | 64.1 | 68.1 | 49.5 | 74.1 | 80.6 |
| | *Babelfy* | 67.0 | 63.5 | 51.6 | 66.4 | 70.3 | 65.5 | 68.6 | 49.9 | 73.2 | 79.8 |
| | *ppr$_{w2w}$* | 68.8 | 66.1 | 53.0 | **68.8** | 70.3 | 67.3 | - | - | - | - |
| | *WSD-TM* | 69.0 | 66.9 | 55.6 | 65.3 | 69.6 | 66.9 | 69.7 | 51.2 | 76.0 | **80.9** |
| | *WSDG* | 68.9 | 65.5 | 54.5 | 67.0 | **72.8** | 67.2 | 70.4 | 51.3 | 75.7 | 80.6 |
| | *PoKED$_\alpha$ (ours)* | 68.8 | **67.4** | 54.6 | 65.2 | 70.5 | 67.0 | 70.1 | 50.8 | 75.5 | 79.7 |
| | *PoKED$_\beta$ (ours)* | 69.5 | 67.0 | 55.8 | 67.3 | **72.8** | 67.4 | 70.5 | 51.4 | **76.2** | 80.8 |
| | *PoKED$_\gamma$ (ours)* | **69.8** | 67.1 | **56.0** | 67.5 | 72.7 | **67.7** | **70.8** | **51.6** | **76.2** | 80.6 |
| **sup.** | *IMS* | 70.9 | 69.3 | 61.3 | 65.3 | 69.5 | 68.9 | 70.5 | 55.8 | 75.6 | 82.9 |
| | *IMS$_{w2v}$* | 72.2 | 70.4 | 62.6 | 65.9 | 71.5 | 70.1 | 71.9 | 56.6 | 75.9 | 84.7 |
| | *Yuan$_{LSTM}$* | 73.8 | 71.8 | 63.5 | 69.5 | 72.6 | 71.5 | - | - | - | - |
| | *Raganato$_{BLSTM}$* | 72.0 | 69.1 | 64.8 | 66.9 | 71.5 | 69.9 | 71.5 | 57.5 | 75.0 | 83.8 |
| | *GAS* | 72.2 | 70.5 | - | 67.2 | 72.6 | - | - | - | - | - |
| | *fastSense* | 73.5 | 73.5 | 62.4 | 66.2 | 73.2 | - | - | - | - | - |
| | *GLU-LW* | 75.5 | 73.4 | 68.5 | 71.0 | 76.2 | - | - | - | - | - |
| | *GlossBERT* | 76.5 | 73.4 | 69.2 | 75.1 | 79.5 | - | - | - | - | - |

*Table 2.* Effect analysis of general knowledge extraction. We report the F1-score performance of PoKED$_\alpha$ on standard benchmarks under each setting for $\tau$. **#average** stands for average number of inter-word connections per word. **Bold** font indicates best performance.

| $\tau$ | #average | S2 | #average | S3 | #average | SE07 | #average | SE13 | #average | SE15 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.51 | 67.1 | 0.47 | 65.6 | 0.30 | 52.7 | 0.29 | 63.2 | 0.36 | 67.1 |
| 1 | 0.92 | 67.7 | 0.84 | 66.2 | 0.51 | 53.4 | 0.41 | 63.6 | 0.68 | 68.4 |
| 2 | 1.47 | 68.1 | 1.32 | 66.5 | 1.93 | 53.9 | 0.93 | 64.1 | 2.03 | 69.6 |
| 3 | 2.35 | 68.4 | 3.06 | 66.9 | **3.28** | **54.6** | 1.58 | 64.7 | **3.97** | **70.5** |
| 4 | **3.89** | **68.8** | **3.97** | **67.4** | 3.86 | 54.0 | **3.67** | **65.2** | 4.87 | 69.9 |
| 5 | 5.79 | 68.3 | 4.28 | 66.8 | 4.77 | 53.2 | 4.22 | 64.5 | 5.52 | 69.3 |
| 6 | 6.22 | 68.0 | 5.45 | 66.4 | 6.02 | 52.6 | 5.19 | 63.4 | 6.45 | 68.2 |

*Table 3.* Effect of knowledge-enhancement. It shows human semantic knowledge plays an important role in our semi-supervised neural WSD system.

| Model | S2 | S3 | SE07 | SE13 | SE15 |
|---|---|---|---|---|---|
| PoKED$_\alpha$ | 68.8 | 67.4 | 54.6 | 65.2 | 70.5 |
| - knowledge-enhancement | 64.3 ▼-4.5 | 63.5 ▼-3.9 | 50.2 ▼-4.4 | 61.4 ▼-3.8 | 65.1 ▼-5.4 |

was overlooked. Therefore, in this paper, we explore an explicit (i.e., understandable and controllable) way to utilize general knowledge. Some inter-word semantic connections, especially those obtained through multi-hop semantic relation chains, enhance the ability of the model to identify the boundaries of word senses and is helpful for the final disambiguation.

Finally, an inter-word semantic connection extracted from a document-sentence pair usually also appears in many other document-sentence pairs; therefore it is very likely that the inter-word semantic connections extracted from a small number of training examples cover a larger amount of training examples. That is, we are using more training examples for model optimization than the available ones.

## 5. Conclusion

In this paper, we propose a semi-supervised neural system (PoKED), which incorporates human semantic knowledge into the neural network architectures for WSD tasks. Specifically, inter-word semantic connections are first extracted from each given document by a WordNet-based data enrichment method, and then provided as general knowledge to an end-to-end WSD model, which explicitly uses the general knowledge to assist its attention mechanisms. Experimental results show that PoKED achieves better performance than the previous state-of-the-art knowledge-based models.

## Acknowledgements

I would like to thank my thesis advisor Professor Uyen Trang Nguyen for her tremendous help and many fruitful discussions. I also thank anonymous reviewers for their thorough constructive comments to improve the paper. Many of my past and present collaborators have made valuable contributions to this work.

## References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 265–283, 2016.

Agirre, E., de Lacalle, O. L., and Soroa, A. The risk of sub-optimal use of open source nlp software: Ukb is inadvertently state-of-the-art in knowledge-based wsd. In *Proceedings of Workshop for NLP Open Source Software*, pp. 29–33, 2018.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

BIRD, S. and LOPER, E. Nltk: The natural language toolkit. Association for Computational Linguistics, 2004.

Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3: 993–1022, 2003.

Callan, J., Hoy, M., Yoo, C., and Zhao, L. Clueweb09 data set, 2009.

Chaplot, D. S. and Salakhutdinov, R. Knowledge-based word sense disambiguation using topic models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Crawl, C. Common crawl. http://commoncrawl.org.

Dasigi, P., Ammar, W., Dyer, C., and Hovy, E. Ontology-aware token embeddings for prepositional phrase attachment. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2089–2098, 2017.

Hadiwinoto, C., Ng, H. T., and Gan, W. C. Improved word sense disambiguation using pre-trained contextualized word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5300–5309, 2019.

Harris, Z. S. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Huang, L., Sun, C., Qiu, X., and Huang, X. Glossbert: Bert for word sense disambiguation with gloss knowledge. *arXiv preprint arXiv:1908.07245*, 2019.

Iacobacci, I., Pilehvar, M. T., and Navigli, R. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 897–907, 2016.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 427–431, 2017.

Kim, Y. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Le, M., Postma, M., Urbani, J., and Vossen, P. Deep dive into word sense disambiguation with lstm. In *Proceedings of 27th International Conference on Computational Linguistics*, pp. 354–365, 2018.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Luo, F., Liu, T., Xia, Q., Chang, B., and Sui, Z. Incorporating glosses into neural word sense disambiguation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2473–2482, 2018.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60, 2014.

Maru, M., Scozzafava, F., Martelli, F., and Navigli, R. Syntagnet: Challenging supervised word sense disambiguation with lexical-semantic combinations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3525–3531, 2019.

Mihalcea, R., Tarau, P., and Figa, E. Pagerank on semantic networks, with application to word sense disambiguation. In *Proceedings of the 20th International Conference on Computational Linguistics*, pp. 1126–1132, 2004.

Miller, G. A. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Miller, G. A., Chodorow, M., Landes, S., Leacock, C., and Thomas, R. G. Using a semantic concordance for sense identification. In *Proceedings of the Workshop on Human Language Technology*, pp. 240–243, 1994.

Moro, A., Raganato, A., and Navigli, R. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014.

Navigli, R. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):10, 2009.

Navigli, R. and Lapata, M. An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):678–692, 2009.

Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Parker, R., Graff, D., Kong, J., Chen, K., and Maeda, K. English gigaword fifth edition. *Linguistic Data Consortium*, 2011.

Raganato, A., Camacho-Collados, J., and Navigli, R. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 99–110, 2017.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Tripodi, R. and Navigli, R. Game theory meets embeddings: a unified framework for word sense disambiguation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 88–99, 2019.

Uslu, T., Mehler, A., Baumartz, D., and Hemati, W. fastsense: An efficient word sense disambiguation classifier. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, 2018.

Wang, C. and Jiang, H. Explicit utilization of general knowledge in machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2263–2272, 2019.

Watcharawittayakul, S., Xu, M., and Jiang, H. Dual fixed-size ordinally forgetting encoding (fofe) for competitive neural language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4725–4730, 2018.

Wei, F., Nguyen, U. T., and Jiang, H. Dual-fofe-net neural models for entity linking with pagerank. In *International Conference on Artificial Neural Networks*, pp. 635–645. Springer, 2019.

Wei, F., Trang Nguyen, U., and Jiang, H. Fonet: A memory-efficient fourier-based orthogonal network for object recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 688–689, 2020.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, pp. 5754–5764, 2019.

Yuan, D., Richardson, J., Doherty, R., Evans, C., and Altendorf, E. Semi-supervised word sense disambiguation with neural models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 1374–1385, 2016.

Zhang, S., Jiang, H., Xu, M., Hou, J., and Dai, L. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of ACL*, 2015.

Zhang, S., Jiang, H., and Dai, L. Hybrid orthogonal projection and estimation (hope): a new framework to learn neural networks. *The Journal of Machine Learning Research*, 17(1):1286–1318, 2016.

Zhong, Z. and Ng, H. T. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010: System Demonstrations*, pp. 78–83, 2010.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 19–27, 2015.

EMNLP 2018 Submission 610. Confidential Review Copy. DO NOT DISTRIBUTE.

# Exploring Machine Reading Comprehension with Explicit Knowledge

**Anonymous EMNLP submission**

## Abstract

To apply general knowledge to machine reading comprehension (MRC), we propose an innovative MRC approach, which consists of a WordNet-based data enrichment method and an MRC model named as Knowledge Aided Reader (KAR). The data enrichment method uses the semantic relations of WordNet to extract semantic level inter-word connections from each passage-question pair in the MRC dataset, and allows us to control the amount of the extraction results by setting a hyper-parameter. KAR uses the extraction results of the data enrichment method as explicit knowledge to assist the prediction of answer spans. According to the experimental results, the single model of KAR achieves an Exact Match (EM) of 72.4 and an F1 Score of 81.1 on the development set of SQuAD, and more importantly, by applying different settings in the data enrichment method to change the amount of the extraction results, there is a 2% variation in the resulting performance of KAR, which implies that the explicit knowledge provided by the data enrichment method plays an effective role in the training of KAR.

## 1 Introduction

Machine reading comprehension (MRC) is a challenging task in artificial intelligence. As the name suggests, MRC requires a machine to read a passage and answer a relevant question. Since the answer to each question is supposed to stem from the corresponding passage, a common solution for MRC is to train an MRC model that predicts for each given passage-question pair an answer span (i.e. the answer start position and the answer end position) in the passage. To encourage the exploration of MRC models, many MRC datasets have been published, such as SQuAD (Rajpurkar et al., 2016) and MS-MARCO (Nguyen et al., 2016). In this paper, we focus on SQuAD.

A lot of MRC models have been proposed for the challenge of SQuAD. Although the top models on the leader-board have achieved almost the same performance as human beings, we are firmly convinced that the way human beings conduct reading comprehension is still worth studying for us to make further innovations in MRC. Therefore, let us briefly review human reading comprehension before diving into MRC. Given a passage and a relevant question, we may wish to match the passage words with the question words, so that we could find the answer around the matched passage words. However, due to the complexity and diversity of natural languages, this naive method is often useless in practice. Instead, we must rely on our reasoning skills to deal with reading comprehension, which makes it necessary for us to obtain enough inter-word connections from each given passage-question pair. Inter-word connections have a wide coverage, they exist not only on the syntactic level (e.g. dependency), but also on the semantic level (e.g. synonymy). The examples provided in Table 1 demonstrate how human reading comprehension could benefit from semantic level inter-word connections.

By roughly analyzing the MRC models proposed for SQuAD, we find that leveraging neural attention mechanisms (Bahdanau et al., 2014) based on recurrent neural networks, such as LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014), is currently the dominant approach. Since neural network models are usually deemed as simulations of human brains, we may as well interpret the training of an MRC model as a process of teaching knowledge to it, where the knowledge comes from the training samples, and thus can be absorbed into the model parameters through gradient descent. However, neural network models are also known as black boxes, that is to say, by just updating model parameters according to training samples, we cannot understand the meaning of the knowledge taught to an

EMNLP 2018 Submission 610. Confidential Review Copy. DO NOT DISTRIBUTE.

| Passage | Question | Answer |
|---|---|---|
| Teachers may use a lesson plan to **facilitate** student learning, providing a course of study which is called the curriculum. | What can a teacher use to **help** students learn? | lesson plan |
| Manufacturing accounts for a significant but declining share of employment, although the city's garment industry is showing a resurgence in **Brooklyn**. | In what **borough** is the garment business prominent? | Brooklyn |

Table 1: Two examples about the effects of semantic level inter-word connections on human reading comprehension. In the first example, we can find the answer because we know "facilitate" and "help" are synonyms. Similarly, in the second example, we can find the answer because we know "borough" is a hypernym of "Brooklyn", or "Brooklyn" is a hyponym of "borough". Both of the two examples are selected from SQuAD.

MRC model, neither can we control the amount of the knowledge taught to it, therefore we name such knowledge as implicit knowledge.

So far, human beings have accumulated a tremendous amount of general knowledge. These general knowledge, despite being an essential component of human intelligence, has never been effectively applied to MRC, which we believe is the biggest gap between MRC and human reading comprehension. We intend to bridge this gap with the help of knowledge bases, which store general knowledge in structured forms. In recent years, many knowledge bases have been established, such as WordNet (Fellbaum, 1998) and Freebase (Bollacker et al., 2008), and they have made it convenient for machines to access and process the general knowledge of human beings. Therefore, it is both meaningful and feasible to integrate the general knowledge in a knowledge base with the training of an MRC model. However, rather than leveraging knowledge base embeddings (Bordes et al., 2011, 2013; Yang et al., 2014; Yang and Mitchell, 2017), we would prefer our MRC model to use general knowledge in an understandable and controllable way, and we name the general knowledge used in this way as explicit knowledge.

In this paper, by using WordNet as our knowledge base, we propose an innovative MRC approach, which consists of two components: a WordNet-based data enrichment method, which uses WordNet to extract semantic level inter-word connections from each passage-question pair in the MRC dataset, and an MRC model named as Knowledge Aided Reader (KAR), which uses the extraction results of the data enrichment method as explicit knowledge to assist the prediction of answer spans. There are two important features in our MRC approach: on the one hand, the data enrichment method allows us to control the amount of the extraction results; on the other hand, this amount in turn affects the performance of KAR. According to the experimental results, by applying different settings in the data enrichment method to change the amount of the extraction results, there is a 2% variation in the resulting performance of KAR, which implies that the explicit knowledge provided by the data enrichment method plays an effective role in the training of KAR.

## 2 Task Description

The MRC task considered in this paper is defined as the following prediction problem: given a passage $P := [p_1, \ldots, p_n]$, which is a sequence of $n$ words, and a relevant question $Q := [q_1, \ldots, q_m]$, which is a sequence of $m$ words, predict an answer start position $a_s$ and an answer end position $a_e$, where $1 \leq a_s \leq a_e \leq n$, so that the fragment $[p_{a_s}, \ldots, p_{a_e}]$ in $P$ is the answer to $Q$.

## 3 WordNet-based Data Enrichment

To provide our MRC model with explicit knowledge, we would like to enrich the content of the MRC dataset by extracting semantic level inter-word connections from each passage-question pair in it, therefore we propose a WordNet-based data enrichment method.

### 3.1 What and how to extract from each passage-question pair

WordNet is a lexical database for English. Words in WordNet are organized into synsets, which in turn are related to each other through semantic relations, such as "hypernym" and "hyponym". In our data enrichment method, we use the semantic relations of WordNet to extract semantic level inter-word connections from each passage-question pair in the MRC dataset. Considering the

EMNLP 2018 Submission 610. Confidential Review Copy. DO NOT DISTRIBUTE.

requirements of our MRC model, we need to represent the extraction results as positional information. Specifically, for each word $w$ in a passage-question pair, we need to obtain a set $Z_w$, which contains the positions of the passage words that $w$ is semantically connected to. Besides, when $w$ itself is a passage word, we also need to ensure that its position is excluded from $Z_w$.

The key problem to obtain the above extraction results is to determine if a subject word is semantically connected to an object word. To solve this problem, we introduce two concepts: the directly-involved synsets and indirectly-involved synsets of a word. Given a word $w$, its directly-involved synsets $\Phi_w$ represents the synsets that $w$ belongs to, and its indirectly-involved synsets $\overline{\Phi}_w$ represents the synsets that the synsets in $\Phi_w$ are related to through semantic relations. Based on the two concepts, we propose the following hypothesis: given a subject word $w_s$ and an object word $w_o$, $w_s$ is semantically connected to $w_o$ if and only if $(\Phi_{w_s} \cup \overline{\Phi}_{w_s}) \cap \Phi_{w_o} \neq \emptyset$. According to the hypothesis, Algorithm 1 describes the process of extracting semantic level inter-word connections from each passage-question pair.

## 3.2 How to obtain the indirectly-involved synsets of each word

The above hypothesis and process can work only if we know how to obtain the directly-involved synsets and indirectly-involved synsets of each word. Given a word $w$, we can easily obtain its directly-involved synsets $\Phi_w$ from WordNet, but obtaining its indirectly-involved synsets $\overline{\Phi}_w$ is much more complicated, because in WordNet, the way synsets are related to each other is flexible and extensible. In some cases, a synset is related to another synset through a single semantic relation. For example, the synset “cold.a.01” is related to the synset “temperature.n.01” through the semantic relation “attribute”. However, in more cases, a synset is related to another synset through a semantic relation chain. For example, first the synset “keratin.n.01” is related to the synset “feather.n.01” through the semantic relation “substance holonym”, then the synset “feather.n.01” is related to the synset “bird.n.01” through the semantic relation “part holonym”, and finally the synset “bird.n.01” is related to the synset “parrot.n.01” through the semantic relation “hyponym”, thus we can say that the synset “keratin.n.01” is related to the synset “parrot.n.01” through the semantic relation chain “substance holonym $\rightarrow$ part holonym $\rightarrow$ hyponym”. We name each semantic relation in a semantic relation chain as a hop, so that a semantic relation chain having $k$ semantic relations is a $k$-hop semantic relation chain. Besides, each single semantic relation is a 1-hop semantic relation chain.

