Short Text Understanding Through Lexical-Semantic Analysis

Wen Hua [§#1], Zhongyuan Wang [§†2], Haixun Wang [‡3], Kai Zheng [#4], Xiaofang Zhou [#5]

[§] *School of Information, Renmin University of China, Beijing, China*
[1] huawen@ruc.edu.cn

[†] *Microsoft Research, Beijing, China*
[2] zhy.wang@microsoft.com

[‡] *Google Research, Mountain View, CA, U.S.A.*
[3] haixun@google.com

[#] *School of Information Technology and Electrical Engineering, University of Queensland, Brisbane, Australia*
[4] kevinz@itee.uq.edu.au [5] zxf@itee.uq.edu.au

***Abstract*—Understanding short texts is crucial to many applications, but challenges abound. First, short texts do not always observe the syntax of a written language. As a result, traditional natural language processing methods cannot be easily applied. Second, short texts usually do not contain sufficient statistical signals to support many state-of-the-art approaches for text processing such as topic modeling. Third, short texts are usually more ambiguous. We argue that knowledge is needed in order to better understand short texts. In this work, we use lexical-semantic knowledge provided by a well-known semantic network for short text understanding. Our knowledge-intensive approach disrupts traditional methods for tasks such as text segmentation, part-of-speech tagging, and concept labeling, in the sense that we focus on semantics in all these tasks. We conduct a comprehensive performance evaluation on real-life data. The results show that knowledge is indispensable for short text understanding, and our knowledge-intensive approaches are effective in harvesting semantics of short texts.**

I. Introduction

In this paper, we focus on short text understanding, which is crucial to many applications, such as web search, microblogging, ads matching, etc. Unlike documents, short texts have some unique characteristics which make them difficult to handle. First, short texts do not always observe the syntax of a written language. This means traditional NLP techniques, ranging from POS tagging to dependency parsing, cannot always apply to short texts. Second, short texts have limited context. The majority of search queries contain less than 5 words, and tweets can have no more than 140 characters. Thus, short texts usually do not possess sufficient signals to support statistical text processing techniques such as topic modeling. Because of the above reasons, short texts give rise to a significant amount of ambiguity, and new approaches must be introduced to handle them.

In the following, we use several examples to illustrate the challenges of short text understanding.

Example 1 (Ambiguity in Text Segmentation):

- "april in paris lyrics" vs. "vacation april in paris"
- "book hotel california" vs. "hotel california eagles"

A short text can often be segmented in multiple ways. We want to choose a semantic coherent one. For instance, two segmentations are possible for "april in paris lyrics", namely {april in paris lyrics} and {april paris lyrics}. The former is better because "lyrics" is semantically related to songs ("april in paris"). The Longest-Cover method for segmentation, which prefers the longest terms in a given vocabulary, ignores such knowledge and thus will lead to incorrect segmentations. Take "vacation april in paris" as an example. The Longest-Cover method segments it as {vacation april in paris}, which is obviously an incoherent segmentation.

An important application of short text understanding is to calculate semantic similarity between short texts. In our previous research [1], semantic similarity has been proven to be much more preferable than surface similarity. However, incorrect segmentation of short texts leads to incorrect semantic similarity. For example, "april in paris lyrics" and "vacation april in paris", although look quite alike, are totally different on the semantic level, as the former searches for lyrics of a song ("april in paris") and the latter vacation information of a city ("paris") during a specific time ("april"). However, when "vacation april in paris" is incorrectly segmented as {vacation april in paris}, it will have a high similarity with "april in paris lyrics". Similarly, telling the difference between "book hotel california" and "hotel california eagles" requires correct segmentation too, as the former is about booking a hotel in California while the latter searches for a song ("hotel california") performed by the Eagles Band.

Example 2 (Ambiguity in Type Detection):

- "pink$_{[e](singer)}$ songs" vs. "pink$_{[adj]}$ shoes"
- "watch$_{[v]}$ free movie" vs. "watch$_{[c]}$ omega"

We tag terms with part of speech or semantic types (e.g., verb, adjective, attribute, concept, and instance). Finding correct types requires knowledge about the terms. In Example 2, "pink" in "pink songs" refers to a famous singer and thus should be labeled as an *instance*, whereas "pink" in "pink shoes" is an *adjective*. Similarly, term "watch" is a verb in "watch free movie" and a concept (category) in "watch

omega". Traditional approaches to Part-Of-Speech tagging (POS tagging) consider only lexical features. In particular, they infer the best type for a term within specific context based on manually defined linguistic rules [2][3] or lexical and sequential probabilities learned from a labeled corpora [4][5][6][7][8][9][10]. However, surface features are insufficient to determine types of terms in short texts. In the case of "pink songs", "pink" will be incorrectly labeled as an adjective using traditional approaches, since both the probability of "pink" as an adjective and the probability of an adjective preceding a noun are relatively high. One of the limitations of state-of-the-art approaches to short text understanding [11][12] is that they do not handle type ambiguity.

Example 3 (Ambiguity in Concept Labeling):

- "hotel california eagles$_{[e](band)}$" vs. "jaguar$_{[e](brand)}$ cars"

An instance may belong to different concepts or correspond to different real-world objects in different contexts. In Example 3, for "hotel california eagles," we may recognize "eagles" to be a *band* rather than an *animal*, given we have the knowledge that a song ("hotel california") is more related to music bands than animals. Without such knowledge, we might consider "hotel california eagles" and "jaguar cars" to be similar since both "eagles" and "jaguar" belong to the category of *animal*.

In this work, we argue that external knowledge is indispensable for short text understanding, which in turn benefits many real-world applications that need to handle large amount of short texts. We harvest lexical-semantic relationships between terms (namely words and phrases) from a well-known probabilistic network and a web corpus, and propose knowledge-intensive approaches to understand short texts effectively and efficiently. Our contributions are threefold:

- We demonstrate the pervasiveness of ambiguity in short texts and the limitations of traditional approaches in handling them;
- We achieve better accuracy of short text understanding, using knowledge-intensive approaches based on lexical-semantic analysis;
- We improve the efficiency of our approaches to facilitate real-time applications.

The rest of this paper is organized as follows: in Section II, we briefly summarize related work in the literature of text processing; then we define the problem of short text understanding formally in Section III, along with a brief introduction of notations adopted in this work; our approaches and experiments are described in Section IV and Section V respectively, followed by a brief conclusion and discussion of future work in Section VI.

II. Related Work

In this section, we discuss related work in three aspects: text segmentation, POS tagging, and concept labeling.

Text Segmentation. The goal of segmentation is to divide a short text into a sequence of meaningful components. Naive approaches used in previous work [13][14][15][16][17] treat the input text as a bag-of-words. However, words on their own are often insufficient to express semantics, as many instances and concepts are composed of multiple words. Some recent approaches [11][12] use the Longest-Cover method for text segmentation, that is, it prefers the longest terms in a given vocabulary. The Longest-Cover method does not understand the semantics of a short text, and fails in cases such as "vacation april in paris" and "book hotel california", which were described in Section I. Thus, a good approach to short text segmentation must take semantics into consideration.

POS Tagging. POS tagging determines the lexical type of a word in a text. Mainstream POS tagging algorithms fall into two categories: rule-based and statistical approaches. Rule-based POS taggers assign tags to unknown words based on a large number of hand-crafted [2][3] or automatically learned [18][19][20] linguistic rules. Statistical POS taggers [21][5] avoid the cost of constructing tagging rules by learning a statistical model automatically from a corpora and then labeling untagged texts based on those learned statistical information. One thing to note is that both rule-based and statistical approaches rely on the assumption that text is correctly structured, which is not always the case for short texts. Besides, all of the aforementioned work only considers lexical features and ignores semantics. This leads to mistakes such as "pink songs" as described in Section I. Besides POS tagging, we also want to disambiguate senses. For example, "country" is a political and geographical concept in "jazz is popular in this country," but an instance of music style in "he likes jazz more than country." In this work, we propose new approaches to determine types of terms including verbs, adjectives, attributes, concepts, and instances.

