Constraint-Based Question Answering with Knowledge Graph

Junwei Bao†*, Nan Duan‡ , Zhao Yan♯ , Ming Zhou‡ , Tiejun Zhao†
†Harbin Institute of Technology, Harbin, China
‡Microsoft Research, Beijing, China
♯Beihang University, Beijing, China
†`baojunwei001@gmail.com`
‡`{nanduan, mingzhou}@microsoft.com`
♯`yanzhao@buaa.edu.cn`
†`tjzhao@hit.edu.cn`

Abstract

WebQuestions and *SimpleQuestions* are two benchmark data-sets commonly used in recent knowledge-based question answering (KBQA) work. Most questions in them are 'simple' questions which can be answered based on a single relation in the knowledge base. Such data-sets lack the capability of evaluating KBQA systems on complicated questions. Motivated by this issue, we release a new data-set, namely *ComplexQuestions*[1], aiming to measure the quality of KBQA systems on 'multi-constraint' questions which require multiple knowledge base relations to get the answer. Beside, we propose a novel systematic KBQA approach to solve multi-constraint questions. Compared to state-of-the-art methods, our approach not only obtains comparable results on the two existing benchmark data-sets, but also achieves significant improvements on the *ComplexQuestions*.

1 Introduction

Knowledge-based question answering is a task that aims to answer natural language questions based on existing knowledge bases (KB). In the last decades, large scale knowledge bases, such as Freebase (Bollacker et al., 2008), have been constructed. Based on Freebase, two benchmark data-sets, *WebQuestions* (Berant et al., 2013) and *SimpleQuestions* (Bordes et al., 2015) are constructed and used in most of KBQA work (Berant and Liang, 2014; Bordes et al., 2014a; Fader et al., 2014; Yang et al., 2014; Bao et al., 2014; Reddy et al., 2014; Dong et al., 2015; Yih et al., 2015).

However, about 85% of questions (Yao, 2015) of *WebQuestions* and all questions in *SimpleQuestions* are 'simple' questions, where a 'simple' question denotes that it can be answered based on a single KB relation. For example in Figure 1, "Which films star by Forest Whitaker" is a simple question that can be answered by the KB triples like ⟨`Forest Whitaker,acted_films,?`⟩ with a single KB relation `acted_films`. This leads to the fact that such data-sets cannot measure the capability of KBQA systems on 'multi-constraint' questions, where 'multi-constraint' means a question containing multiple semantic constraints expressed with different expressions to restrict the answer set. To answer a multi-constraint question, we have to base on multiple KB relations. For example in Figure 1, "Which films star by Forest Whitaker and are directed by Mark Rydell" is a multi-constraint question with a constraint "directed by Mark Rydell", which requires multiple

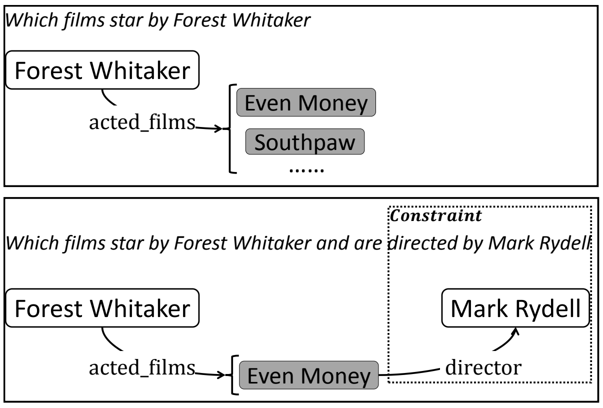


Figure 1: Simple and multi-constraint questions.

*This work was finished while the author was visiting Microsoft Research Asia.
[1]https://github.com/JunweiBao/MulCQA/tree/ComplexQuestions

Constraint Category	Example	Percentage
Multi-Entity	which films star by **Forest Whitaker** and are directed by **Mark Rydell**?	30.6%
Type	which **city** did Bill Clinton born?	38.8%
Explicit Temporal	who is the governor of Kentucky **2012**?	10.4%
Implicit Temporal	who is the us president **when the Civil War started**?	3.5%
Ordinal	what is the **second longest** river in China?	5.1%
Aggregation	**how many** children does bill gates have?	1.2%

Table 1: Constraint categories, examples, and distributions over a set of web queries. Note, a multi-constraint question may belong to not only one constraint category. And there are still other relatively low frequency type of complex questions existing which we don't take consideration in this work. So the sum of the Percentage for these constraints are not guaranteed to be 1.

KB triples ⟨`Forest Whitaker,acted_films,Even Money`⟩ and ⟨`Even Money,director,Mark Rydell`⟩ with two KB relations `acted_films` and `director` to get the exact answer set.

Motivated by this issue, this work contributes to QA research in the following two aspects: (1) We propose a novel systematic KBQA approach to solve multi-constraint questions by translating a **mul**ti-**c**onstraint **q**uestion (MulCQ) to a **mul**ti-**c**onstraint query **g**raph (MulCG); (2) A new QA data-set, namely *ComplexQuestions*, is released, aiming to measure the quality of KBQA systems on multi-constraint questions. Compared to state-of-the-art approaches, our method obtains comparable results on the existing benchmark data-sets *WebQuestions* and *SimpleQuestions*. Furthermore, we achieve significant improvement on the newly created *ComplexQuestions* data-set.

2 Multi-Constraint Question

2.1 Constraint Classification

A MulCQ is defined as a question which requires multiple KB relations or special operations to get the answer. Based on web query analysis, we classify constraints into 6 categories as follows:

(1) *Multi-entity constraint.* A question in this category denotes that multiple entities occur in it, which restrict the answer. For example, "Forest Whitaker" and "Mark Rydell" are two entity constraints in the first question in Table 1.