Let us use $\Gamma := \{\gamma_1, \gamma_2, \ldots\}$ to represent the semantic relations of WordNet, and use $\Omega_\phi^{\gamma_i}$ to represent the synsets that a synset $\phi$ is related to through a single semantic relation $\gamma_i \in \Gamma$. Since $\Omega_\phi^{\gamma_i}$ is easy to obtain from WordNet, we can further obtain the synsets that $\phi$ is related to through 1-hop semantic relation chains: $\Psi_\phi^1 = \bigcup_{\gamma_i \in \Gamma} \Omega_\phi^{\gamma_i}$, the synsets that $\phi$ is related to through 2-hop semantic relation chains: $\Psi_\phi^2 = \bigcup_{\hat{\phi} \in \Psi_\phi^1} \bigcup_{\gamma_i \in \Gamma} \Omega_{\hat{\phi}}^{\gamma_i}$, and by induction, the synsets that $\phi$ is related to through $k$-hop semantic relation chains: $\Psi_\phi^k = \bigcup_{\hat{\phi} \in \Psi_\phi^{k-1}} \bigcup_{\gamma_i \in \Gamma} \Omega_{\hat{\phi}}^{\gamma_i}$. In theory, if we do not limit the hop counts of semantic relation chains, $\phi$ can be related to all other synsets in WordNet, which is meaningless in many cases. Therefore, we use a hyper-parameter $\chi \in \mathbb{N}$ to represent the maximum hop count of semantic relation chains, and only consider the semantic relation chains that have no more than $\chi$ hops. Based on the above descriptions, given a word $w$ and its directly-involved synsets $\Phi_w$, we can obtain its indirectly-involved synsets: $\overline{\Phi}_w = \bigcup_{\phi \in \Phi_w} \bigcup_{k=1}^{\chi} \Psi_\phi^k$.

## 3.3 About controlling the amount of the extraction results

The hyper-parameter $\chi$ is crucial in controlling the amount of the extraction results. When we set $\chi$ to 0, the indirectly-involved synsets of each word contains no synset, so that semantic level inter-word connections only exist between synonyms. As we increase $\chi$, the indirectly-involved synsets of each word usually contains more synsets, so that semantic level inter-word connections are likely to exist between more words. As a result, by increasing $\chi$ within a certain range, we can extract more semantic level inter-word connections from the MRC dataset, and thus provide our MRC model with more explicit knowledge. However, due to the limitations of WordNet, only a part of the extraction results are useful explicit knowledge, while the rest are useless for the prediction of answer spans. According to our observation, the proportion of the useless explicit knowledge

EMNLP 2018 Submission 610. Confidential Review Copy. DO NOT DISTRIBUTE.

**Algorithm 1** Extract semantic level inter-word connections from each passage-question pair

```
procedure EXTRACT($P, Q$)                                          ▷ Given a passage $P$ and a relevant question $Q$
    for $p_i$ in $P$ do                                            ▷ For each passage word $p_i$
        $Z_{p_i} \leftarrow \{j \in \{1,\dots,n\} \backslash \{i\} : (\Phi_{p_i} \cup \overline{\Phi}_{p_i}) \cap \Phi_{p_j} \neq \emptyset\}$    ▷ Obtain the extraction results $Z_{p_i}$
    end for
    for $q_i$ in $Q$ do                                            ▷ For each question word $q_i$
        $Z_{q_i} \leftarrow \{j \in \{1,\dots,n\} : (\Phi_{q_i} \cup \overline{\Phi}_{q_i}) \cap \Phi_{p_j} \neq \emptyset\}$    ▷ Obtain the extraction results $Z_{q_i}$
    end for
end procedure                                                      ▷ Return the extraction results on P and Q
```

increases as $\chi$ gets larger. Therefore, there exists an optimal setting for $\chi$, which can result in the best performance of our MRC model.

## 4 Knowledge Aided Reader

As depicted in Figure 1, our MRC model, Knowledge Aided Reader (KAR), consists of five layers: given a passage-question pair, the lexical embedding layer encodes the lexical features of each word to generate the passage lexical embeddings and the question lexical embeddings; based on the lexical embeddings, the contextual embedding layer encodes the contextual clues about each word to generate the passage contextual embeddings and the question contextual embeddings; based on the contextual embeddings, the memory generation layer performs passage-to-question attention and question-to-passage attention to generate the preliminary memories over the passage-question pair; based on the preliminary memories, the memory refining layer performs self-matching attention to generate the refined memories over the passage-question pair; based on the refined memories and the question contextual embeddings, the answer span prediction layer generates the answer start position distribution and the answer end position distribution. KAR is quite different from the existing MRC models in that it uses the semantic level inter-word connections, which are pre-extracted from the MRC dataset by the WordNet-based data enrichment method, as explicit knowledge to assist the prediction of answer spans. On the one hand, the memory generation layer uses the explicit knowledge to assist the passage-to-question attention and the question-to-passage attention. On the other hand, the memory refining layer uses the explicit knowledge to assist the self-matching attention. Besides, to better utilize the explicit knowledge, the lexical embedding layer encodes dependency and synonymy information into the lexical embedding of each word.

### 4.1 Lexical Embedding Layer

For each word, the lexical embedding layer generates its lexical embedding by merging the following four basic embeddings:

**1. Word-level Embedding.** We define our vocabulary as the intersection between the words in all training samples and those in the pre-trained 300-dimensional GloVe (Pennington et al., 2014). Given a word $w$, if it is in the vocabulary, we set its word-level embedding $\alpha_w$ to its GloVe word vector, which is fixed during the training, otherwise we have $\alpha_w = \alpha_o \in \mathbb{R}^{300}$, where $\alpha_o$ is a trainable parameter serving as the shared word vector of all out-of-vocabulary (OOV) words.

**2. Character-level Embedding.** We represent each character as a separate 150-dimensional vector, which is a trainable parameter. Given a word $w$ consisting of a sequence of $k$ characters, whose vectors are represented as $U_\beta \in \mathbb{R}^{150\times k}$, we use a bidirectional FOFE (Zhang et al., 2015) to process $U_\beta$, concatenate the forward FOFE output ($\mathbb{R}^{150\times k}$) and the backward FOFE output ($\mathbb{R}^{150\times k}$) across rows to obtain $F_\beta \in \mathbb{R}^{300\times k}$, and perform self attention on $F_\beta$ to obtain the character-level embedding of $w$:

$$\beta_w = F_\beta \text{softmax}(\tanh(W_\beta F_\beta)^\top v_\beta) \in \mathbb{R}^{300}$$

where $W_\beta$ and $v_\beta$ are trainable parameters. Applying character-level embedding is helpful in representing OOV words.

**3. Dependency Embedding.** Inspired by Liu et al. (2017a), we use a dependency parser to obtain the dependent words of each word. Given a word $w$ having $k$ dependent words, whose word-level embeddings are represented as $U_\eta \in \mathbb{R}^{300\times k}$, we perform self attention on $U_\eta$ to obtain the dependency embedding of $w$:

$$\eta_w = U_\eta \text{softmax}(\tanh(W_\eta U_\eta)^\top v_\eta) \in \mathbb{R}^{300}$$

EMNLP 2018 Submission 610. Confidential Review Copy. DO NOT DISTRIBUTE.

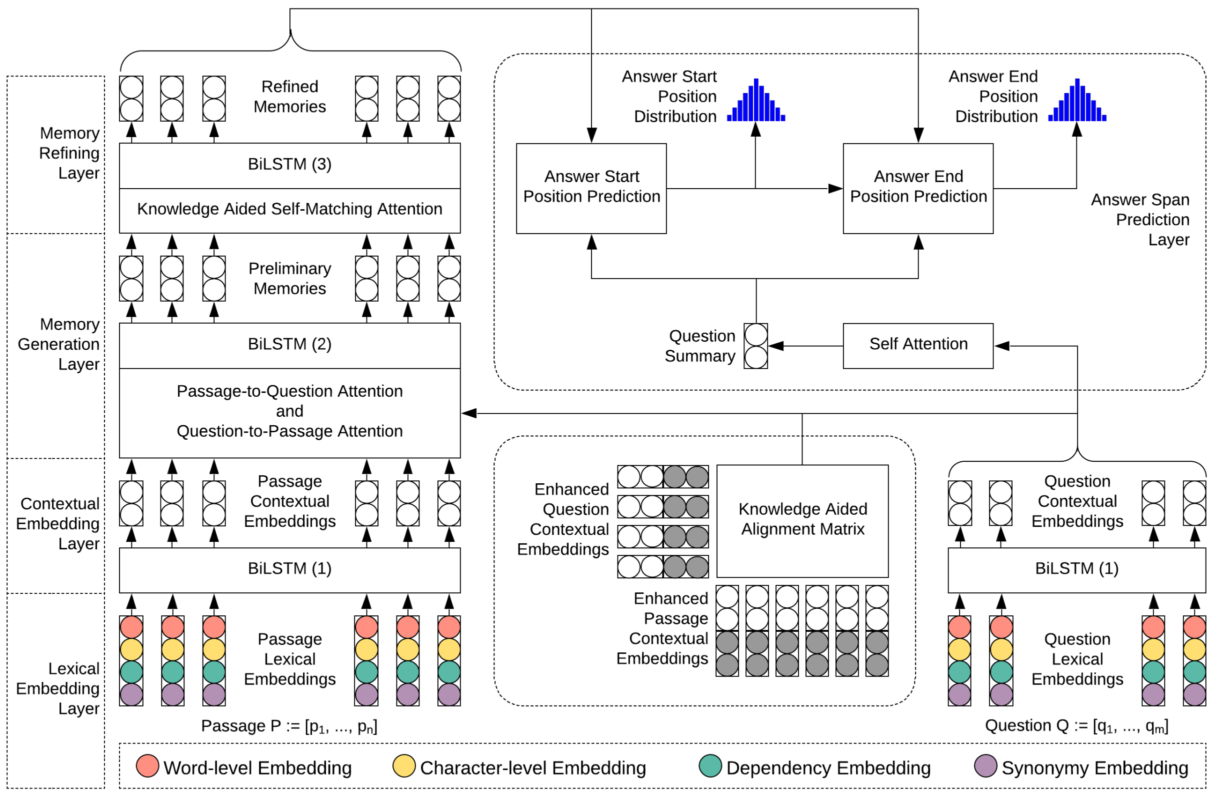

Refined Memories
Memory Refining Layer
BiLSTM (3)
Knowledge Aided Self-Matching Attention
Preliminary Memories
Memory Generation Layer
BiLSTM (2)
Passage-to-Question Attention and Question-to-Passage Attention
Passage Contextual Embeddings
Contextual Embedding Layer
BiLSTM (1)
Passage Lexical Embeddings
Lexical Embedding Layer
Passage P := [p1, ..., pn]
Answer Start Position Distribution
Answer End Position Distribution
Answer Start Position Prediction
Answer End Position Prediction
Answer Span Prediction Layer
Question Summary
Self Attention
Enhanced Question Contextual Embeddings
Knowledge Aided Alignment Matrix
Enhanced Passage Contextual Embeddings
Question Contextual Embeddings
BiLSTM (1)
Question Lexical Embeddings
Question Q := [q1, ..., qm]
Word-level Embedding
Character-level Embedding
Dependency Embedding
Synonymy Embedding

Figure 1: Our MRC model: Knowledge Aided Reader (KAR)

where $W_\eta$ and $v_\eta$ are trainable parameters. By applying dependency embedding, we make use of syntactic level inter-word connections, which serve as a supplement to the pre-extracted semantic level inter-word connections.

**4. Synonymy Embedding.** In the scope of the vocabulary, we use WordNet to obtain the synonyms of each word. Given a word $w$ having $k$ synonyms, whose word-level embeddings are represented as $U_\mu \in \mathbb{R}^{300\times k}$, we perform self attention on $U_\mu$ to obtain the synonymy embedding of $w$:

$$\mu_w = U_\mu \text{softmax}(\tanh(W_\mu U_\mu)^\top v_\mu) \in \mathbb{R}^{300}$$

where $W_\mu$ and $v_\mu$ are trainable parameters. By applying synonymy embedding, we improve the vector-space similarity between synonyms, and thus promote the effects of the pre-extracted semantic level inter-word connections.

Based on the above descriptions, given a word $w$, we obtain $\alpha_w$, $\beta_w$, $\eta_w$, and $\mu_w$, and concatenate them across rows to obtain $\pi_w \in \mathbb{R}^{1200}$. In this way, for all passage words, we obtain $\Pi_P = [\pi_{p_1}, \ldots, \pi_{p_n}] \in \mathbb{R}^{1200\times n}$, and for all question words, we obtain $\Pi_Q = [\pi_{q_1}, \ldots, \pi_{q_m}] \in \mathbb{R}^{1200\times m}$. We put $\Pi_P$ through a 1-layer highway network (Srivastava et al., 2015) to obtain the passage lexical embeddings: $L_P = [l_{p_1}, \ldots, l_{p_n}] \in \mathbb{R}^{1200\times n}$, and put $\Pi_Q$ through the same highway network to obtain the question lexical embeddings: $L_Q = [l_{q_1}, \ldots, l_{q_m}] \in \mathbb{R}^{1200\times m}$.

## 4.2 Contextual Embedding Layer

For each word, the contextual embedding layer fuses its lexical embedding with those of its surrounding words to generate its contextual embedding. Specifically, we use a bidirectional LSTM (BiLSTM), whose hidden state size is $d$, to process $L_P$ and $L_Q$ separately. For $L_P$, we concatenate the forward LSTM output ($\mathbb{R}^{d\times n}$) and the backward LSTM output ($\mathbb{R}^{d\times n}$) across rows to obtain the passage contextual embeddings: $C_P = [c_{p_1}, \ldots, c_{p_n}] \in \mathbb{R}^{2d\times n}$. For $L_Q$, we concatenate the forward LSTM output ($\mathbb{R}^{d\times m}$) and the backward LSTM output ($\mathbb{R}^{d\times m}$) across rows to obtain the question contextual embeddings: $C_Q = [c_{q_1}, \ldots, c_{q_m}] \in \mathbb{R}^{2d\times m}$.

## 4.3 Memory Generation Layer

For each passage word, the memory generation layer fuses its contextual embedding with both the

EMNLP 2018 Submission 610. Confidential Review Copy. DO NOT DISTRIBUTE.

passage contextual embeddings and the question contextual embeddings to generate its preliminary memory over the passage-question pair. Specifically, the task of this layer is decomposed into the following four steps:

**1. Generating enhanced contextual embeddings.** We enhance the contextual embedding of each word according to the pre-extracted semantic level inter-word connections. Given a word $w$, whose contextual embedding is $c_w \in \mathbb{R}^{2d}$, suppose we have obtained $Z_w$ through Algorithm 1, then we gather the columns in $C_P$ whose positions are contained in $Z_w$, represent these columns as $U_\tau \in \mathbb{R}^{2d\times|Z_w|}$, and perform attention on $U_\tau$ to obtain the $c_w$-attended contextual embedding:

$$\tau_w = U_\tau \text{softmax}(\tanh(W_\tau(c_w \rtimes U_\tau))^\top v_\tau) \in \mathbb{R}^{2d}$$

where $W_\tau$ and $v_\tau$ are trainable parameters, and $x \rtimes X$ represents concatenating a vector $x$ with each column in a matrix $X$ across rows. Based on the above descriptions, given a word $w$, we concatenate $c_w$ and $\tau_w$ across rows to obtain $\lambda_w \in \mathbb{R}^{4d}$. In this way, for all passage words, we obtain $\Lambda_P = [\lambda_{p_1}, \ldots, \lambda_{p_n}] \in \mathbb{R}^{4d\times n}$, and for all question words, we obtain $\Lambda_Q = [\lambda_{q_1}, \ldots, \lambda_{q_m}] \in \mathbb{R}^{4d\times m}$. We put $\Lambda_P$ through a 1-layer highway network to obtain the enhanced passage contextual embeddings: $B_P = [b_{p_1}, \ldots, b_{p_n}] \in \mathbb{R}^{4d\times n}$, and put $\Lambda_Q$ through the same highway network to obtain the enhanced question contextual embeddings: $B_Q = [b_{q_1}, \ldots, b_{q_m}] \in \mathbb{R}^{4d\times m}$.

**2. Constructing knowledge aided alignment matrix.** Based on the enhanced contextual embeddings, we construct an alignment matrix $A \in \mathbb{R}^{n\times m}$, where each element $A[i,j]$ (i.e. the $i$-th row and $j$-th column in $A$) represents the similarity between the enhanced contextual embedding $b_{p_i} \in \mathbb{R}^{4d}$ of the passage word $p_i$ and the enhanced contextual embedding $b_{q_j} \in \mathbb{R}^{4d}$ of the question word $q_j$. We use the similarity function proposed by Seo et al. (2016) to obtain each element in $A$:

$$A[i,j] = {v_A}^\top (b_{p_i} \bowtie b_{q_j} \bowtie (b_{p_i} \odot b_{q_j})) \in \mathbb{R}$$

where $v_A \in \mathbb{R}^{12d}$ is a trainable parameter, $\bowtie$ represents concatenation across rows, and $\odot$ represents element-wise multiplication. Since the enhanced contextual embeddings are generated according to the pre-extracted semantic level inter-word connections, the alignment matrix $A$ is named as knowledge aided alignment matrix.

**3. Performing passage-to-question attention and question-to-passage attention.** On the one hand, following Seo et al. (2016), we perform passage-to-question attention to obtain the passage-attended question representations:

$$R_Q = C_Q \text{softmax}_r(A)^\top \in \mathbb{R}^{2d\times n}$$

where $\text{softmax}_r(X)$ represents normalizing each row in a matrix $X$ by softmax. On the other hand, following Xiong et al. (2016), we perform question-to-passage attention to obtain the question-attended passage representations:

$$R_P = C_P \text{softmax}_c(A) \text{softmax}_r(A)^\top \in \mathbb{R}^{2d\times n}$$

where $\text{softmax}_c(X)$ represents normalizing each column in a matrix $X$ by softmax.

**4. Generating preliminary memories.** We concatenate $C_P$, $R_Q$, $C_P \odot R_Q$, and $C_P \odot R_P$ across rows, put this concatenation ($\mathbb{R}^{8d\times n}$) through a 1-layer highway network, use a BiLSTM, whose hidden state size is $d$, to process the output of the highway network ($\mathbb{R}^{8d\times n}$), and concatenate the forward LSTM output ($\mathbb{R}^{d\times n}$) and the backward LSTM output ($\mathbb{R}^{d\times n}$) across rows to obtain the preliminary memories over the passage-question pair: $G = [g_{p_1}, \ldots, g_{p_n}] \in \mathbb{R}^{2d\times n}$.

