IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 10, 2019, accepted September 26, 2019, date of publication September 30, 2019,
date of current version October 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2944559

A Set Space Model to Capture Structural Information of a Sentence

YANPING CHEN[1], GUORONG WANG[1], QINGHUA ZHENG[2], YONGBIN QIN[1], RUIZHANG HUANG[1], AND PING CHEN[3]

[1]College of Computer Science and Technology, Guizhou University, Guizhou 550025, China
[2]Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China
[3]Department of Computer Science, University of Massachusetts Boston, Boston, MA 02125, USA

Corresponding author: Yanping Chen (ypench@gmail.com)

This work was supported in part by the Guizhou Provincial Key Laboratory of Public Big Data, in part by the Shanxi Province Key Laboratory of Satellite and Terrestrial Network Technology Research and Development, in part by the National Natural Science Foundation of China under Grant U1836205 and Grant 91746116, and in part by the Science and Technology Projects of Guizhou Province under Grant [2017]3002 and Grant [2018]1035.

ABSTRACT The context of a sentence is composed of a limited number of words. This leads to the feature sparsity problem whereby the sentence's meaning is easily influenced by language phenomena such as polysemy, ambiguity and puns. To resolve these problems, the set space model (SSM) uses language characteristics to group features of a sentence into different sets. Afterwards, the proposed feature calculus is used to capture the structural information of the sentence. Experiments have shown that this approach to the relation recognition task is effective. However, at least three weaknesses remain. First, due to the lack of a probabilistic explanation, several aspects of SSM (e.g., filter selection) have not yet been covered. Second, the existing studies have only provided an outline of SSM, and many issues remain unclear. To understand this approach, it is necessary to discuss a suitable example in detail. Third, SSM has been applied only to the task of relation recognition. Case studies of more typical topics (e.g., named entity recognition) will help illustrate the use of SSM's methodology to manipulate features. This paper develops SSM to cover these problems. It describes a systematic and novel approach to manipulating features of a sentence. In the experimental part, two typical information extraction tasks are performed to demonstrate SSM's capabilities. Two case studies are considered, and favorable improvements are observed. All of the obtained results surpass those of compared approaches. The experiments also show the influence of sentence structural information on information extraction.

INDEX TERMS Set space model, information extraction.

I. INTRODUCTION

Information retrieval (IR) focuses on document-level information search. It ranks documents relative to a query, whereby a document or a query is modeled as a bag-of-words. Accordingly, the vector space model (VSM) has been developed to support document-level processing. Under VSM, a document is represented as a vector of a fixed length. A corpus is represented by a matrix, where each column refers to a document, and a row represents the distribution of a word among documents. VSM maps documents into a measure space. Documents are represented as scattered dots in that space. The distance between dots corresponds to similarity between documents. It is computed by using predefined measure functions, e.g., cosine similarity or the Manhattan distance.

IR is an effective approach to document-based retrieval. However, the main deficiency of an IR system is that it cannot model the level of syntactic or semantic units in a document [1]. Given a query, an IR system usually returns thousands or millions of documents, which leads to the ''information overload'' problem. To reduce the cost of searching for information, the information extraction (IE) approach provides an effective context-based information search method. It aims at extracting linguistic units with concrete concepts [1], [2], such as named entities, relations, quantifiers and events. It is widely used to support extraction of structured data from semi-structured or unstructured data.

The associate editor coordinating the review of this manuscript and approving it for publication was Xin Luo.

It is hoped that the output of an IE system can be used to construct a knowledge base automatically.

The main challenge for an IE system is that it is implemented at the sentence level, where the context is composed of a few words. Because the bag-of-words assumption is unable to capture semantic information of words and word ordering information [3], [4], it can easily lead to a significant feature sparsity problem. Furthermore, polysemy and ambiguity are common language phenomena. These phenomena have a lesser influence on term weighting in document-level processing tasks (e.g., IR) due to the presence of many comparatively frequently occurring words. On the other hand, for sentence-level tasks, the meaning of a word in a sentence is strongly influenced by the context. The influence of polysemy and ambiguity becomes increasingly important due to a limited number of words. For example, in two sentence fragments ''in Peking'' and ''Peking announces'', the word ''in'' is informative in indicating that ''Peking'' is a place name in the phrase ''in Peking'', while the verb ''announces'' reveals that ''Peking'' refers to an organization in the phrase ''Peking announces''.

In summary, several challenges arise in construction of an IE system: (a) the meaning of a word is strongly influenced by its context; (b) the impact of polysemy and ambiguity of words becomes increasingly important due to sentences' context having a short range; (c) because of the problem of heterogeneity, the performance of using an external resource is unpredictable, and (d) some techniques used to filter terms may reduce performance. For instance, in the above example ''in Peking'', the word ''in'' is often removed as a stop word.

In the IE domain, a sentence can be considered as a container that holds interrelated words. The positioning of words in a sentence is not random. It is governed by grammar rules and aims to express the sentence's meaning. Due to some linguistic phenomena (e.g., polysemy, ambiguity and puns), the meaning of a sentence is not a simple combination of words' meanings. To understand a sentence, word sense disambiguation should be considered. It also critical in implementing an IE task. In our previous study, we proposed the set space model (SSM) for the task of entity relation recognition [5]. The intuition of SSM is that for a specific IE task, some linguistic units make a sentence structurally valid. Based on the structural characteristics of sentences, such linguistic units can be used to group features into different sets. Afterwards, based on set operations, grouped features can be used to generate combined features for capturing structural information of sentences. SSM provides a formal and systematic method of generating combined features. Experiments in Chen *et al.* [5] showed that structural information of sentences was very helpful for the task of relation recognition.