Concept Labeling. Concept labeling determines the most appropriate concepts of an instance within specific context. Named Entity Recognition (NER) is a special case of concept labeling, which only focuses on named entities. Specially, it seeks to locate named entities in a text and classifies them into predefined categories using statistical models like CRF [22] and HMM [23]. However, the number of predefined categories is extremely limited. Besides, traditional approaches to NER cannot be directly applied to short texts which are informal and error-prone. Recent work attempts to link instances to concepts in a knowledgebase. For example, Song [11] developed a Bayesian Inference mechanism to conceptualize terms and short texts, and tried to eliminate instance ambiguity based on other homogeneous instances. Kim [12] noticed that related instances can also help with disambiguation. Hence, they tried to capture semantic relations between terms using LDA, and improved the accuracy of short text conceptualization by taking context semantics into consideration. Whereas other terms, such as verbs, adjectives, and attributes, can also help eliminating instance ambiguity. For example, "harry potter" is a *book* in "read harry potter", while a *movie* in "watch harry potter". Therefore, we incorporate type detection into our framework of short text understanding, and conduct instance disambiguation based on all types of context information.

III. Problem Statement

We briefly introduce some concepts and notations employed in the paper. Then we define the short text understand-

ing problem, and give an overview of our framework.

A. Preliminary Concepts

Definition 1 (vocabulary): A vocabulary is a collection of words and phrases (of a certain language).

We download lists of English verbs and adjectives from an online dictionary - YourDictionary[1], and harvest a collection of attributes , concepts, and instances from a well-known probabilistic knowledgebase - Probase [24]. Altogether, they constitute our vocabulary.

Definition 2 (term): A term t is an entry in the vocabulary.

We represent a term as a sequence of words, and denote $|t|$ as the length (number of words) of term t. Example terms are "hotel", "california" and "hotel california" etc.

Definition 3 (segmentation): A segmentation p of a short text is a sequence of terms $p = \{t_i | i = 1, ..., l\}$ such that:

1) terms cannot overlap with each other, i.e., $t_i \cap t_{i+1} = \emptyset, \forall i$;

2) every non-stopword in the short text should be covered by a term, i.e., $s - \bigcup_{i=1}^{l} t_i \subset stopwords$.

For example, a possible segmentation of "vacation april in paris" is {vacation april paris}, where only stopword "in" is ignored from the original short text. For "new york times square," although both "new york times" and "times square" are terms in our vocabulary, {new york times times square} is invalid according to our restriction because the two terms overlap with each other.

Definition 4 (type and typed-term): A term can be mapped to multiple types including verb, adjective, attribute, concept, and instance. A typed-term $\bar{t}$ refers to a term with a specific type $\bar{t}.r$.

We denote the set of possible typed-terms for a term as $\mathbb{T} = \{\bar{t}_i | i = 1, ..., m\}$, which can be obtained directly from the vocabulary. For example, we observe that term "book" appears in verb-list, concept-list as well as instance-list of our vocabulary, thus the possible typed-terms of "book" are {book$_{[v]}$,book$_{[c]}$,book$_{[e]}$}.

Definition 5 (concept vector and concept cluster vector): During concept labeling, we map a typed-term to a concept vector denoted as $\bar{t}.\vec{c} = (\langle c_1, w_1 \rangle, \langle c_2, w_2 \rangle, ..., \langle c_n, w_n \rangle)$, where c_i represents a concept in the knowledgebase, and w_i the weight of c_i. We can also map a typed-term to a concept cluster vector $\bar{t}.\vec{C} = (\langle C_1, W_1 \rangle, \langle C_2, W_2 \rangle, ..., \langle C_N, W_N \rangle)$, where C_i represents a concept cluster and W_i the weight-sum of containing concepts.

Take "disneyland" as an example. We can map it to a concept vector ($\langle$*themepark*, 0.0351$\rangle$, $\langle$*amusementpark*, 0.0336$\rangle$, $\langle$*company*, 0.0179$\rangle$, $\langle$*park*, 0.0178$\rangle$, $\langle$*bigcompany*, 0.0178$\rangle$), as well as a concept cluster vector ($\langle$ {*theme park*, *amusement park*, *park*}, 0.0865 $\rangle$, $\langle$ {*company*, *big company*}, 0.0357 $\rangle$). We describe concept clustering later in Section IV-B.

[1]http://www.yourdictionary.com/

TABLE I. SUMMARY OF NOTATIONS.

	Definition	**Example**
s	short text	book hotel california
p	segmentation	{book hotel california}
t	term	hotel,california,hotel california
$\bar{t}$	typed-term	book$_{[v]}$,book$_{[c]}$,book$_{[e]}$
$\bar{t}.r$	type	v,adj,att,c,e
$\bar{t}.\vec{c}$	concept vector	(theme park,company,park...)
$\bar{t}.\vec{C}$	concept cluster vector	({theme park,park},{company}...)

B. Problem Definition

Given a query "book disneyland hotel california", we want to know that the user is searching for hotels close to Disneyland Theme Park in California. In order to do this, we take several steps as shown in Figure 1.

1. Using a vocabulary, we detect all candidate terms that appear in a short text. For the query "book disneyland hotel california," we get {"book","disneyland","hotel carlifornia","hotel","california"}. Based on our definition, we obtain two possible segmentations: {book disneyland hotel california} and {book disneyland hotel california}. We determine the latter is better because it is more semantically coherent (see Section IV-A for more details);

2. Although "book" has multiple types, namely {book$_{[v]}$, book$_{[c]}$, book$_{[e]}$}, we recognize that it should be a verb within such a context. Analogously, we label "hotel" as a concept, "disneyland" and "california" as instances.

3. We find that "disneyland" has multiple senses, since it can be either a theme park or a company. We determine that it refers to the famous theme park within this short text, because we know that the concept *hotel* is more semantically related to the concept *theme park* than the concept *company*.

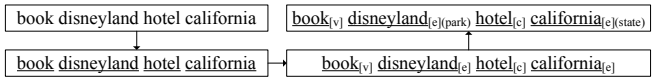


Fig. 1. Examples of steps in short text understanding.

From the above example, we observe that the basic way to understand a short text is to divide it into a collection of terms and try to understand the semantics of each term. Therefore, we formulate the task of short text understanding as follows:

Definition 6 (Short Text Understanding): For a short text s in natural language, generate a semantic interpretation of s, which is represented as a sequence of typed-terms, namely $\bar{s} = \{\bar{t}_i | i = 1, ..., l\}$.

As illustrated in Figure 1, the semantic interpretation of short text "book disneyland hotel california" is {book$_{[v]}$ disneyland$_{[e](park)}$ hotel$_{[c]}$ california$_{[e](state)}$}. Note that we can obtain semantics from attributes associated with typed-terms namely $\bar{t}.\vec{C}$. Therefore, we divide the task of short text understanding into three subtasks that correspond to the aforementioned three steps respectively:

1. **Text Segmentation**. Given a short text s, find the best segmentation p^*.

2. **Type Detection**. For term t, find the best typed-term $\bar{t}^*$ in the context.

3. **Instance Disambiguation**. For any instance $\bar{t}$ with possible senses (concept clusters) $\vec{C} = (C_1, C_2, ..., C_N)$, rank the senses with regard to the context.

C. Framework Overview

Figure 2 illustrates our framework for short text understanding. In the offline part, we acquire knowledge from the web and existing knowledgebases. Then, we pre-calculate some scores and probabilities which will be used for inferencing. In online part, we perform text segmentation, type detection, and instance disambiguation, and generate a semantically coherent interpretation of a given short text.

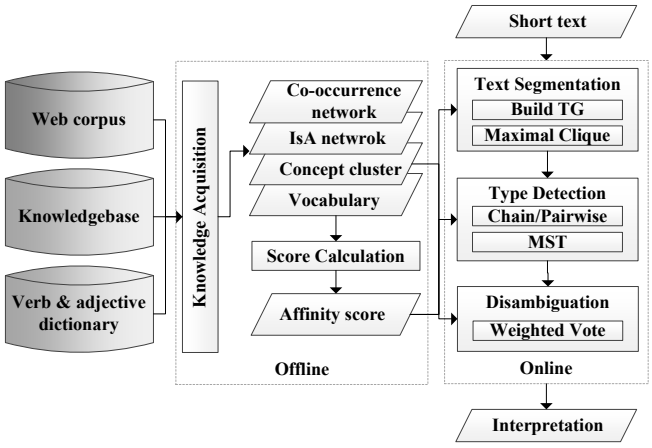


Fig. 2. Framework overview.

Q1: What knowledge to acquire: We need three types of knowledge for short text understanding: 1) A vocabulary of verbs, adjectives, attributes, concepts and instances; 2) Hypernym-hyponym relations that tell the concepts of an instance. For example, we need to know that "disneyland" refers to a theme park as well as a company. We obtain this knowledge directly from the is-a network in Probase; 3) A co-occurrence network. In order to determine the most appropriate concepts of "disneyland" in "book disneyland hotel california", we need to know that the concept *hotel* is more related to the concept *theme park* than the concept *company*. We construct a co-occurrence network for this purpose.