## 4.4 Memory Refining Layer

For each passage word, the memory refining layer fuses its preliminary memory with those of some other passage words to generate its refined memory over the passage-question pair. Inspired by Wang et al. (2017), we perform self-matching attention on the preliminary memories. However, we are different from Wang et al. (2017) in that for each passage word, we only match its preliminary memory with those of a corresponding subset of other passage words, which are selected according to the pre-extracted semantic level inter-word connections, therefore our self-matching attention is named as knowledge aided self-matching attention. Specifically, given a passage word $p_i$, whose preliminary memory is $g_{p_i} \in \mathbb{R}^{2d}$, suppose we have obtained $Z_w$ through Algorithm 1, then we gather the columns in $G$ whose positions are contained in $Z_w$, represent these columns as $U_\zeta \in \mathbb{R}^{2d\times|Z_w|}$, and perform attention on $U_\zeta$ to obtain the $g_{p_i}$-attended preliminary memory:

$$\zeta_{p_i} = U_\zeta \text{softmax}(\tanh(W_\zeta(g_{p_i} \rtimes U_\zeta))^\top v_\zeta) \in \mathbb{R}^{2d}$$

where $W_\zeta$ and $v_\zeta$ are trainable parameters. Based on the above descriptions, given a passage word

EMNLP 2018 Submission 610. Confidential Review Copy. DO NOT DISTRIBUTE.

$p_i$, we concatenate $g_{p_i}$ and $\zeta_{p_i}$ across rows to obtain $\delta_{p_i} \in \mathbb{R}^{4d}$. In this way, for all passage words, we obtain $\Delta = [\delta_{p_1}, \ldots, \delta_{p_n}] \in \mathbb{R}^{4d \times n}$. We put $\Delta$ through a 1-layer highway network, use a BiLSTM, whose hidden state size is $d$, to process the output of the highway network ($\mathbb{R}^{4d \times n}$), and concatenate the forward LSTM output ($\mathbb{R}^{d \times n}$) and the backward LSTM output ($\mathbb{R}^{d \times n}$) across rows to obtain the refined memories over the passage-question pair: $H = [h_{p_1}, \ldots, h_{p_n}] \in \mathbb{R}^{2d \times n}$.

### 4.5 Answer Span Prediction Layer

In the answer span prediction layer, we first perform self attention on $C_Q$ to obtain a summary of the question:

$$\epsilon = C_Q \mathrm{softmax}(\tanh(W_\epsilon C_Q)^\top v_\epsilon) \in \mathbb{R}^{2d}$$

where $W_\epsilon$ and $v_\epsilon$ are trainable parameters. Then with $\epsilon$ as the query, we perform attention on $H$ to obtain the answer start position distribution:

$$d_s = \mathrm{softmax}(\tanh(W_s(\epsilon \rtimes H))^\top v_s) \in \mathbb{R}^n$$

where $W_s$ and $v_s$ are trainable parameters. Next we concatenate $\epsilon$ and $Hd_s \in \mathbb{R}^{2d}$ across rows to obtain $\xi \in \mathbb{R}^{4d}$. Finally with $\xi$ as the query, we perform attention on $H$ again to obtain the answer end position distribution:

$$d_e = \mathrm{softmax}(\tanh(W_e(\xi \rtimes H))^\top v_e) \in \mathbb{R}^n$$

where $W_e$ and $v_e$ are trainable parameters.
Based on the above descriptions, for the training, we minimize the sum of the negative log probabilities of the ground truth answer start position and the ground truth answer end position by the predicted distributions, and for the inference, the answer start position $a_s$ and the answer end position $a_e$ are chosen such that the product of $d_s[a_s]$ and $d_e[a_e]$ is maximized and $a_s \leq a_e$.

## 5 Related Works

Data enrichment has been widely used in the existing MRC models. For example, Yu et al. (2018) use translation models to paraphrase the original passages so as to generate extra training samples; Yang et al. (2017) generate extra training samples by training a generative model that generates questions based on unlabeled text; Chen et al. (2017), Liu et al. (2017b), and Pan et al. (2017) append linguistic tags, such as POS tag and NER tag, to each word in the original passages; and Liu et al. (2017a) generate a syntactic tree for each sentence in the original passage-question pairs. However, the above works just enrich the original MRC dataset with the outputs of certain external models or systems, therefore their MRC models can only make use of machine generated data, but cannot utilize human knowledge explicitly.
Attention mechanism has also been widely used in the existing MRC models. For example, Xiong et al. (2016) use a coattention encoder and a dynamic pointer decoder to address the local maximum problem; Seo et al. (2016) use a bidirectional attention flow mechanism to obtain the question-aware passage representation; and Wang et al. (2017) use a self-matching attention mechanism to refine the question-aware passage representation. The passage-to-question attention, question-to-passage attention, and self-matching attention in KAR draw on the ideas of the above works, but are different from them in that we integrate explicit knowledge with these attentions.

## 6 Experiments

### 6.1 MRC Dataset

The MRC dataset used in this paper is SQuAD, which contains over $100,000$ passage-question pairs and their answers. All questions and answers in SQuAD are human generated, and the answer to each question is a fragment in the corresponding passage. SQuAD has been randomly partitioned into three parts: the training set ($80\%$), the development set ($10\%$), and the test set ($10\%$). Both the training set and the development set are publicly available, while the test set is confidential. Besides, SQuAD adopts both Exact Match (EM) and F1 Score as the evaluation metrics.

### 6.2 Implementation Details

To implement KAR, we first preprocess SQuAD. Specifically, we put each passage and question in SQuAD through a Stanford CoreNLP (Manning et al., 2014) pipeline, which performs tokenization, sentence splitting, POS tagging, lemmatization, and dependency parsing in order. With the outputs of the pipeline, we use the WordNet interface provided by NLTK (Bird and Loper, 2004) to perform the WordNet-based data enrichment method, and thus obtain an enriched MRC dataset. Based on the data preprocessing, we implement KAR using TensorFlow (Abadi et al., 2016). For the character-level embedding, we set the forget-

EMNLP 2018 Submission 610. Confidential Review Copy. DO NOT DISTRIBUTE.

| $\chi$ | **Amount of the Extraction Results (connections per word)** | **Performance of KAR (EM / F1)** |
|---|---|---|
| 0 | 0.39 | 70.8 / 79.8 |
| 1 | 0.63 | 71.1 / 80.0 |
| 2 | 1.24 | 71.6 / 80.4 |
| **3** | **2.21** | **72.4 / 81.1** |
| 4 | 3.68 | 71.9 / 80.7 |
| 5 | 5.58 | 71.8 / 80.5 |

Table 2: The amount of the extraction results and the performance of KAR under each setting for $\chi$.

ting factor of FOFE to 0.7. For each BiLSTM, we set its hidden state size $d$ to 300. For the training, we use ADAM (Kingma and Ba, 2014) as our optimizer, set the learning rate to 0.0005, and set the mini-batch size to 40. To avoid overfitting, we apply dropout (Srivastava et al., 2014) to the word vectors, the character vectors, the input to each BiLSTM, and the linear transformation before each softmax in the answer span prediction layer, with a dropout rate of 0.2, and apply early stopping with a patience of 5. To avoid the exploding gradient problem, we apply gradient clipping (Pascanu et al., 2013) with a cutoff threshold of 2. Besides, we also apply exponential moving average with a decay rate of 0.999.

## 6.3 Experimental Process and Results

In this paper, we only consider the single model performance of MRC models on the development set of SQuAD. On this premise, we perform the following two experiments:

**1. Verifying the effects of explicit knowledge.** We obtain six enriched MRC datasets by setting $\chi$ to 0, 1, 2, 3, 4, and 5 separately, and train a different KAR on each enriched MRC dataset. As shown in Table 2, the amount of the extraction results increases monotonically as we increase $\chi$ from 0 to 5, but during this process, the performance of KAR first rises by 2% until $\chi$ reaches 3, and then begins to drop gradually. Thus it can be seen that the explicit knowledge provided by the WordNet-based data enrichment method plays an effective role in the training of KAR.

**2. Verifying the effects of dependency embedding and synonymy embedding.** By applying the optimal setting for $\chi$ (i.e. 3), we perform ablation analysis on the dependency embedding and the synonymy embedding. As shown in Table 3,

| **Ablation Part** | **Performance (EM / F1)** |
|---|---|
| Dependency Embedding | 71.6 / 80.2 |
| Synonymy Embedding | 70.9 / 79.5 |
| **No Ablation** | **72.4 / 81.1** |

Table 3: The ablation analysis on the dependency embedding and the synonymy embedding.

| **MRC Models** | **Performance (EM / F1)** |
|---|---|
| GDAN (Yang et al., 2017) | – / 67.2 |
| DCN (Xiong et al., 2016) | 65.4 / 75.6 |
| BiDAF (Seo et al., 2016) | 67.7 / 77.3 |
| SEDT (Liu et al., 2017a) | 68.1 / 77.5 |
| DrQA (Chen et al., 2017) | 69.5 / 78.8 |
| MEMEN (Pan et al., 2017) | 70.9 / 80.3 |
| R-NET (Wang et al., 2017) | 72.3 / 80.6 |
| **KAR (ours)** | **72.4 / 81.1** |
| QANet (Yu et al., 2018) | 75.1 / 83.8 |
| SAN (Liu et al., 2017b) | 76.2 / 84.0 |

Table 4: The comparison of different MRC models (published single model performance on the development set of SQuAD).

both of the two basic embeddings contribute to the performance of KAR, but the synonymy embedding seems to be more important.

Besides, we also compare the best performance of KAR with the published performance of the MRC models mentioned in the related works. As shown in Table 4, although KAR has achieved fairly good performance, there is still some way to go to catch up with the cutting-edge MRC models. This is because the scope of the general knowledge in WordNet is very limited, so that KAR cannot obtain enough useful explicit knowledge.

## 7 Conclusion

In this paper, we explore how to apply the general knowledge in WordNet as explicit knowledge to the training of an MRC model, and thereby propose the WordNet-based data enrichment method and KAR. Based on the explicit knowledge provided by the data enrichment method, KAR has achieved fairly good performance on SQuAD, and more importantly, the performance of KAR varies with the amount of the explicit knowledge. In the future work, we will use larger knowledge bases, such as Freebase, to improve the quality of the explicit knowledge provided to KAR.

EMNLP 2018 Submission 610. Confidential Review Copy. DO NOT DISTRIBUTE.

## References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

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

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *AAAI*, volume 6, page 6.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Rui Liu, Junjie Hu, Wei Wei, Zi Yang, and Eric Nyberg. 2017a. Structural embedding of syntactic trees for machine comprehension. *arXiv preprint arXiv:1703.00572*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017b. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv preprint arXiv:1707.09098*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

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

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1436–1446.

EMNLP 2018 Submission 610. Confidential Review Copy. DO NOT DISTRIBUTE.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. 2017. Semi-supervised qa with generative domain-adaptive nets. *arXiv preprint arXiv:1702.02206*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 495–500.

# Explicit Utilization of General Knowledge in Machine Reading Comprehension

**Chao Wang and Hui Jiang**
Department of Electrical Engineering and Computer Science
Lassonde School of Engineering, York University
4700 Keele Street, Toronto, Ontario, Canada
$\{chwang, hj\}$@eecs.yorku.ca

## Abstract

To bridge the gap between Machine Reading Comprehension (MRC) models and human beings, which is mainly reflected in the hunger for data and the robustness to noise, in this paper, we explore how to integrate the neural networks of MRC models with the general knowledge of human beings. On the one hand, we propose a data enrichment method, which uses WordNet to extract inter-word semantic connections as general knowledge from each given passage-question pair. On the other hand, we propose an end-to-end MRC model named as Knowledge Aided Reader (KAR), which explicitly uses the above extracted general knowledge to assist its attention mechanisms. Based on the data enrichment method, KAR is comparable in performance with the state-of-the-art MRC models, and significantly more robust to noise than them. When only a subset (20%–80%) of the training examples are available, KAR outperforms the state-of-the-art MRC models by a large margin, and is still reasonably robust to noise.

## 1 Introduction

Machine Reading Comprehension (MRC), as the name suggests, requires a machine to read a passage and answer its relevant questions. Since the answer to each question is supposed to stem from the corresponding passage, a common MRC solution is to develop a neural-network-based MRC model that predicts an answer span (i.e. the answer start position and the answer end position) from the passage of each given passage-question pair. To facilitate the explorations and innovations in this area, many MRC datasets have been established, such as SQuAD (Rajpurkar et al., 2016), MS MARCO (Nguyen et al., 2016), and TriviaQA (Joshi et al., 2017). Consequently, many pioneering MRC models have been proposed, such as BiDAF (Seo et al., 2016), R-NET (Wang et al., 2017), and QANet (Yu et al., 2018). According to the leader board of SQuAD, the state-of-the-art MRC models have achieved the same performance as human beings. However, does this imply that they have possessed the same reading comprehension ability as human beings?

OF COURSE NOT. There is a huge gap between MRC models and human beings, which is mainly reflected in the hunger for data and the robustness to noise. On the one hand, developing MRC models requires a large amount of training examples (i.e. the passage-question pairs labeled with answer spans), while human beings can achieve good performance on evaluation examples (i.e. the passage-question pairs to address) without training examples. On the other hand, Jia and Liang (2017) revealed that intentionally injected noise (e.g. misleading sentences) in evaluation examples causes the performance of MRC models to drop significantly, while human beings are far less likely to suffer from this. The reason for these phenomena, we believe, is that MRC models can only utilize the knowledge contained in each given passage-question pair, but in addition to this, human beings can also utilize general knowledge. A typical category of general knowledge is inter-word semantic connections. As shown in Table 1, such general knowledge is essential to the reading comprehension ability of human beings.

A promising strategy to bridge the gap mentioned above is to integrate the neural networks of MRC models with the general knowledge of human beings. To this end, it is necessary to solve two problems: extracting general knowledge from passage-question pairs and utilizing the extracted general knowledge in the prediction of answer spans. The first problem can be solved with knowledge bases, which store general knowledge in structured forms. A broad variety of knowledge bases are available, such as WordNet (Fellbaum, 1998) storing semantic knowledge, ConceptNet (Speer et al., 2017) storing commonsense knowledge, and

| Passage | Question | Answer |
|---|---|---|
| Teachers may use a lesson plan to **facilitate** student learning, providing a course of study which is called the curriculum. | What can a teacher use to **help** students learn? | lesson plan |
| Manufacturing accounts for a significant but declining share of employment, although the city's garment industry is showing a resurgence in **Brooklyn**. | In what **borough** is the garment business prominent? | Brooklyn |

Table 1: Two examples about the importance of inter-word semantic connections to the reading comprehension ability of human beings: in the first one, we can find the answer because we know "facilitate" is a synonym of "help"; in the second one, we can find the answer because we know "Brooklyn" is a hyponym of "borough".

Freebase (Bollacker et al., 2008) storing factoid knowledge. In this paper, we limit the scope of general knowledge to inter-word semantic connections, and thus use WordNet as our knowledge base. The existing way to solve the second problem is to encode general knowledge in vector space so that the encoding results can be used to enhance the lexical or contextual representations of words (Weissenborn et al., 2017; Mihaylov and Frank, 2018). However, this is an implicit way to utilize general knowledge, since in this way we can neither understand nor control the functioning of general knowledge. In this paper, we discard the existing implicit way and instead explore an explicit (i.e. understandable and controllable) way to utilize general knowledge.

The contribution of this paper is two-fold. On the one hand, we propose a data enrichment method, which uses WordNet to extract inter-word semantic connections as general knowledge from each given passage-question pair. On the other hand, we propose an end-to-end MRC model named as Knowledge Aided Reader (KAR), which explicitly uses the above extracted general knowledge to assist its attention mechanisms. Based on the data enrichment method, KAR is comparable in performance with the state-of-the-art MRC models, and significantly more robust to noise than them. When only a subset (20%–80%) of the training examples are available, KAR outperforms the state-of-the-art MRC models by a large margin, and is still reasonably robust to noise.

## 2 Data Enrichment Method

In this section, we elaborate a WordNet-based data enrichment method, which is aimed at extracting inter-word semantic connections from each passage-question pair in our MRC dataset. The extraction is performed in a controllable manner, and the extracted results are provided as general knowledge to our MRC model.

### 2.1 Semantic Relation Chain

WordNet is a lexical database of English, where words are organized into synsets according to their senses. A synset is a set of words expressing the same sense so that a word having multiple senses belongs to multiple synsets, with each synset corresponding to a sense. Synsets are further related to each other through semantic relations. According to the WordNet interface provided by NLTK (Bird and Loper, 2004), there are totally sixteen types of semantic relations (e.g. hypernyms, hyponyms, holonyms, meronyms, attributes, etc.). Based on synset and semantic relation, we define a new concept: semantic relation chain. A semantic relation chain is a concatenated sequence of semantic relations, which links a synset to another synset. For example, the synset "keratin.n.01" is related to the synset "feather.n.01" through the semantic relation "substance holonym", the synset "feather.n.01" is related to the synset "bird.n.01" through the semantic relation "part holonym", and the synset "bird.n.01" is related to the synset "parrot.n.01" through the semantic relation "hyponym", thus "substance holonym $\rightarrow$ part holonym $\rightarrow$ hyponym" is a semantic relation chain, which links the synset "keratin.n.01" to the synset "parrot.n.01". We name each semantic relation in a semantic relation chain as a hop, therefore the above semantic relation chain is a 3-hop chain. By the way, each single semantic relation is equivalent to a 1-hop chain.

### 2.2 Inter-word Semantic Connection

The key problem in the data enrichment method is determining whether a word is semantically connected to another word. If so, we say that there exists an inter-word semantic connection between them. To solve this problem, we define another new concept: the extended synsets of a word. Given a word $w$, whose synsets are represented as a set $S_w$, we use another set $S_w^*$ to represent its extended synsets, which includes all the synsets that are in $S_w$ or that can be linked to from $S_w$ through semantic relation chains. Theoretically, if there is no limitation on semantic relation chains, $S_w^*$ will include all the synsets in WordNet, which is meaningless in most situations. Therefore, we use a hyper-parameter $\kappa \in \mathbb{N}$ to represent the permitted maximum hop count of semantic relation chains. That is to say, only the chains having no more than $\kappa$ hops can be used to construct $S_w^*$ so that $S_w^*$ becomes a function of $\kappa$: $S_w^*(\kappa)$ (if $\kappa = 0$, we will have $S_w^*(0) = S_w$). Based on the above statements, we formulate a heuristic rule for determining inter-word semantic connections: a word $w_1$ is semantically connected to another word $w_2$ if and only if $S_{w_1}^*(\kappa) \cap S_{w_2} \neq \emptyset$.

### 2.3 General Knowledge Extraction

Given a passage-question pair, the inter-word semantic connections that connect any word to any passage word are regarded as the general knowledge we need to extract. Considering the requirements of our MRC model, we only extract the positional information of such inter-word semantic connections. Specifically, for each word $w$, we extract a set $E_w$, which includes the positions of the passage words that $w$ is semantically connected to (if $w$ itself is a passage word, we will exclude its own position from $E_w$). We can control the amount of the extracted results by setting the hyper-parameter $\kappa$: if we set $\kappa$ to 0, inter-word semantic connections will only exist between synonyms; if we increase $\kappa$, inter-word semantic connections will exist between more words. That is to say, by increasing $\kappa$ within a certain range, we can usually extract more inter-word semantic connections from a passage-question pair, and thus can provide the MRC model with more general knowledge. However, due to the complexity and diversity of natural languages, only a part of the extracted results can serve as useful general knowledge, while the rest of them are useless for the prediction of answer spans, and the proportion of the useless part always rises when $\kappa$ is set larger. Therefore we set $\kappa$ through cross validation (i.e. according to the performance of the MRC model on the development examples).