In this paper, following the motivation in Chen *et al.* [5], we extend many aspects of SSM. The contributions of this paper include the following:

1) Due to the lack of a probabilistic explanation, some aspects of SSM (e.g., filter selection) cannot be supported. In this paper, we provide a *probability analysis* under the framework of set theory. It helps bridge the gap between statistical learning and logic representation.

2) In the previous paper, many aspects (e.g., feature grouping and feature calculus) of SSM are difficult to understand due to a lack of examples. In this paper, a detailed discussion with a suitable example is provided, which is helpful in constructing an SSM-based system.

3) In addition to the relation recognition task, a typical information extraction task (named entity recognition) is considered to demonstrate the methodology of SSM applied to implementing feature grouping, feature calculus, etc. Two case studies are presented that help illustrate the ability of SSM to capture structural information of sentences.

The remainder of the paper is organized as follows. Related studies are introduced in Section II, where current approaches to capturing structural information of sentences are discussed. The SSM framework is presented in Section III, where many aspects of this model are discussed. We describe experiments in Sections IV and V. Conclusions are presented in Section VI.

II. RELATED WORK

In the information extraction domain, various techniques have been proposed to make better use of sentence structural information. They can be roughly divided into four categories: n-gram feature, parse tree, sequence model and combined features.

N-gram is the simplest approach whereby consecutive words are combined into a single term. Because adjacent words often have no dependency relationship, many n-gram features are fragmental and noisy, especially if n is large. To avoid these problems, n-gram features can be derived by analytical methods. For example, Ren and Li [6] presented a wrapper method to deduce n-gram feature templates. Another way to reduce the fragmental problem is to combine n-grams with relevant information, e.g., latent topic variables, and PageRank [3], [7].

Parse tree (or dependency tree) is a fine-grained method of modeling sentence structure. It originates from a linguistic theory that provides a formal method of representing the structure of a sentence. In related studies, a kernel method based on parse trees was widely used to capture sentence structural information for relation recognition [8]. The main challenge for parse tree-based systems is that their performance is often hurt by inaccurate clunking or parsing [9]. These shortcomings can easily lead to poor performance if heterogeneous, noisy and fragmental data are being analyzed.

Sequence models, e.g., hidden Markov models (HMM) and conditional random field (CRF), are the most commonly used method of modeling dependencies between words. For a specific task, a sequence model often outputs a maximized labeling sequence. Such a sequence can effectively capture the structural information of a sentence. However, some tasks (e.g., coreference resolution) entail trying to determine the semantic relationship between two linguistic units that may

be scattered across a document. Under these conditions, it is difficult to model relationships by sequence labeling. Another shortcoming of sequence models is that they produce decisions mainly based on features around each word. Because a first-order Markov dependency is often assumed, the respective models cannot capture global features appropriately [10].

Combined features can also effectively capture structural information of sentences [11], [12]. Unlike n-grams that use consecutive words directly, combined features are generated by using syntactic or semantic rules. Combined features are often generated by greedy methods that try to identify new and additional features to improve performance. Chen *et al.* [13] proposed a feature assembly method that provided a formal method of combining features. Another regularized method is the kernel method, where kernel substitution is used to generate combined features [14].

Capturing semantic information, in addition to structural information, of sentences is also important for information extraction. One way to obtain semantic information is to use external knowledge directly. For example, Freebase [15] and patterns [16] are often used in relation recognition to guide the extraction process. In named entity recognition, various external resources have been proposed, e.g., gazetteer, lexicon, thesaurus, and WordNet [17]. Many special features, e.g., ontology [18], heuristic information (e.g., transliterated names) [19], embedded logic rules [20] and bilingual information [21] have also been explored in this field.

Neural network-based models can also effectively capture structural information of sentences. In such methods, a word is represented as a vector that can be automatically pretrained based on a large corpus [22]–[24]. The word representation (also known as word embedding) encodes syntactic or semantic information about a word. In neural network-based models, long short-term memory (LSTM) and the attention mechanism are widely used to model dependencies between words. For example, Luo *et al.* [25] modeled a highway network as a recurrent neural network (RNN), and Ali *et al.* [26] proposed a multiattention structure. To capture structural information, position embedding that embeds word positions into a distributed representation, and subsequently concatenates them with word embeddings, creating inputs of a neural network model [27], [28], is also widely used. In neural network-based models, parsing trees can be used to learn a sentence representation by an RNN [29]. For example, Kalchbrenner *et al.* [30] generated a sentence representation from a parse tree by a k-max pooling method.

Several themes can be deduced from the models discussed above. First, techniques used by document-level processing are mainly based on term frequencies, where the bag-of-words assumption is made. In sentence-level processing, linguistic roles and word positions are more important. Second, sentence-level processing often results in a sparse feature representation. It is important in such tasks to capture structural information and use external knowledge. However, linguistic functions are rarely considered in related studies using such information. As a result, performance of using external resources becomes unpredictable. For example, there is a high probability that a location name follows a preposition. However, this information can not be captured, if a sentence is represented as a bag-of-words. Third, some techniques used to filter stop words may reduce performance. In contrast, combining such words with other information may improve performance [31]. Fourth, many techniques use greedy methods to generate combined features. Because no linguistic information is considered, the resulting features are difficult to interpret, and performance is not robust.

