The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Inference on Syntactic and Semantic Structures for Machine Comprehension

Chenrui Li,[1] Yuanbin Wu,[1, 2] Man Lan[1, 2]
[1]School of Computer Science and Software Engineering, East China Normal University
[2]Shanghai Key Laboratory of Multidimensional Information Processing
{lcrr2011}@163.com, {ybwu, mlan}@cs.ecnu.edu.cn

Abstract

Hidden variable models are important tools for solving open domain machine comprehension tasks and have achieved remarkable accuracy in many question answering benchmark datasets. Existing models impose strong independence assumptions on hidden variables, which leaves the interaction among them unexplored. Here we introduce linguistic structures to help capturing global evidence in hidden variable modeling. In the proposed algorithms, question-answer pairs are scored based on structured inference results on parse trees and semantic frames, which aims to assign hidden variables in a global optimal way. Experiments on the MCTest dataset demonstrate that the proposed models are highly competitive with state-of-the-art machine comprehension systems.

Introduction

Being of great practical use, open domain machine comprehension attracts a long lasting research interest. Both world knowledge and linguistic analysis are important for the task. Modern machine comprehension algorithms (especially, deep learning based algorithms) show that how to represent and organize external world knowledge (e.g., distributed representation) is key to gain high performances. On the other side, to establish a deeper understanding of answer inference process, exploring linguistic structures remains critical and fundamental. In this work, we focus on answer reasoning with limited world knowledge following the setting of MCTest (Richardson, Burges, and Renshaw 2013). Given a fictional story written for elementary school students, MCTest is designed to test systems' ability of natural language inference based on the given texts.

Hidden variable models are powerful tools for building robust and interpretable machine comprehension systems. Intuitively, it is always helpful that a system could identify some critical hidden information in text, such as the most relevant story sentences to the question, or the most plausible alignments between words in the question and words in the story. In fact, although simple lexical matching methods (Smith et al. 2015) are essential and complex deep learning models (Yin, Ebert, and Schütze 2016; Wang et al. 2016a) are competitive, most state-of-the-art systems are based on hidden variables (Wang et al. 2015; Narasimhan and Barzilay 2015; Sachan et al. 2015; Sachan and Xing 2016).

Method	Hidden Variable	Inference
Direct lexical matching	no	no
(Sachan et al. 2015) (Wang et al. 2015)	independent	direct
This work	dependent	tractable

Table 1: Comparisons with related work.

In existing formulations, hidden variables are assumed to be independent. For example, Sachan et al. (2015) use hidden variables to describe word alignments between questions and stories. Suppose that we have two question words q_u and q_v, and both of them have multiple possible alignments to the given story. In (Sachan et al. 2015), the alignments of q_u, q_v are assumed to be independent, which might lose some important information. For instance, if q_u is the subject of q_v in the question sentence, their correct alignments would also keep certain "agent-action" relation in the story. Hence, assignments of the two hidden variables should be correlated. The same problem also appears in (Wang et al. 2015) which uses hidden variables to describe the most relevant story sentences to a question.

In this work, we try to enlarge the expressiveness of hidden variables by capturing dependencies among them. Two new models are proposed based on structured inference on syntactic trees and semantic frames. Specifically, in the tree model, we organize hidden variables according to the dependency tree structure. For each node (word) in the dependency tree of a question, we associate it with a variable to indicate its aligned story sentence. The tree edges will describe the dependencies among hidden variables. In the frame model, we group hidden variables according to semantic frames. Each frame element is associated with a variable. Assignments of the variables are thus guided by the semantic frame structures. In both models, question-answer pairs are scored according to structured inference results: the structured inference first finds the optimal assignment of hidden variables, then the final answer is scored based on the assignment.

The inference algorithms of the two models are exact and fast. Combined with the lexical matching baseline and the hidden variable perceptron algorithm, our models are able to achieve state-of-the-art performances on MCTest dataset (72.7%). To summarize, our main contributions are

- Introducing linguistic structures to describe hidden variable dependencies.
- Applying effective inference algorithms for decoding the proposed models.
- Achieving state-of-the-art performances on the MCTest task (outperforming previous hidden variable and deep learning models).

Related Work

There are many settings of machine comprehension tasks which vary in corpus size (10^2 to 10^5 words), domain (open or close), question source (synthetic or crowdsourced) and answer form (cloze or multiple-choice). Example datasets include CNN/Daily Mail (Hermann et al. 2015), CBT (Hill et al. 2015), Algebra (Kushman et al. 2014), Science exam (Clark and Etzioni 2016), and SQuAD (Rajpurkar et al. 2016). They evaluate different aspects of text understanding systems.

In this work, we choose the MCTest (Richardson, Burges, and Renshaw 2013) setting. Comparing with other machine comprehension settings, the MCTest task evaluates systems when external knowledge is absent. Taking the SQuAD for comparison, questions in MCTest dataset are more likely to refer to multiple story sentences (54.3% of the questions, SQuAD is 13.6%). Hence, reasoning across sentence boundaries is more important in MCTest. Another difference is that for SQuAD dataset, lexicalized features are crucial (e.g., lexicalized dependency path features in the logistic regression baseline (Rajpurkar et al. 2016), embeddings of question words and story words (Lee et al. 2016; Wang et al. 2016b; 2017)). We could think that the tighter dependency on lexicalized features, the more importance of the domain related knowledge.