## 3 Knowledge Aided Reader

In this section, we elaborate our MRC model: Knowledge Aided Reader (KAR). The key components of most existing MRC models are their attention mechanisms (Bahdanau et al., 2014), which are aimed at fusing the associated representations of each given passage-question pair. These attention mechanisms generally fall into two categories: the first one, which we name as mutual attention, is aimed at fusing the question representations into the passage representations so as to obtain the question-aware passage representations; the second one, which we name as self attention, is aimed at fusing the question-aware passage representations into themselves so as to obtain the final passage representations. Although KAR is equipped with both categories, its most remarkable feature is that it explicitly uses the general knowledge extracted by the data enrichment method to assist its attention mechanisms. Therefore we separately name the attention mechanisms of KAR as knowledge aided mutual attention and knowledge aided self attention.

### 3.1 Task Definition

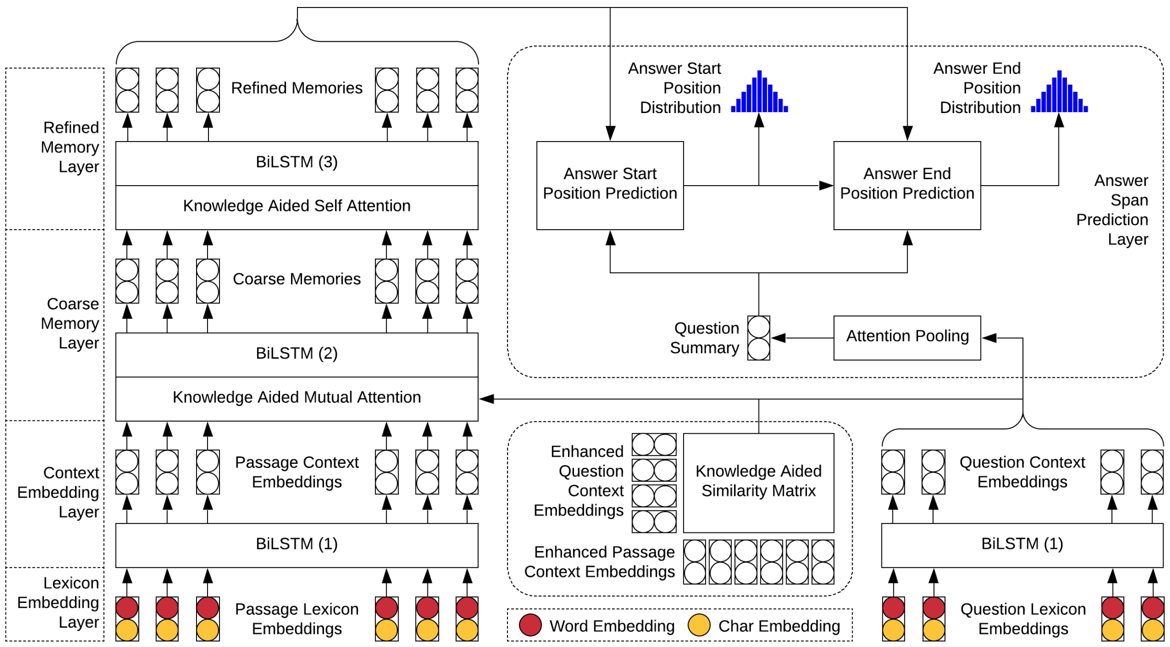
Given a passage $P = \{p_1, \ldots, p_n\}$ and a relevant question $Q = \{q_1, \ldots, q_m\}$, the task is to predict an answer span $[a_s, a_e]$, where $1 \leq a_s \leq a_e \leq n$, so that the resulting subsequence $\{p_{a_s}, \ldots, p_{a_e}\}$ from $P$ is an answer to $Q$.

### 3.2 Overall Architecture

As shown in Figure 1, KAR is an end-to-end MRC model consisting of five layers:

**Lexicon Embedding Layer.** This layer maps the words to the lexicon embeddings. The lexicon embedding of each word is composed of its word embedding and character embedding. For each word, we use the pre-trained GloVe (Pennington et al., 2014) word vector as its word embedding, and obtain its character embedding with a Convolutional Neural Network (CNN) (Kim, 2014). For both the passage and the question, we pass the concatenation of the word embeddings and the character embeddings through a shared dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the passage lexicon embeddings $L_P \in \mathbb{R}^{d \times n}$ and the question lexicon embeddings $L_Q \in \mathbb{R}^{d \times m}$.

**Context Embedding Layer.** This layer maps the lexicon embeddings to the context embeddings.



Refined Memory Layer
Coarse Memory Layer
Context Embedding Layer
Lexicon Embedding Layer
Refined Memories
BiLSTM (3)
Knowledge Aided Self Attention
Coarse Memories
BiLSTM (2)
Knowledge Aided Mutual Attention
Passage Context Embeddings
BiLSTM (1)
Passage Lexicon Embeddings
Answer Start Position Distribution
Answer End Position Distribution
Answer Start Position Prediction
Answer End Position Prediction
Answer Span Prediction Layer
Question Summary
Attention Pooling
Enhanced Question Context Embeddings
Knowledge Aided Similarity Matrix
Enhanced Passage Context Embeddings
Question Context Embeddings
BiLSTM (1)
Question Lexicon Embeddings
Word Embedding
Char Embedding

Figure 1: An end-to-end MRC model: Knowledge Aided Reader (KAR)

For both the passage and the question, we process the lexicon embeddings (i.e. $L_P$ for the passage and $L_Q$ for the question) with a shared bidirectional LSTM (BiLSTM) (Hochreiter and Schmidhuber, 1997), whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the passage context embeddings $C_P \in \mathbb{R}^{d\times n}$ and the question context embeddings $C_Q \in \mathbb{R}^{d\times m}$.

**Coarse Memory Layer.** This layer maps the context embeddings to the coarse memories. First we use knowledge aided mutual attention (introduced later) to fuse $C_Q$ into $C_P$, the outputs of which are represented as $\tilde{G} \in \mathbb{R}^{d\times n}$. Then we process $\tilde{G}$ with a BiLSTM, whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the coarse memories $G \in \mathbb{R}^{d\times n}$, which are the question-aware passage representations.

**Refined Memory Layer.** This layer maps the coarse memories to the refined memories. First we use knowledge aided self attention (introduced later) to fuse $G$ into themselves, the outputs of which are represented as $\tilde{H} \in \mathbb{R}^{d\times n}$. Then we process $\tilde{H}$ with a BiLSTM, whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the refined memories $H \in \mathbb{R}^{d\times n}$, which are the final passage representations.

**Answer Span Prediction Layer.** This layer predicts the answer start position and the answer end position based on the above layers. First we obtain the answer start position distribution $o_s$:

$$t_i = v_s^\top \tanh(W_s h_{p_i} + U_s r_Q) \in \mathbb{R}$$

$$o_s = \mathrm{softmax}(\{t_1, \ldots, t_n\}) \in \mathbb{R}^n$$

where $v_s$, $W_s$, and $U_s$ are trainable parameters; $h_{p_i}$ represents the refined memory of each passage word $p_i$ (i.e. the $i$-th column in $H$); $r_Q$ represents the question summary obtained by performing an attention pooling over $C_Q$. Then we obtain the answer end position distribution $o_e$:

$$t_i = v_e^\top \tanh(W_e h_{p_i} + U_e[r_Q; H o_s]) \in \mathbb{R}$$

$$o_e = \mathrm{softmax}(\{t_1, \ldots, t_n\}) \in \mathbb{R}^n$$

where $v_e$, $W_e$, and $U_e$ are trainable parameters; $[;]$ represents vector concatenation. Finally we construct an answer span prediction matrix $O = \mathrm{uptri}(o_s o_e^\top) \in \mathbb{R}^{n\times n}$, where $\mathrm{uptri}(X)$ represents the upper triangular matrix of a matrix $X$. Therefore, for the training, we minimize $-\log(O_{a_s,a_e})$ on each training example whose labeled answer span is $[a_s, a_e]$; for the inference, we separately take the row index and column index of the maximum element in $O$ as $a_s$ and $a_e$.

## 3.3 Knowledge Aided Mutual Attention

As a part of the coarse memory layer, knowledge aided mutual attention is aimed at fusing the question context embeddings $C_Q$ into the passage context embeddings $C_P$, where the key problem is to calculate the similarity between each passage context embedding $c_{p_i}$ (i.e. the $i$-th column in $C_P$) and each question context embedding $c_{q_j}$ (i.e. the $j$-th column in $C_Q$). To solve this problem, Seo et al. (2016) proposed a similarity function:

$$f(c_{p_i}, c_{q_j}) = v_f^\top [c_{p_i}; c_{q_j}; c_{p_i} \odot c_{q_j}] \in \mathbb{R}$$

where $v_f$ is a trainable parameter; $\odot$ represents element-wise multiplication. This similarity function has also been adopted by several other works (Clark and Gardner, 2017; Yu et al., 2018). However, since context embeddings contain high-level information, we believe that introducing the pre-extracted general knowledge into the calculation of such similarities will make the results more reasonable. Therefore we modify the above similarity function to the following form:

$$f^*(c_{p_i}, c_{q_j}) = v_f^\top [c^*_{p_i}; c^*_{q_j}; c^*_{p_i} \odot c^*_{q_j}] \in \mathbb{R}$$

where $c^*_x$ represents the enhanced context embedding of a word $x$. We use the pre-extracted general knowledge to construct the enhanced context embeddings. Specifically, for each word $w$, whose context embedding is $c_w$, to construct its enhanced context embedding $c^*_w$, first recall that we have extracted a set $E_w$, which includes the positions of the passage words that $w$ is semantically connected to, thus by gathering the columns in $C_P$ whose indexes are given by $E_w$, we obtain the matching context embeddings $Z \in \mathbb{R}^{d \times |E_w|}$. Then by constructing a $c_w$-attended summary of $Z$, we obtain the matching vector $c^+_w$ (if $E_w = \emptyset$, which makes $Z = \{\}$, we will set $c^+_w = 0$):

$$t_i = v_c^\top \tanh(W_c z_i + U_c c_w) \in \mathbb{R}$$

$$c^+_w = Z \,\mathrm{softmax}(\{t_1, \ldots, t_{|E_w|}\}) \in \mathbb{R}^d$$

where $v_c$, $W_c$, and $U_c$ are trainable parameters; $z_i$ represents the $i$-th column in $Z$. Finally we pass the concatenation of $c_w$ and $c^+_w$ through a dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the enhanced context embedding $c^*_w \in \mathbb{R}^d$.

Based on the modified similarity function and the enhanced context embeddings, to perform knowledge aided mutual attention, first we construct a knowledge aided similarity matrix $A \in \mathbb{R}^{n \times m}$, where each element $A_{i,j} = f^*(c_{p_i}, c_{q_j})$. Then following Yu et al. (2018), we construct the passage-attended question summaries $R_Q$ and the question-attended passage summaries $R_P$:

$$R_Q = C_Q \,\mathrm{softmax}_r^\top(A) \in \mathbb{R}^{d \times n}$$

$$R_P = C_P \,\mathrm{softmax}_c(A) \,\mathrm{softmax}_r^\top(A) \in \mathbb{R}^{d \times n}$$

where $\mathrm{softmax}_r$ represents softmax along the row dimension and $\mathrm{softmax}_c$ along the column dimension. Finally following Clark and Gardner (2017), we pass the concatenation of $C_P$, $R_Q$, $C_P \odot R_Q$, and $R_P \odot R_Q$ through a dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the outputs $\tilde{G} \in \mathbb{R}^{d \times n}$.

## 3.4 Knowledge Aided Self Attention

As a part of the refined memory layer, knowledge aided self attention is aimed at fusing the coarse memories $G$ into themselves. If we simply follow the self attentions of other works (Wang et al., 2017; Huang et al., 2017; Liu et al., 2017b; Clark and Gardner, 2017), then for each passage word $p_i$, we should fuse its coarse memory $g_{p_i}$ (i.e. the $i$-th column in $G$) with the coarse memories of all the other passage words. However, we believe that this is both unnecessary and distracting, since each passage word has nothing to do with many of the other passage words. Thus we use the pre-extracted general knowledge to guarantee that the fusion of coarse memories for each passage word will only involve a precise subset of the other passage words. Specifically, for each passage word $p_i$, whose coarse memory is $g_{p_i}$, to perform the fusion of coarse memories, first recall that we have extracted a set $E_{p_i}$, which includes the positions of the other passage words that $p_i$ is semantically connected to, thus by gathering the columns in $G$ whose indexes are given by $E_{p_i}$, we obtain the matching coarse memories $Z \in \mathbb{R}^{d \times |E_{p_i}|}$. Then by constructing a $g_{p_i}$-attended summary of $Z$, we obtain the matching vector $g^+_{p_i}$ (if $E_{p_i} = \emptyset$, which makes $Z = \{\}$, we will set $g^+_{p_i} = 0$):

$$t_i = v_g^\top \tanh(W_g z_i + U_g g_{p_i}) \in \mathbb{R}$$

$$g^+_{p_i} = Z \,\mathrm{softmax}(\{t_1, \ldots, t_{|E_{p_i}|}\}) \in \mathbb{R}^d$$

where $v_g$, $W_g$, and $U_g$ are trainable parameters. Finally we pass the concatenation of $g_{p_i}$ and $g^+_{p_i}$ through a dense layer with ReLU activation, whose output dimensionality is $d$. Therefore we obtain the fusion result $\tilde{h}_{p_i} \in \mathbb{R}^d$, and further the outputs $\tilde{H} = \{\tilde{h}_{p_1}, \ldots, \tilde{h}_{p_n}\} \in \mathbb{R}^{d \times n}$.

## 4 Related Works

**Attention Mechanisms.** Besides those mentioned above, other interesting attention mechanisms include performing multi-round alignment to avoid the problems of attention redundancy and attention deficiency (Hu et al., 2017), and using mutual attention as a skip-connector to densely connect pairwise layers (Tay et al., 2018).

**Data Augmentation.** It is proved that properly augmenting training examples can improve the performance of MRC models. For example, Yang et al. (2017) trained a generative model to generate questions based on unlabeled text, which substantially boosted their performance; Yu et al. (2018) trained a back-and-forth translation model to paraphrase training examples, which brought them a significant performance gain.

**Multi-step Reasoning.** Inspired by the fact that human beings are capable of understanding complex documents by reading them over and over again, multi-step reasoning was proposed to better deal with difficult MRC tasks. For example, Shen et al. (2017) used reinforcement learning to dynamically determine the number of reasoning steps; Liu et al. (2017b) fixed the number of reasoning steps, but used stochastic dropout in the output layer to avoid step bias.

**Linguistic Embeddings.** It is both easy and effective to incorporate linguistic embeddings into the input layer of MRC models. For example, Chen et al. (2017) and Liu et al. (2017b) used POS embeddings and NER embeddings to construct their input embeddings; Liu et al. (2017a) used structural embeddings based on parsing trees to constructed their input embeddings.

**Transfer Learning.** Several recent breakthroughs in MRC benefit from feature-based transfer learning (McCann et al., 2017; Peters et al., 2018) and fine-tuning-based transfer learning (Radford et al., 2018; Devlin et al., 2018), which are based on certain word-level or sentence-level models pre-trained on large external corpora in certain supervised or unsupervised manners.

## 5 Experiments

### 5.1 Experimental Settings

**MRC Dataset.** The MRC dataset used in this paper is SQuAD 1.1, which contains over $100,000$ passage-question pairs and has been randomly partitioned into three parts: a training set ($80\%$), a development set ($10\%$), and a test set ($10\%$). Besides, we also use two of its adversarial sets, namely AddSent and AddOneSent (Jia and Liang, 2017), to evaluate the robustness to noise of MRC models. The passages in the adversarial sets contain misleading sentences, which are aimed at distracting MRC models. Specifically, each passage in AddSent contains several sentences that are similar to the question but not contradictory to the answer, while each passage in AddOneSent contains a human-approved random sentence that may be unrelated to the passage.

**Implementation Details.** We tokenize the MRC dataset with spaCy 2.0.13 (Honnibal and Montani, 2017), manipulate WordNet 3.0 with NLTK 3.3, and implement KAR with TensorFlow 1.11.0 (Abadi et al., 2016). For the data enrichment method, we set the hyper-parameter $\kappa$ to 3. For the dense layers and the BiLSTMs, we set the dimensionality unit $d$ to 600. For model optimization, we apply the Adam (Kingma and Ba, 2014) optimizer with a learning rate of $0.0005$ and a mini-batch size of $32$. For model evaluation, we use Exact Match (EM) and F1 score as evaluation metrics. To avoid overfitting, we apply dropout (Srivastava et al., 2014) to the dense layers and the BiLSTMs with a dropout rate of $0.3$. To boost the performance, we apply exponential moving average with a decay rate of $0.999$.

### 5.2 Model Comparison in both Performance and the Robustness to Noise

We compare KAR with other MRC models in both performance and the robustness to noise. Specifically, we not only evaluate the performance of KAR on the development set and the test set, but also do this on the adversarial sets. As for the comparative objects, we only consider the single MRC models that rank in the top 20 on the SQuAD 1.1 leader board and have reported their performance on the adversarial sets. There are totally five such comparative objects, which can be considered as representatives of the state-of-the-art MRC models. As shown in Table 2, on the development set and the test set, the performance of KAR is on par with that of the state-of-the-art MRC models; on the adversarial sets, KAR outperforms the state-of-the-art MRC models by a large margin. That is to say, KAR is comparable in performance with the state-of-the-art MRC models, and significantly more robust to noise than them.

| Single MRC model | Dev set (EM / F1) | Test set (EM / F1) | AddSent (F1) | AddOneSent (F1) |
|---|---|---|---|---|
| FusionNet (Huang et al., 2017) | 75.3 / 83.6 | 76.0 / 83.9 | 51.4 | 60.7 |
| RaSoR+TR+LM (Salant and Berant, 2017) | 77.0 / 84.0 | 77.6 / 84.2 | 47.0 | 57.0 |
| SAN (Liu et al., 2017b) | 76.2 / 84.1 | 76.8 / 84.4 | 46.6 | 56.5 |
| R.M-Reader (Hu et al., 2017) | **78.9 / 86.3** | 79.5 / 86.6 | 58.5 | 67.0 |
| QANet (with data augmentation) (Yu et al., 2018) | 75.1 / 83.8 | **82.5 / 89.3** | 45.2 | 55.7 |
| **KAR (ours)** | 76.7 / 84.9 | 76.1 / 83.5 | **60.1** | **72.3** |

Table 2: Model comparison based on SQuAD 1.1 and two of its adversarial sets: AddSent and AddOneSent. All the numbers are up to date as of October 18, 2018. Note that SQuAD 2.0 (Rajpurkar et al., 2018) is not involved in this paper, because it requires MRC models to deal with the problem of answer triggering, but this paper is aimed at improving the hunger for data and robustness to noise of MRC models.