III. SET SPACE MODEL

This section discusses SSM in greater depth, and many aspects of the model are described in more detail. Furthermore, two important topics – probability analysis and feature selection – that help extend the theory of SSM are presented. Probability analysis provides a unified representation for combining probability and logic. Feature selection entails a method of filtering noisy and inconsistent features. For ease of understanding, we first introduce the notation in Table 1.

A. DEFINITIONS

Let $\mathbf{A} = \{a_1, a_2, \cdots\}$ be a feature set that contains features extracted for a specific task (e.g., relation recognition). In this paper, we refer to the minimal granularity of features as *atomic features*. Set $\mathbf{A}$ is used to refer to an atomic feature space. Using structural characteristics of sentences, elements in $\mathbf{A}$ can be partitioned into grouped feature sets $\mathbf{X} = \{\mathbf{X}_1, \cdots, \mathbf{X}_n\}$, and $\mathbf{X}$ is called a partition of $\mathbf{A}$. In this paper, subscripts are used to distinguish grouped feature sets. For $\forall \mathbf{X}_i \in \mathbf{X}$, it is assumed that all elements in $\mathbf{X}_i$ satisfy a certain property.

Elements of $\mathbf{X}_i$ can be represented as $\mathbf{X}_i = \{x_m^i | 1 \leq m \leq |\mathbf{X}_i|\}$, where $|\mathbf{X}_i|$ is the cardinality of $\mathbf{X}_i$. A dot operation is used to represent elements of $\mathbf{X}_i$ (e.g., $\mathbf{X}_i.x_j$). If $x_m^i \in \mathbf{X}_i$, then $x_m^i \in \mathbf{X}$. Therefore, $\mathbf{X}$ can also be represented as

$$\mathbf{X} = \{x_1^1, \cdots, x_{|\mathbf{X}_1|}^1, x_1^2, \cdots, x_{|\mathbf{X}_2|}^2, x_1^3, \cdots\}$$

If there is no ambiguity, elements of $\mathbf{X}$ or $\mathbf{X}_i$ are all referred to as $\{x_1, \cdots x_n\}$. Because $\mathbf{X}_i$ is grouped according to structural characteristics of sentences, we assume that $\forall x_i \forall x_j (x_i \in \mathbf{X}_i, x_j \in \mathbf{X}_j)$, if $\mathbf{X}_i \neq \mathbf{X}_j$ then $x_i \neq x_j$.

Function "*Grouping*" is defined to represent the process of grouping a feature set: $\mathbf{X} = \mathit{Grouping}(\mathbf{A})$. Function $\mathit{Atomizing}(x_i)$ is defined to return atomic features. Therefore, $\mathbf{X}_{\text{Atom}} = \{z : z = \mathit{Atomizing}(x_i), x_i \in \mathbf{X}\}$ represents the process of generating bag-of-words features from $\mathbf{X}$. We assume that functions can be implemented on elements and sets. If a function implemented on an element, it returns the result for the element; e.g., $\mathit{Atomizing}(x_i) = x$. If a function's argument is a set, the function is individually applied to every element of the set; e.g., $\mathbf{X}_{\text{Atom}} = \mathit{Atomizing}(\mathbf{X}) = \{z : z = \mathit{Atomizing}(x_i), x_i \in \mathbf{X}\}$.

TABLE 1. Symbol table.

Symbol	Meaning	Symbol	Meaning
$\mathbf{A}$	an atomic feature set	$\mathbf{X}$	a grouped feature set
$\mathbf{X}_i$	a feature set satisfying a certain property	$\mathbf{KB}$	a knowledge base
$\mathbf{S}$	a sentence set	S_i	a sentence in $\mathbf{S}$
$\mathbf{C}$	a constant set	c_i	an object in the problem domain
$\mathbf{P}$	a predicate set	$\mathbf{F}$	a function set
$\mathbf{P}_i$	a predicate constant	$\mathbf{F}_i$	a function constant
$\mathcal{P}(\mathbf{X})$	the power set of $\mathbf{X}$	$\mathscr{P}$	the map of a feature predicate
$\mathscr{F}$	the map of a feature function	$\mathscr{T}$	a many-to-many mapping
$\mathbf{Z}$	the transformed feature set	$\mathbf{Y}$	a label set
z_i	a feature	ω_{z_i}	a configuration space of z_i
$\mathbf{Z}_i$	a subset of $\mathbf{Z}$	Ω	the configuration space of $\mathbf{Z}$ on $\mathbf{Y}$
$\mathscr{R}$	a one-to-many mapping	φ	a probability density function
λ_c	a feature weighting	Δ	the simplex on Ω
$\mathcal{F}_C$	a candidate feature set	$\widetilde{p}$	the training samples' empirical distribution
$\mathscr{R}\restriction_{\mathbf{Z}}$	the restriction of $\mathscr{R}$ to $\mathbf{Z}$	$\widehat{\mathbf{X}}$	the support of $\mathbf{X}$
$\mathbf{R}$	a relation set	$\widehat{\mathbf{Z}}$	the support of $\mathbf{Z}$
$x_{e1}\mathbf{R}x_{e2}$	relations between x_{e1} and x_{e2}	T_i	random variables over a label set