(2) *Type constraint.* A question in this category denotes that its answer should follow a type, which is explicitly mentioned by the question. For example, the answer to the second question in Table 1 should be a "city" name, instead of locations with other types such as country, town, etc.

(3) *Explicit temporal constraint.* A question in this category denotes that it contains explicit temporal expressions, such as "2012" in the third question in Table 1. Such questions are very common in web queries, which means handling them well will bring about significant improvements.

(4) *Implicit temporal constraint.* A question in this category denotes that it contains implicit temporal expressions. For example, "when the Civil War started" denotes an implicit temporal constraint in the fourth question in Table 1. We should transform it into an explicit temporal constraint before answering the question. Such constraints are usually expressed by subordinate clauses.

(5) *Ordinal constraint.* A question in this category denotes that its answer should be selected from a ranked set, based on ordinal numbers or superlative phrases as ranking criteria. For example, "second longest" in the fifth question in Table 1 denotes that the answer should be the second item in the ranked Chinese river set, based on their lengths.

(6) *Aggregation constraint.* A question in this category denotes that it asks for the number of a set, which often starts with phrases "how many" or contains "number of", "count of", etc.

2.2 Question Selection

We perform the following steps to select suitable multi-constraint question candidates for human annotators to label, based on Freebase.

Firstly, a three month (2015.1.1-2015.4.1) query log from a practical search engine is used as the raw

Operation	**Trigger**	**Description**
$Equal(y_0, y_1)$	N/A	Return `True` if y_0 is equal to y_1, otherwise `False`
$< (y_0, y_1)$	"after" or "later then"	Return `True` if y_0 is smaller than y_0, otherwise `False`
$> (y_0, y_1)$	"before" or "earlier then"	Return `True` if y_0 is larger than y_0, otherwise `False`
`MaxAtN`(x, n)	Maximize superlatives in WordNet	Rank items of x in descending order, return the n^{th} one
`MinAtN`(x, n)	Minimize superlatives in WordNet	Rank items of x in ascending order, return the n^{th} one
`Count`(x)	"how many", "count of" or "number of"	Return the number of entity set x.

Table 2: Functional predicates defined in this work.

query set, which contains 20,999,951 distinct 5W1H questions[2] that satisfy the following two rules: (i) each query should not contain pronouns (e.g., 'you', 'my', etc.), as questions with such words are usually non-factual questions, and (ii) each query's length is between 7 and 20, as short queries seldom contain multi-constraints, and long queries are usually difficult to answer. Then, we further sample 10 percent of questions, and use an entity linking method (Yang and Chang, 2015) to detect entities. If no entity can be detected from a query, we simply remove it. Next, both *WebQuestions* and *SimpleQuestions* are used to extract a set of words, without considering stop words and entity words. If a query does not contain any word in this word set, we simply remove it. This is intuitive, as *WebQuestions* and *SimpleQuestions* are our training data, and we only consider queries that can be covered by the training data as query candidates. Last, we classify the remaining queries based on the following rules:

(1) If a question contains at least two non-overlap entities, then it belongs in the Multi-Entity category;

(2) If a question contains a type phrase that comes from Freebase, then it belongs in the Type category;

(3) If a question contains a time expression detected by an Named Entity Recognizer (NER) (Finkel et al., 2005), then it belongs in the Explicit Temporal category;

(4) If a question contains keywords "when", "before", "after" and "during" in the middle, then it belongs in the Implicit Temporal category;

(5) If a question contains ordinal number or superlative phrase from WordNet (Miller, 1995), then it belongs in the Ordinal category;

(6) If a question starts with "how many", or includes "number of" or "count of", then it belongs in the Aggregation category.

Note, a multi-constraint question may contain multiple types of constraints. We show constraint types, examples, and distributions in Table 1. Ten thousand questions from the above 6 categories are selected, according to their distributions. By manually labeling these questions according to Freebase, we obtain 878 multi-constraint question answer pairs.

2.3 Question Annotation

We release the *ComplexQuestions* data-set, which consists of 2100 multi-constraint question answer pairs coming from 3 sources:

(1) 596 QA pairs selected from *WebQuestions* training set, and 326 from the test set,

(2) 300 QA pairs released by (Yin et al., 2015),

(3) 878 manually labeled QA pairs based on Section 2.2.

We then split it into two parts: a training set containing 1300 QA pairs and a test set including 800 QA pairs[3].

3 Definition

3.1 Knowledge base

$\mathcal{K}$ denotes a knowledge base[4] (KB) that stores a set of facts. Each fact $t \in \mathcal{K}$ is a triple $\langle s, p, o \rangle$, where p represents a predicate (e.g., `birthday`), and s, o (e.g., `BarackObama`, `1961`) represent an entity or a value,

[2]5W1H questions are ones the start with "what", "where", "when", "who", "which" or "how".

[3]We put QA pairs from the training (testing) set of *WebQuestions* still in the training (testing) set of *ComplexQuestions*, and the same for the test part.

[4]In this work, we use Freebase, which is a large knowledge base with more than 46 million entities and 2.6 billion facts. In Freebase setting, CVT, namely *compound value type* is a special entity category, which is not a real world entity, but is used to collect multiple fields of an event.

which are the subject and object of t.

3.2 Multi-Constraint Query Graph

Before introducing multi-constraint query graph (MulCG), we first define four basic elements:

Vertex There are two types of vertices: constant vertex (rectangle) and variable vertex (circle). A constant vertex represents a grounded KB entity or a value, such as `Barack Obama` or `1961`. A variable vertex represents ungrounded entities or unknown values.