Q2: Why text segmentation before type detection: In traditional NLP, chunking relies on POS tagging, which in turn relies on the fact that the sentences being processed observe the grammar of a written language. This is however not the case for short texts. Our approach exploits external knowledge and infers the best segmentation based on the semantics among the terms, which reduces its dependency on POS tagging. Furthermore, in order to calculate semantic relatedness, the set of terms (namely the segmentation of a short text) should be determined first, which raises the necessity to accomplish segmentation first.

IV. Methodology

As shown in Figure 2, our methodology consists of two parts: an online inference part for short text understanding and an offline part for knowledge acquisition. We describe the details in this section.

A. Online Inference

There are basically three tasks in online processing of short texts, namely text segmentation, type detection, and instance disambiguation.

1) Text Segmentation: We organize the vocabulary in a hash index so that we can detect all possible terms in a short text efficiently. But the real question is how to obtain a coherent segmentation from the set of terms. We use two examples in Figure 3 to illustrate our approach of text segmentation. Obviously, {april in paris lyrics} is a better segmentation of "april in paris lyrics" than {april paris lyrics}, since "lyrics" is more semantically related to songs than to months or cities. Similarly, {vacation april paris} is a better segmentation of "vacation april in paris", due to higher coherence among "vacation", "april", and "paris" than that between "vacation" and "april in paris".

We segment a short text into a sequence of terms. We give the following heuristics in determining a good segmentation.

- Except for stop words, each word belongs to one and only one term;
- Terms are coherent (i.e., terms mutually reinforce each other).

We use a graph to represent candidate terms and their relationships. In this work, we define two types of relations among candidate terms:

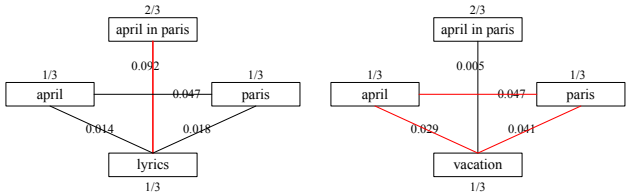
- **Mutual Exclusion** - Candidate terms that contain a same word are mutually exclusive. For example, "april in paris" and "april" in Figure 3 are mutually exclusive, because they cannot co-exist in the final segmentation;
- **Mutual Reinforcement** - Candidate terms that are related mutually reinforce each other. For example, in Figure 3, "april in paris" and "lyrics" reinforce each other because they are semantically related.

Based on these two types of relations, we construct a Term Graph (TG, as shown in Figure 3) where each node is a candidate term. We associate each node with a weight representing its coverage of words in the short text excluding stop words. We add an edge between two candidate terms when they are not mutually exclusive, and set the edge weight to reflect the strength of mutual reinforcement as follows:

$$w(x, y) = \max(\epsilon, \max_{i,j} S(\bar{x}_i, \bar{y}_j)) \quad (1)$$

where $\epsilon > 0$ is a small positive weight, $\{\bar{x}_1, \bar{x}_2, ..., \bar{x}_m\}$ is the set of typed-terms for term x, $\{\bar{y}_1, \bar{y}_2, ..., \bar{y}_n\}$ is the set of typed-terms for term y, and $S(\bar{x}, \bar{y})$ reflects semantic coherence between typed-terms $\bar{x}$ and $\bar{y}$. We call $S(\bar{x}, \bar{y})$ **Affinity Score** and we calculate affinity scores in the offline process (We describe it in detail in Section IV-B). Since a term may potentially map to multiple typed-terms, we define the edge weight between two candidate terms as the maximum Affinity Score between their corresponding typed-terms. When two

terms are not related, the edge weight is set to be slightly larger than 0 (to guarantee the feasibility of a Monte Carlo algorithm).



(a) coherent segmentation of "april in paris lyrics" is {april in paris, lyrics}.

(b) coherent segmentation of "vacation april in paris" is {vacation, april, paris}.

Fig. 3. Examples of text segmentation.

Now, the problem of finding the best segmentation is transformed into the problem of finding a sub-graph in the original TG such that the sub-graph

- is a complete graph (clique) - The selected terms are not mutually exclusive;
- has 100% word coverage excluding stop words;
- has the largest average edge weight - We choose average edge weight rather than total edge weight as the measure of a sub-graph, since the latter usually prefers shorter terms (i.e., more nodes and edges in the sub-graph), which is contradictory with the intuition of the widely-used Longest-Cover algorithm.

Given that an edge exists between each pair of nodes as long as the corresponding terms are not mutually exclusive, we can arrive at the following theorem:

Theorem 1: Finding a clique with 100% word coverage is equivalent to retrieving a Maximal Clique from the TG.

Proof: If the retrieved clique G' is not a Maximal Clique of the original TG, then we can find another node v such that after inserting v and the corresponding edges into G', the resulting sub-graph is still a clique. Due to the special structure of TG, v is not mutually exclusive with any other node in G'. In other words, they do not cover the same word. Therefore, adding v into G' will increase the total word coverage to be larger than 100%, which is obviously impossible. ■

Now we need to find a Maximal Clique with the largest average edge weight from the original TG. However, this problem is NP-hard, since it requires to enumerate every possible subset of nodes, determine whether the resulting subgraph is a Maximal Clique or not, calculate its average edge weight, and then find the one with the largest weight. Consequently, the time complexity of this problem is $O(2^{n_v} \cdot n_v^2)$, where n_v is the number of nodes in TG. Though n_v is not too large in the case of short texts, we still need to reduce the exponential time requirement into polynomial, since short text understanding is usually regarded as an online task or an underlying step of many other applications like classification or clustering. Therefore, we propose a randomized algorithm to obtain an approximate solution more efficiently, as described in Algorithm 1 and Algorithm 2.

Algorithm 1 runs as follows: First, it randomly selects an edge $e = (u, v)$ *with probability proportional to its weight*.

Algorithm 1 Maximal Clique by Monte Carlo (MaxCMC)

Input:
$G = (V, E)$; $W(E) = \{w(e)|e \in E\}$
Output:
$G' = (V', E')$; $s(G')$
$V' = \emptyset$; $E' = \emptyset$
while $E \neq \emptyset$ **do**
  randomly select $e = (u, v)$ from E with probability proportional to its weight
  $V' = V' \cup \{u, v\}$; $E' = E' \cup \{e\}$
  $V = V - \{u, v\}$; $E = E - \{e\}$
  for each $t \in V$ **do**
    if $e' = (u, t) \notin E$ or $e' = (v, t) \notin E$ **then**
      $V = V - \{t\}$
      remove edges linked to t from E: $E = E - \{e' = (t, *)\}$
    end if
  end for
end while
calculate average edge weight: $s(G') = \frac{\sum_{e \in E'} w(e)}{|E'|}$

Algorithm 2 Chunking by Maximal Clique (CMaxC)

Input:
$G = (V, E)$; $W(E) = \{w(e)|e \in E\}$
number of times to run Algorithm 1: k
Output:
$G'_{best} = (V'_{best}, E'_{best})$
$s_{max} = 0$
for $i = 1; i \leq k; i++$ **do**
  run Algorithm 1 with ¡$G'_i = (V'_i, E'_i)$,$s(G'_i)$¿ as output
  if $s(G'_i) > s_{max}$ **then**
    $G'_{best} = G'_i$; $s_{max} = s(G'_i)$
  end if
end for

In other words, the larger the edge weight, the higher the probability to be selected. After picking an edge, it removes all nodes that are disconnected (namely mutually exclusive) with the picked nodes u or v. At the same time, it removes all edges that are linked to the deleted nodes. This process is repeated until no edges can be selected. The obtained sub-graph G' is obviously a Maximal Clique of the original TG. Finally, it evaluates G' and assigns it with a score representing the average edge weight. Since edges are randomly selected according to their weights in this process, it will intuitively result in high probability to achieve a Maximal Clique with the largest average edge weight. In order to further improve the accuracy of the above algorithm, we repeat it for k times, and choose the Maximal Clique with the highest score as the final segmentation. Obviously, the larger k is, the larger accuracy we can achieve. The parameter k can be manually defined or automatically learned using existing machine learning methods. However, due to lack of large labeled dataset, we have to set k manually. The experimental results in Section V verify the effectiveness of this randomized algorithm, and we found that our framework works very well even when k is 3.