B. FEATURE GROUPING

For a specific task, experience or prior knowledge are required to partition atomic features $\mathbf{A}$ into grouped feature sets $\mathbf{X}$. To achieve good performance, it is better to utilize structural or functional linguistic units to partition sentences. For example, the entity relation recognition task tries to find semantic relationships between two named entities in a sentence. Given a sentence set $\mathbf{S} = \{S_1, S_2, \cdots, S_n\}$, each sentence $S_i \in \mathbf{S}$ can be partitioned into five parts by two named entities. In each part, grouped features can be extracted independently. Therefore, for entity relation recognition, grouped feature sets can be generated as

$$\mathbf{X} = \{\mathbf{X}_l, \mathbf{X}_f, \mathbf{X}_m, \mathbf{X}_s, \mathbf{X}_r\}$$

where $x_i^f \in \mathbf{X}_f$ and $x_i^s \in \mathbf{X}_s$ are features extracted from the first and second entity mentions. In the task of relation recognition, two named entities are manually annotated. They can precisely partition a sentence into different parts, which reduces errors caused by parsing or segmenting the entire sentence.

To take advantage of SSM, precisely partitioning atomic features to support feature calculus is very important. However, there is currently no regular method that supports feature grouping. Given a specific task, experience or domain knowledge are required to guarantee good performance. One recommended strategy for partitioning a sentence is to use linguistic units that can be identified precisely (e.g., function words) or that are relevant to a specific task. For example, some tasks (e.g., relation recognition or coreference resolution) entail trying to identify semantic relationships between two named entities. Entities can be effectively used to group features around them. In the task of named entity recognition, boundaries of named entities can be identified precisely [32]. They are also useful in feature grouping. In what follows, to demonstrate the use of the SSM methodology to group features, the task of relation recognition is given as an example.

TABLE 2. Descriptions of named entity and relation types.

Task	Type	Description
Named Entity Rec.	PER	Name of a person (or a set of people).
	ORG	Name of an organization (or a set of organizations).
	GPE	A reference to geographic or social entities.
	LOC	Names of places.
	FAC	A facility, i.e., a man-made structure.
	VEH	A reference to vehicles.
	WEA	A reference to weapons.
Relation Recognition	PHYS	A physical relation is a relation between physical locations of entities.
	PAR-WHO	The part-whole captures the ownership and other hierarchical relationships between two entities.
	PER-SOC	Personal-social relations describe the relationships between persons.
	ORG-AFF	Organization-affiliation represents a relation between a PER or ORG (e.g., founder) entities.
	ART	Agent-artifact relation exists if an agent owns or uses an artifact.
	GEN-AFF	General-affiliation: the citizenship (or origin) relation between PER (or ORG) or GEO entities.

Before the details of this example are discussed, Table 2 presents the descriptions of relation types and named entity types. Descriptions follow the definitions in the automatic content extraction (ACE) annotation guidelines [33] and are used throughout this paper.

In relation recognition, a relation has two named entities as arguments. Two named entities constitute a structured relation mention (a sentence where a relation occurred). Two named entities can be used to precisely partition a relation mention into at most five parts. Features can be extracted from each part independently. For example, consider the following two sentences:

$S_1 =$ '*Kelly arrived in Seoul from Beijing*'
$S_2 =$ '*Work with the leadership in the houses to discuss legacies*'

In this example, relation mention S_1 contains a '*PHYS*' relation between *Kelly* and *Seoul*. '*Kelly*' is a '*PER*' entity, and '*Seoul*' is a '*GPE*' entity. S_2 contains an

TABLE 3. Grouped feature set.

Set	Description and Grouped Features
$\mathbf{X}_{E1}$	Entity mentions of the first argument: { *Kelly*, *the_leadership* }
$\mathbf{X}_{E2}$	Entity mentions of the second argument: { *Seoul*, *the_houses* }
$\mathbf{X}_{T1}$	Entity types of the first argument: { *PER*}
$\mathbf{X}_{T2}$	Entity types of the second argument: {*GPE*, *ORG*}
$\mathbf{X}_{L}$	Words on the left before $\mathbf{X}_{E1}$: {*work*, *with*}
$\mathbf{X}_{M}$	Words between $\mathbf{X}_{E1}$ and $\mathbf{X}_{E2}$: {*arrived*, *in*}
$\mathbf{X}_{R}$	Words on the right after $\mathbf{X}_{E2}$: {*from*, *Beijing*, *to*, *discuss*, *legacies*}

'*ORG-AFF*' relation, where '*the_leadership*' is a *PER*' entity, and '*the_houses*' is an '*ORG*' entity.

In this example, seven grouped feature sets can be extracted. Elements of them are listed in Table 3, where spaces between words in a feature are replaced by underscores, e.g., '*the_leadership*'.

In Table 3, the grouped feature set is $\mathbf{X} = \{\mathbf{X}_{E1}, \mathbf{X}_{E2}, \mathbf{X}_{T1}, \mathbf{X}_{T2}, \mathbf{X}_{L}, \mathbf{X}_{M}, \mathbf{X}_{R}\}$. In following sections, this set is used as an example to demonstrate aspects of feature calculus, set space transformation, etc. To distinguish features belonging to different feature sets, subscripts are used. For example, in Table 3 elements of $\mathbf{X}_M$ are marked with subscript "M" (e.g., '$arrived_M$', 'in_M').