Edge There are two types of edges: relational edge and functional edge. A relational edge represents a predicate in the KB, such as `birthday`. A functional edge represents a functional predicate of a truth, such as $<$ in the truth $\langle$`2000`, `<`, `2001`$\rangle$. Functional predicates are defined in Table 2.

Basic Query Graph A basic query graph is defined as a triple $\langle v_s, p, v_o\rangle$, where v_s denotes a constant vertex as the subject that occurs in a given question, v_o (shaded circle) denotes a variable vertex as hidden answers of the question, p denotes the 'path' that links v_s and v_o by one or two edges [5] (e.g., `officials-holder`).

Constraint A constraint is defined as a triple $\langle v_s, r, v_o\rangle$, where v_s is a constant vertex, v_o is a variable vertex, r is a functional edge, and after instantiation based on a knowledge base, all instantiated entities from v_o should satisfy the predicate of r with regard to v_s.

MulCG A MulCG is constructed based on a basic query graph $\mathcal{B}$ of a question and an ordered constraint sequence $\mathcal{C} = \{\mathcal{C}_1, ..., \mathcal{C}_N\}$ by the following operations: (1) Treat the basic query graph $\mathcal{B}$ of the given question as $\mathcal{G}_0$; (2) Iteratively add $\mathcal{C}_i$ to $\mathcal{G}_{i-1}$ to generate $\mathcal{G}_i$, by linking the variable vertex of $\mathcal{C}_i$ to a vertex of $\mathcal{G}_{i-1}$ with some possible path, or directly merge them as one variable vertex. (3) Output $\mathcal{G}_N$.

Given a MulCG of a question, we can execute it based on the KB by instantiating all variable vertices according to the constraints in order. Specifically, we start from the constant vertex in the basic query graph and instantiate all variable vertices according to the constraints in order. During this procedure, each instantiated paired entities connected by an edge should satisfy the predicate of the edge based on $\mathcal{K}$ and commonsense knowledge.

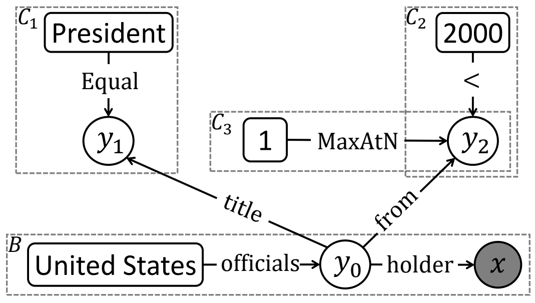


Figure 2: MulCG for question "Who was the first president of United States after 2000?"

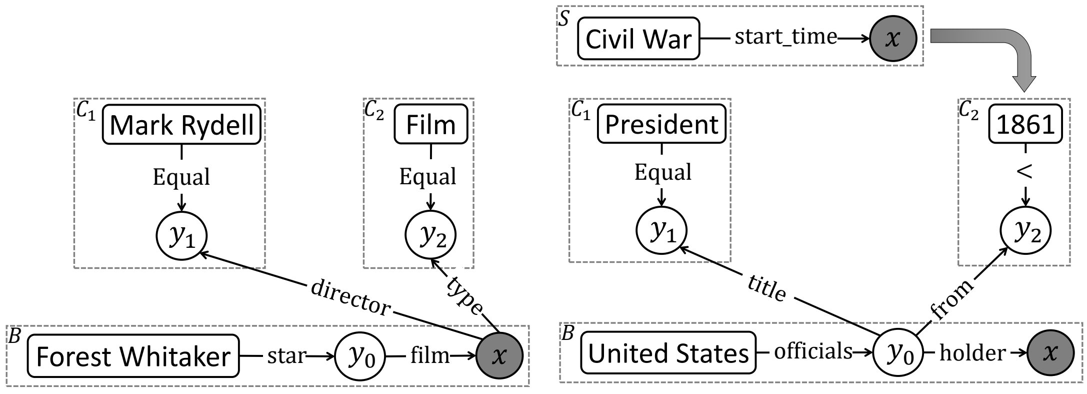
Figure 2 shows one possible MulCG for the given question. $\mathcal{B}$ is a basic query graph with a constant vertex `United States`, variable vertices y_0 and x, and two edges `officials` and `holder`. $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$ is an ordered constraint sequence detected based on the question, where $\mathcal{C}_1 = \langle$`President`,`Equal`,$y_1\rangle$, $\mathcal{C}_2 = \langle$`2000`,`<`,$y_2\rangle$, $\mathcal{C}_3 = \langle$`1`,`MaxAtN`,$y_2\rangle$. By adding $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ in order, we can construct the MulCG in Figure 2. Note, different constraint order can result in different MulCGs. We will introduce how to generate a MulCG in Section 4.

Compared to the stage graph in Yih et al. (2015), our MulCG has the following two differences: (1) Entity constraints can be added beyond single KB fact, while stage graph only considers entities that connect to the CVT node of a single KB fact as constraints. (2) Non-entity constraints are defined and handled in a systematic way, while stage graph only considers limited non-entity constraints, i.e., type and gender.

4 Our Approach

Problem Formalization Given a MulCQ $\mathcal{Q}$ and a KB $\mathcal{K}$, the question is parsed into a set of MulCGs $\mathcal{H}(\mathcal{Q})$. For each MulCG $\mathcal{G} \in \mathcal{H}(\mathcal{Q})$, a feature vector $\mathcal{F}(\mathcal{Q}, \mathcal{G})$ is extracted and the one with the highest

[5] If p contains two edges, then the vertex between must represent a CVT entity in KB. We call an edge or two edges with a CVT variable vertex 'path' in this work.