In algorithm 1, the *while* loop will be repeated for at most n_e times, since each time the algorithm removes at least one edge from the original TG. Here, n_e is the total number of edges in TG. Similarly, the *for* loop in each *while* loop will be repeated for at most n_v times. Therefore, the total time complexity of this randomized algorithm is $O(k \cdot n_e \cdot n_v)$ or $O(k \cdot$

n_v^3). In other words, the algorithm successfully reduces the time requirement of finding best segmentations from exponential to polynomial.

2) Type Detection: Recall that we can obtain the collection of typed-terms for a term directly from the vocabulary. For example, term "watch" appears in instance-list, concept-list, as well as verb-list of our vocabulary, thus the possible typed-terms of "watch" are $\{watch_{[c]}, watch_{[e]}, watch_{[v]}\}$. Analogously, the collections of possible typed-terms for "free" and "movie" are $\{free_{[adj]}, free_{[v]}\}$ and $\{movie_{[c]}, movie_{[e]}\}$ respectively, as illustrated in Figure 4. For each term derived from a short text, type detection determines the best typed-term from the set of possible typed-terms. In the case of "watch free movie", the best typed-terms for "watch", "free", and "movie" are $\text{watch}_{[v]}$, $\text{free}_{[adj]}$, and $\text{movie}_{[c]}$ respectively.

The Chain Model (CM): Traditional approaches to POS tagging consider lexical features only. Most of them adopt Markov Model [4][5][6][7][8][9][10] which learns lexical probabilities ($P(word|tag)$) as well as sequential probabilities ($P(tag_i|tag_{i-1}, ..., tag_{i-n})$) from a labeled corpora of sentences, and tags a new sentence by searching for tag sequence that maximizes the combination of lexical and sequential probabilities. However, such surface features are insufficient to determine types of terms in the case of short texts. As we have discussed in Section I, "pink" in "pink songs" will be mistakenly recognized as an adjective using traditional POS taggers, since both the probability of "pink" as an adjective and that of an adjective preceding a noun are relatively high. Whereas, "pink" is actually a famous singer and thus should be labeled as an instance, considering the fact that the concept *song* is much more semantically related with the concept *singer* than the color-describing adjective "pink". Furthermore, the sequential feature ($P(tag_i|tag_{i-1}, ..., tag_{i-n})$) fails in short texts. In other words, the type of a term does not necessarily depend on types of preceding terms only, as illustrated in the query "microsoft office download". Therefore, better approaches should be invented to improve the accuracy of type detection.

Our intuition is that although lexical features are insufficient to determine types of terms derived from a short text, errors can be reduced substantially by taking into consideration semantic relations with surrounding context. We believe that the preferred result of type detection is a sequence of typed-terms where each typed-term has a high prior score obtained by considering traditional lexical features, and typed-terms in a short text are semantically coherent with each other.

More formally, we define **Singleton Score (SS)** to measure the correctness of a typed-term considering lexical features. To simplify implementation, we calculate Singleton Scores directly based on the results of traditional POS taggers. Specifically, we first obtain the POS tagging result of a short text using an open source POS tagger - Stanford Tagger[2] [25][26]. Then we assign Singleton Scores to terms by comparing theirs types and POS tags. Specifically, terms whose types are consistent with their POS tags will get a slightly larger Singleton Score than those whose types are different from their POS tags. Since traditional POS tagging methods cannot distinguish among attributes, concepts, and instances, we treat all of them as nouns. This guarantees types and POS tags to be comparable.

$$S_{sg}(\bar{x}) = \begin{cases} 1+\theta & \bar{x}.r = pos(\bar{x}) \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

In Equation 2, $\bar{x}.r$ and $pos(\bar{x})$ are the type and POS tag of typed-term $\bar{x}$ respectively.

Based on Singleton Score which represents lexical features of typed-terms and Affinity Score which models semantic coherence between typed-terms (will be described in Section IV-B), we formulate the problem of type detection into a graph model - the Chain Model. Figure 4 (a) illustrates an example of the Chain Model.

We borrow the idea of *first order bilexical grammar*, and consider topical coherence between adjacent typed-terms, namely the preceding and the following one. In particular, we build a *chain-like* graph where nodes are typed-terms retrieved from the original short text, edges are added between each pair of typed-terms mapped from *adjacent* terms, and the edge weight between typed-terms $\bar{x}$ and $\bar{y}$ is calculated by multiplying the Affinity Score with the corresponding Singleton Scores.

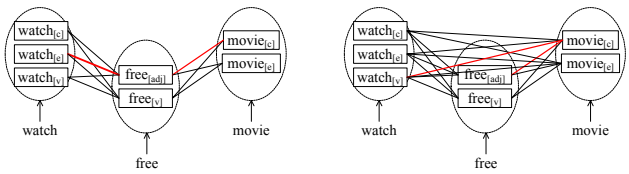
$$w(\bar{x}, \bar{y}) = S_{sg}(\bar{x}) \cdot S(\bar{x}, \bar{y}) \cdot S_{sg}(\bar{y}) \quad (3)$$

Here, $S_{sg}(\bar{x})$ is the Singleton Score of typed-term $\bar{x}$ defined in Equation 2, and $S(\bar{x}, \bar{y})$ is the Affinity Score between typed-terms $\bar{x}$ and $\bar{y}$ reflecting their semantic coherence.

Now the problem of type detection is transformed into finding the best sequence of typed-terms collectively, which maximizes the total weight of the resulting sub-graph. That is, given a sequence of terms $\{t_1, t_2, ..., t_l\}$ derived from the original short text, we need to find a corresponding sequence of typed-terms $\{\bar{t}_1^*, \bar{t}_2^*, ..., \bar{t}_l^*\}$ that maximize:

$$\sum_{i=1}^{l-1} w(\bar{t}_i, \bar{t}_{i+1}) \quad (4)$$

In the case of "watch free movie", the best sequence of typed-terms detected using the Chain Model is $\{watch_{[e]}, free_{[adj]}, movie_{[c]}\}$, as illustrated in Figure 4 (a).



(a) type detection result of "watch free movie using the *Chain Model* is $\{watch_{[e]}, free_{[adj]}, movie_{[c]}\}$.

(b) type detection result of "watch free movie using the *Pairwise Model* is $\{watch_{[v]}, free_{[adj]}, movie_{[c]}\}$.

Fig. 4. Difference between *Chain Model* and *Pairwise Model*.

The Pairwise Model (PM): In fact, terms that are most related in a short text might not always be adjacent. Therefore, if we only consider semantic relations between consecutive terms, like in the Chain Model, it will lead to mistakes. In the case of "watch free movie" in Figure 4 (a), the Chain Model incorrectly recognizes "watch" to be an instance, since "watch" is an instance of the concept *product* in our knowledgebase, and the probability of adjective "free" co-occurring with concept *product* is relatively high. However,

[2]http://nlp.stanford.edu/software/tagger.shtml

when relatedness between "watch" and "movie" is considered, "watch" should be labeled as a verb. The Pairwise Model is able to capture such cross-term relations. More specifically, the Pairwise Model adds edges between typed-terms mapped from each pair of terms rather than adjacent terms only. In Figure 4 (b), there are edges between nonadjacent terms "watch" and "movie", in addition to those between "watch" and "free" as well as those between "free" and "movie".

Like the assumption of Chain Model, the best sequence of typed-terms should be semantically coherent. One thing to note is that although cross-term relations are considered in the Pairwise model, a typed-term is not required to be related with every other typed-term. Instead, we assume that it should be semantically coherent with at least one other typed-term. Therefore, the goal of the Pairwise Model is to find the best sequence of typed-terms which guarantees that the *Maximum Spanning Tree (MST)* of the resulting sub-graph has the largest weight. In Figure 4 (b), as long as the total weight of edge between $watch_{[v]}$ and $movie_{[c]}$ and that between $free_{[adj]}$ and $movie_{[c]}$ is the largest, {$watch_{[v]}$,$free_{[adj]}$,$movie_{[c]}$} can be successfully recognized as the best sequence of typed-terms for "watch free movie", regardless of relations between $watch_{[v]}$ and $free_{[adj]}$.

We employ the Pairwise Model in our prototype system as the approach to type detection. But we present the accuracy of both models in the experiments, in order to verify the superiority of Pairwise Model over Chain Model.

3) Instance Disambiguation: Instance disambiguation is the process of eliminating inappropriate concepts behind an ambiguous instance. We accomplish this task by re-ranking concept clusters of the target instance based on context information in a short text (i.e., remaining terms), so that the most appropriate concept clusters are ranked higher and the incorrect ones lower.