C. FEATURE CALCULUS

Studies have shown that combined features are very useful for information extraction [13]. The main reason is that high-frequency features with uniform distributions exhibit no preference for predicting a relation type. Combining such features with others leads to skewed distributions that are more helpful in identifying a specific type [31], [34], [35]. However, related studies of generating combined features are mainly based on greedy methods or personal experiences. Such approaches may generate many noisy and fragmental features. On the other hand, feature calculus provides a formal and systematic method of combining features. Based on SSM, five set operations and two logical operations are defined to support feature calculus that uses grouped features to generate combined features for capturing structural information of sentences.

1) SET OPERATION

Five feature operations, developed according to set theory, are defined to manipulate grouped features: concatenation ("||"), product ("$*$"), sum ("$+$"), implication ("$\rightarrow$") and equivalence ("$\leftrightarrow$"). In what follows, we assume $x_i \in \mathbf{X}_i$, $x_j \in \mathbf{X}_j$, and **KB** is a knowledge base.

1. **Feature concatenation** : $\mathbf{X}_i||\mathbf{X}_j = \{z : z = \langle x_i, x_j\rangle\}$

A concatenated feature is represented as an ordered feature pair $\langle x_i, x_j\rangle$. For example, let $\mathbf{Z} = \mathbf{X}_{T1}||\mathbf{X}_{T2}$; then, the result of $\mathbf{X}_{T1}||\mathbf{X}_{T2}$ is $\mathbf{Z} = \{\langle$'*PER*', '*GPE*'$\rangle$, $\langle$'*PER*', '*ORG*'$\rangle\}$, where the order of atomic features is required. Concatenated features can also be written using the underscore symbol, e.g., '*PER_ORG*'. In the information extraction domain, feature concatenation is a common way of generating combined features [11], [36].

2. **Feature product** : $\mathbf{X}_i * \mathbf{X}_j = \{z : z = x_i \wedge x_j\}$

Feature product formalizes the notion that feature $z = x_i \wedge x_j$ is confirmed until two features x_i and x_j occur at the same time. The difference between feature concatenation and feature product is that in feature product, two atomic features are interchangeable, i.e., $x_i \wedge x_j = x_j \wedge x_i$. Feature product can also be denoted by $z = \{x_i, x_j\}$. Unlike the bag-of-words representation, $z = \{x_i, x_j\}$ is a feature that combines x_i and x_j.

This operation is used to combine two features if their order is unimportant. For example, the task of coreference resolution entails trying to group entity mentions (e.g., x_{e1} and x_{e2}) referring to the same entity. Because the coreference relation between two entities is symmetric, the ordinal information of some features is not critical. For example, let $Singular(x_{e1})$ be the function used to determine the singular form of x_{e1}. Then, $z = Singular(x_{e1}) \wedge Singular(x_{e2})$ determines whether two named entities have the same singular form. In this scenario, feature z is order-insensitive.

In our example, let $\mathbf{Z} = \mathbf{X}_{E1} \wedge \mathbf{X}_M$; then, the feature product operation can generate feature set $\mathbf{Z} = \{z_1 =$ '*Kelly*' $\wedge$ '*arrived*', $z_2 =$ '*Kelly*' $\wedge$ '*in*', $z_3 =$ '*the_leadership*' $\wedge$ '*arrived*', $z_4 =$ '*the_leadership*' $\wedge$ '*in*'$\}$.

3. **Feature sum** : $\mathbf{X}_i + \mathbf{X}_j = \{z : z = x_i \vee x_j\}$

Feature sum combines two atomic features with a disjunction operation, where occurrence of one of them activates the combined feature. For example, either a '*father*' feature or a '*mother*' feature can indicate the '*parent*' feature, i.e., '*parent*' = '*father*' $\vee$ '*mother*'. Feature sum can be used to merge two homologous feature sets. For example, family names and transliterated names can be merged into a person name set.

4. **Feature implication** : $\rightarrow\mathbf{X}_i = \{z : \exists x_i (x_i \in \mathbf{X}_i \wedge \mathbf{KB} \models x_i \rightarrow z)\}$

Above, $\mathbf{KB} \models$ P means that P is satisfied under knowledge base **KB**. To induce an implied feature, a knowledge base **KB** (e.g., an ontology or a thesaurus) is required to support the operation. Feature implication can be used to deduce syntactic and semantic information of a sentence. For example, given the word '*apple*' in a sentence, according to its context, we can deduce that it may be a '*fruit*' or a '*subject*' or may have the '*noun*' part-of-speech tag, etc. In Table 3, based on $\mathbf{X}_{E1}$ and $\mathbf{X}_{T1}$, we can derive that '*Kelly*'$\rightarrow$'*PER*' and '*the_leadership*'$\rightarrow$'*PER*', etc. For simplicity, $\mathbf{KB} \models \mathbf{X}_i \rightarrow \mathbf{Z}_i$ can be abbreviated as $\rightarrow\mathbf{X}_i$.

As another example, let $\mathbf{KB}_{\text{mor}}$ be a thesaurus containing morphemes, and $\mathbf{X}_i$ be a word set. Then, morphemes can be induced by $\mathbf{KB}_{\text{mor}} \models x \rightarrow z$, where z is the implied morpheme. Hence, we can generate a morpheme set as $\mathbf{Z}_{\text{mor}} = \rightarrow\mathbf{X}_i = \{z : \exists x_i (x_i \in \mathbf{X}_i \wedge \mathbf{KB}_{\text{mor}} \models x_i \rightarrow z)\}$.