Early studies on MCTest task show that simple lexical level similarity comparisons can achieve reasonable performances. The baseline method in (Richardson, Burges, and Renshaw 2013) uses a sliding window to count the number of overlapping words among the question, the answer and the story, and then select the answer with the maximum count. Smith et al. (2015) show that running the sliding window baseline with different window sizes can further improve the result. It is clear that the simple matching method could not handle complex questions, thus more expressive hidden variable models are introduced. Narasimhan and Barzilay (2015) and Wang et al. (2015) use hidden variables to indicate candidate answer sentences. Wang et al. (2015) incorporate syntax, frames and semantic features, and Narasimhan and Barzilay (2015) focus on discourse relation features. Sachan et al. (2015) and Sachan and Xing (2016) propose hidden variables to represent alignments between questions and stories. They extract global features for alignments based on rhetorical structure, coreference links and abstract meaning representations. Our models depart from existing work by introducing structures in the inference process rather than only using them as features.

Another line of work applies deep learning methods. In general, due to the limited corpus size, existing deep learning models perform badly on MCTest (Yin, Ebert, and Schütze 2016). Two exceptions are (Trischler et al. 2016; Wang et al. 2016a). Trischler et al. (2016) combine multiple similarity measurements based on word and sentence representations. Wang et al. (2016a) introduce external RTE and answer selection datasets for learning the neural networks. It is worth noting that choosing proper initial values of parameters is crucial for above two models. In fact, according to the experiments there, without tuning initial values, the performances could be far below state-of-the-art models.

Among many studies on text semantic matching, the answer selection task (Yih et al. 2013) is closely related to machine comprehension. Researchers have applied tree alignment models (synchronized grammar) (Wang, Smith, and Mitamura 2007) and tree edit models (Wang and Manning 2010; Heilman and Smith 2010; Yao et al. 2013). Different from existing answer selection models, we won't align two parse trees directly, but use tree structures for organizing hidden variables and conducting structured inference.

The Problem

Given a story in raw unstructured text, the task of machine comprehension is to answer questions according to the content of the story. In this work, we assume that a set of candidate answers are provided, and the task is reduced to select a correct answer from the given set. Formally, let $T = \mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{|T|}$ represent a story, where $\mathbf{t}_i$ is the ith sentence in T. Define $\mathbf{q} = q_1, q_2, \ldots, q_{|\mathbf{q}|}$ to be a question, where q_i is a question word, and $\mathbf{a} = a_1, a_2, \ldots, a_{|\mathbf{a}|}$ be an answer, where a_i is a answer word. Denote $A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{|A|}\}$ to be a candidate answer set. For a story and a question, we assign a score for each answer in the candidate set to measure its probability of being the correct answer. The answer with the highest score is the model's output. To be concrete, let $x = (T, \mathbf{q})$ and $y = \mathbf{a} \in A$, the model assigns a score $h(x, y) \in \mathbf{R}$, and outputs $\hat{y} = \arg\max_{y \in A} h(x, y)$.

The Approach

In hidden variable models, the scoring function $h(x, y)$ relies on some unobserved variables z. Two types of z are often used: *hidden sentence alignment* and *hidden word alignment*. In hidden sentence alignment models, a variable z represents a story sentence which may support the question-answer pair (Wang et al. 2015; Narasimhan and Barzilay 2015). Being a sentence level model, various sentence similarity measurements can be applied in the score function h. The hidden word alignment models, on the other side, are more fine-grained: for each question (or answer) word, a variable z indicates which story word will align with it (Sachan et al. 2015; Sachan and Xing 2016; Yih et al. 2013).

In this work, we combine the two models: we aim to find support sentences for an input question-answer pair, and these sentences are obtained by locating support sentences of individual question (or answer) words. Formally, for a word q_u in $\mathbf{q}$ (or a_u in $\mathbf{a}$), we define a hidden variable z_u

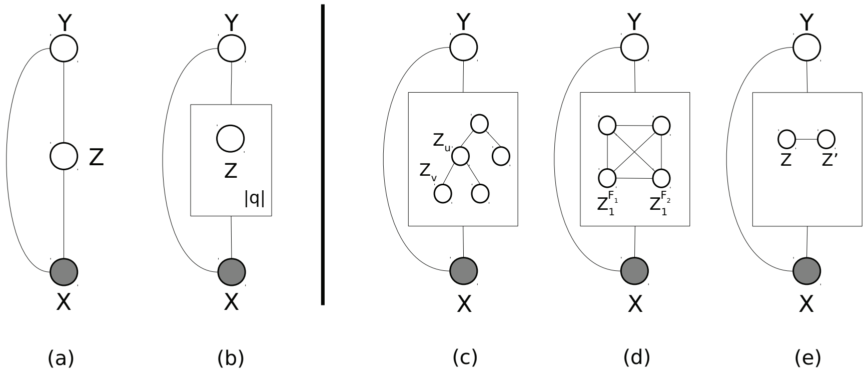

Figure 1: Graphical model representations of hidden variable models. (a) is the model with one hidden sentence alignment variable, (b) is the model with independent hidden word alignment variables, (c) is the proposed tree model, (d) is the frame model and (e) is the model with two hidden sentence alignment variables. In (a) and (e), the hidden variables are allocated to the question string (i.e., the support sentences of the question string). In (b), (c) and (d), z variables represent support words (in (b)) or support sentences (in (c) and (d)) for each question word.

which indicates a candidate support sentence of the word. Instead of treating hidden variables z_u independently, we are going to capture dependencies among them using syntactic and semantic structures.