To verify the effectiveness of general knowledge, we first study the relationship between the amount of general knowledge and the performance of KAR. As shown in Table 3, by increasing $\kappa$ from 0 to 5 in the data enrichment method, the amount of general knowledge rises monotonically, but the performance of KAR first rises until $\kappa$ reaches 3 and then drops down. Then we conduct an ablation study by replacing the knowledge aided attention mechanisms with the mutual attention proposed by Seo et al. (2016) and the self attention proposed by Wang et al. (2017) separately, and find that the F1 score of KAR drops by 4.2 on the development set, 7.8 on AddSent, and 9.1 on AddOneSent. Finally we find that after only one epoch of training, KAR already achieves an EM of 71.9 and an F1 score of 80.8 on the development set, which is even better than the final performance of several strong baselines, such as DCN (EM / F1: 65.4 / 75.6) (Xiong et al., 2016) and BiDAF (EM / F1: 67.7 / 77.3) (Seo et al., 2016). The above empirical findings imply that general knowledge indeed plays an effective role in KAR.

To demonstrate the advantage of our explicit way to utilize general knowledge over the existing implicit way, we compare the performance of KAR with that reported by Weissenborn et al. (2017), which used an encoding-based method to utilize the general knowledge dynamically retrieved from Wikipedia and ConceptNet. Since their best model only achieved an EM of 69.5 and an F1 score of 79.7 on the development set, which is much lower than the performance of KAR, we have good reason to believe that our explicit way works better than the existing implicit way.

| $\kappa$ | Average number of inter-word semantic connections per word | Dev set (EM / F1) |
|---|---|---|
| 0 | 0.39 | 74.2 / 82.8 |
| 1 | 0.63 | 74.6 / 83.1 |
| 2 | 1.24 | 75.1 / 83.5 |
| **3** | **2.21** | **76.7 / 84.9** |
| 4 | 3.68 | 75.9 / 84.3 |
| 5 | 5.58 | 75.3 / 83.8 |

Table 3: With $\kappa$ set to different values in the data enrichment method, we calculate the average number of inter-word semantic connections per word as an estimation of the amount of general knowledge, and evaluate the performance of KAR on the development set.

### 5.3 Model Comparison in the Hunger for Data

We compare KAR with other MRC models in the hunger for data. Specifically, instead of using all the training examples, we produce several training subsets (i.e. subsets of the training examples) so as to study the relationship between the proportion of the available training examples and the performance. We produce each training subset by sampling a specific number of questions from all the questions relevant to each passage. By separately sampling 1, 2, 3, and 4 questions on each passage, we obtain four training subsets, which separately contain 20%, 40%, 60%, and 80% of the training examples. As shown in Figure 2, with KAR, SAN (re-implemented), and QANet (re-implemented without data augmentation) trained on these training subsets, we evaluate their performance on the development set, and find that KAR

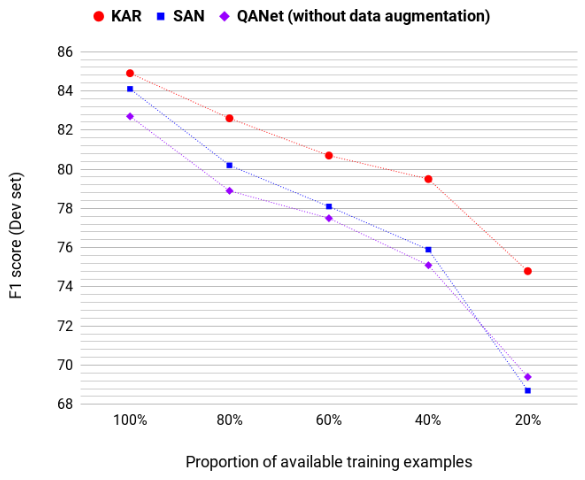


Figure 2: With KAR, SAN, and QANet (without data augmentation) trained on the training subsets, we evaluate their performance on the development set.

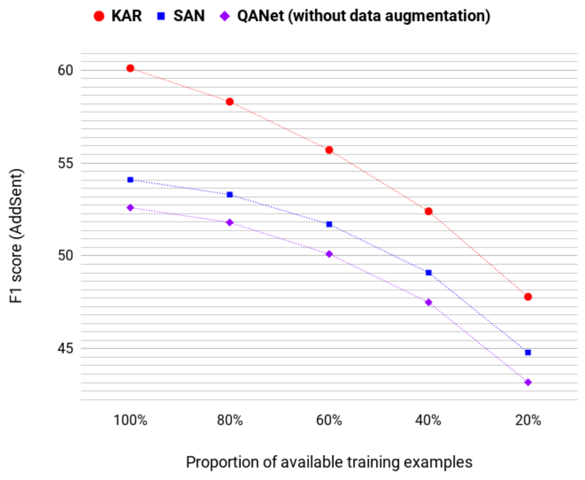


Figure 3: With KAR, SAN, and QANet (without data augmentation) trained on the training subsets, we evaluate their performance on AddSent.

performs much better than SAN and QANet. As shown in Figure 3 and Figure 4, with the above KAR, SAN, and QANet trained on the same training subsets, we also evaluate their performance on the adversarial sets, and still find that KAR performs much better than SAN and QANet. That is to say, when only a subset of the training examples are available, KAR outperforms the state-of-the-art MRC models by a large margin, and is still reasonably robust to noise.

## 6 Analysis

According to the experimental results, KAR is not only comparable in performance with the state-of-the-art MRC models, but also superior to them in terms of both the hunger for data and the robustness to noise. The reasons for these achievements, we believe, are as follows:

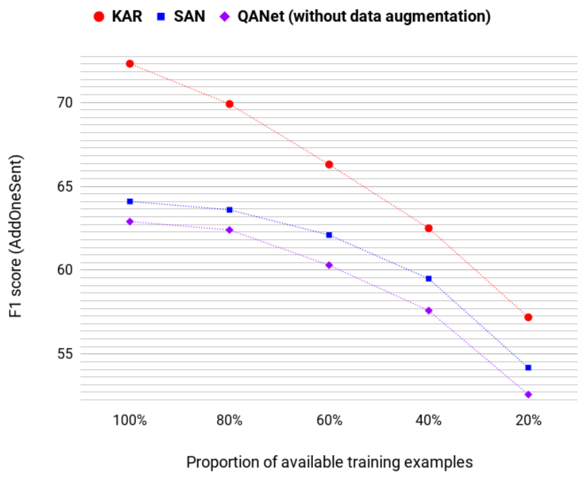


Figure 4: With KAR, SAN, and QANet (without data augmentation) trained on the training subsets, we evaluate their performance on AddOneSent.

- KAR is designed to utilize the pre-extracted inter-word semantic connections from the data enrichment method. Some inter-word semantic connections, especially those obtained through multi-hop semantic relation chains, are very helpful for the prediction of answer spans, but they will be too covert to capture if we simply leverage recurrent neural networks (e.g. BiLSTM) and pre-trained word vectors (e.g. GloVe).

- An inter-word semantic connection extracted from a passage-question pair usually also appears in many other passage-question pairs, therefore it is very likely that the inter-word semantic connections extracted from a small amount of training examples actually cover a much larger amount of training examples. That is to say, we are actually using much more training examples for model optimization than the available ones.

- Some inter-word semantic connections are distracting for the prediction of answer spans. For example, the inter-word semantic connection between "bank" and "waterside" makes no sense given the context "the bank manager is walking along the waterside". It is the knowledge aided attention mechanisms that enable KAR to ignore such distracting inter-word semantic connections so that only the important ones are used.

## 7 Conclusion

In this paper, we innovatively integrate the neural networks of MRC models with the general knowledge of human beings. Specifically, inter-word semantic connections are first extracted from each given passage-question pair by a WordNet-based data enrichment method, and then provided as general knowledge to an end-to-end MRC model named as Knowledge Aided Reader (KAR), which explicitly uses the general knowledge to assist its attention mechanisms. Experimental results show that KAR is not only comparable in performance with the state-of-the-art MRC models, but also superior to them in terms of both the hunger for data and the robustness to noise. In the future, we plan to use some larger knowledge bases, such as ConceptNet and Freebase, to improve the quality and scope of the general knowledge.

## Acknowledgments

This work is partially supported by a research donation from iFLYTEK Co., Ltd., Hefei, China, and a discovery grant from Natural Sciences and Engineering Research Council (NSERC) of Canada.

## References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

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

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2017. Reinforced mnemonic reader for machine reading comprehension. *arXiv preprint arXiv:1705.02798*.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2017. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Rui Liu, Junjie Hu, Wei Wei, Zi Yang, and Eric Nyberg. 2017a. Structural embedding of syntactic trees for machine comprehension. *arXiv preprint arXiv:1703.00572*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017b. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. *arXiv preprint arXiv:1805.07858*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Shimi Salant and Jonathan Berant. 2017. Contextualized word representations for reading comprehension. *arXiv preprint arXiv:1712.03609*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. 2018. Densely connected attention propagation for reading comprehension. In *Advances in Neural Information Processing Systems*, pages 4906–4917.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Dirk Weissenborn, Tomáš Kočiský, and Chris Dyer. 2017. Dynamic integration of background knowledge in neural nlu systems. *arXiv preprint arXiv:1706.02596*.

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

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. 2017. Semi-supervised qa with generative domain-adaptive nets. *arXiv preprint arXiv:1702.02206*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

# Learning Convolutional Neural Networks using Hybrid Orthogonal Projection and Estimation

**Hengyue Pan** PANHY@CSE.YORKU.CA

**Hui Jiang** HJ@CSE.YORKU.CA

*iFLYTEK Laboratory for Neural Computing and Machine Learning (iNCML), Department of Electrical Engineering and Computer Science, York University, 4700 Keele Street, Toronto, Ontario, M3J 1P3, CANADA*

**Editors:** Yung-Kyun Noh and Min-Ling Zhang

## Abstract

Convolutional neural networks (CNNs) have yielded the excellent performance in a variety of computer vision tasks, where CNNs typically adopt a similar structure consisting of convolution layers, pooling layers and fully connected layers. In this paper, we propose to apply a novel method, namely Hybrid Orthogonal Projection and Estimation (HOPE), to CNNs in order to introduce orthogonality into the CNN structure. The HOPE model can be viewed as a hybrid model to combine feature extraction using orthogonal linear projection with mixture models. It is an effective model to extract useful information from the original high-dimension feature vectors and meanwhile filter out irrelevant noises. In this work, we present three different ways to apply the HOPE models to CNNs, i.e., *HOPE-Input*, *single-HOPE-Block* and *multi-HOPE-Blocks*. For *HOPE-Input* CNNs, a HOPE layer is directly used right after the input to de-correlate high-dimension input feature vectors. Alternatively, in *single-HOPE-Block* and *multi-HOPE-Blocks* CNNs, we consider to use HOPE layers to replace one or more blocks in the CNNs, where one block may include several convolutional layers and one pooling layer. The experimental results on CIFAR-10, CIFAR-100 and ImageNet databases have shown that the orthogonal constraints imposed by the HOPE layers can significantly improve the performance of CNNs in these image classification tasks (we have achieved one of the best performance when image augmentation has not been applied, and top 5 performance with image augmentation).

**Keywords:** Deep Learning, Neural Networks, HOPE

## 1. Introduction

Convolutional neural networks (CNNs) (LeCun et al., 1990) currently play an important role in the deep learning and computer vision fields. In the past several years, researchers have revealed that CNNs can give the state-of-the-art performance in many computer vision tasks, especially for image classification and object detection tasks (Krizhevsky et al., 2012a; Szegedy et al., 2014; Simonyan and Zisserman, 2015; Pan and Jiang, 2017). Comparing with the fully connected deep neural networks (DNNs), CNNs are superior in exploiting spatial constraints and in turn extracting better local features from input images using the convolution layers and weight sharing, and furthermore may provide better invariance through the pooling mechanism. All of these make CNNs very suitable for image-related tasks (LeCun and Bengio, 1995). Moreover, large-scale deep CNNs can be effectively learned end-to-end in a supervised way from a large amount of labelled images.

In the past several years, a tremendous amount of research efforts have been devoted to further improve the performance of deep CNNs. In (Hinton et al., 2012; Srivastava et al., 2014), the dropout method has been proposed to prevent CNNs from overfitting by randomly dropping a small portion of hidden nodes in the network during the training procedure. Many experiments have confirmed that the dropout technique can significantly improve the network performance, especially when only a small training set is available. Besides, a similar idea, called dropconnect (Wan et al., 2013), has been proposed to drop connections between layers instead of hidden nodes during the training stage. Another interesting research field is to design good nonlinear activation functions for neural networks beyond the popular rectified linear function (ReLU), such as maxout (Goodfellow et al., 2013) and PReLU (He et al., 2015), which are also demonstrated to yield improvement in terms of classification performance. On the other hand, another important path to improve model performance is to search for some new CNN structures. For example, in (Lin et al., 2013), Network in Network (NIN) has been proposed, in which one micro neural network is used to replace the regular linear convolutional filter. Recurrent Convolutional Neural Network (RCNN) (Liang and Hu, 2015) is another new CNN structure, which introduces recurrent connections into the convolution layers.

More recently, a novel model, called Hybrid Orthogonal Projection and Estimation (HOPE) (Zhang et al., 2016), has been proposed to learn fully-connected deep neural networks in either supervised or unsupervised ways. This model introduces a linear orthogonal projection to reduce the dimensionality of the raw high-dimension data and then uses a finite mixture distribution to model the extracted features. By splitting the feature extraction and data modeling into two separate stages, it may derive a good feature extraction model that can generate better low-dimension features for the further learning process. More importantly, based on the analysis in (Zhang et al., 2016), the HOPE model has a tight relationship with neural networks since each hidden layer of DNNs can also be viewed as a HOPE model being composed of the feature extraction layer and data modeling layer. Therefore, the maximum likelihood based unsupervised learning as well as the minimum cross-entropy error based supervised learning algorithms can be used to learn neural networks under the HOPE framework for deep learning. In this case, the standard back-propagation method may be used to optimize the objective function to learn the models except that the orthogonal constraints are imposed for all projection layers during the training procedure.

However, (Zhang et al., 2016) has not taken CNNs into account but merely investigated the HOPE models for the fully connected neural networks and demonstrated good performance in the small MNIST data set. To make the HOPE model work for more image-related tasks, we need to consider how to combine the basic idea of the HOPE model with non-fully connected CNNs. In this paper, we extend the HOPE model to the popular CNNs by considering the special model structures of both convolution and pooling layers, and further consider how to introduce the orthogonal constraints into the CNN model structure and learn CNNs under the HOPE framework. The main contribution of this paper is to propose a suitable method to split one convolution layer into one HOPE projection layer and one HOPE model layer. The projection layer introduces orthogonal constraints into the convolution filters and removes the correlations from the feature maps of convolution layers, and the model layer can then model the projected vectors. Moreover, the proposed HOPE CNNs can be learned end-to-end via a modified error back-propagation algorithm. Specifically, we force the convolution filter in the projection layer becomes orthogonal during the weight updating process, then with the moving of the orthogonal convolution filter, most noise signals of each local part in the input feature maps can be removed. The proposed HOPE CNN framework is technically novel and significantly differs from previous HOPE DNN models, because the projection layers of HOPE CNNs work on the convolution filters and the model layers not only consider single projected vector, but also its neighbors. The most straightforward idea is to use a HOPE layer as the first hidden layer in CNNs to de-correlate the high-dimension input CNN features and remove the irrelevant noises, which is called a HOPE-Input layer. This idea is similar as the original formulation in (Zhang et al., 2016) except the HOPE model is applied to each convolutional filter. Moreover, we may introduce even more HOPE layers into the CNNs for better performance. Generally speaking, we can split one CNN into several building blocks, and each block may include several convolutional layers and end with one pooling layer. In practice, we can either replace one block (single-HOPE-Block CNNs) or multiple blocks (multi-HOPE-Blocks CNNs) by using HOPE layers for better performance.

Our experimental results on CIFAR-10, CIFAR-100 and ImageNet databases have shown that the application of HOPE layers results in significant performance improvement over the regular CNN baseline models.

## 2. Hybrid Orthogonal Projection and Estimation (HOPE) Framework

In the original Hybrid Orthogonal Projection and Estimation (HOPE) formulation (Zhang et al., 2016), it is assumed that any high-dimension feature vector can be modelling by a hybrid model consisting of feature extraction using a linear orthogonal projection and statistic modeling using a finite mixture model. Assuming that each high-dimension feature vector $\mathbf{x}$ is of dimension $D$, then the linear orthogonal projection maps $\mathbf{x}$ to an $M$-dimension feature space ($M < D$), and the projected vector may retain the most useful information of $\mathbf{x}$. Specifically, we define a $D \times D$ orthogonal matrix $[\mathbf{U}; \quad V]$ which satisfies:

$$[\mathbf{z}; \quad \mathbf{n}] = \begin{bmatrix} \mathbf{U} \\ V \end{bmatrix} \mathbf{x} \tag{1}$$

where $\mathbf{z}$ is an $M$-dimension vector, called the signal component, and $\mathbf{n}$ is the residual noise vector with the dimensionality of $D - M$.

In practice, $\mathbf{z}$ is heavily de-correlated but it may still locate in a rather high dimension feature space. In the HOPE formulation, it is proposed to model $\mathbf{z}$ with a finite mixture model:

$$p(\mathbf{z}) = \sum_{k=1}^{K} \pi_k \cdot f_k(\mathbf{z}|\theta_k) \tag{2}$$

where $K$ is the number of mixture components, $\pi_k$ is the mixture weight of the $k$th component ($\sum_{k=1}^{K} \pi_k = 1$), $f_k()$ denotes a selected distribution from the exponential family, and $\theta_k$ denotes all model parameters of $f_k()$. As discussed in (Zhang et al., 2016), if the von Mises-Fisher (vMF) distribution is chosen for $f_k()$, the resultant HOPE model is equivalent in mathematical formulation to a hidden layer in neural networks using the popular rectified linear units (ReLU).

The HOPE model combines a linear orthogonal projection and a finite mixture model under a unified generative modeling framework. It can be learned unsupervisingly based on maximum likelihood estimation from unlabelled data as well as discriminatively from labelled data. In (Zhang et al., 2016), the HOPE model has been applied to the fully connected DNNs, and the models can be learned in either supervised or unsupervised ways. For one hidden layer with input vector $\mathbf{x}$ ($\mathbf{x} \in R^D$) and output vector $\mathbf{y}$ ($\mathbf{y} \in R^G$), it is first split into two layers: i) The first layer is a linear orthogonal projection layer, which is used to project $\mathbf{x}$ to a feature vector $\mathbf{z}$ ($\mathbf{z} \in R^M, M < D$) and remove the noise signals by using an orthogonal projection matrix $\mathbf{U}$:

$$\mathbf{z} = \mathbf{U}\mathbf{x}. \tag{3}$$

ii) The second layer is a non-linear model layer, which convert $\mathbf{z}$ to the output vector $\mathbf{y}$ following the selected model $f_k()$ and a nonlinear log-likelihood pruning operation (in supervised learning the model can be learned automatically from the training dataset). An example of a HOPE layer in DNNs is shown in Figure 1.