5. **Feature equivalence** : $\leftrightarrow\mathbf{X}_i = \{z : \exists x_i (x_i \in \mathbf{X}_i \wedge \mathbf{KB} \models x_i \leftrightarrow z)\}$

The equivalence operation expresses the notion that two features have the same semantic meaning (are synonymous). It means that they are interchangeable. To induce synonyms, a knowledge base (e.g., a thesaurus denoted by $\mathbf{KB}_{\text{syn}}$) should

be used to support the operation. If z is a synonym of x, then $\mathbf{KB}_{\text{syn}} \models x \leftrightarrow z$. $\mathbf{Z}_{\text{syn}}$ can be induced by $\mathbf{Z}_{\text{syn}} = \leftrightarrow\mathbf{X}_i = \{z : \exists x_i(x_i \in \mathbf{X}_i \wedge \mathbf{KB}_{\text{syn}} \models x_i \leftrightarrow z)\}$, where $\leftrightarrow\mathbf{X}_i$ is an abbreviation of $\mathbf{KB} \models \mathbf{X}_i \leftrightarrow \mathbf{Z}_i$.

The equivalence operation is implemented by comparing semantic information of features. In practice, we can use an equivalent feature with a skewed distribution to replace a feature that exhibits a weaker discriminative ability. For example, some words are rarely used. Occurrences of such words are noisy, and may reduce performance. On the other hand, frequently occurring words are often evenly distributed. Such words have no predictive ability. In information retrieval, they are removed during term selection. However, if information is being extracted at the sentence level, due to the feature sparsity problem discarding these features may have some negative impact. Therefore, instead of filtering the respective words, we can replace them by the feature equivalence operation. It is implemented by using a thesaurus or an ontology to group all synonyms into the same set. The word with the highest discriminative power can be selected to represent the synonym set by the term weighting method.

Following the above definitions, more complex operations can be defined. For example, $\mathbf{X}_{i+1}||, \cdots, ||\mathbf{X}_{i+n}$ concatenates several features and generates n-tuple features. Operation $\mathbf{X}_{i+1}*, \cdots, *\mathbf{X}_{i+n}$ generates a set feature (e.g., $\langle x_{i+1}, \cdots, x_{i+n}\rangle$). It is used as a single feature. Combined features can be assembled in a more sophisticated way, e.g., $\rightarrow(\mathbf{X}_i||\mathbf{X}_j)$. The following is an example given to illustrate the methodology used to generate sophisticated features:

$$\begin{aligned}\mathbf{Z} &= \rightarrow(\mathbf{X}_{E1})\| \rightarrow(\mathbf{X}_M)\| \rightarrow(\mathbf{X}_{E2}) \\ &= \{z : z = \rightarrow(x_1)\| \rightarrow(x_2)\| \rightarrow(x_3), \\ &\qquad x_1 \in \mathbf{X}_{E1}, x_2 \in \mathbf{X}_M, x_3 \in \mathbf{X}_{E2}\}\end{aligned}$$

If we let x_1 = '*Kelly*', x_2 = '*arrived*', and x_3 = '*Seoul*', based on external knowledge we know that '*Kelly*' is a person's name, and '*Seoul*' is a geographic entity. Then, '*PER*'_'*arrive*'_'*GPE*' (or $\langle$'*PER*', '*arrive*', '*GPE*'$\rangle$) is informative for predicting a physical relation.

2) LOGICAL CALCULUS

SSM, developed using set theory, has the ability to use logic techniques (e.g., propositional logic, first order logic or description logic). In this section, based on the logical calculus, we define two feature operators (feature function and feature predicate) to support feature calculus. Before discussing this, we introduce the background of logical calculus.

Let $\mathbf{C} = \{c_1, c_2, \cdots\}$ be a constant set. Each $c_i \in \mathbf{C}$ denotes an object in the problem domain. $\mathbf{X} = \{x_1, \cdots, x_n\}$ is a variable set, the elements of which take values from range $\mathbf{C}$. Constants and variables are terms of a logic language. Furthermore, a logic language also contains vocabularies or signatures used to refer to concepts of a logical system. They are also known as symbols. Rules are defined to determine the validity of symbols. In a logic system, predicates and functions are used to manipulate terms. A predicate is often defined to evaluate attributes or relations between terms. A function maps a tuple of objects to objects. In a logic system, all predicates and functions can be referred to as $\mathbf{P} = \{\mathrm{P}_i^n(x_1, \cdots, x_n) : 0 \le n \le \infty, 0 \le i \le n\}$ and $\mathbf{F} = \{\mathrm{F}_i^n(x_1, \cdots, x_n) : 0 \le n \le \infty, 0 \le i \le m\}$, where n is the arity and represents the number of parameters of a predicate (or a function). P_i and F_i are symbols that are also known as predicate constants and function constants.

In the knowledge representation domain, knowledge base **KB** often contains four types of symbols: variables, constants, predicates and functions. If $\exists(x_1, \cdots, x_n) \in \mathbf{X}$ (where x_i takes values from $\mathbf{C}$), such that $\mathrm{P}_i^n(x_1, \cdots, x_n) = \mathit{true}$, then P_i^n is *satisfiable*. If knowledge base $\mathbf{KB} = \mathit{true}$, and P_i^n is satisfiable, then **KB** entails P_i^n. This case is denoted by $\mathbf{KB} \models \mathrm{P}_i^n$. Otherwise, **KB** cannot entail P_i^n ($\mathbf{KB} \not\models \mathrm{P}_i^n$).

Utilizing techniques developed in knowledge representation, we define another two operations to support feature calculus; below, $1 \le s, t \le n$.