Tree Model

A natural way to capture word dependencies is by their syntactic relations. If two words have close syntactic relation in the question (or answer), the support sentences of them might also share some information to keep the relation. Further benefit would be obtained if we can assign z_u by considering all other words and their hidden variable assignments. However, to find the global optimum of all possible word pairs, we need to keep the consistency among all local assignments, which is NP-hard (i.e., MAP inference of general pairwise Markov random field (MRF)).

In order to develop a tractable model, we propose to use dependency tree structures to organize hidden variables (Figure 1.(c)). First, dependency trees (especially projective trees) have encoded global syntactic relations properly in the head-modifier structure: the head is a proxy of its modifiers on their relations with other words. Second, the simple tree structure helps to reduce model complexity and make the exact inference tractable [1]. We will consider both dependency trees of questions and answers. In the following, we first take the question as example to illustrate the model.

For question $\mathbf{q}$, denote $D_{\mathbf{q}}$ to be its dependency tree, and $e = (q_u, q_v) \in D_{\mathbf{q}}$ to be a tree edge. For a word q_u, we set candidate support sentences (i.e., the possible value of z_u) to be those story sentences containing word q_u [2]. Define the linear score function $h_t(x, y, z)$:

$$h_t(x, y, z) = \alpha^{\mathsf{T}} \varphi(x, y, z), \tag{1}$$

where α is the model parameter, $\varphi(x, y, z)$ is the feature function. To obtain the score for the answer y, we take the maximum score over all hidden variables

$$h_t(x, y) = \max_z h_t(x, y, z).$$

Following the dependency tree structure, we decompose $h_t(x, y, z)$ (i.e., α and $\varphi(x, y, z)$) on $D_{\mathbf{q}}$'s nodes and edges:

$$\sum_{q_u \in \mathbf{q}} \alpha_n^{\mathsf{T}} \varphi_n(x, y, z_u) + \sum_{(q_u, q_v) \in D_{\mathbf{q}}} \alpha_e^{\mathsf{T}} \varphi_e(x, y, z_u, z_v),$$

where $(\varphi_n, \alpha_n), (\varphi_e, \alpha_e)$ are feature functions and model parameters on nodes and edges respectively. The node feature $\varphi_n(x, y, z_u)$ describes how plausible the support sentence z_u is according to q_u and y. For example, the number of words in z_u also appearing in $\mathbf{q}$ (or y). In fact, all features in existing hidden word alignment models (Yih et al. 2013) are applicable in φ_n, which makes our model a superset of those models.

The edge feature $\varphi_e(x, y, z_u, z_v)$ is new to machine comprehension models. It describes dependencies between the hidden variable z_u and z_v. For example, if z_u, z_v are the same story sentence $\mathbf{t}$, $\varphi_e(x, y, z_u, z_v)$ could ask whether the word q_u, q_v keep a similar relation in $\mathbf{t}$ as in $\mathbf{q}$. In fact, if all question words are aligned to a same story sentence, $\sum_{(q_u, q_v) \in D_{\mathbf{q}}} \varphi_e(x, y, z_u, z_v)$ can mimic the alignment of two dependency trees. In another case, if z_u, z_v are two different story sentences, φ_e could include features about whether the combination of the two sentences is able to support the answer. It is helpful to tackle questions whose answers are supported by multiple story sentences [3].

[1]Comparing with linear chain MRF, the tree structure may help us to capture long distance dependencies.

[2]We've experimented with larger candidate sentence sets (e.g., using word embedding based similarity threshold), but got negative results. One reason might be that the vocabulary size of MCTest data set is limited, and when we enlarge the candidate set, we always introduce random noise words rather than informative words.

[3]Note that features in φ_e (also φ_n) fall into two types according to whether it depends on the answer y. For those features are not relevant to y, their weights in α_e will not be updated during the

Algorithm 1 Message passing for the tree model

$\pi(u)$: the parent of u in $D_{\mathbf{q}}$
$\chi(u)$: the children of u in $D_{\mathbf{q}}$
z_r: the root of $D_{\mathbf{q}}$
z_0: an artificial node being the parent of z_r
$S = \{z_0, z_1, z_2, \ldots, z_{|\mathbf{q}|}\}$
repeat
  select z_u with all $\chi(u)$ are removed from S
  $m_{u\to\pi(u)} = \max_{z_u} \Big(\alpha_n^\intercal \varphi_n(x, y, z_u) + \alpha_e^\intercal \varphi_e(x, y, z_u, z_{\pi(u)}) + \sum_{k\in\chi(u)} m_{k\to u}\Big)$
  remove z_u from S
until S is empty
output $h_t(x, y) = \max_{z_r} m_{r\to 0}$

To handle syntactic structures of answers, we also include a tree model based on dependency trees of answers $D_{\mathbf{a}}$

$$h'_t(x, y, z) = \alpha'^\intercal \varphi'(x, y, z), \tag{2}$$

where the feature function φ' is based on the same φ_n, φ_e, but decomposed according to $D_{\mathbf{a}}$, and the paramter α' is also decomposed to α'_n, α'_e.

The inference problem in Equation 1 is tractable ($O(|\mathbf{q}||T|^2)$) by the classical message passing algorithm on the dependency tree $D_{\mathbf{q}}$ (Pearl 1982; Quattoni et al. 2007). For completeness, we list it in Algorithm 1.