(a) MulCG with entity and type constraint for question "Which <u>films</u> star by <u>Forest Whitaker</u> and are directed by <u>Mark Rydell</u>".

(b) MulCG with implicit temporal constraint for question "Who was <u>U.S.</u> <u>president</u> <u>after the Civil War started</u>".

Figure 3: MulCGs examples for different type of constraints.

ranking score is selected. Finally, by executing the MulCG, we get the answers $\mathcal{A}$.

4.1 Basic Query Graph Generation

We use the entity linking approach proposed by (Yang and Chang, 2015) to detect entities mentioned by the given question. For each detected entity s, we treat it as a subject constant vertex. Based on the KB, for each unique KB 'path' from s, where a KB 'path' means one hop predicate p_0 or two hop predicates p_1-p_2[6], we construct a basic query graph $\langle s, p_0, x\rangle$ or $\langle s, p_1\text{-}y_{cvt}\text{-}p_2, x\rangle$. y_{cvt} and x are variable vertices, and x denotes the answer. For example, the basic query graph $\mathcal{B}$ in Figure 2 can be represented as $\langle$`United States`, `officials`-y_0-`holder`, $x\rangle$.

To measure the quality of each basic query graph constructed, we leverage a convolutional neural network (CNN)-based model that is similar to (Gao et al., 2015; Shen et al., 2014b; Shen et al., 2014a; Yih et al., 2015) to calculate the similarity between question and the path of the basic query graph. We will describe the training resource in Section 4.4.

4.2 Constraint Detection and Binding

Basic query graph is fit for single relation questions (Yih et al., 2014; Bordes et al., 2015), but not suffices to express a question with multiple constraints, such as the question in Figure 2. Hence, we propose to use constraints to restrict the answer set by adding them into the basic query graph.Adding a constraint contains two steps: *Constraint Detection* and *Constraint Binding*. We explain how to add each of the six kinds of constraints respectively in the following parts.

Entity Constraint Entity constraint is designed to understand entities and relations which are often expressed by noun phrases and verb phrases. A constraint with an entity as its constant vertex is an entity constraint. For instance, Figure 3(a) is a question with multiple entities such as "Forest Whitaker" and "Mark Rydell". After the basic query graph $\mathcal{G}_0 = \mathcal{B}$ is generated, we detect a constraint $\mathcal{C}_1 = \langle$`Mark Rydell`, `Equal`, $y_1\rangle$ and bind it to $\mathcal{G}_0$ by an edge `director`. Generally, the two steps to add an entity constraint are as follows: (1) *Constraint Detection*: For a detected entity $e \in \mathcal{E}$ (e.g., `Mark Rydell`), we construct a constraint $\mathcal{C}_i = \langle e$, `Equal`, $y_i\rangle$ (e.g., $\langle$`Mark Rydell`, `Equal`, $y_1\rangle$); (2) *Constraint Binding*: Given a MulCG $\mathcal{G}_{i-1}$ (e.g., $\mathcal{B}$), and a detected constraint $\mathcal{C}_i$ (e.g., $\langle$`Mark Rydell`, `Equal`, $y_1\rangle$), we try to bind $\mathcal{C}_i$ to $\mathcal{G}_{i-1}$ by linking the variable vertex of $\mathcal{C}_i$ (e.g., y_1 of $\mathcal{C}_1$) to a vertex of $\mathcal{G}_{i-1}$ (e.g., x of $\mathcal{B}$) by a possible path p (e.g., `director`). To measure the similarity between the path p (e.g., `director`) and the 'context pattern'[7] (e.g., "directed by e1") of constraint $\mathcal{C}_i$, we adopt a convolutional neural network (CNN) model which is described in Section 4.4.

[6]A KB 'path' containing two hop predicates means the entity between the two predicates is a CVT entity in the KB.

[7]A 'context pattern' is a 2-word context with the entity mention replaced by a slot "*e1*"

Note, a constraint can be linked to any vertex in the basic query graph. For example, the constraint $\mathcal{C}_1$ in Figure 2 is binding to a variable vertex y_0. When a constraint is binding to a constant vertex, then it is usually used for entity disambiguation.

Type Constraint Answer type is often explicitly expressed by nouns in questions. For instance, "film" is the answer type of the question in Figure 3(a), which denotes a type constraint. (1) *Constraint Detection*: Different from an entity constraint, we detect an answer type with simple but efficient rules as Yao and Durme (2014) for the question. For each KB type[8], we construct a type constraint (e.g., $\mathcal{C}_2 = \langle \texttt{Film}, \texttt{Equal}, y_2 \rangle$); (2) *Constraint Binding*: A type constraint is only added to the variable vertex which denotes the answers with a specific edge `type`.

Explicit Temporal Constraint Temporal constraint is designed to understand temporal expressions, which are often expressed by numericals, prepositional phrases or clauses. An explicit time expression, such as "after 2000" in the question in Figure 2 indicates a temporal constraint $\mathcal{C}_2$. Through linking the vertex y_0 to y_2 with a predicate `from`, functional constraint $\mathcal{C}_2 = \langle \texttt{2000}, <, y_2 \rangle$ selects the subset of the grounded entities for y_0 whose taking office time is later than `2000`. Generally, the two steps to add an explicit temporal constraint are as follows: (1) *Constraint Detection*: We use Stanford NER (Finkel et al., 2005) to detect a time phrase. If a time t (e.g., `2000`) is detected, and a functional operation r (e.g., $<$) is triggered by a lexicon which is partially listed in Table 2, we then construct a constraint $\mathcal{C}_i = \langle t, r, y_i \rangle$ (e.g., $\mathcal{C}_2 = \langle \texttt{2000}, <, y_2 \rangle$); (2) *Constraint Binding*: If an explicit temporal constraint $\mathcal{C}_i$ (e.g., $\mathcal{C}_2$) is detected, then we execute $\mathcal{G}_{i-1}$ (e.g., $\mathcal{B}$ with $\mathcal{C}_1$) on the KB. If there is a KB path p from grounded entities of the linking vertex (e.g., y_0) in $\mathcal{G}_{i-1}$ satisfying the restriction that p's object KB type is `Date_Time`, then the temporal constraint $\mathcal{C}_i$ is bound to $\mathcal{G}_{i-1}$ by an edge denoting p.