6. **Feature function** : $\mathscr{F}\{x_1, \cdots, x_n\} = \{z : \exists(\mathrm{F}_i, x_i, x_s, \cdots, x_t)(\mathrm{F}_i^m \in \mathbf{F} \wedge z = \mathrm{F}_i^m(x_s, \cdots, x_t))\}$

A feature function generates new features from tuples of features. It can take atomic features, feature sets, or a sentence as arguments. Then, a set of features is returned. For example, we can define function *TheFirstArgumentOf* (S_i) that returns the first entity mention of S_i; then, $\mathbf{X}_{E1} = \{z : z = \mathit{TheFirstArgumentOf}(S_i), S_i \in \mathbf{S}\}$ contains all first instances of named entities in **S**. To implement feature function, external knowledge (in the form of patterns or rules) is required. The operation can also be represented by a classifier that extracts designated features.

7. **Feature predicate** : $\mathscr{P}(x_1, \cdots, x_n) = \{z : z = \mathrm{P}_i, \exists(x_s, \cdots, x_t)(\mathrm{P}_i \in \mathbf{P} \wedge \mathbf{KB} \models \mathrm{P}_i^m(x_s, \cdots, x_t))\}$

In traditional logical calculus, predicate P_i represents an attribute or relation among objects. A predicate takes a true value if it is satisfied. Otherwise, it returns false. In the SSM framework, predicates are mainly used to generate features. If a predicate is satisfiable, then symbol P_i is collected as a feature.

The above process can also be formalized as $\mathscr{P}(x_1, \cdots, x_n) = \{z : z = \langle x_s, \cdots, x_t\rangle, \exists(x_s, \cdots, x_t)(\mathrm{P}_i \in \mathbf{P} \wedge \mathbf{KB} \models \mathrm{P}_i^m(x_s, \cdots, x_t))\}$. In this condition, predicate P_i is used to select features that satisfy the predicate.

Traditionally, to run a logic system, many hard constraints must be satisfied (e.g., decidability, completeness). Because NLP tasks often entail processing a large number of features, many of them are noisy and fragmental. It is difficult to apply logical calculus to an NLP learning task directly. The main reason for supporting feature calculus is that instead of logical reasoning, SSM only uses logical calculus to generate raw features that can perform weighting and filtering in later stages. Therefore, hard constraints that would apply to a traditional logic system can be relaxed.

Another way to utilize logical calculus is to generate new features by formal methods, e.g., *clausal constraints*, *inductive logic programming* and *feature description*

logics [37]–[39]. Many of these methods must be implemented in a knowledge base without contradictions. They can be used to induce new predicates from **KB** or subset of **KB**.

D. SET SPACE TRANSFORMATION

In the above discussion, based on set operations and logical calculus, seven feature operations have been defined to manipulate atomic features. In this section, to introduce probability analysis for SSM, feature calculus is redefined as mappings. They are as follows:

$$\begin{array}{ll}(a)\mathscr{T}_{\|}:\mathbf{X}\to\mathbf{X^n} & (b)\mathscr{T}_{*}:\mathbf{X}\to\mathcal{P}(\mathbf{X})\\ (c)\mathscr{T}_{+}:\mathbf{X}\to\mathcal{P}(\mathbf{X}) & (d)\mathscr{T}_{\rightarrow}:\mathbf{X}\to\mathbf{X}'\\ (e)\mathscr{T}_{\leftrightarrow}:\mathbf{X}\to\mathbf{X}'' & (f)\mathscr{F}:\mathbf{X}\to\mathbf{X}'''\\ (g)\mathscr{P}:\mathbf{X}\to\mathbf{P} & \end{array}$$

where $\mathcal{P}(\mathbf{X})$ denotes the power set of $\mathbf{X}$. To support operations $\mathscr{T}_{\rightarrow}$, $\mathscr{T}_{\leftrightarrow}$ and $\mathscr{F}$, external resources should be used. Therefore, $\mathbf{X}\subset\mathbf{X}'$, $\mathbf{X}\subset\mathbf{X}''$ and $\mathbf{X}\subset\mathbf{X}'''$ can be induced. $\mathscr{P}$ and $\mathscr{F}$ denote the results generated by implementing feature predicates and feature functions, respectively.

Let $\mathbf{Z}=\mathbf{X^n}\cup\mathcal{P}(\mathbf{X})\cup\mathbf{X}'\cup\mathbf{X}''\cup\mathbf{X}'''\cup\mathbf{P}$; then, we can define a mapping as

$$\mathscr{T}(\mathbf{X})=\mathscr{T}_{\|}(\mathbf{X})\cup\mathscr{T}_{*}(\mathbf{X})\cup\mathscr{T}_{+}(\mathbf{X})\cup\mathscr{T}_{\rightarrow}(\mathbf{X})\cup\mathscr{T}_{\leftrightarrow}(\mathbf{X})\cup\mathscr{F}(\mathbf{X})\cup\mathscr{P}(\mathbf{X})$$

where $\mathscr{T}$ is a many-to-many mapping. The mappings from (a) to (g) can be combined as

$$\mathscr{T}:\mathbf{X}\to\mathbf{Z} \tag{1}$$

Equation (1) maps the original feature set $\mathbf{X}$ into a higher-dimensional feature space $\mathbf{Z}$. Space $\mathbf{Z}=\{z_1,z_2,\cdots\}$ represents the transformed feature set. In a high-dimensional space, a more flexible hyperplane can flexibly support classification.