Frame Model

Frame semantics is another widely used linguistic structure in text analysis. A semantic frame is a description of a situation with a trigger and several arguments. For example, the sentence "What did Sarah buy from the shop?" contains a frame "COMMERCE_BUY", with trigger word "buy" and arguments "Sarah" (Buyer), "shop" (Seller) and "what" (Goods). Frame semantics assumes that the meaning of a sentence could be read from its frames. Previous machine comprehension systems have included semantic frames as features (Wang et al. 2015; Sachan and Xing 2016). But, like the dependency tree structure, frames also provide structured information about dependencies among different hidden variables. We can also ask whether it is possible to use frames as inference structures for hidden variables.

We propose a model focusing on trigger-trigger relations in frames (Figure 1.(d)). Note that the triggers are core components of frames, a good alignment of trigger words may suggest a good alignment of frames, thus a large possibility of containing the correct answer.

Given a question $\mathbf{q}$, define its frames to be $\mathbf{F} = \{F_1, F_2, \ldots, F_{|\mathbf{F}|}\}$. For the trigger or an argument q_u in a frame F, we associate it with a variable z_u^F to represent its support sentence. Let z_1^F always be the variable of the trigger, and $z_2^F, z_3^F, \ldots, z_{|F|}^F$ be the variables of arguments. We consider candidate support sentences of a trigger to be those either containing the trigger word or the same type frame. Define a feature function on hidden variables of trigger words $\phi_t(x, y, \mathbf{F}) = \phi_t(x, y, z_1^{F_1}, z_1^{F_2}, \ldots, z_1^{F_{|\mathbf{F}|}})$. Features in ϕ_t will help searching the best alignment sentences of different frames in $\mathbf{q}$ to support the correct answer. We obtain the score function $h_f(x, y) = \max_z h_f(x, y, z)$, where

$$h_f(x, y, z) = \beta_t^\intercal \phi_t(x, y, \mathbf{F}). \tag{3}$$

We also develop a model using semantic relations within a frame. A feature function $\phi_a(x, y, F) = \phi_a(x, y, z_1^F, z_2^F, \ldots, z_{|F|}^F)$ is designed to find the best support sentences for the frame F. But it turns out that ϕ_a does not improve the overall performances in the MCTest dataset. (dropping about 1% of accuracy comparing with ϕ_t). And different from the tree models, the frame model on answers doesn't provide improvement in experiment results. We think that a more careful selection on frames would help to refine the model.

The inference of the frame model is accomplished by enumeration. Since the number of triggers are limited for questions in MCTest dataset, the cost of time is tolerable in our experiments.

Baseline Models

We incorporate a hidden sentence alignment model from (Narasimhan and Barzilay 2015) (Model 2 there). Define a feature function $\vartheta(x, y, z, z')$ which selects two support sentences z, z' (Figure 1.(e)). We have a score function $h_s(x, y) = \max_{z,z'} h_s(x, y, z, z')$, where

$$h_s(x, y, z, z') = \gamma^\intercal \vartheta(x, y, z, z'). \tag{4}$$

We also include the lexical matching baseline in (Richardson, Burges, and Renshaw 2013; Smith et al. 2015). Its scoring function does not rely on hidden variables:

$$h_g(x, y) = \delta^\intercal \eta(x, y), \tag{5}$$

where $\eta(x, y)$ contains features about lexical similarities between x and y.

Training

We can combine the score functions in previous section to get the final $h(x, y)$. First, to aggregate notations, denote Θ to represent all model parameters ($\alpha_n, \alpha_e, \alpha'_n, \alpha'_e, \beta_t, \gamma, \delta$), Φ to represent all feature functions (φ_n, φ_e, ϕ_t, ϑ, η) and $\mathbf{z}$ to represent all hidden variables (i.e., hidden variables in Equation 1, 2, 3, and 4). Then, we define

$$\begin{aligned} h(x, y) =& \max_{\mathbf{z}} h(x, y, \mathbf{z}) = h_t(x, y) + h'_t(x, y) \\ &+ h_f(x, y) + h_s(x, y) + h_g(x, y), \end{aligned} \tag{6}$$

and the finial prediction $\hat{y} = \arg\max_{y\in A} h(x, y)$.

An assumption in Equation 6 is that the hidden variables of different models are independent, thus, for example, the tree model and the frame model could align a question word to different story sentences. We make the assumption mainly

learning process of answer ranking models. These features can be seen as invariant prior knowledge: for example, one feature in φ_e could be the distance between z_u, z_v in T. We would think that the closer distance, the closer relation between them, thus we set a negative weight for it in α_e. Similar features have also been applied in (Sachan et al. 2015; Wang et al. 2015).

Algorithm 2 Hidden Variable Perceptron

Input: the training set $\{(T^i, \mathbf{q}^i, A^i, \mathbf{a}^i)\}_{i=1}^N$
Input: the iteration number M, the learning rate C
Initialization: randomly set entries of Θ in $[0, 1]$
for j=1,2,...,M **do**
  for i=1,2,...,N **do**
    $x \triangleq (T^i, \mathbf{q}^i)$, $y^* \triangleq \mathbf{a}^i$
    $\mathbf{z}_y \triangleq \arg\max_{\mathbf{z}} h(x, y, \mathbf{z}), \quad \forall y$
    $\hat{y} = \arg\max_{y \in A^i} h(x, y, \mathbf{z}_y)$
    if $\hat{y} \neq y^*$ **then**
      $\Theta_{j,i} = \Theta_{j,i-1} + C\Phi(x, y^*, \mathbf{z}_{y^*}) - C\Phi(x, \hat{y}, \mathbf{z}_{\hat{y}})$
    end if
  end for
end for
return $\frac{1}{M*N} \sum_{j,i} \Theta_{j,i}$

for simplifying the inference, and the joint inference among different models is left for future work.