Implicit Temporal Constraint Time expressions such as the clause "after the Civil War started" in the question in Figure 3(b) can also trigger a temporal constraint, but it is expressed with an implicit temporal adverbial clause. (1) *Constraint Detection*: A NER can not detect this kind of implicit temporal expressions, so the dependency information is adopted to detect temporal clause starting with predefined keywords[9]. We first use our system to answer the clause to get an explicit time (e.g., by a MulCG $\mathcal{S}$ to get `1861`), then the detection falls into the same procedure as an explicit temporal constraint; (2) *Constraint Binding*: It is same as an explicit temporal constraint.

Ordinal Constraint An ordinal constraint aims to understand the numerals and superlative forms of adjectives or adverbs. For example, the question in Figure 2 contains an expression "first" which denotes that an ordinal constraint $\mathcal{C}_3$ should be added to graph $\mathcal{G}_2$ ($\mathcal{B}$ with constraints $\mathcal{C}_1$ and $\mathcal{C}_2$). After executing $\mathcal{G}_2$, we rank the grounded values of vertex y_2 and pick up the $\texttt{1}^{st}$ one by functional operation `MaxAtN`. Generally, two steps are adopted to add an ordinal constraint: (1) *Constraint Detection*: We use a manually collected ordinal number list and superlative vocabulary from WordNet (Miller, 1995) to detect an ordinal number n (e.g., 1) and a functional operation *op* (e.g., `MaxAtN` or `MinAtN`) to construct an ordinal constraint $\mathcal{C}_i = \langle n, op, y_i \rangle$; (2) *Constraint Binding*: If an edge p (e.g., from) linking a vertex of $\mathcal{G}_{i-1}$ (e.g., y_0 of $\mathcal{G}_2$) to the variable vertex of a constraint $\mathcal{C}_i$ (e.g., y_2[10] of $\mathcal{C}_3$) satisfies the restriction that the object entity of the predicate of p is a numerical or time value, and the word embedding similarity between the superlative word and the binding path's last word is the largest, then $\mathcal{C}_i$ is bound to $\mathcal{G}_{i-1}$ with edge p (e.g., $\mathcal{C}_3$ is bound to $\mathcal{G}_2$ with `from`).

Aggregation Constraint An aggregation constraint is added when the question starts with phrase "how many", or contains "number of", "count of". For instance, the phrase "how many" in the question "how many children does bill gates have" trigger an aggregation constraint. We treat it specially by counting the number of the grounded entities of the answer vertex.

4.3 Search Space Generation

[8]A KB type is the type of an entity in the KB, such as `People`, `Film`. We extract entire KB types from Freebase.

[9]Such as "when","before","after","during",etc.

[10]Note, since the linking edges for $\mathcal{C}_2$ and $\mathcal{C}_3$ are both `from`, we bind $\mathcal{C}_3$ with $\mathcal{G}_2$ by merging y_3 and y_2 as y_2.

Given a MulCG $\mathcal{Q}$, Algorithm 1 explains how to generate the search space $\mathcal{H}(\mathcal{Q})$. Firstly, we set $\mathcal{H}(\mathcal{Q})$ and a temp set $\mathcal{T}$ empty. A set of entities $\mathcal{E}$ are detected by entity linking component which takes $\mathcal{Q}$ as input. Secondly, for each entity $e \in \mathcal{E}$, we generate all possible basic query graphs $\mathcal{G}_b$ based on the knowledge base $\mathcal{K}$. Each basic query graph $g_b \in G_b$ is added into both $\mathcal{T}$ and $\mathcal{H}(\mathcal{Q})$. Then, for each basic query graph $g_b \in \mathcal{T}$, based on $\mathcal{Q}, \mathcal{E}, \mathcal{K}$, a set of constraints $\mathcal{C}$ are detected through constraint detection component. Function $Permutation(\mathcal{C})$ returns all possible index sequences of permutations of $\mathcal{C}$. For each index sequence $\mathcal{I} \in Permutation(\mathcal{C})$, constraints are bound recurrently. For each constraint $\mathcal{C}_{\mathcal{I}_i}$, we try to bind $\mathcal{C}_{\mathcal{I}_i}$ into a temporary MulCG g_c. Finally, a set of MulCG candidates $\mathcal{H}(\mathcal{Q})$ is generated.