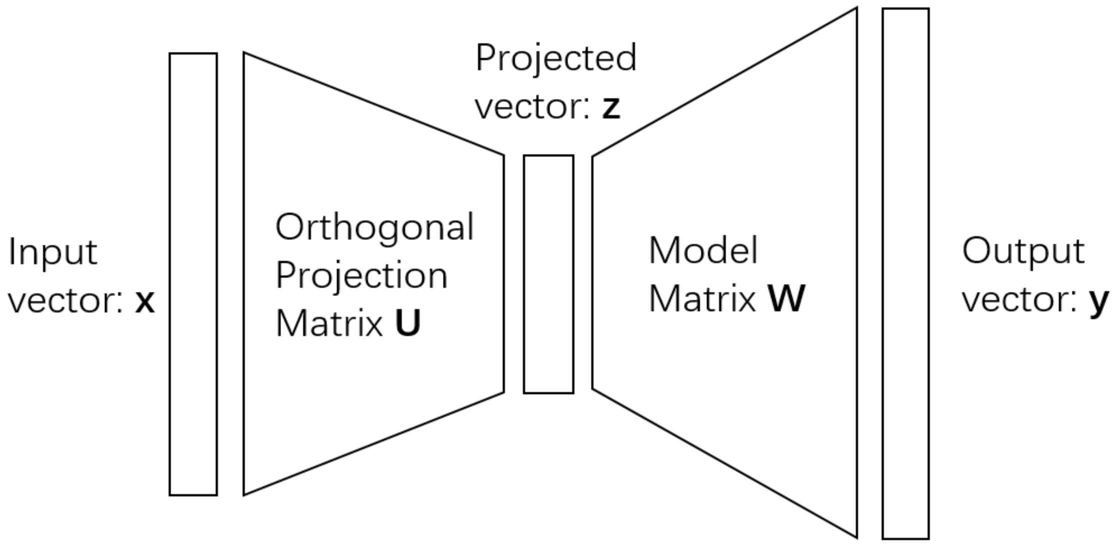


Figure 1: The HOPE model is viewed as a hidden layer in DNNs.

As in (Zhang et al., 2016), all HOPE model parameters, including the projection matrix $U$ and the model matrix $W$, can be learned, using the error back-propagation algorithm with stochastic gradient descent, to optimize an objective function subject to an orthogonal constraint, $\mathbf{U}\mathbf{U}^T = \mathbf{I}$, for each projection layer. As in (Zhang et al., 2016), for computational simplicity, the constraint is cast as the following penalty term to gradually de-correlate the matrix $\mathbf{U}$ during the learning process:

$$P(\mathbf{U}) = \sum_{i=1}^{M} \sum_{j=i+1}^{M} \frac{|\mathbf{u}_i \cdot \mathbf{u}_j|}{|\mathbf{u}_i| \cdot |\mathbf{u}_j|}. \tag{4}$$

In (Zhang et al., 2016), both unsupervised learning and supervised learning are studied for DNNs under the HOPE framework. The above orthogonal constraint is found to be equally important in both scenarios. In this paper, we will study how to supervisingly learn CNNs under the HOPE formulation and more specifically investigate how to introduce the orthogonality into the CNN model structure.

## 3. Our Proposed Method

In (Zhang et al., 2016), the authors have applied the HOPE model to the fully connected DNNs and have achieved good performance in experiments on small databases like MNIST. However, more widely used neural models in computer vision, i.e. convolutional neural networks (CNNs), have not been considered. Unlike DNNs, CNNs adopt some unique model structures and have achieved huge successes in many large-scale image classification tasks. Therefore, it is interesting to consider how to combine the HOPE model with CNNs to further improve image classification performance.

## 3.1. Applying the HOPE model to CNNs

To introduce the HOPE model to CNNs, the most straightforward solution is to split each convolution layer into a concatenation of a projection layer and a model layer and impose the orthogonal constraints onto the projection layer as in (Zhang et al., 2016). Assuming that we have a regular convolution layer in CNNs, which uses some $S \times S$ linear filters to map from $C_i$ input feature maps to $C_m$ output feature maps. As shown in Figure 2, under the HOPE framework, we propose to split this convolution layer into the two separate layers:

i) One linear orthogonal projection layer with the projection matrix $\mathbf{U}$: in the projection layer, we use each local region that is 'covered' by the orthogonal convolution filter in the input feature maps as the basic unit of the orthogonal projection. Specifically, the orthogonal convolution filter linearly maps a 3-dimension tensor with the size of $S \times S \times C_i$ into a vector $1 \times 1 \times C_p$, $C_p$ denotes the number of feature maps to be used in the projection layer. As the projection filters convolve with the input layer, it generates a total of $C_p$ feature maps in the projection layer. The projection filter itself is a 4-dimension tensor with the size of $S \times S \times C_i \times C_p$. Based on the definition of the convolution procedure and follow the formulation in (Zhang et al., 2016), we can reshape this 4-dimension tensor as a matrix $\mathbf{U}$ with the size of $(S \cdot S \cdot C_i) \times C_p$, as shown in Figure 2. Notice that we do not apply any non-linear activation function in the linear orthogonal projection layer.

ii) One non-linear model layer with the weight matrix $W$: it has exactly same structure as a regular convolutional layer, which maps the $C_p$ projected feature maps into $C_m$ output feature maps. Differing from (Zhang et al., 2016), instead of only mapping the projected vector, the proposed model layer here takes all projected vectors within each $S \times S$ region into account and map all projected features within this region into the final output feature maps. We have found that this modification is critical in CNNs for better performance in image classification since it helps the network to extract local features. In our implementation, we use ReLU as the non-linear activation function. Since we apply supervised learning method to learn HOPE CNNs, we do not need to explicitly define the mixture model, and the mixture model can be learned automatically from training data.

Figure 2 shows the whole structure of one HOPE layer in CNNs. Since the projection layer is linear, we may collapse these two layers to derive a normal convolution layer in CNNs. However, as argued in (Zhang et al., 2016), the HOPE framework provides many advantages by explicitly define the feature extraction stage and data modeling stage.

Note that $C_p$ is always far less than $S \times S \times C_i$ in the above HOPE formulation, it implies that the orthogonal projection may help to remove irrelevant noises in this step.

In this paper, we only consider the supervised learning of CNNs under the HOPE framework. In this case, the model parameters in the model layer can be learned in the same way as in the convolutional CNNs. However, for the projection layers, we need to impose the orthogonal constraint, i.e. $\mathbf{U}\mathbf{U}^T = \mathbf{I}$, during the learning process. Following (Zhang et al., 2016), we cast this constraint as a penalty term in Eq. (4).

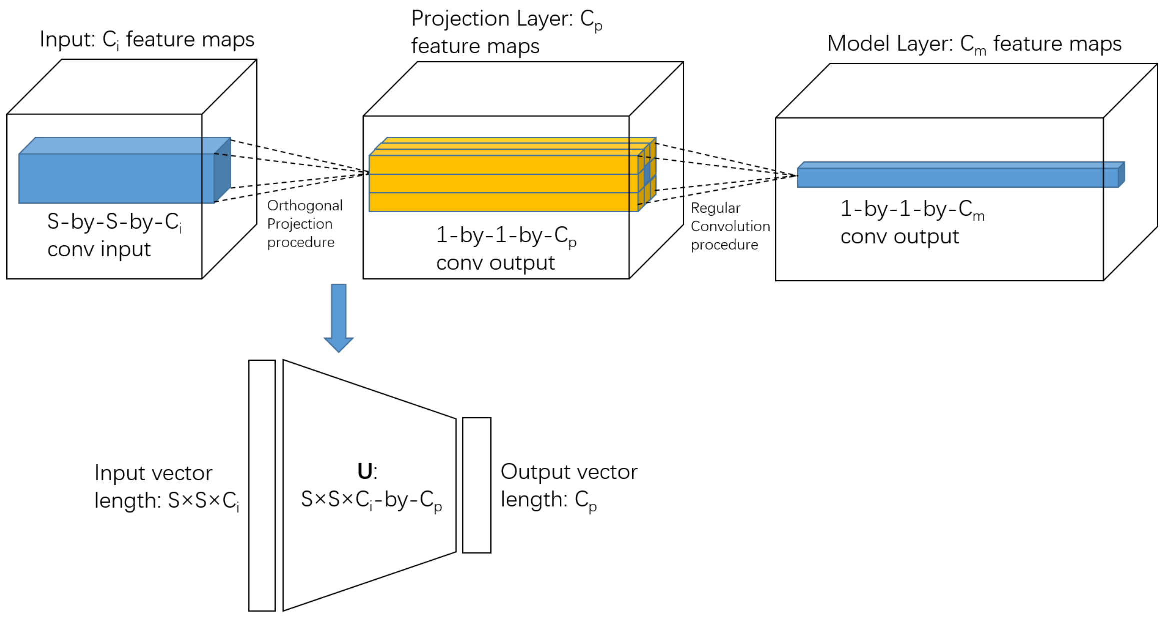

Input: C_i feature maps
S-by-S-by-C_i conv input
Orthogonal Projection procedure
Projection Layer: C_p feature maps
1-by-1-by-C_p conv output
Regular Convolution procedure
Model Layer: C_m feature maps
1-by-1-by-C_m conv output
Input vector length: S×S×C_i
U: S×S×C_i-by-C_p
Output vector length: C_p

Figure 2: A convolution layer in CNNs can be converted into a HOPE model. We do not need to explicitly define the data distribution of the model layer since it can be learned automatically via supervised learning criteria.

First of all, we need to derive the gradient of the penalty term $P(\mathbf{U})$ with respect to $\mathbf{U}$ as follows:

$$\frac{\partial P(\mathbf{U})}{\partial \mathbf{u}_i} = \sum_{j=1}^{M} (\frac{|\mathbf{u}_i \cdot \mathbf{u}_j|}{|\mathbf{u}_i| \cdot |\mathbf{u}_j|}) \cdot \Big( (\frac{\mathbf{u}_j}{\mathbf{u}_i \cdot \mathbf{u}_j}) - (\frac{\mathbf{u}_i}{\mathbf{u}_i \cdot \mathbf{u}_i}) \Big) \tag{5}$$

To facilitate the above computation in GPUs, we may equivalently represent the above gradient computation as a matrix form, i.e., essentially a multiplication of the two matrices $\mathbf{D}$ and $\mathbf{B}$ as follows:

$$\frac{\partial P(\mathbf{U})}{\partial \mathbf{U}} = (\mathbf{D} - \mathbf{B})\mathbf{U} \tag{6}$$

where $\mathbf{D}$ is an $M$-by-$M$ matrix of $d_{ij} = \frac{\text{sign}(\mathbf{u}_i \cdot \mathbf{u}_j)}{|\mathbf{u}_i| \cdot |\mathbf{u}_j|}$ $(1 < i, j < M)$ and $\mathbf{B}$ is another $M$-by-$M$ diagonal matrix of $b_{ii} = \frac{\sum_j g_{ij}}{\mathbf{u}_i \cdot \mathbf{u}_i}$ with $g_{ij} = \frac{|\mathbf{u}_i \cdot \mathbf{u}_j|}{|\mathbf{u}_i| \cdot |\mathbf{u}_j|}$ $(1 < i, j < M)$.

Secondly, we can combine the above $\frac{\partial P(\mathbf{U})}{\partial \mathbf{U}}$ with the gradient $\Delta\mathbf{U}$, which is calculated from the objective function:

$$\widetilde{\Delta\mathbf{U}} = \Delta\mathbf{U} + \beta \cdot \frac{\partial P(\mathbf{U})}{\partial \mathbf{U}} \tag{7}$$

where $\beta$ is a pre-defined parameter to balance the orthogonal penalty term. Finally, the projection matrix $\mathbf{U}$ can be updated as follows:

$$\mathbf{U}^{(n)} = \mathbf{U}^{(n-1)} - \gamma \cdot \widetilde{\Delta\mathbf{U}} \tag{8}$$

where $\gamma$ is the learning rate for the weight update. During the learning process, $\mathbf{U}$ is gradually de-correlated and eventually becomes an orthogonal matrix. In Figure 3, we display all correlation coefficients, i.e., $\frac{|u_i \cdot u_j|}{|u_i| \cdot |u_j|}$, of the HOPE orthogonal projection matrix $\mathbf{U}$ and the corresponding linear projection matrix, in which the orthogonal constraints are removed. The two images in Figure 3 clearly show that the HOPE projection matrix removes most of the correlation and thus becomes orthogonal. As shown in the results, the proposed HOPE layer in CNNs may remove the noise signals from the feature maps.

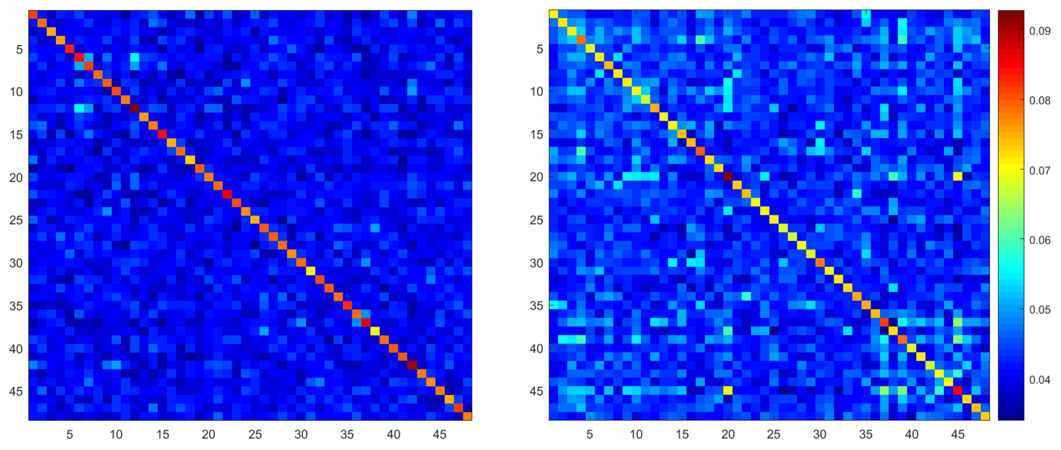


Figure 3: The correlation coefficients of the HOPE orthogonal projection matrix (left) and the corresponding linear projection matrix (right). We can see the HOPE projection matrix is more orthogonal compare with its linear counterpart. Here $S = 3$, $C_i = 64$ and $C_p = 48$.

## 3.2. The HOPE-Input Layer

The first way to apply the HOPE model to CNNs is to use the above HOPE layer to replace the first convolution layer right after the image pixel input. The HOPE formulation may help to de-correlate the raw image pixel inputs and filter out irrelevant noises in the first place. This is called as one HOPE-Input layer.

## 3.3. HOPE-Blocks

In many cases, simply applying one HOPE-Input layer is not enough to remove noise signals from features and achieve good performance. Therefore, we need to introduce more HOPE layers into the baseline CNN. In practice, one CNN can be divided into some building blocks, and each block may include several convolutional layers and end with one pooling layer. We can use these blocks as the basic units to introduce HOPE layers. Figure 4 shows an example of one HOPE-Block, and here we apply three HOPE layers to replace the corresponding convolutional layers (for the first convolutional layer, the projection layer is from the last block).

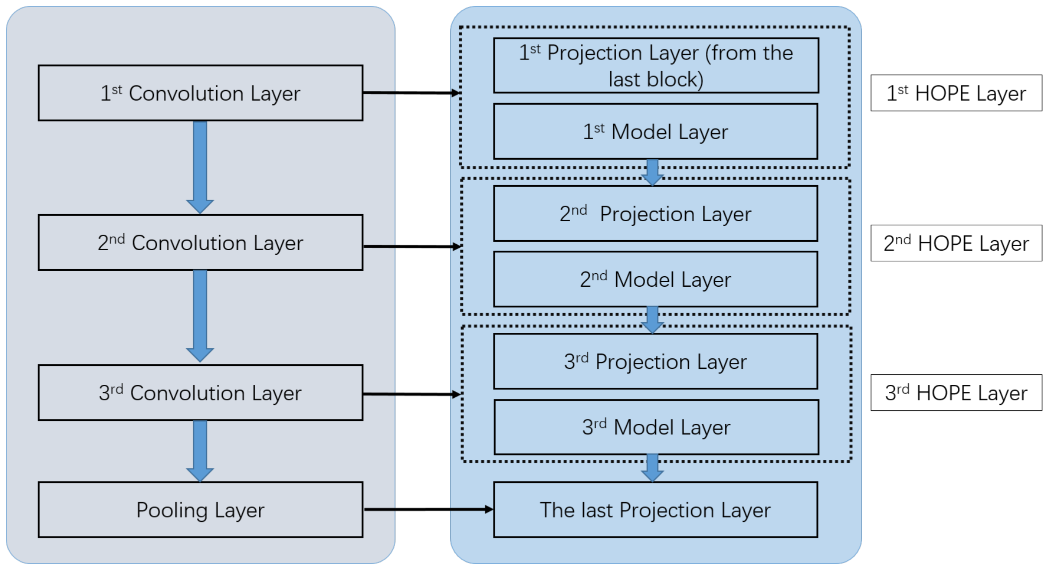


Figure 4: One HOPE-Block

For the pooling layer, we may need to consider a slightly different way to apply HOPE model. In CNNs, the pooling layers (Krizhevsky et al., 2012b) are traditionally considered as an important part for good performance. (Springenberg et al., 2014) has shown that the pooling layers result in the reduction of feature dimensionality, which help the CNNs to *view* much larger regions of the input feature maps, and generate more stable and invariant high level features.

Since the projection layer in HOPE models shares the similar objection with pooling layers, i.e., reducing the feature dimensionality, remove noise and increase the stability of the feature, we can just use one HOPE projection layer to replace one pooling layer, and view the next convolutional layer as the model layer. Comparing with the regular pooling layers, we believe that the HOPE projection layer may be advantageous in feature extraction since the learnable linear orthogonal projection may help to de-correlate the input feature maps more precisely and generate better features for the upper layers.

In practice, we can just introduce one HOPE-Block to replace the first building block in the baseline CNN (single-HOPE-Block), or apply multiple HOPE-Blocks (multi-HOPE-Blocks).

## 4. Experiments

In this paper, we use three widely used image classification databases, namely CIFAR-10. CIFAR-100 (Krizhevsky and Hinton, 2009) and ImageNet (Deng et al., 2009), to evaluate the performance of our proposed HOPE methods [1].