Our intuition is that a concept cluster is appropriate for an instance only if it is a common sense of that instance and it achieves support from surrounding context at the same time. Take "hotel california eagles" described in Section I as an example. Although both *animal* and *music band* are popular senses of "eagles", only *music band* is semantically coherent (i.e., frequently co-occurs) with the concept *song* and thus can be kept as the final semantics of "eagles".

We have mentioned before that a term is not necessarily related with every other term in the short text. If irrelevant terms are used to disambiguate a target instance, most of its concept clusters will obtain little support, which will in turn lead to over-filtering. Therefore, we decide to use only the most related term to help with disambiguation. In the Chain Model and Pairwise Model, we have obtained the best sequence of typed-terms together with the weighted edges in-between, hence the most related term can be retrieved straightforwardly by comparing weights of edges connecting to the target instance.

Based on the aforementioned intuition, we model the process of instance disambiguation using a **Weighted-Vote** approach. Assume that the target ambiguous instance is $\bar{x}$ whose concept cluster vector is $\bar{x}.\vec{C} = (\langle C_1, W_1\rangle, ..., \langle C_N, W_N\rangle)$, and the most related typed-term used for disambiguation is $\bar{y}$. Then the importance of each concept cluster in $\bar{x}$'s disambiguated concept cluster vector $\bar{x}.\vec{C'} = (\langle C_1, W'_1\rangle, ..., \langle C_N, W'_N\rangle)$ is a combination of Self-Vote and Context-Vote. More formally,

$$\bar{x}.W'_i = V_{self}(C_i) \cdot V_{context}(C_i) \tag{5}$$

Here, Self-Vote $V_{self}(C_i)$ is defined as the original weight of concept cluster C_i, namely $V_{self}(C_i) = \bar{x}.W_i$; Context-Vote $V_{context}(C_i)$ represents the probability of C_i as a co-occurrence neighbor of the context $\bar{y}$. In other words, Context-Vote $V_{context}(C_i)$ is the weight of C_i in $\bar{y}$'s co-occur concept cluster vector. The concept cluster vector as well as the co-occur concept cluster vector of a typed-term can be obtained offline. We will describe it in detail in Section IV-B.

In the case of "hotel california eagles", the original concept cluster vector of "eagles" is (⟨*animal*,0.2379⟩,⟨*band*,0.1277⟩, ⟨*bird*,0.1101⟩,⟨*celebrity*,0.0463⟩...) and the co-occur concept cluster vector of "hotel california" is (⟨*singer*,0.0237⟩, ⟨*band*,0.0181⟩, ⟨*celebrity*,0.0137⟩, ⟨*album*,0.0132⟩...). After disambiguation using Weighted-Vote, the final concept cluster vector of "eagles" (after normalization) is (⟨band,0.4562⟩, ⟨celebrity,0.1583⟩,⟨animal,0.1317⟩,⟨singer,0.0911⟩...).

B. Offline Knowledge Acquisition

A prerequisite to short text understanding is knowledge about instance semantics as well as relatedness between terms. Therefore, we build an is-a network and a co-occurrence network between words and phrases. We also pre-calculate some essential scores for online inference.

1) Harvesting Is-A Network from Probase: Probase [24] is a huge semantic network of concepts (e.g., *country* and *president*), instances (e.g., *china* and *barac obama*) and attributes (e.g., *population* and *age*). It mainly focuses on two types of relationships, namely the *isA* relationship between instances and concepts (e.g., *china* isA *country* and *barac obama* isA *president*) and the *isAttributeOf* [27] relationship between attributes and concepts (e.g., *population* isAttributeOf *country* and *age* isAttributeOf *president*).

We use Probase[3] for two reasons. First, Probase's broad coverage of concepts makes it more general, in comparison with other knowledgebases such as Freebase [28], WordNet [29], WikiTaxonomy [30], DBPedia [31], etc. Knowledge in Probase is acquired automatically from a corpus of 1.68 billion webpages, and it contains 2.7 million concepts and 16 million instances, which results in more than 20.7 million is-a pairs[4]. Second, the probabilistic information contained in Probase enables probabilistic reasoning and thus makes short text understanding feasible. Unlike traditional knowledgebases that simply treat knowledge as black or white, Probase quantifies many measures such as popularity, typicality, basic level of categorization, etc. which are important to cognition.

2) Constructing Co-occurrence Network: We construct a co-occurrence network to model semantic relatedness. The co-occurrence network can be regarded as an indirected graph, where nodes are typed-terms and edge weight $w(\bar{x}, \bar{y})$ formulates the strength of relatedness between typed-terms $\bar{x}$ and $\bar{y}$. We observe that

[3]Probase data is publicly available at http://probase.msra.cn/dataset.aspx

[4]http://research.microsoft.com/en-us/projects/probase/statistics.aspx

- Terms of different types occurs in different contexts. Therefore, the co-occurrence network should be constructed between typed-terms instead of terms;
- Common terms (e.g., "item" and "object") which co-occur with almost every other term are meaningless in modeling semantic relatedness, thus the corresponding edge weights should be penalized.

Based on these observations, we build a co-occurrence network as follows: 1) We scan every distinct sentence from a web corpus, and obtain part-of-speech tags using Stanford POS tagger. For words tagged as verbs or adjectives, we derive their stems and get a collection of verbs and adjectives. For noun phrases, we check them in the vocabulary and determine their types (attribute, concept, instance) collectively by minimizing topical diversity. Our intuition is that the number of topics mentioned in a sentence is usually limited. For example, "population" can be an attribute of *country* as well as an instance of *geographical data*. Assume that the collection of noun phrases parsed from a sentence is {"china","population"}, then "population" should be labeled as an attribute in order to limit the topic of the sentence to be *country* only. Using this approach, we can obtain a set of attributes, concepts and instances. Take "Outlook.com is a free personal email from Microsoft" as another example. The collection of typed-terms we get after analyzing this sentence is $\{outlook_{[e]}, free_{[adj]}, personal_{[adj]}, email_{[c]}, microsoft_{[e]}\}$. 2) Given the set of typed-terms derived from a sentence, we add a co-occur edge between each pair of typed-terms. To estimate edge weight, we first calculate the frequency of two typed-terms appearing together using the following formula:

$$f_s(\bar{x}, \bar{y}) = n_s \cdot e^{-dist_s(\bar{x},\bar{y})} \tag{6}$$

Here, n_s is the number of times sentence s appears in the web corpus, and $dist_s(\bar{x}, \bar{y})$ is the distance between typed-terms $\bar{x}$ and $\bar{y}$ (i.e., number of typed-terms in-between) in that sentence. $e^{-dist_s(\bar{x},\bar{y})}$ is used to penalize long distance co-occurrence. We then aggregate frequencies among sentences, and weigh each edge by a modified tf-idf formula.

$$f(\bar{x}, \bar{y}) = \sum_s f_s(\bar{x}, \bar{y}) \tag{7}$$

$$w(\bar{x}, \bar{y}) = \frac{f(\bar{x}, \bar{y})}{\sum_{\bar{z}} f(\bar{x}, \bar{z})} \cdot \log \frac{N}{N_{nei(\bar{y})}} \tag{8}$$

In Equation 8, $\frac{f(\bar{x},\bar{y})}{\sum_{\bar{z}} f(\bar{x},\bar{z})}$ reflects the probability that humans think of typed-term $\bar{y}$ when seeing $\bar{x}$. N is the total number of typed-terms contained in the co-occurrence network, and $N_{nei(\bar{y})}$ is the number of co-occurrence neighbors of $\bar{y}$. Therefore, the *idf* part of this formula penalizes typed-terms that co-occur with almost every other typed-term.

There are some obvious drawbacks in the above approach. First, the number of typed-terms is extremely large. Recall that Probase contributes 2.7 million concepts and 16 million instances to our vocabulary. This will increase storage cost and affect the efficiency of probabilistic inference on the network. Second, concept-level co-occurrence is more useful for short text understanding, when semantic coherence is considered. Therefore, we compress the original co-occurrence network by retrieving concepts of each instance from the is-a network, and then grouping similar concepts together into concept clusters. The nodes in the reduced version of the co-occurrence network are verbs, adjectives, attributes and concept clusters, and the edge weights (i.e., $w(\bar{x}, C)$ and $w(C_1, C_2)$) are aggregated from the original network. We use the reduced network in the remaining of this work to estimate semantic relatedness.