Algorithm 1: MulCG Generation

```
1  $\mathcal{H}(\mathcal{Q}) = \emptyset$;
2  $\mathcal{T} = \emptyset$;
3  $\mathcal{E} = EntityLinking(\mathcal{Q})$;
4  foreach $s \in \mathcal{E}$ do
5      $\mathcal{G}_b = BasicQueryGraphGenration(s, \mathcal{K})$;
6      foreach $g_b \in \mathcal{G}_b$ do
7          insert $g_b$ to $\mathcal{T}$;
8          insert $g_b$ to $\mathcal{H}(\mathcal{Q})$;
9      end
10 end
11 foreach $g_b \in \mathcal{T}$ do
12     $\mathcal{C} = ConstraintDetection(g_b, \mathcal{E}, \mathcal{Q}, \mathcal{K})$;
13     foreach $\mathcal{I} \in Permutation(\mathcal{C})$ do
14         $g_c = g_b$;
15         for $i = 0$ to $|\mathcal{I}| - 1$ do
16             $g_c = ConstraintBinding(g_c, \mathcal{C}_{\mathcal{I}_i})$;
17         end
18         insert $g_c$ to $\mathcal{H}(\mathcal{Q})$;
19     end
20 end
21 return $\mathcal{H}(\mathcal{Q})$.
```

4.4 Features and Ranking

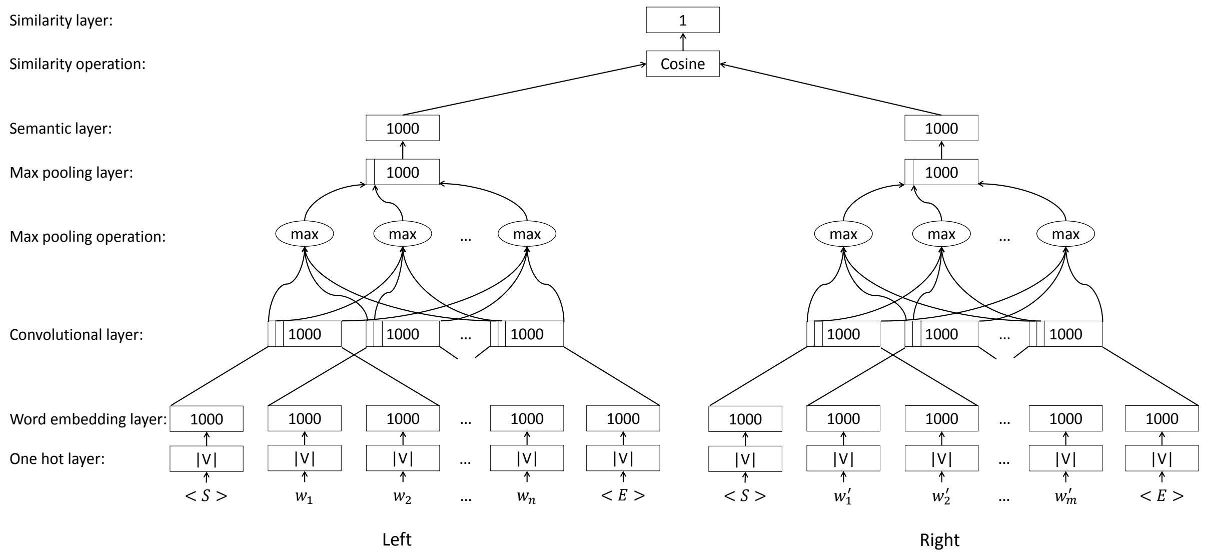


Figure 4: Siamese Convolutional Neural Network

In this work, we propose using Siamese convolutional neural networks (CNN) in Figure 4 to calculate the similarity of two sequences. The model consists of two neural networks taking two sequences as input and maps both of them to k-dimensional vectors. Similar models, such as CDSSM (Shen et al., 2014b; Gao et al., 2015) has been proved for web search. Besides, Yih et al. (2015) use similar frameworks for semantic parsing and question answering. This continuous space representation approach has shown better results compared to lexical matching approaches (e.g., word-alignment models).

Specifically, for two sequences $\mathcal{S}_l = (w_1, w_2, ..., w_n)$ and $\mathcal{S}_r = (w^{'}_1, w^{'}_2, ..., w^{'}_m)$, we add "$\langle S \rangle$" and "$\langle E \rangle$" to the head and tail of them respectively to form an input $\mathcal{S}^{'}_l$ and $\mathcal{S}^{'}_r$ of the network. Firstly, a word hashing layer is adopted to hash a word $w \in \mathcal{S}_l$ (or $\mathcal{S}_r$) to a one-hot vector $\mathcal{H}_h = OneHot(w, \mathcal{V})$ based on the vocabulary $\mathcal{V}$. Then by looking up the word embedding table $\mathcal{W}_e$, each word w is embedded into a k-dimensional vector $\mathcal{H}_e = LookUp(\mathcal{H}_h, \mathcal{W}_e)$. A convolutional matrix $\mathcal{W}_c$ is used to project the embedding of words within a context window of 3 words to a local contextual feature vector, and a max pooling layer follows which extracts the most salient local features to get a fix length global feature vector. With a multi-layer perceptron (MLP) to map the max pooling layer to a semantic layer, we get $\mathcal{H}_l$ and $\mathcal{H}_r$ as the distributed representation of the left and right side. Finally, the semantic similarity is computed as $cosine(\mathcal{H}_l, \mathcal{H}_r)$. To conclude, this model can be defined as $Sim(\mathcal{S}_l, \mathcal{S}_r)$.

Based on the CNN model described above, we design four features in Table 3 for the basic query graph, and four types of features for each kind of constraints, namely indicating features $\mathcal{I}_{s/t/e/i/o/a}$, count features $\mathcal{N}_{s/t/e/i/o/a}$, constraint detection features $\mathcal{V}_{s/t/e/i/o/a}$ and constraint binding features $\mathcal{P}_{s/t/e/i/o/a}$.