Given a training set $\{(T^i, \mathbf{q}^i, A^i, \mathbf{a}^i)\}_{i=1}^N$, where $\mathbf{a}^i$ is the correct answer of question $\mathbf{q}^i$. We estimate Θ using averaged hidden variable perceptron (Sun et al. 2009) (Algorithm 2). We tune the algorithm parameter on the development set, and set $C = 0.001, M = 10$.

Features

Table 2 lists features used in different models. To build the features, the key problem is to measure similarities between two text spans σ, σ'. We use following operators:

- $\texttt{lex}(\sigma, \sigma')$: the number of overlapping unigram, bigram and trigram between σ and σ'.
- $\texttt{dep}(\sigma, \sigma')$: the number of overlapping tree edges (and connected tree edge bigrams) in dependency trees of σ, σ' (applicable when σ, σ' are sentences).
- $\texttt{cos}(\sigma, \sigma')$, the cosine similarities between vector representations (i.e., the sum of word vectors in the text span) of σ and σ'.

To mimic the sliding window baseline (Richardson, Burges, and Renshaw 2013; Smith et al. 2015), we denote a symbol $\sigma|\texttt{wd}$ to project the text span σ onto the window $\texttt{wd}$ (i.e., $\sigma|\texttt{wd}$ is a subsequence of σ). We also apply heuristic rules to combine $\mathbf{q}$ and $\mathbf{a}$ into one sentence $\mathbf{q} \circ \mathbf{a}$ (i.e., the *hypothesis* sentence in (Sachan et al. 2015)).

η	$\sum_{\texttt{wd}} \texttt{lex}(\mathbf{q} \circ \mathbf{a}, T\|\texttt{wd})$, $\sum_{\texttt{wd}} \texttt{cos}(\mathbf{q} \circ \mathbf{a}, T\|\texttt{wd})$ $\max_{\mathbf{t}_i \in T} \texttt{lex}(\mathbf{q} \circ \mathbf{a}, \mathbf{t}_i)$ $\max_{\mathbf{t}_i \in T} \texttt{lex}(\mathbf{q} \circ \mathbf{a}, \mathbf{t}_i \cup \mathbf{t}_{i+1})$ $\sum_{\mathbf{t}_i \in T} \texttt{cos}(\mathbf{q} \circ \mathbf{a}, \mathbf{t}_i)$, $\sum_{\mathbf{t}_i \in T} \texttt{dep}(\mathbf{q} \circ \mathbf{a}, \mathbf{t}_i)$ $\|\arg\max_i \texttt{lex}(\mathbf{a}, \mathbf{t}_i) - \arg\max_i \texttt{lex}(\mathbf{q}, \mathbf{t}_i)\|$
φ_n	$\texttt{lex}(\mathbf{q} \circ \mathbf{a}, z_u)$, $\texttt{lex}(\mathbf{a}, z_u)$ $\texttt{cos}(\mathbf{q} \circ \mathbf{a}, z_u)$, $\texttt{cos}(\mathbf{a}, z_u)$ $\texttt{dep}(\mathbf{q} \circ \mathbf{a}, z_u)$. $\texttt{dep}(\mathbf{a}, z_u)$ apply above 6 features to the sentence before/after z_u whether z_u overlaps with both $\mathbf{q}$ and $\mathbf{a}$
φ_e	$\texttt{lex}(\mathbf{q} \circ \mathbf{a}, z_u \cup z_v)$, $\texttt{lex}(\mathbf{a}, z_u \cup z_v)$ $\texttt{cos}(\mathbf{q} \circ \mathbf{a}, z_u \cup z_v)$, $\texttt{cos}(\mathbf{a}, z_u \cup z_v)$ $\texttt{dep}(\mathbf{q} \circ \mathbf{a}, z_u \cup z_v)$, $\texttt{dep}(\mathbf{a}, z_u \cup z_v)$ whether $z_u \cup z_v$ overlaps with both $\mathbf{q}$ and $\mathbf{a}$ if $z_u = z_v$, whether q_u, q_v keep the same relation in z_u † distance between z_u and z_v
ϕ_t	$\texttt{lex}(\mathbf{q} \circ \mathbf{a}, \cup_i z_1^{F_i})$, $\texttt{lex}(\mathbf{a}, \cup_i z_1^{F_i})$ whether $\cup_i z_1^{F_i}$ overlaps with both $\mathbf{q}$ and $\mathbf{a}$

Table 2: Features. For sliding window methods (i.e., features with wd), the window size equals to the length of $\mathbf{q} \circ \mathbf{a}$. The union "$\cup$" of sentences concatenates their words. Features in ϑ are similar to φ_e by setting hidden variables accordingly. All features are selected on the development set.

† If there are multiple occurrences of q_u in z_u, we take them as different assignments w.r.t. the different occurrences of q_u.