3) Concept Clustering by K-Mediods: To represent the semantics of an instance in a more compact manner, and to reduce the size of the original co-occurrence network at the same time, we employ the K-Mediods [32] algorithm to cluster similar concepts contained in Probase (k is set as 5000 in this work). We believe that if two concepts share many instances, they are similar to each other. Therefore, we define the distance between two concepts c_1 and c_2 as

$$d(c_1, c_2) = 1 - \mathbf{cosine}(E(c_1), E(c_2)) \tag{9}$$

where $E(c)$ is the instance distribution of concept c, which can be obtained directly from Probase's is-a network. Readers can refer to [33] for more details on concept clustering.

Given a typed-term $\bar{t}$, we can determine its semantics (i.e., concept cluster vector $\bar{t}.\vec{C}$) from the is-a network and the concept clustering result.

$$\bar{t}.\vec{C} = \begin{cases} \emptyset & \bar{t}.r \in \{v, adj, att\} \\ (< C, 1 > | \bar{t} \in C) & \bar{t}.r = c \\ (< C_i, W_i > | i = 1, ..., N) & \bar{t}.r = e \end{cases} \tag{10}$$

In Equation 10, we distinguish among three circumstances: 1) verbs, adjectives, and attributes have no hypernyms in the is-a network, thus we specifically define their concept cluster vectors as empty; 2) for a concept, only the concept cluster it belongs to will be assigned with the weight 1 and all the other concept clusters will be assigned with the weight 0; 3) for an instance, we retrieve its concepts from the is-a network, and weigh each concept cluster by the summation of weights of containing concepts. More formally, $W_i = \sum_{c \in C_i} p(c|\bar{t})$ where $p(c|\bar{t})$ is the popularity score harvested by Probase. For example, the concept vector of "eagles" contained in Probase is ($\langle themepark, 0.0351\rangle$, $\langle amusementpark, 0.0336\rangle$, $\langle company, 0.0179\rangle$, $\langle park, 0.0178\rangle$, $\langle bigcompany, 0.0178\rangle$). After concept clustering, we obtain a concept cluster vector (⟨ {*theme park*, *amusement park*, *park*}, 0.0865 ⟩, ⟨ {*company*, *big company*}, 0.0357 ⟩).

4) Scoring Semantic Coherence: We define **Affinity Score (AS)** to measure semantic coherence between typed-terms. In this work, we consider two types of coherence: similarity and relatedness (co-occurrence). We believe that two typed-terms are coherent if they are semantically similar or they often co-occur on the web. Therefore, the Affinity Score between typed-terms $\bar{x}$ and $\bar{y}$ can be calculated as follows:

$$S(\bar{x}, \bar{y}) = \mathbf{max}(S_{sim}(\bar{x}, \bar{y}), S_{co}(\bar{x}, \bar{y})) \tag{11}$$

Here, $S_{sim}(\bar{x}, \bar{y})$ is the semantic similarity between typed-terms $\bar{x}$ and $\bar{y}$, which can be calculated directly as *cosine* similarity between their concept cluster vectors.

$$S_{sim}(\bar{x}, \bar{y}) = \mathbf{cosine}(\bar{x}.\vec{C}, \bar{y}.\vec{C}) \tag{12}$$

$S_{co}(\bar{x}, \bar{y})$ measures semantic relatedness between typed-terms $\bar{x}$ and $\bar{y}$. We denote the co-occur concept cluster vector of typed-term $\bar{x}$ as $\vec{C}_{co(\bar{x})}$, and the concept cluster vector of typed-term $\bar{y}$ as $\bar{y}.\vec{C}$. We observe that the larger the overlapping between

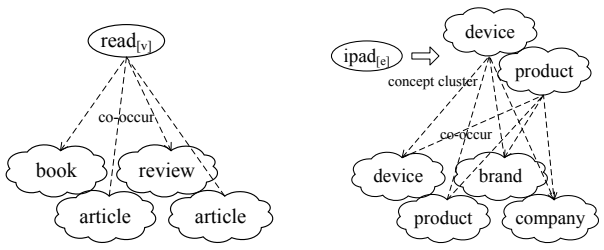
these two concept cluster vectors, the stronger the relatedness between typed-terms $\bar{x}$ and y. Therefore, we calculate $S_{co}(\bar{x}, \bar{y})$ as follows:

$$S_{co}(\bar{x}, \bar{y}) = \textbf{cosine}(\vec{C}_{co(\bar{x})}, \bar{y}.\vec{C}) \quad (13)$$

An important question is how to get the co-occur concept clusters of a typed-term (namely $\vec{C}_{co(\bar{x})}$) from the reduced co-occurrence network. Figure 5 shows two examples: 1) for verbs, adjectives, and attributes, their co-occur concept clusters can be retrieved directly; 2) for instances and concepts, we aggregate the co-occur concept cluster vectors of their concept clusters. More formally, we denote the co-occur concept clusters of a typed-term as a vector $\vec{C}_{co(\bar{x})} = (< C_1, W_1 >, < C_2, W_2 >, ..., < C_N, W_N >)$, and calculate the weight of each concept cluster as follows:

$$W_i = \begin{cases} w(\bar{x}, C_i) & \bar{x}.r \in \{v, adj, att\} \\ \sum_C w(C, \bar{x}.\vec{C}) \cdot w(C, C_i) & \bar{x}.r \in \{c, e\} \end{cases} \quad (14)$$

In Equation 14, $w(\bar{x}, C_i)$ and $w(C, C_i)$ represent edge weights between typed-terms and concept clusters and that between concept clusters respectively in the reduced co-occurrence network. As mentioned before, these information are aggregated from edge weights in the original co-occurrence network. $w(C, \bar{x}.\vec{C})$ refers to the weight of C in $\bar{x}$'s concept cluster vector defined in Equation 10.



(a) for typed-term $read_{[v]}$. (b) for typed-term $ipad_{[e]}$.

Fig. 5. Examples of retrieving co-occur concept clusters.

V. Experiment

We conducted comprehensive experiments on real-world dataset to evaluate the performance of our approach to short text understanding. All the algorithms were implemented in C#, and all the experiments were conducted on a server with 2.90GHz Intel Xeon E5-2690 CPU and 192GB memory.

A. Benchmark

One of the most notable advantages of our framework over current state-of-the-art approaches [11][12] to short text understanding is that we build a generalized framework that can recognize best segmentations, conduct type detection, and eliminate instance ambiguity explicitly based on various types of context information. Therefore, we manually picked 11 terms that have ambiguity in segmentations, types, or concepts (i.e., "april in paris", "hotel california", "watch", "book", "pink", "blue", "orange", "population", "birthday", "apple", "fox"), and randomly selected 1100 queries containing one of these terms from one day's querylog (100 queries for each term). Furthermore, in order to verify the effectiveness of our framework on general short texts, we randomly sampled another 400 queries without any restriction. We removed 22 queries containing only one word which cannot be recognized by Probase. Altogether, we obtained 1478 queries through this process.

We divided the original dataset into 5 disjoint parts, and invited 15 colleagues to label them (3 for each part). We defined three labeling tasks, namely labeling the correctness of text segmentation, type detection and concept labeling respectively. Note that different people might refer to the same topic with different expressions or in different levels which all make sense. For example, some might label "barack obama" as a'*president* while others label him as a *politician*. Besides, although we have clustered Probase's concepts into 5000 concept clusters, it is still infeasible for annotators to manually select one from thousands of concept clusters to label an instance. Therefore, we decided to run our algorithms first, provide annotators with the segmentation of each query as well as types and top-1 concept clusters of terms in that query, and then ask them to determine the correctness of provided results. In order to eliminate conflicts, final labels were based on majority vote.

B. Effectiveness of Text Segmentation

In order to incorporate context semantics into the framework of text segmentation, we construct a Term Graph (TG) between candidate terms and conduct segmentation by searching for the Maximal Clique with the largest average edge weight in TG. We propose a randomized algorithm to reduce time complexity of the naive Brute Force search. Therefore, we compare the accuracy of three models for text segmentation in this part, namely Longest-Cover, MaxCBF (Maximal Clique by Brute Force) and MaxCMC (Maximal Clique by Monte Carlo).

TABLE II. Accuracy of text segmentation.

	Longest-Cover	MaxCBF	MaxCMC
accuracy	0.954	**0.984**	0.979

From the results in Table II, we can see that the Maximal Clique approach to text segmentation achieves better performance than the Longest-Cover algorithm by taking into consideration context semantics in addition to traditional surface features like length. Furthermore, the randomized algorithm used to improve efficiency also achieves comparable accuracy to that of the Brute Force search. Therefore, we decide to adopt the randomized Maximal Clique algorithm (MaxCMC) as the approach to text segmentation in the rest of the experiments.