Given a MulCQ $\mathcal{Q}$ and a MulCG candidate $\mathcal{G}_i \in \mathcal{H}(\mathcal{Q})$, $\mathcal{F}_k(\mathcal{Q}, \mathcal{G}_i)$ represents the k^{th} feature of $\langle \mathcal{Q}, \mathcal{G}_i \rangle$. A liner scoring function is adopted to calculate the reward score of each $\langle \mathcal{Q}, \mathcal{G}_i \rangle$ as $Score(\mathcal{Q}, \mathcal{G}_i) = \sum_k \lambda_k \cdot \mathcal{F}_k(\mathcal{Q}, \mathcal{G}_i)$. A learning to rank method lambda-rank (Burges, 2010) is used to learn the each feature weight λ_k.

Features	**Description**
Basic Query Graph	$\mathcal{S}_{ent}$: Entity linking score ('EntityLinking')
	$\mathcal{S}_{pc}$: CNN score between question pattern and path ('PatChain')
	$\mathcal{S}_{qep}$: CNN score between question and entity+path ('QuesEP')
	$\mathcal{S}_{cw}$: CNN score between question and path, where the model is trained on ClueWeb ('ClueWeb')
Constraint	$\mathcal{I}_{s/t/e/i/o/a}$: **I**ndicating features for each kind of constraints, each value is 1 or 0
	$\mathcal{N}_{s/t/e/i/o/a}$: **N**umber of each kind of constraints, each value is a positive integer
	$\mathcal{V}_s$: Sum of entity linking scores for constant **V**ertex in each entity constraint.
	$\mathcal{V}_t$: Sum of CNN scores between answer type and KB type of the constant **V**ertex in each type constraint
	$\mathcal{V}_i$: Sum of reward scores of temporal clause for constant **V**ertex in each implicit temporal constraint
	$\mathcal{P}_s$: Sum of CNN scores between context pattern and binding **P**ath for each entity constraint
	$\mathcal{P}_o$: Sum of embedding similarities between superlative phrase and binding **P**ath for each ordinal constraint

Table 3: Features and their description.

5 Experiment

We introduce experiment part on these aspects. We first introduce the settings of our experiments, especially the three data sets containing question/answer (QA) pairs. On these data sets, the results of our method are given, and based on the results we analyze drawbacks.

5.1 Set Up

System Components We use the entire Freebase dump which is same as Berant et al. (2013) and host it with Virtuoso engine[11]. Besides, an entity linker (Yang and Chang, 2015), the Stanford NER (Finkel et al., 2005), and an in-house implementation of shift-reduce dependency parser (Zhang and Nivre, 2011) with Stanford dependency (De Marneffe et al., 2006) which is used in detecting temporal clause are adopted in this work.

Data Sets We evaluate our approach on three data sets.

(i) *ComplexQuestions* (CompQ): It is a new data set which includes 2100 QA pairs released by this work with the details in Section 2.

(ii) *WebQuestions* (WebQ): It contains 3778 QA pairs on training set and 2032 on test set which is released by Berant et al. (2013). The questions are collected from query log and the answers are manually labeled based on Freebase.

(iii) *SimpleQuestions* (SimpQ): Each question in *SimpleQuestions* is written by a human with reference to a knowledge base triple.

CNN Training Data To train a CNN model, we first use our system S_0 to enumerate all possible basic query graphs for each question, and pick up the ones with the F_1 score larger then 0. We then get a set of question-path pairs to train the initial CNN models. Then we use these CNN models to train a system S_1. Given S_1, we use it to answer each question to get all MulCGs and pick up the ones with the F_1 score larger then 0.5. Finally we get a set of question-path pairs as our CNN training data.

METHOD	SETTING	CNN TRAINING SOURCE			TEST (Average F_1)	
	Constraint	**CompQ-Train**	**WebQ-Train**	**SimpQ**	**CompQ-Test**	**WebQ-Test**
STAGG	√	√	-	-	36.89	-
	√	-	√	-	-	52.36
	√	√	√	-	37.42	52.35
	√	√	√	√	37.69	54.30
This Work	-	√	-	-	30.31	-
	-	-	√	-	-	50.98
	-	√	√	-	31.58	51.69
	-	√	√	√	31.42	54.20
	√	√	-	-	40.94	-
	√	-	√	-	-	52.43
	√	√	√	-	41.75	52.49
	√	√	√	√	**42.33**	**54.36**

Table 4: Average F_1 score on **CompQ-Test** and **WebQ-Test** which stand for the test sets of *ComplexQuestions* and *WebQuestions* respectively. **CompQ-Train**, **WebQ-Train** stand for the training sets of *ComplexQuestions* and *WebQuestions* respectively, and **SimpQ** represents the *SimpleQuestions*. **Constraint** means whether to add constraints or not.

5.2 Results and Analysis

We re-implement STAGG method (Yih et al., 2015) as our baseline. STAGG method considers some constraints, such as entity constraint on CVT vertex, type constraint and ordinal constraint triggered by “first” and “oldest”. To evaluate our method compared to the baseline on different settings, we design experiments shown in Table 4. The results show that our method outperforms the baseline on the test set of *ComplexQuestions* and have comparable result on the test set of *WebQuestions*.

Specifically, the **Constraint** column in Table 4 indicates using constraints or not. Through these settings, we can see how important the constraints are for answering multi-constraint questions. Besides, $\mathcal{S}_{pc}$ and $\mathcal{S}_{qep}$ are in-domain features that the CNN models they relay on vary from the training data. So we train different CNN models for $\mathcal{S}_{pc}$ and $\mathcal{S}_{qep}$ on different combinations of training resource. By these settings, we can know how does the amount of CNN training data effect on the results.