Experiments

The Dataset

The MCTest dataset (Richardson, Burges, and Renshaw 2013) contains 660 stories written for elementary grade school level students. For each story, four multiple choice questions are posed (2640 questions), and each of them contains four candidate answers. Since the stories are all fictional, the answers could only be found from the stories themselves. Questions are annotated with two types: one (only one story sentence is sufficient to answer it) and multiple (need multiple story sentences). Two subsets of MCTest data set are MC160 (160 stories) and MC500 (500 stories). We use the official training, development and testing splitting.

For preprocessing, we use simple rules for sentence splitting and word segmentation, Stanford CoreNLP for coreference resolution (Lee et al. 2011). Stanford parser for dependency parsing (Klein and Manning 2002), and SEMAFOR frame-semantic parser (Kshirsagar et al. 2015) for frame parsing. For word similarities, we use the pre-trained word embeddings from word2vec [4].

Settings

We compare our models with following baselines (the first eight rows of Table 3):

- lexical matching baselines (row 1, 2), which use a sliding window to count overlapping words between $\mathbf{q} \cup \mathbf{a}$ and T.
- hidden sentence alignment models (row 3, 4).
- hidden word alignment models (row 5, 6).
- deep learning models (row 7, 8).

Besides the full model in Equation 6, we consider different configurations of the proposed models. Recall that h_t, h'_t are the question and answer tree model, h_f is the frame model, h_s is the hidden sentence alignment model with two hidden variables, and h_g is the lexical matching method. We try different combinations of them for controlled comparisons.

Method	MCTest-160 accuracy(%)			MCTest-500 accuracy(%)		
	One(112)	Multiple(128)	All	One(272)	Multiple(328)	All
(Richardson, Burges, and Renshaw 2013)+RTE	76.78	62.50	69.16	68.01	59.45	63.33
(Smith et al. 2015)	78.79	70.31	74.27	69.12	63.34	65.96
(Narasimhan and Barzilay 2015)	82.36	65.23	73.23	68.38	59.90	63.75
(Wang et al. 2015)	84.22	67.85	75.27	72.05	67.94	69.94
(Sachan et al. 2015)	-	-	-	67.65	67.99	67.83
(Sachan and Xing 2016)	-	-	-	72.05	**68.90**	70.33
(Wang et al. 2016a)	**88.39**	64.84	75.83	79.04	63.51	70.96
(Trischler et al. 2016)	79.46	70.31	74.58	74.26	68.29	71.00
h_g	82.04	67.19	74.58	77.57	62.80	69.50
$h_g + h_t$	80.35	**72.66**	**76.25**	78.68	64.33	70.83
$h_g + h'_t$	83.93	62.50	72.50	79.04	63.11	70.33
$h_g + h_t + h'_t$	83.04	65.63	73.75	78.31	63.72	70.33
$h_g + h_f$	83.82	65.63	74.58	77.57	65.55	71.00
$h_g + h_s$	80.25	67.97	74.17	76.10	64.33	69.67
$h_g + h_t + h'_t + h_f$	81.25	66.41	73.33	76.84	65.55	70.67
$h_g + h_t + h'_t + h_f$(VOTE)	83.04	67.19	74.58	80.88	63.72	71.50
$h_g + h_t + h'_t + h_f + h_s$	78.57	67.19	72.50	79.41	65.24	71.67
$h_g + h_t + h'_t + h_f + h_s$(VOTE)	81.25	68.75	74.58	**80.15**	66.46	**72.67**

Table 3: Experimental results on the MCTest dataset.

Results

Table 3 lists the experiment results. The proposed algorithms are able to achieve the best performances on both MCTest-160 and MCTest-500. The testing time is about 0.5s per question on a single CPU with 8G RAM. The following are some discussions.

Firstly, models with hidden variables are always better than the lexical matching baseline h_g on MCTest-500 [5]. However, on the smaller MCTest-160, h_g is competitive. We think that the more expressiveness hidden models are necessary for the task, and they also require more training data to explore. On the other side, if we apply hidden variable models without the lexical matching features, their performances will decreased 10% in average (omitted in Table 3 due to the lack of space). Thus hidden variable models alone are also not sufficient to achieve the state-of-the-art results. It suggests that both the high level semantic similarity measurements and the detailed answer inference structures analysis are important for the reading comprehension task.

Secondly, we find that both the tree model $h_g + h_t$ and the frame model $h_g + h_f$ are better than existing hidden word alignment models (row 4, 5) and hidden sentence alignment models (row 2, 3). Since the main difference of our models to previous work is the appearance of dependencies among hidden variables, we would think that they are helpful for ranking answers, and the prior syntactic and semantic structures can be a proper way to capture these dependencies.

Thirdly, by comparing the results on MCTest-160 and MCTest-500, we find that the tree model $h_g + h_t$ is better than the full model on the smaller dataset. Thus, the full model may need more training data to fit. Another observation is that, for "Multiple" questions in MCTest-500, our models have lower scores than previous hidden word alignment models. One reason could be that the candidate aligned words are unconstrained there, while we need a constrained candidate support sentence set for controlling inference complexity.

	all	`-lex`	`-cos`	`-dep`	-others
w/o VOTE	68.0	62.5	66.5	66.0	64.5
w/ VOTE	69.5	63.5	67.5	67.5	65.5

Table 4: Ablation analysis of model $h_g + h_t$ on the development set of MCTest-500 (200 questions). "lex", "dep", "cos" represent features in $\eta, \varphi_n, \varphi_e$ which involve operator `lex`, `dep`, `cos` respectively (Table 2). "others" represents the remaining features.