C. Effectiveness of Type Detection

In this part, we compare our approaches to type detection (i.e., the Chain Model and Pairwise Model) with a widely-used, non-commercial POS tagger - Stanford Tagger. Since traditional POS taggers do not distinguish among attributes, concepts and instances, we need to address this problem first in order to make a reasonable comparison. We consider two situations here: 1) if the recognized term contains multiple words or its POS tag is *noun*, then we check the frequency of that term as an attribute, a concept and an instance respectively in our knowledgebase, and choose the type with the highest frequency

as its label; 2) otherwise, we label the term according to its POS tag.

Table III demonstrates the accuracies of Stanford Tagger (ST), Chain Model (CM) and Pairwise Model (PM) for type detection. We use four kinds of accuracies to measure the effectiveness of these models:

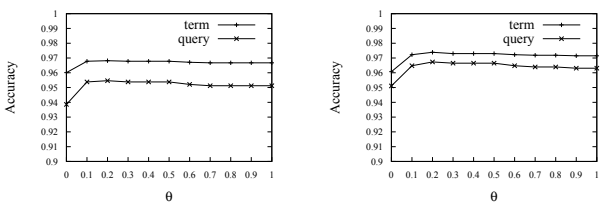
- lexical-level: the percentage of correct lexical (i.e., verb and adjective) term-type pairs;
- semantic-level: the percentage of correct semantic (i.e., attribute, concept and instance) term-type pairs;
- term-level: the percentage of correct term-type pairs;
- query-level: the percentage of queries whose term-type pairs are all correct.

From Table III, we can see that the Pairwise Model performs better than the Chain Model on all kinds of accuracy measures, which in turn provides better accuracies than the Stanford Tagger. This is consistent with our expectations. Since the Stanford Tagger only pays attention to lexical features, it will mistakenly recognize "pink" in "pink songs" as an adjective, which is actually an instance of *singer*. The Chain Model and Pairwise Model, on the contrary, take context semantics into consideration and thus can solve the above problem. Note that the Chain Model has the limitation that it only considers semantic relations between adjacent terms, which makes it incomparable with the Pairwise Model.

TABLE III. ACCURACY OF TYPE DETECTION.

	ST	CM	PM
lexical-level	0.865	0.967	**0.978**
semantic-level	0.944	0.969	**0.973**
term-level	0.932	0.968	**0.974**
query-level	0.876	0.955	**0.967**

Recall that we employ a Singleton Score to incorporate the result of traditional POS taggers in the Chain Model and Pairwise Model. We assign Singleton Score of a typed-term as $1 + \theta$ when its type is consistent with its POS tag, and 1 otherwise. In other words, the variable θ represents the amount of impact lexical features have on type detection results. Figure 6 depicts the variation of type detection accuracies on terms and queries, when θ ranges from 0 to 1.



(a) CM (b) PM

Fig. 6. Accuracy of type detection when θ increases.

From Figure 6, we can see that the accuracy of type detection increases dramatically when context semantics and lexical features are combined to estimate best types (from $\theta = 0$ to $\theta = 0.1$). However, as lexical features play an increasingly important role in the Chain Model and Pairwise Model, the accuracy decreases slightly (from $\theta = 0.2$ to $\theta = 1$). Most notably, the accuracy of type detection using Chain Model and Pairwise Model when $\theta = 0$ (namely only semantic features are considered) is larger than that of the Standford Parser depicted in Table III. This proves that context semantics are more important than lexical features for determining types of terms in short texts.

D. Effectiveness of Short Text Understanding

As we have mentioned before, one of the most notable contributions of our work is that we propose a generalized framework that can recognize best segmentations, conduct type detection, and eliminate instance ambiguity based on various types of context information. Therefore, we examine the effectiveness of short text understanding as a whole in this part. More specifically, we compare the performance of our framework with current state-of-the-art approaches to mining semantics from short texts:

- Song [11] - conduct text segmentation by Longest-Cover, and disambiguate based on similar instances;
- Kim [12] - conduct text segmentation by Longest-Cover, and disambiguate based on related instances;
- Our approach - conduct text segmentation by finding Maximal Clique, and disambiguate using various types of context information.

We consider accuracy from two perspectives to measure the effectiveness of short text understanding:

- term-level: the percentage of instances whose top-1 concept cluster is correct;
- query-level: the percentage of queries whose instances are all correctly conceptualized.

As depicted in Table IV, our approach performs dramatically better than current state-of-the-art approaches, since it takes into consideration both surface features and context semantics for text segmentation and type detection, and at the same time utilizes various context information to conduct instance disambiguation.

TABLE IV. ACCURACY OF SHORT TEXT UNDERSTANDING.

	Song	Kim	Our Approach
term-level	0.694	0.701	**0.943**
query-level	0.525	0.526	**0.890**

It is worth mentioning that the most crucial prerequisite to concept labeling (or instance disambiguation) is that the types of contextual terms should be recognized correctly. This is because the same term with various types might co-occur with different concept clusters. Therefore, we examine the correlation between type detection and concept labeling in the remaining part.

First, we compare the accuracy of concept labeling after applying Stanford Tagger with that achieved after applying Chain Model and Pairwise Model. In order to conduct a meaningful comparison, we extend Stanford Tagger to two

models, ST-AC and ST-NAC corresponding to CM-WV (Chain Model + Weighted-Vote) and PM-WV (Pairwise Model + Weighted-Vote) respectively.

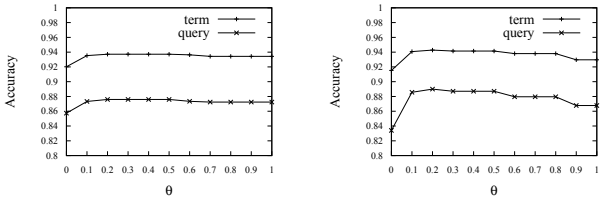
- Stanford Tagger + Adjacent Context (ST-AC) - disambiguate using most related and adjacent contextual terms after applying Stanford Tagger;
- Stanford Tagger + Nonadjacent Context (ST-NAC) - disambiguate using most related contextual terms which can be nonadjacent, after applying Stanford Tagger.

Table V demonstrates the accuracies of ST-AC, ST-NAC, CM-WV, and PM-WV for concept labeling. We can see that the performance of ST-AC and ST-NAC is incomparable with that of CM-WV and PM-WV respectively. This is mainly because the Chain Model and Pairwise Model perform better than the Stanford Tagger in determining types of terms parsed from a short text, as shown in Table III.

TABLE V. IMPACT OF TYPE DETECTION ON CONCEPT LABELING.

	ST-AC	CM-WV	ST-NAC	PM-WV
term-level	0.831	**0.937**	0.834	**0.943**
query-level	0.679	**0.876**	0.683	**0.890**

We then examine the variation of concept labeling accuracies on terms and queries when θ ranges from 0 to 1. From Figure 7, we can see a similar trend with that in Figure 6. Specifically, the accuracy of concept labeling increases dramatically, when our type detection module achieves better performance by combining context semantics and lexical features (from $\theta = 0$ to $\theta = 0.1$). However, as lexical features play an increasingly important role in type detection (from $\theta = 0.2$ to $\theta = 1$), its accuracy decreases slightly, which in turn leads to performance degradation in concept labeling.



(a) CM-WV (b) PM-WV

Fig. 7. Accuracy of concept labeling when θ increases.

E. Efficiency of Short Text Understanding

As we know, short text understanding is usually regarded as an online task or an underlying step of many other text mining applications like classification and clustering. These applications usually need to handle millions of short texts at a time, which makes the efficiency of short text understanding extremely critical. Therefore, we examine the time requirement of our framework to verify its efficiency.

Figure 8 depicts the variation of average time requirement of our framework for short text understanding (i.e., conduct text segmentation using Monte Carlo algorithm, type detection using the Pairwise Model, and instance disambiguation using Weighted-Vote algorithm) when query length (number of words) increases. Most of the queries in our dataset (1275 out of 1278) contain no more than 11 words. From Figure 8, we can see that our framework can efficiently interpret a short text within hundreds of milliseconds, and that the time requirement increases linearly with the ascending of query length.

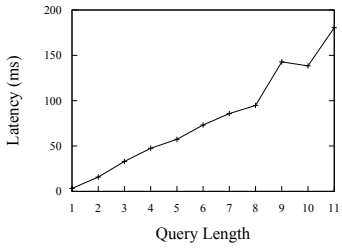


Fig. 8. Average time requirement of short text understanding when length (number of words) increases.