### 4.1. Databases

CIFAR-10 and CIFAR-100 are two popular databases that are widely used in computer vision. Both databases contain 50,000 32-by-32 RGB images for training and 10,000 images for validation. The

1. The codes of the proposed method can be downloaded via: `https://github.com/mowangphy/HOPE-CNN`.

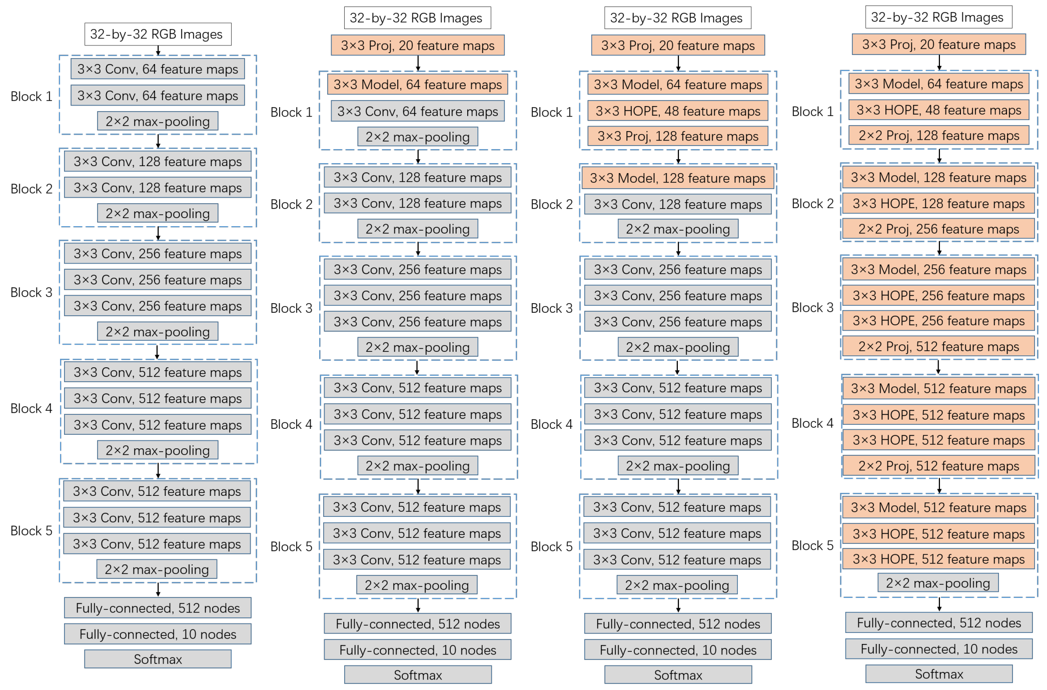

32-by-32 RGB Images
Block 1
3×3 Conv, 64 feature maps
3×3 Conv, 64 feature maps
2×2 max-pooling
Block 2
3×3 Conv, 128 feature maps
3×3 Conv, 128 feature maps
2×2 max-pooling
Block 3
3×3 Conv, 256 feature maps
3×3 Conv, 256 feature maps
3×3 Conv, 256 feature maps
2×2 max-pooling
Block 4
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
2×2 max-pooling
Block 5
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
2×2 max-pooling
Fully-connected, 512 nodes
Fully-connected, 10 nodes
Softmax
32-by-32 RGB Images
3×3 Proj, 20 feature maps
Block 1
3×3 Model, 64 feature maps
3×3 Conv, 64 feature maps
2×2 max-pooling
Block 2
3×3 Conv, 128 feature maps
3×3 Conv, 128 feature maps
2×2 max-pooling
Block 3
3×3 Conv, 256 feature maps
3×3 Conv, 256 feature maps
3×3 Conv, 256 feature maps
2×2 max-pooling
Block 4
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
2×2 max-pooling
Block 5
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
2×2 max-pooling
Fully-connected, 512 nodes
Fully-connected, 10 nodes
Softmax
32-by-32 RGB Images
3×3 Proj, 20 feature maps
Block 1
3×3 Model, 64 feature maps
3×3 HOPE, 48 feature maps
3×3 Proj, 128 feature maps
Block 2
3×3 Model, 128 feature maps
3×3 Conv, 128 feature maps
2×2 max-pooling
Block 3
3×3 Conv, 256 feature maps
3×3 Conv, 256 feature maps
3×3 Conv, 256 feature maps
2×2 max-pooling
Block 4
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
2×2 max-pooling
Block 5
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
2×2 max-pooling
Fully-connected, 512 nodes
Fully-connected, 10 nodes
Softmax
32-by-32 RGB Images
3×3 Proj, 20 feature maps
Block 1
3×3 Model, 64 feature maps
3×3 HOPE, 48 feature maps
2×2 Proj, 128 feature maps
Block 2
3×3 Model, 128 feature maps
3×3 HOPE, 128 feature maps
2×2 Proj, 256 feature maps
Block 3
3×3 Model, 256 feature maps
3×3 HOPE, 256 feature maps
3×3 HOPE, 256 feature maps
2×2 Proj, 512 feature maps
Block 4
3×3 Model, 512 feature maps
3×3 HOPE, 512 feature maps
3×3 HOPE, 512 feature maps
2×2 Proj, 512 feature maps
Block 5
3×3 Model, 512 feature maps
3×3 HOPE, 512 feature maps
3×3 HOPE, 512 feature maps
2×2 max-pooling
Fully-connected, 512 nodes
Fully-connected, 10 nodes
Softmax

Figure 5: From Left to Right: Baseline CNN, HOPE-Input CNN, single-HOPE-Block CNN, and multi-HOPE-Blocks CNN (Using CIFAR experiments as examples), where *Proj* denotes one HOPE projection layer, *Model* denotes one HOPE model layer, and *HOPE* denotes one whole HOPE layer (includes one projection layer and one model layer).

main difference between these two databases is that CIFAR-10 only divides all images into 10 coarse classes, but CIFAR-100 divides them into 100 fine classes. All CIFAR data should be zero-mean normalized, and we did not apply data whitening to pre-process the training and test data.

To expand the training sample size and reduce over-fitting, we also consider to use data augmentation techniques on the two databases. Specifically, In each mini-batch, we will randomly select half of the images to apply four kinds of augmentation methods respectively:

- **Translation:** The selected images will be randomly translate horizontally and vertically for at most 5 pixels.
- **Rotation:** The selected images will be randomly rotated by -5 to 5 degrees. The rotated images should be cropped to keep the original size.
- **Scaling:** We firstly randomly extract a patch from the input image (the patch size is pre-defined), and resize the patch to the original image size. This procedure equals to zoom-in.
- **Color space translation:** For each channel of one image, we define a translation matrix. Each element in the translation matrix is corresponding to one pixel in the corresponding color channel, and the value lays between 0.95 and 1.05. The image will element-wise multiply with the translation matrix for the color space translation.

Figure 6 displays some examples of the selected data augmentation methods.

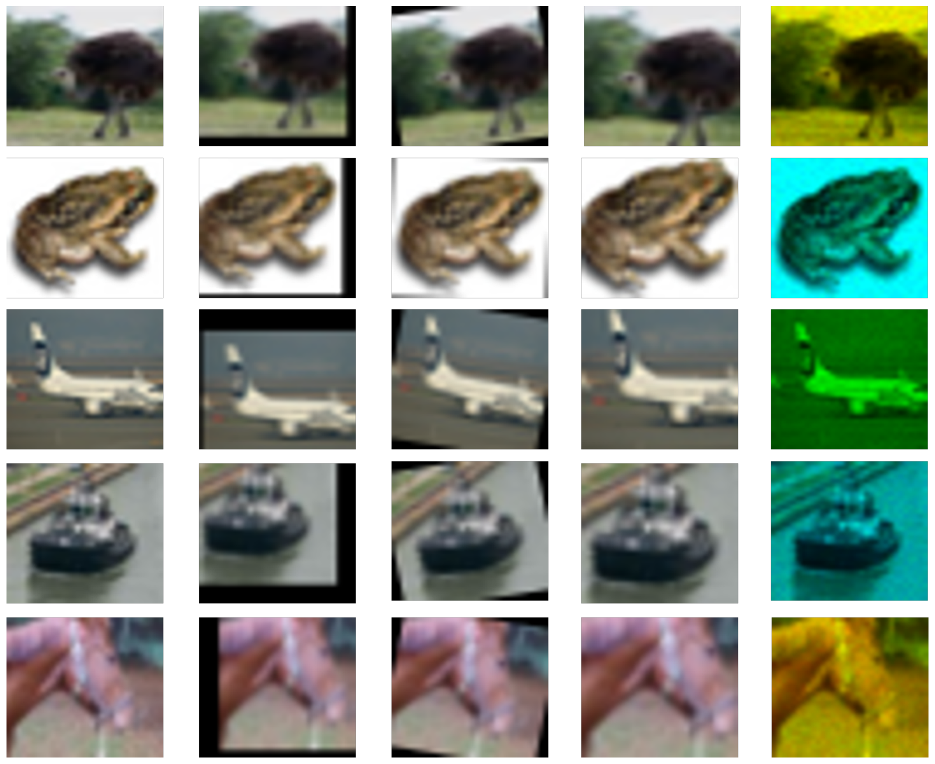


Figure 6: From Left to Right: original images, translated images, rotated images, scaled images and color space translated images.

Comparing with CIFAR-10 and CIFAR-100, ImageNet is an image database with much larger sample size (nearly 1.3 million training images and 50,000 validation images in the classification tasks), and all images are divided into 1000 classes. During the experiments, all images should be re-scaled to 224-by-224 to feed into CNNs.

## 4.2. CIFAR Experiments

In our CIFAR-10 and CIFAR-100 experiments, we consider several different CNN structures as specified in Figure 5 in detail. Firstly, we follow the CNN structure that is defined by Sergey Zagoruyko as our baseline CNNs.[2] Then we evaluate the HOPE-Input CNN, single-HOPE-Block and multi-HOPE-Block CNNs as discussed in section 3, and compare them with the baseline models. In Figure 5, we have provided the detailed description of the structure of 4 CNNs (baseline, HOPE-Input, single-HOPE-Block and multi-HOPE-Blocks) used in our experiments.

2. See https://github.com/szagoruyko/cifar.torch for more information. According to the website, without using data augmentation, the best performance on the CIFAR-10 test set is 8.7% in error rate. By using RGB color channel instead of YUV, our reproduced baseline performance is 8.30% in this paper.

In Figure 5, for the HOPE-Input layer, we use 20 feature maps in the projection layer. For the whole HOPE layer in the Block 1 (single-HOPE-Block and multi-HOPE-Blocks CNNs), the projection layer contains 48 feature maps. For the rest HOPE layers in the multi-HOPE-Blocks CNN, the projection layers should have the same number of feature maps as the corresponding model layers. In practice, the feature map number of the input layer and the first block should be tuned very carefully. The reason of this fact is that the shallow layers tend to have highly correlated data, which may more sensitive to the size of projection layers. It is easy to learn that too many feature maps or too few feature maps can both lead to worse performance. We did several tests on HOPE-Input and single-HOPE-Block CNNs to determine the best scale of the projection layers, and the results are shown in Table 1.

Table 1: The relationship of the size of projection layers and classification performance (using CIFAR-10 tests as examples).

| | Proj Layer 1 | Proj Layer 2 | Test Err |
|---|---|---|---|
| HOPE-Input | 10 | - | 8.01% |
| | 15 | - | 7.86% |
| | 20 | - | 7.77% |
| | 25 | - | 7.92% |
| Single-HOPE-Block | 20 | 24 | 8.12% |
| | 20 | 32 | 7.71% |
| | 20 | 48 | **7.06%** |
| | 20 | 64 | 7.57% |

To further investigate the performance of the HOPE CNNs (HOPE-Input, single-HOPE-Block and multi-HOPE-Blocks), we also consider the model configurations called as LIN CNNs (Lin-Input, single-Lin-Block and multi-Lin-Blocks), which uses the same model structure as the HOPE CNNs except that the orthogonal constraint in Eq. (4) is NOT applied in training.

In all CIFAR experiments, we use the mini-batch SGD with a batch size of 100 images to perform 350 epochs of network training. The initial learning rate is $0.065$, and the learning rate should be halved after every 30 epochs. We also use momentum of 0.9 and weight decay rate of 0.0005. In batch normalization (Ioffe and Szegedy, 2015), we set $\epsilon = 0.001$. For the HOPE layers, we use an initial $\beta$ that equals to $0.15$, and the $\beta$ should be divided by $1.75$ after every 25 epochs. As shown in Table 2, the choices hyper parameter $\beta$ and its decay rate may have some impacts on the final classification performance. Generally speaking, it is not very sensitive once we choose a good value for $\beta$. All weights in CNNs are initialized by using the method proposed by He et al (He et al., 2015). For the multi-HOPE-Blocks CNN experiments, we introduce 5 HOPE-Blocks (overall 6 blocks), since the last block only includes fully connected layers. In CNNs, fully-connected layers are applied at the end of the network. Our previous experiments show that adding HOPE layers on fully-connected layers cannot bring about positive influences on classification accuracy. One important reason is that the previous CNN layers (or HOPE CNN layers) already remove most of data correlation.

Table 2: The corresponding test error of different combination of initial $\beta$ and its decay rate (the $\beta$ should be decayed every 25 epochs).

| Initial $\beta$ | 0.015 | 0.05 | 0.1 | 0.15 | 0.15 | 0.15 | 0.2 |
|---|---|---|---|---|---|---|---|
| Decay Rate | 1.75 | 1.75 | 1.75 | 1.50 | 1.75 | 2.00 | 1.75 |
| Test Error | 8.43% | 8.31% | 8.06% | 7.46% | **7.06%** | 7.37% | 7.63% |

### 4.2.1. LEARNING SPEED

We firstly consider the computational efficiency of the proposed HOPE methods in learning CNNs. Our computing platform includes Intel Xeon E5-1650 CPU (6 cores), 64 GB memory and a Nvidia Geforce TITAN X GPU (12 GB memory). Our method is implemented with MatConvNet (Vedaldi and Lenc, 2015), which is a CUDA based CNN toolbox in Matlab. The learning speed of all CNNs are listed in Table 3.

From Table 3, we can see that using the more complicated HOPE layers in CNNs will slow down the computation of CNNs in GPUs, but the speeds are still reasonably good. Moreover, the learning speed of the HOPE methods is similar with the corresponding LIN methods, which implies that the computational overhead for the orthogonal projection constraint is negligible in training.

Table 3: The learning speed of different CNNs on CIFAR experiments.

| Methods | Learning Speed |
|---|---|
| Baseline | 220 images/s |
| LIN-Input | 206 images/s |
| HOPE-Input | 203 images/s |
| Single-LIN-Block | 200 images/s |
| Single-HOPE-Block | 195 images/s |
| Multi-LIN-Blocks | 149 images/s |
| Multi-HOPE-Blocks | 141 images/s |

### 4.2.2. PERFORMANCE ON CIFAR-10 AND CIFAR-100

We use the classification error rate on the test sets of the selected databases to evaluate the performance of all CNN models. Besides the 7 CNN configurations we mentioned above, we also include several well-known CNN models from the previous work to compare with our methods, including Tree-Pooling (Lee et al., 2015), Deep Networks for Global Optimization (DNGO) (Snoek et al., 2015), Fitnet4-LSUV (Mishkin and Matas, 2015), RCNN (Liang and Hu, 2015), ALL-CNN (Springenberg et al., 2014), Maxout Networks (Goodfellow et al., 2013) and Network in Network (Lin et al., 2013). All selected models have the comparable model size with our proposed CNNs.

From all results summarized in Table 4, we can see that the proposed HOPE-based CNNs models work well in both CIFAR-10 and CIFAR-100 databases. In the cases that the data augmentation methods are not applied, the single-HOPE-Block CNNs can achieve the best performances on both CIFAR-10 and CIFAR-100, which is the state-of-the-art performance without data augmentation,

Table 4: The classification error rates of all examined CNNs on the test set of CIFAR-10 and CIFAR-100. CIFAR-10+ and CIFAR-100+ denote the results using data augmentation.

| | CIFAR-10 | CIFAR-10+ | CIFAR-100 | CIFAR-100+ |
|---|---|---|---|---|
| Baseline | 8.30% | 6.96% | 30.71% | 29.38% |
| LIN-Input | 7.97% | 6.88% | 30.13% | 28.91% |
| HOPE-Input | 7.77% | 6.65% | 29.96% | 28.79% |
| Single-LIN-Block | 8.30% | 7.21% | 31.85% | 30.27% |
| Single-HOPE-Block | **7.06%** | **5.89%** | **29.47%** | **26.99%** |
| Multi-LIN-Blocks | 9.29% | 8.16% | 39.28% | 35.52% |
| Multi-HOPE-Blocks | 7.88% | 6.47% | 31.01% | 27.59% |
| Tree-Pooling (Lee et al., 2015) | 7.62% | 6.05% | 32.37% | - |
| DNGO (Snoek et al., 2015) | - | 6.37% | - | 27.40% |
| Fitnet4-LSUV (Mishkin and Matas, 2015) | - | 6.06% | - | 27.66% |
| RCNN (Liang and Hu, 2015) | 8.69% | 7.09% | 31.75% | - |
| ALL-CNN (Springenberg et al., 2014) | 9.08% | 7.25% | 33.71% | - |
| Maxout (Goodfellow et al., 2013) | 11.68% | 9.38% | 34.54% | - |
| Network in Network (Lin et al., 2013) | 10.41% | 8.81% | 35.68% | - |

and much better compare with the selected strong baselines. Moreover, we can see that HOPE-Input, single-HOPE-Block and multi-HOPE-Blocks CNNs consistently outperform the counterpart LIN models that do not use the orthogonal constraints. This implies that the orthogonality introduced by the HOPE methods is quite useful to improve the performance of CNNs in the image classification tasks.

Table 4 also shows that after data augmentation the proposed HOPE method can also achieve state-of-the-art performance on both CIFAR-10 and CIFAR-100 databases.

In the supervised learning of HOPE CNNs, the projection layers in the shallow layers (from input layer to the first block) are most important since those layers can remove most of residual noises and data correlation from the raw data. Therefore, adding more HOPE layers (multi-HOPE-Blocks) may not bring about obvious improvements. Moreover, introducing HOPE layers in the top may introduce a lot of extra model parameters due to the large number of feature maps used in those layers. This may further lead to over-fitting and decrease the performance on the test sets. This can explain why single-HOPE-Block CNNs show better performance compare with multi-HOPE-Blocks CNNs.

## 4.3. ImageNet Experiments

In our ImageNet experiments, we directly apply VGG-16 (Simonyan and Zisserman, 2014) as the baseline network [3] Then we evaluate the HOPE-Input CNN and single-HOPE-Block CNN as discussed in section 3, and compare them with the original VGG-16 model. Similar to CIFAR experiments, we also take the corresponding LIN CNNs into account to further demonstrate the effectiveness of HOPE.

3. See http://http://www.vlfeat.org/matconvnet/pretrained/ for more information. According to the website, the top-5 error of VGG-16 on ImageNet validation set is 9.5% on MatConvNet platform.

Table 5: The learning speed and top-5 classification error (validation set) of different CNNs on ImageNet experiments.

| Methods | Learning Speed | Top-5 Error |
|---|---|---|
| VGG-16 | 58 images/s | 9.5% |
| LIN-Input | 56 images/s | 9.5% |
| HOPE-Input | 55 images/s | 9.3% |
| Single-LIN-Block | 53 images/s | 9.6% |
| Single-HOPE-Block | 52 images/s | **9.0%** |

In all ImageNet experiments, we use 128 mini-batch size to train the CNNs for 50 epochs. The initial learning rate and $\beta$ are 0.01 and 0.02 respectively, and both of them should be halved after every 5 epochs. The momentum and weight decay rate are 0.9 and 0.0005 respectively. In batch normalization (Ioffe and Szegedy, 2015), we set $\epsilon = 0.001$. All weights in CNNs will be initialized by using the method proposed by He et al (He et al., 2015).