Fourthly, although the full model $h_g + h_t + h'_t + h_f + h_s$ already outperforms the best previous result, we find that a simple voting strategy could further improve the results (denoted by "VOTE" in Table 3). During the testing, instead of taking $\hat{y} = \arg\max_{y \in A} h(x, y)$ as the output, we compute predictions of h_t, h'_t, h_f, h_s, h_g individually, and if any hidden variable model (i.e., h_t, h'_t, h_f, h_s) has the same prediction with h_g, we let $\hat{y}$ be the prediction of h_g. If predictions of all hidden variable models are different with h_g, we fall back on the original $h(x, y)$. Due to the size of MCTest dataset, underfitting may occur in models. Empirically, the voting could be seen as an ensemble of different hidden variable models. A more thorough study of ensemble learning for machine comprehension is left as future work.

Finally, the proposed models performs better than deep learning based models (Trischler et al. 2016; Wang et al.

[4]https://code.google.com/archive/p/word2vec/

[5]By introducing features from `cos`, `dep`, our implementation of h_g is better than previous work (row 1, 2).

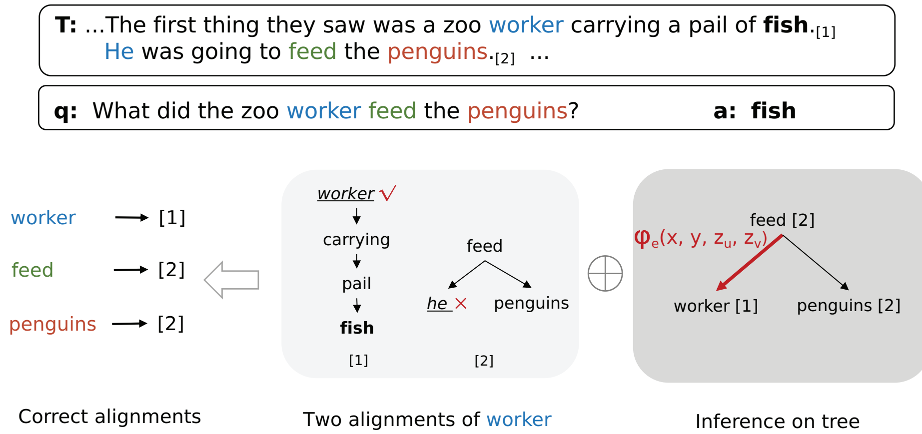

Figure 2: An example of answer inference process in the tree model. We have two story sentences. The bottom left part shows the correct assignment of variables, and the bottom right part shows how features in η, φ_e help to obtain the correct assignment.

1	**T:** The squirrel flew so high that it passed the buildings. It passed the birds, it passed the planes and stopped in the clouds. **Q:** What was the second thing the squirrel passed? **A:** A) the buildings B) the clouds *C) the birds D) the planes
2	**T:** . . . He came to a fast stop when he saw the dog. He'd seen a dog before...and he used to live with a black dog named Henry...He jumped on his favorite chair and looked down as Maggie ran under it. She was kind of cute for a dog... **Q:** What was the dog's name? **A:** *A) Maggie B) Henry C) Pester D) Linda

Table 5: Error analysis. Row 1&2 are examples that our models fail. The question in row 1 needs counting, and row 2 needs powerful anaphora resolution.

2016a). As mentioned in (Yin, Ebert, and Schütze 2016), learning the embedding layers for fictional stories could be hard in neural network models, while the prior syntactic and semantic structures could reduce the requirement on human annotations for the proposed models. The proposed models can also reduce the number of model parameters comparing with deep learning algorithms. On the other side, as suggested by (Sachan et al. 2015; Wang et al. 2016a), it is also possible to learn knowledge independent features from related tasks (e.g., text entailment, answer selection), and it might be a promising method to augment hidden variable models with deep representation learning.

Table 4 lists ablation analysis on features (we only give results on the setting of $h_g + h_t$ due to the lack of space). We can observe that, while the lexical matching features ("lex") are essential for the task, "cos", "dep" and "other" feature groups could also provide significant performance gain. For example, we would think that features such as whether two question words keep the same relation in the aligned sentence (in "others") are crucial for the correct alignment.

We give an example to illustrate the answer inference in Figure 2. In the example, we have two possible alignments of the question word "worker" (i.e., story sentence 1 and sentence 2). By considering the answer information ("fish"), we may prefer to align it to the first sentence instead of the second one (specifically, the sliding window features in η can tell that the first sentence has more overlaps with the hypothesis $\mathbf{q} \circ \mathbf{a}$). Then, with the help of edge feature φ_e, we can connect the syntactic information of the two sentences (specifically, `dep` features in φ_e can tell that "feed" and "work" hold the same type of dependency in sentence 2 and the question). Finally, the optimal assignment of hidden alignment variable give a high ranking score for the correct answer ("fish"). We also give some failed questions in Table 5. It shows that questions such as counting are still hard for our models.

Conclusion

We studied dependencies among hidden variables for the machine comprehension. Two novel methods based on syntactic trees and semantic frames were proposed. The models achieved state-of-the-art performances on standard MCTest dataset. For future work, we plan to investigate the joint inference of the proposed models, and also incorporate knowledge and data from other related tasks.