5.2.1 Results on *ComplexQuestions*

Table 4 shows that, when using the same CNN training sources, our method outperforms the STAGG method about 4.35 ± 0.30 points (40.94-36.89, 41.75-37.42 and 42.33-37.69) on *ComplexQuestions*. This result indicates that our systematic constraint solving method is more suitable for answering questions with multiple constraints than the baseline. Besides, adding constraints can bring about 10.54 ± 0.37 points’ (40.94-30.31, 41.75-31.58 and 42.33-31.42) gain on *ComplexQuestions* which tells that constraints as an important feature can help to bring a significant improvement for multi-constraint questions. Since the training set of *ComplexQuestions* contains a relative small size of QA pairs (1300), we add *WebQuestions* and *SimpleQuestions* to enlarge the CNN training data. So another improvement is gained from adding new CNN training resource. STAGG achieves a 0.8 points’ gain, our method improves with 1.39 and 1.11 points with or without constraint respectively.

5.2.2 Results on *WebQuestions* and *SimpleQuestions*

We also evaluate our method on the test set *WebQuestions*. From Table 4 we can see that, our method has comparable results with STAGG by comparing 52.43 to 52.36[12], 52.49 to 52.35, and 54.36 to 54.30. This indicates that adding constraints can get a small improvement on the *WebQuestions* because most of the questions in the *WebQuestions* are simple questions, each of which can be solved by a single KB relation. So adding more CNN training resource bring about 1.93 (54.36-52.43) point improvement for our method. Table 5 also shows the results of recent work on *WebQuestions*, and this work outperforms the others.

[11] http://virtuoso.openlinksw.com/

[12] We get a similar result with Yih (2015)’s 52.50 F_1 score.

Besides, the result also shows that our system performs stable, which not only works well on multi-constraint questions, but also simple questions. To further prove the stability of our system, we also test on the test set of *SimpleQuestions*, by using a dictionary based entity linker, our method achieves 72.78 on accuracy. This is because all the *SimpleQuestions* are single relation questions.

6 Related Work and Discussion

Knowledge-based question answering (KBQA) works with lexical features (Yao, 2015) or convolutional neural network features (Yih et al., 2015) already achieve good results on single relation questions. *WebQuestions* (Berant et al., 2013) and *SimpleQuestions* (Bordes et al., 2015) are two data sets for KBQA task. Based on our analysis, more than 84% and almost all questions in *WebQuestions* and *SimpleQuestions* are single relation questions.

Method	Average F_1
(Berant et al., 2013)	35.70
(Bordes et al., 2014b)	29.70
(Yao and Durme, 2014)	33.00
(Bao et al., 2014)	37.50
(Berant and Liang, 2014)	39.90
(Yang et al., 2014)	41.30
(Wang et al., 2014)	45.30
(Bordes et al., 2015)	39.90
(Yao, 2015)	44.30
(Berant and Liang, 2015)	49.70
(Yih et al., 2015)	52.50
(Reddy et al., 2016)	50.30
(Xu et al., 2016)	53.30
This work	**54.36**

Table 5: QA result on **WebQ**.

Previous KBQA work (Yao and Durme, 2014; Bordes et al., 2014a; Yang et al., 2014; Fader et al., 2014; Reddy et al., 2014; Dong et al., 2015; Bordes et al., 2015) testing on these two data sets do not solve multiple constraints systematically. Different methods such as specific *bridging* operator (Berant et al., 2013), semantic template $p.(p1.e1 \sqcap p2.e2)$ (Berant and Liang, 2014) are used to treat questions with multiple entities specially. Yih et al. (2015) have already done some work to handle questions with constraints, such as considering entity constraints on CVT vertice or ordinal constraints triggered by "first" or "oldest". But these methods haven't specifically evaluated their KBQA systems or presented solutions to multi-constraint questions in a systematic manner.

We propose a novel method to answer questions with multi-constraints by multiple-constraint query graphs. By evaluating it on the *ComplexQuestions* released by this work, we find our method works well on multi-constraint questions.

7 Conclusion

We release a QA data-set *ComplexQuestions* which contains multi-constraint questions, and propose a novel systematic KBQA method using multi-constraint query graph to answer multi-constraint questions. Experiments show that, compared to state-of-the-art approaches, our method obtains comparable results on the existing benchmark data-sets *WebQuestions* and *SimpleQuestions*. Furthermore, we achieve significant improvement on the newly created *ComplexQuestions* data-set. Besides, we put learning the matching between constraint expressions and semantic constraints from massive data in future work.

References

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 967–976, Baltimore, Maryland, June. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1415–1425.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October. 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, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 615–620.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *Machine Learning and Knowledge Discovery in Databases*, pages 165–180. Springer.

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

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11:23–581.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 260–269, Beijing, China, July. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165. ACM.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Jianfeng Gao, Li Deng, Michael Gamon, Xiaodong He, and Patrick Pantel. 2015. Modeling interestingness with deep neural networks, December 17. US Patent 20,150,363,688.

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

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014a. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014b. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 373–374. International World Wide Web Conferences Steering Committee.

Zhenghao Wang, Shengquan Yan, Huaming Wang, and Xuedong Huang. 2014. An overview of microsoft deep qa system on stanford webquestions benchmark. Technical report, Technical report, Microsoft Research.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2326–2336, Berlin, Germany, August. Association for Computational Linguistics.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *EMNLP*, pages 645–650.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 956–966.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *Proceedings of NAACL-HLT*, pages 66–70.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *ACL (2)*, pages 643–648. Citeseer.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China, July. Association for Computational Linguistics.

Pengcheng Yin, Nan Duan, Ben Kao, Junwei Bao, and Ming Zhou. 2015. Answering questions with complex semantic constraints on open knowledge bases. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1301–1310. ACM.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA - Short Papers*, pages 188–193.