VI. CONCLUSION

In this work, we propose a generalized framework to understand short texts effectively and efficiently. More specifically, we divide the task of short text understanding into three subtasks: text segmentation, type detection, and concept labeling. We formulate text segmentation as a weighted Maximal Clique problem, and propose a randomized approximation algorithm to maintain accuracy and improve efficiency at the same time. We introduce a Chain Model and a Pairwise Model which combine lexical and semantic features to conduct type detection. They achieve better accuracy than traditional POS taggers on the labeled benchmark. We employ a Weighted-Vote algorithm to determine the most appropriate concepts for an instance when ambiguity is detected. The experimental results demonstrate that our proposed framework outperforms existing state-of-the-art approaches in the field of short text understanding.

We observe that the three steps of short text understanding, namely text segmentation, type detection, and concept labeling are actually related with each other. For example, the quality of text segmentation has an direct influence on that of type detection and concept labeling. Similarly, disambiguated instances can improve the accuracy of measuring semantic coherence between terms, which in turn leads to better performance of text segmentation and type detection. Therefore, a better framework for short text understanding should be one with feedbacks. We leave it as future work.

ACKNOWLEDGEMENT

This work was partially supported by the National 863 High-tech Program under Grant No. 2012AA011001, the National Key Basic Research Program (973 Program) of China under Grant No. 2014CB340403, and the National Natural Science Foundation of China under Grant No. M13210007.

REFERENCES

[1] W. Hua, Y. Song, H. Wang, and X. Zhou, "Identifying users' topical tasks in web search," in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, ser. WSDM '13. New York, NY, USA: ACM, 2013, pp. 93–102. [Online]. Available: http://doi.acm.org/10.1145/2433396.2433410

[2] S. Klein and R. F. Simmons, "A computational approach to grammatical coding of english words," *J. ACM*, vol. 10, no. 3, pp. 334–347, Jul. 1963. [Online]. Available: http://doi.acm.org/10.1145/321172.321180

[3] B. B. Greene and G. M. Rubin, *Automatic grammatical tagging of English*. Department of Linguistics, Brown University, 1971.

[4] K. W. Church, "A stochastic parts program and noun phrase parser for unrestricted text," in *Proceedings of the second conference on Applied natural language processing*, ser. ANLC '88. Stroudsburg, PA, USA: Association for Computational Linguistics, 1988, pp. 136–143. [Online]. Available: http://dx.doi.org/10.3115/974235.974260

[5] S. J. DeRose, "Grammatical category disambiguation by statistical optimization," *Comput. Linguist.*, vol. 14, no. 1, pp. 31–39, Jan. 1988. [Online]. Available: http://dl.acm.org/citation.cfm?id=49084.49087

[6] C. G. de Marcken, "Parsing the lob corpus," in *Proceedings of the 28th annual meeting on Association for Computational Linguistics*, ser. ACL '90. Stroudsburg, PA, USA: Association for Computational Linguistics, 1990, pp. 243–251. [Online]. Available: http://dx.doi.org/10.3115/981823.981854

[7] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun, "A practical part-of-speech tagger," in *Proceedings of the third conference on Applied natural language processing*, ser. ANLC '92. Stroudsburg, PA, USA: Association for Computational Linguistics, 1992, pp. 133–140. [Online]. Available: http://dx.doi.org/10.3115/974499.974523

[8] R. Weischedel, R. Schwartz, J. Palmucci, M. Meteer, and L. Ramshaw, "Coping with ambiguity and unknown words through probabilistic models," *Comput. Linguist.*, vol. 19, no. 2, pp. 361–382, Jun. 1993. [Online]. Available: http://dl.acm.org/citation.cfm?id=972470.972477

[9] B. Merialdo, "Tagging english text with a probabilistic model," *Comput. Linguist.*, vol. 20, no. 2, pp. 155–171, Jun. 1994. [Online]. Available: http://dl.acm.org/citation.cfm?id=972525.972526

[10] H. Schütze and Y. Singer, "Part-of-speech tagging using a variable memory markov model," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ser. ACL '94. Stroudsburg, PA, USA: Association for Computational Linguistics, 1994, pp. 181–187. [Online]. Available: http://dx.doi.org/10.3115/981732.981757

[11] Y. Song, H. Wang, Z. Wang, H. Li, and W. Chen, "Short text conceptualization using a probabilistic knowledgebase," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, ser. IJCAI'11. AAAI Press, 2011, pp. 2330–2336. [Online]. Available: http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-388

[12] D. Kim, H. Wang, and A. Oh, "Context-dependent conceptualization," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, ser. IJCAI'13. AAAI Press, 2013, pp. 2654–2661. [Online]. Available: http://dl.acm.org/citation.cfm?id=2540128.2540511

[13] R. Mihalcea and A. Csomai, "Wikify! linking documents to encyclopedic knowledge," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, ser. CIKM '07. New York, NY, USA: ACM, 2007, pp. 233–242. [Online]. Available: http://doi.acm.org/10.1145/1321440.1321475

[14] D. Milne and I. H. Witten, "Learning to link with wikipedia," in *Proceedings of the 17th ACM conference on Information and knowledge management*, ser. CIKM '08. New York, NY, USA: ACM, 2008, pp. 509–518. [Online]. Available: http://doi.acm.org/10.1145/1458082.1458150

[15] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti, "Collective annotation of wikipedia entities in web text," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 457–466. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557073

[16] X. Han and J. Zhao, "Named entity disambiguation by leveraging wikipedia semantic knowledge," in *Proceedings of the 18th ACM conference on Information and knowledge management*, ser. CIKM '09. New York, NY, USA: ACM, 2009, pp. 215–224. [Online]. Available: http://doi.acm.org/10.1145/1645953.1645983

[17] X. Han, L. Sun, and J. Zhao, "Collective entity linking in web text: A graph-based method," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '11. New York, NY, USA: ACM, 2011, pp. 765–774. [Online]. Available: http://doi.acm.org/10.1145/2009916.2010019

[18] E. Brill, "A simple rule-based part of speech tagger," in *Proceedings of the workshop on Speech and Natural Language*, ser. HLT '91. Stroudsburg, PA, USA: Association for Computational Linguistics, 1992, pp. 112–116. [Online]. Available: http://dx.doi.org/10.3115/1075527.1075553

[19] E. BRILL, "Some advances in transformation-based part of speech tagging," in *National Conference on Artificial Intelligence, 1994*, 1994, pp. 722–727.

[20] E. Brill, "Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging," *Comput. Linguist.*, vol. 21, no. 4, pp. 543–565, Dec. 1995. [Online]. Available: http://dl.acm.org/citation.cfm?id=218355.218367

[21] R. Garside, "The claws word-tagging system," *The Computational analysis of English: A corpus-based approach. London: Longman*, pp. 30–41, 1987.

[22] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, ser. CONLL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 188–191. [Online]. Available: http://dx.doi.org/10.3115/1119176.1119206

[23] G. Zhou and J. Su, "Named entity recognition using an hmm-based chunk tagger," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 473–480. [Online]. Available: http://dx.doi.org/10.3115/1073083.1073163

[24] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: a probabilistic taxonomy for text understanding," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '12. New York, NY, USA: ACM, 2012, pp. 481–492. [Online]. Available: http://doi.acm.org/10.1145/2213836.2213891

[25] K. Toutanova and C. D. Manning, "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger," in *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, ser. EMNLP '00. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000, pp. 63–70. [Online]. Available: http://dx.doi.org/10.3115/1117794.1117802

[26] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 173–180. [Online]. Available: http://dx.doi.org/10.3115/1073445.1073478

[27] T. Lee, Z. Wang, H. Wang, and S. won Hwang, "Attribute extraction and scoring: A probabilistic approach," in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, April 2013, pp. 194–205.

[28] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 1247–1250.

[29] F. Christiane, "Wordnet: an electronic lexical database," *Cambrige, MIT Press, Language, Speech, and Communication*, 1998.

[30] S. P. Ponzetto and M. Strube, "Deriving a large scale taxonomy from wikipedia," in *AAAI*, vol. 7, 2007, pp. 1440–1445.

[31] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The semantic web*. Springer, 2007, pp. 722–735.

[32] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[33] P. Li, H. Wang, K. Q. Zhu, Z. Wang, and X. Wu, "Computing term similarity by large probabilistic isa knowledge," in *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, ser. CIKM '13. New York, NY, USA: ACM, 2013, pp. 1401–1410. [Online]. Available: http://doi.acm.org/10.1145/2505515.2505567