### 4.3.1. LEARNING SPEED

Since ImageNet database has very large sample size, and the CNNs are much deeper comparing with their counterparts in CIFAR experiments, we use 4 Nvidia Geforce TITAN X GPUs to do network training, and the learning speeds are listed in Table 5. In our platform, one single training round of ImageNet takes nearly 2 weeks. Therefore, we can not try all possible combinations of HOPE model as CIFAR experiments, but just the best configuration obtained from CIFAR (Single-HOPE-Block).

### 4.3.2. PERFORMANCE ON IMAGENET

From Table 5, we can also see that HOPE models also work well on the ImageNet database. And Single-HOPE-Block CNN shows the best performance among all selected models. HOPE-Input CNN can also yield improvement over the VGG-16 baseline model.

## 5. Conclusion and Future Work

In this paper, we have proposed several methods to apply the recent HOPE model to CNNs for image classification. Experimental results on the CIFAR-10, CIFAR-100 and ImageNet databases have shown that our proposed HOPE methods work well with CNNs, and can yield the state-of-the-art classification performance in these databases. This study has confirmed that the orthogonal constraints imposed by the HOPE models can significantly improve the performance of CNNs in these image classification tasks.

Due to the limitation of time and computing resources, we haven't applied the propose HOPE CNN methods to the more recent deep residual nets (He et al., 2016). However, the application of HOPE CNN is straightforward since the HOPE model can also be inserted in the shortcut connections in deep residual nets. We will continue to investigate along this line as a possible direction to extend this work in the future.

## References

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1319–1327, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012a.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012b.

Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361, 1995.

Yann LeCun, Boser B., JS Denker, D Henderson, Richard E. Howard, W. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems (NIPS)*. Citeseer, 1990.

Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. *arXiv preprint arXiv:1509.08985*, 2015.

Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375, 2015.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

Hengyue Pan and Hui Jiang. A fast method for saliency detection by back-propagating a convolutional neural network and clamping its partial outputs. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 3585–3592. IEEE, 2017.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*. 2015.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mostofa Ali, Ryan P Adams, et al. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, 2015.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *arXiv preprint arXiv:1409.4842*. 2014.

Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

Shiliang Zhang, Hui Jiang, and Lirong Dai. Hybrid orthogonal projection and estimation (HOPE): A new framework to learn neural networks. *Journal of Machine Learning Research*, 17(37):1–33, 2016. URL `http://jmlr.org/papers/v17/15-335.html`.

# Fixed-Size Ordinally Forgetting Encoding Based Word Sense Disambiguation

**Xi Zhu, Mingbin Xu, Hui Jiang**
Department of Electrical Engineering and Computer Science
Lassonde School of Engineering, York University
4700 Keele Street, Toronto, Ontario, Canada
{xzhu, xmb, hj}@eecs.yorku.ca

## Abstract

In this paper, we present our method of using fixed-size ordinally forgetting encoding (FOFE) to solve the word sense disambiguation (WSD) problem. FOFE enables us to encode variable-length sequence of words into a theoretically unique fixed-size representation that can be fed into a feed forward neural network (FFNN), while keeping the positional information between words. In our method, a FOFE-based FFNN is used to train a pseudo language model over unlabelled corpus, then the pre-trained language model is capable of abstracting the surrounding context of polyseme instances in labelled corpus into context embeddings. Next, we take advantage of these context embeddings towards WSD classification. We conducted experiments on several WSD data sets, which demonstrates that our proposed method can achieve comparable performance to that of the state-of-the-art approach at the expense of much lower computational cost.

## 1 Introduction

Words with multiple senses commonly exist in many languages. For example, the word *bank* can either mean a “financial establishment” or “the land alongside or sloping down to a river or lake”, based on different contexts. Such a word is called a “polyseme”. The task to identify the meaning of a polyseme in its surrounding context is called word sense disambiguation (WSD). Word sense disambiguation is a long-standing problem in natural language processing (NLP), and has broad applications in other NLP problems such as machine translation (Taghipour and Ng, 2015). Lexical sample task and all-word task are the two main branches of WSD problem. The former focuses on only a pre-selected set of polysemes whereas the later intends to disambiguate every polyseme in the entire text. Numerous works have been devoted in WSD task, including supervised, unsupervised, semi-supervised and knowledge based learning (Iacobacci et al., 2016). Our work focuses on using supervised learning to solve all-word WSD problem.

Most supervised approaches focus on extracting features from words in the context. Early approaches mostly depend on hand-crafted features. For example, IMS by Zhong and Ng (2010) uses POS tags, surrounding words and collections of local words as features. These approaches are later improved by combining with word embedding features (Taghipour and Ng, 2015), which better represents the words’ semantic information in a real-value space. However, these methods neglect the valuable positional information between the words in the sequence (Kågebäck and Salomonsson, 2016). The bi-directional Long-Short-Term-Memory (LSTM) approach by Kågebäck and Salomonsson (2016) provides one way to leverage the order of words. Recently, Yuan et al. (2016) improved the performance by pre-training a LSTM language model with a large unlabelled corpus, and using this model to generate sense vectors for further WSD predictions. However, LSTM significantly increases the computational complexity during the training process.

The development of the so called “fixed-size ordinally forgetting encoding” (FOFE) has enabled us to consider more efficient method. As firstly proposed in (Zhang et al., 2015), FOFE provides a way to encode the entire sequence of words of variable length into an almost unique fixed-size representation, while also retain the positional information for words in the sequence. FOFE has been applied to several NLP problems in the past, such as language model (Zhang et al., 2015), named entity recognition (Xu et al., 2017), and word embedding (Sanu et al., 2017). The promising results demonstrated by the FOFE approach in these areas inspired us to apply FOFE in solving the WSD problem. In this paper, we will first describe how FOFE is used to encode sequence of any length into a fixed-size representation. Next, we elaborate on how a pseudo language model is trained with the FOFE encoding from unlabelled data for the purpose of context abstraction, and how a classifier for each polyseme is built from context abstractions of its labelled training data. Lastly, we provide the experiment results of our method on several WSD data sets to justify the equivalent performance as the state-of-the-art approach.

## 2 Fixed-size Ordinally Forgetting Encoding

The fact that human languages consist of variable-length sequence of words requires NLP models to be able to consume variable-length data. RNN/LSTM addresses this issue by recurrent connections, but such recurrence consequently increases the computational complexity. On the contrary, feed forward neural network (FFNN) has been widely adopted in many artificial intelligence problems due to its powerful modelling ability and fast computation, but is also limited by its requirement of fixed-size input. FOFE aims at encoding variable-length sequence of words into a fixed-size representation, which subsequently can be fed into an FFNN.

Given vocabulary $V$ of size $|V|$, each word can be represented by a one-hot vector. FOFE can encode a sequence of words of any length using linear combination, with a forget factor to reflect the positional information. For a sequence of words $S = w_1, w_2, .., w_T$ from V, let $\mathbf{e}_i$ denote the one-hot representation for the $i^{th}$ word, then the FOFE code of S can be recursively obtained using following equation (set $\mathbf{z}_0 = \mathbf{0}$):

$$\mathbf{z}_t = \alpha \cdot \mathbf{z}_{t-1} + \mathbf{e}_t \quad (1 \leq t \leq T)$$

where $\alpha$ is a constant between 0 and 1, called forgetting factor. For example, assuming A, B, C are three words with one-hot vectors $[1, 0, 0]$, $[0, 1, 0]$, $[0, 0, 1]$ respectively. The FOFE encoding from left to right for ABC is [$\alpha^2$,$\alpha$,1] and for ABCBC is [$\alpha$,$\alpha + \alpha$,$1 + \alpha$]. It becomes evident that the FOFE code is in fixed size, which is equal to the size of the one-hot vector, regardless of the length of the sequence $S$.

The FOFE encoding has the property that the original sequence can be unequivocally recovered from the FOFE encoding. According to Zhang et al. (2015), the uniqueness for the FOFE encoding of a sequence is confirmed by the following two theorems:

**Theorem 1** *If the forgetting factor $\alpha$ satisfies $0 \leq \alpha < 0.5$, FOFE is unique for any sequence of finite length $T$ and any countable vocabulary $V$.*

**Theorem 2** *If the forgetting factor $\alpha$ satisfies $0.5 \leq \alpha \leq 1$, FOFE is almost unique for any finite value of $T$ and vocabulary $V$, except only a finite set of countable choices of $\alpha$.*

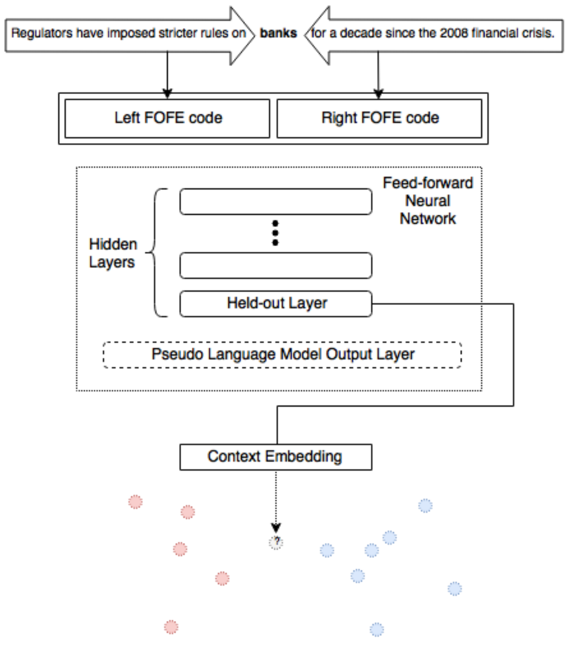
Even for situations described by Theorem 2 where uniqueness is not strictly guaranteed, the probability for collision is extremely low in practice. Therefore, FOFE can be safely considered as an encoding mechanism that converts variable-length sequence into a fixed-size representation theoretically without any loss of information.

## 3 Methodology

The linguistic distribution hypothesis states that words that occur in close contexts should have similar meaning (Harris, 1954). It implies that the particular sense of a polyseme is highly related to its surrounding context. Moreover, human decides the sense of a polyseme by firstly understanding its occurring context. Likewise, our proposed model has two stages, as shown in Figure 1: training a FOFE-based pseudo language model that abstracts context as embeddings, and performing WSD classification over context embeddings.

### 3.1 FOFE-based Pseudo Language Model

A language model is trained with large unlabelled corpus by Yuan et al. (2016) in order to overcome the shortage of WSD training data. A language model represents the probability distribution of a given sequence of words, and it is commonly used in predicting the subsequent word given preceding sequence. Zhang et al. (2015) proposed a FOFE-based neural network language model by feeding FOFE code of preceding sequence into FFNN. WSD is different from language model in terms of that the sense prediction of a target word depends on its surrounding sequence rather than only preceding sequence. Hence, we build a pseudo language model that uses both preceding and suc-



Regulators have imposed stricter rules on
banks
for a decade since the 2008 financial crisis.
Left FOFE code
Right FOFE code
Feed-forward Neural Network
Hidden Layers
Held-out Layer
Pseudo Language Model Output Layer
Context Embedding
?

Figure 1: Context abstraction through FOFE-based pseudo language model and WSD classification over context embeddings

ceeding sequence to accommodate the purpose of WSD tasks.

The preceding and succeeding sequences are separately converted into FOFE codes. As shown in Figure 1, the words preceding the target word are encoded from left to right as the left FOFE code, and the words succeeding the target word are encoded from right to left as the right FOFE code. The forgetting factor that underlies the encoding direction reflects the reducing relevance of a word due to the increasing distance relative to the target word. Furthermore, the FOFE is scalable to higher orders by merging tailing partial FOFE codes. For example, a second order FOFE of sequence $S = w_1, w_2, .., w_T$ can be obtained as $[z_{T-1}, z_T]$. Lastly, the left and right FOFE codes are concatenated into one single fixed-size vector, which can be fed into an FFNN as an input.

FFNN is constructed in fully-connected layers. Each layer receives values from previous layer as input, and produces values through a function over weighted input values as its output. FFNN increasingly abstracts the features of the data through the layers. As the pseudo language model is trained to predict the target word, the output layer is irrelevant to WSD task and hence can be discarded. However, the remaining layers still have learned the ability to generalize features from word to context during the training process. The values of the held-out layer (the second last layer) are extracted as context embedding, which provides a nice numerical abstraction of the surrounding context of a target word.

### 3.2 WSD Classification

Words with the same sense mostly appear in similar contexts, hence the context embeddings of their contexts are supposed to be close in the embedding space. As the FOFE-based pseudo language model is capable of abstracting surrounding context for any target word as context embeddings, applying the language model on instances in annotated corpus produces context embeddings for senses.

A classifier can be built for each polyseme over the context embeddings of all its occurring contexts in the training corpus. When predict the sense of a polyseme, we similarly extract the context embedding from the context surrounding the predicting polyseme, and send it to the polyseme's classifier to decide the sense. If a classifier cannot be built for the predicting polyseme due to the lack of training instance, the first sense from the dictionary is used instead.

For example, word $w$ has two senses $s_i$ for $i = 1, 2$ occurring in the training corpus, and each sense has $n_i$ instances. The pseudo language model converts all the instances into context embeddings $\mathbf{c}^i_j$ for $j = 1, \ldots, n_i$, and these embeddings are used as training data to build a classifier for $w$. The classifier can then be used to predict the sense of an instance of $w$ by taking the predicting context embedding $\mathbf{c}'$.

The context embeddings should fit most traditional classifiers, and the choice of classifier is empirical. Yuan et al. (2016) takes the average over context embeddings to construct sense embeddings $\mathbf{s}_i = \frac{\sum_{j=i} \mathbf{c}^i_j}{n_i}$, and selects the sense whose sense embedding is closest to the predicting context embedding measured by cosine similarity. In practice, we found k-nearest neighbor (kNN) algorithm, which predicts the sense to be the majority of k nearest neighbors, produces better performance on the context embeddings produced by our FOFE-based pseudo language model.

## 4 Experiment

To evaluate the performance of our proposed model, we implemented our model using Tensorflow (Abadi et al., 2015) and conducted experi-

| Model | Corpus Size | Vocab. | Training Time | Senseval2 | SemEval13 |
|---|---|---|---|---|---|
| IMS * | - | - | - | 0.625 | - |
| IMS + Word2vec * | - | - | - | 0.634 | - |
| LSTM (Yuan et al., 2016) | 100B | 1M | - | 0.736 | 0.670 |
| LSTM (Le et al., 2017) | 2B | 1M | 4.5 months | 0.700 | 0.666 |
| LSTM (our training) † | 0.8B | 100K | 2 weeks | 0.661 | 0.633 |
| **FOFE (this work)** | 0.8B | 100K | 3 days | 0.693 | 0.650 |

Table 1: The corpus size, vocabulary size and training time when pre-training the language models, and F1 scores of different models on multiple WSD tasks using SemCor as training data. The asterisk ($*$) indicates the results are from (Iacobacci et al., 2016). Our training (†) uses code published by (Le et al., 2017) with Google1B (Chelba et al., 2014) as training data.

ments on standard SemEval data that are labelled by senses from WordNet 3.0 (Fellbaum, 1998). We built the classifier using SemCor (Miller et al., 1993) as training corpus, and evaluated on Senseval2 (Edmonds and Cotton, 2001), and SemEval-2013 Task 12 (Navigli et al., 2013).

### 4.1 Experiment settings

When training our FOFE-based pseudo language model, we use Google1B (Chelba et al., 2014) corpus as the training data, which consists of approximately 0.8 billion words. The 100,000 most frequent words in the corpus are chosen as the vocabulary. The dimension of word embedding is chosen to be 512. During the experiment, the best results are produced by the 3rd order pseudo language model. The concatenation of the left and right 3rd order FOFE codes leads to a dimension of 512 * 3 * 2 = 3072 for the FFNN's input layer. Then we append three hidden layers of dimension 4096. Additionally, we choose a constant forgetting factor $\alpha = 0.7$ for the FOFE encoding and $k = 8$ for our k-nearest neighbor classifier.

### 4.2 Results

Table 1 presents the micro F1 scores from different models. Note that we use a corpus with 0.8 billion words and vocabulary of 100,000 words when training the language model, comparing with Yuan et al. (2016) using 100 billion words and vocabulary of 1,000,000 words. The context abstraction using the language model is the most crucial step. The sizes of the training corpus and vocabulary significantly affect the performance of this process, and consequently the final WSD results. However, Yuan et al. (2016) did not publish the 100 billion words corpus used for training their LSTM language model.

Recently, Le et al. (2017) reimplemented the LSTM-based WSD classifier. The authors trained the language model with a smaller corpus Gigaword (Graff and Cieri, 2003) of 2 billion words and vocabulary of 1 million words, and reported the performance. Their published code also enabled us to train an LSTM model with the same data used in training our FOFE model, and compare the performances at the equivalent conditions.

Additionally, the bottleneck of the LSTM approach is the training speed. The training process of the LSTM model by Le et al. (2017) took approximately 4.5 months even after applying optimization of trimming sentences, while the training process of our FOFE-based model took around 3 days to produce the claimed results.

## 5 Conclusion

In this paper, we propose a new method for word sense disambiguation problem, which adopts the fixed-size ordinally forgetting encoding (FOFE) to convert variable-length context into almost unique fixed-size representation. A feed forward neural network pseudo language model is trained with FOFE codes of large unlabelled corpus, and used for abstracting the context embeddings of annotated instance to build a k-nearest neighbor classifier for every polyseme. Compared to the high computational cost induced by LSTM model, the fixed-size encoding by FOFE enables the usage of a simple feed forward neural network, which is not only much more efficient but also equivalently promising in numerical performance.

## References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2635–2639.

Philip Edmonds and Scott Cotton. 2001. Senseval-2: Overview. In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, SENSEVAL '01, pages 1–5, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

David Graff and Christopher Cieri. 2003. English gigaword 2003. *Linguistic Data Consortium, Philadeplhia*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907, Berlin, Germany. Association for Computational Linguistics.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional LSTM. *CoRR*, abs/1606.03568.

Minh Le, Marten Postma, and Jacopo Urbani. 2017. Word sense disambiguation with lstm: Do we really need 100 billion words? *arXiv preprint arXiv:1712.03376*.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, pages 303–308, Stroudsburg, PA, USA. Association for Computational Linguistics.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. Semeval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 222–231.

Joseph Sanu, Mingbin Xu, Hui Jiang, and Quan Liu. 2017. Word embeddings based on fixed-size ordinally forgetting encoding. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 310–315.

Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 314–323.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1237–1247. Association for Computational Linguistics.

Dayu Yuan, Ryan Doherty, Julian Richardson, Colin Evans, and Eric Altendorf. 2016. Word sense disambiguation with neural language models. *CoRR*, abs/1603.07012.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 495–500.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL*.