Acknowledgements

The authors wish to thank all reviewers for their helpful comments and suggestions. The corresponding authors are Yuanbin Wu and Man Lan. This research is (partially) supported by NSFC (61402175), STCSM (15ZR1410700) and Shanghai Key Laboratory of Trustworthy Computing (07dz22304201604). Yuanbin Wu is supported by a Microsoft Research Asia Collaborative Research Program.

References

Clark, P., and Etzioni, O. 2016. My computer is an honor student - but how intelligent is it? standardized tests as a measure of AI. *AI Magazine* 37(1):5–12.

Heilman, M., and Smith, N. A. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proc. of NAACL*, 1011–1019. Los Angeles, California: Association for Computational Linguistics.

Hermann, K. M.; Kociský, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *NIPS*, 1693–1701.

Hill, F.; Bordes, A.; Chopra, S.; and Weston, J. 2015. The goldilocks principle: Reading children's books with explicit memory representations. In *Proc. of ICLR*.

Klein, D., and Manning, C. D. 2002. Fast exact inference with a factored model for natural language parsing. In *NIPS*, 3–10.

Kshirsagar, M.; Thomson, S.; Schneider, N.; Carbonell, J.; Smith, N. A.; and Dyer, C. 2015. Frame-semantic role labeling with heterogeneous annotations. In *Proc. of ACL*, 218–224. Beijing, China: Association for Computational Linguistics.

Kushman, N.; Artzi, Y.; Zettlemoyer, L.; and Barzilay, R. 2014. Learning to automatically solve algebra word problems. In *Proc. of ACL*, 271–281. Baltimore, Maryland: Association for Computational Linguistics.

Lee, H.; Peirsman, Y.; Chang, A.; Chambers, N.; Surdeanu, M.; and Jurafsky, D. 2011. Stanfords multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proc. of CoNLL: Shared Task*, 28–34. Portland, Oregon, USA: Association for Computational Linguistics.

Lee, K.; Kwiatkowski, T.; Parikh, A. P.; and Das, D. 2016. Learning recurrent span representations for extractive question answering. *CoRR* abs/1611.01436.

Narasimhan, K., and Barzilay, R. 2015. Machine comprehension with discourse relations. In *Proc. of ACL*, 1253–1262.

Pearl, J. 1982. Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the National Conference on Artificial Intelligence. Pittsburgh, PA, August 18-20, 1982.*, 133–136.

Quattoni, A.; Wang, S. B.; Morency, L.; Collins, M.; and Darrell, T. 2007. Hidden conditional random fields. *IEEE TPAMI* 29(10):1848–1852.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proc. of EMNLP*, 2383–2392. Austin, Texas: Association for Computational Linguistics.

Richardson, M.; Burges, C. J.; and Renshaw, E. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proc. of EMNLP*, 193–203.

Sachan, M., and Xing, E. 2016. Machine comprehension using rich semantic representations. In *Proc. of ACL*, 486–492.

Sachan, M.; Dubey, K.; Xing, E.; and Richardson, M. 2015. Learning answer-entailing structures for machine comprehension. In *Proc. of ACL*, 239–249.

Smith, E.; Greco, N.; Bosnjak, M.; and Vlachos, A. 2015. A strong lexical matching method for the machine comprehension test. In *Proc. of EMNLP*, 1693–1698.

Sun, X.; Matsuzaki, T.; Okanohara, D.; and Tsujii, J. 2009. Latent variable perceptron algorithm for structured classification. In *Proc. of IJCAI*, 1236–1242.

Trischler, A.; Ye, Z.; Yuan, X.; He, J.; and Bachman, P. 2016. A parallel-hierarchical model for machine comprehension on sparse data. In *Proc. of ACL*, 432–441.

Wang, M., and Manning, C. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proc. of COLING*, 1164–1172. Beijing, China: Coling 2010 Organizing Committee.

Wang, H.; Bansal, M.; Gimpel, K.; and McAllester, D. 2015. Machine comprehension with syntax, frames, and semantics. In *Proc. of ACL*, 700–706.

Wang, B.; Guo, S.; Liu, K.; He, S.; and Zhao, J. 2016a. Employing external rich knowledge for machine comprehension. In *Proc. of IJCAI*, 2929–2925.

Wang, Z.; Mi, H.; Hamza, W.; and Florian, R. 2016b. Multi-perspective context matching for machine comprehension. *CoRR* abs/1612.04211.

Wang, W.; Yang, N.; Wei, F.; Chang, B.; and Zhou, M. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proc. of ACL*, 189–198. Vancouver, Canada: Association for Computational Linguistics.

Wang, M.; Smith, N. A.; and Mitamura, T. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proc. of EMNLP-CoNLL*, 22–32. Prague, Czech Republic: Association for Computational Linguistics.

Yao, X.; Van Durme, B.; Callison-Burch, C.; and Clark, P. 2013. Answer extraction as sequence tagging with tree edit distance. In *Proc. of NAACL-HLT*, 858–867. Atlanta, Georgia: Association for Computational Linguistics.

Yih, W.-t.; Chang, M.-W.; Meek, C.; and Pastusiak, A. 2013. Question answering using enhanced lexical semantic models. In *Proc. of ACL*, 1744–1753. Sofia, Bulgaria: Association for Computational Linguistics.

Yin, W.; Ebert, S.; and Schütze, H. 2016. Attention-based convolutional neural network for machine comprehension. In *Proceedings of the Workshop on Human-Computer Question Answering*, 15–21.