Feature calculus increases the number of features considerably. Many of them are fragmental and noisy. There are classifiers that claim the ability to handle arbitrary features. However, having a large number of features increases computational complexity, and processing such features is resource-intensive. Therefore, feature weighting and feature selection should be introduced to improve performance [31]. This issue will be discussed in Section III-F.

E. PROBABILITY ANALYSIS

In this section, probability analysis is developed under the framework of SSM. To analyze the data distribution of SSM, we follow the *principle of insufficient reason* used in maximum entropy [40] and random field model [41]. It is assumed that if we know nothing about probabilities of events, the best approach is to consider them equally likely [42].

Let $\mathbf{Y}$ be a label set. For every vertex $z_i\in\mathbf{Z}$, if there exists $y_j\in\mathbf{Y}$ such that z_i can be labeled by y_j, then we call (z_i,y_j) a **configuration** of z_i. A **configuration space** of z_i is composed of every configuration of z_i and is denoted by $\omega_{z_i}=\{(z_i,y_j):y_j\in\mathbf{Y}\}$. Set ω_{z_i} is a set of pairs, where z_i is labeled by every $y_j\in\mathbf{Y}$. A subset of $\mathbf{Z}$, denoted by $\mathbf{Z}_i$, can also be labeled by $y_j\in\mathbf{Y}$, which generates a configuration $(\mathbf{Z}_i,y_j)=\{(z_i,y_j):z_i\in\mathbf{Z}_i\}$. A configuration space of $\mathbf{Z}_i$ on $\mathbf{Y}$ is represented as $\omega_{\mathbf{Z}_i}=\{(\mathbf{Z}_i,y_j):y_j\in\mathbf{Y}\}=\{(z_i,y_j):z_i\in\mathbf{Z}_i,y_j\in\mathbf{Y}\}$. We denote by Ω the configuration space of $\mathbf{Z}$ on $\mathbf{Y}$ ($\Omega=\omega_{\mathbf{Z}}$):

$$\begin{aligned}\Omega&=\{(\mathbf{Z}_i,y_j):\mathbf{Z}_i\in\mathcal{P}(\mathbf{Z}),y_j\in\mathbf{Y}\}\\&=\{(z_i,y_i):z_i\in\mathbf{Z},y_j\to\mathbf{Y}\}\end{aligned} \tag{2}$$

$\mathcal{P}(\mathbf{Z})$ is the power set of $\mathbf{Z}$. Elements of Ω are sets, denoted by $\{\omega_1,\omega_2,\cdots\}$. Each element of Ω contains a configuration of $\mathbf{Z}_i$.

Another approach is to define the map between $\mathbf{Z}$ and $\mathbf{Y}$, such that

$$\mathscr{R}:\mathbf{Z}\to\mathbf{Y} \tag{3}$$

$\mathscr{R}$ is a one-to-many mapping, and can be regarded as a binary relation. Then, the *domain* and *range* of $\mathscr{R}$ are

$$\begin{aligned}dom(\mathscr{R})&=\{z:\exists y((z,y)\in\Omega),y\in\mathbf{Y}\}\\ran(\mathscr{R})&=\{y:\exists z((z,y)\in\Omega),z\in\mathbf{Z}\}\end{aligned}$$

The *restriction* of $\mathscr{R}$ to set $\mathbf{Z}_i$ is a mapping

$$\mathscr{R}\restriction_{\mathbf{Z}_i}=\{(z,y):z\in\mathbf{Z}_i,y\in\mathbf{Y},(z,y)\in\mathscr{R}\} \tag{4}$$

Therefore, $\mathscr{R}\restriction_{\mathbf{Z}_i}\subset\Omega$. The field of $\mathscr{R}$ is the set $field(\mathscr{R})=dom(\mathscr{R})\cup ran(\mathscr{R})$. We define a probabilistic model on $field(\mathscr{R}\restriction_{\mathbf{Z}_i})$ as

$$p(\omega_{\mathbf{Z}_i})=\int_{\omega\in\omega_{\mathbf{Z}_i}}\varphi(\omega)d\omega \tag{5}$$

where $\varphi:\Omega\to R$ is a probability density function.

In practice, $\mathbf{Z}$ represents the transformed feature space. Each $z_i\in\mathbf{Z}$ is a feature. Features belonging to a predicated case are denoted by $\mathbf{Z}_i\subset\mathbf{Z}$. Labeled instances are denoted by $\{(\mathbf{Z}_1,y_1),(\mathbf{Z}_2,y_2),\cdots\}$. Because we focus on categorical variables, characteristic function $f_c(z_i,y_j)\in\{0,1\}$ is used to indicate whether the respective configuration occurs.[1] Variable c ranges from 1 to $|\mathbf{Z}|\times|\mathbf{Y}|$ because vertex z_i may be labeled by all elements in $\mathbf{Y}$. The difference is that the probabilities of its occurrences are unequal. Therefore, each configuration has a parameter λ_c indicating the predictive power (feature weighting).

Using the Gibbs distribution, the distribution of $\omega=\{(\mathbf{Z}_i,y):y\in\mathbf{Y}\}$ can be computed as

$$p(y|\mathbf{Z}_i)=\frac{1}{Z}\exp\left(\sum_{z_i\in\mathbf{Z}_i}\lambda_c f_c(z_i,y)\right) \tag{6}$$

where $Z=\sum_y\exp(\sum_z\lambda_c f_c(z_i,y))$ is the partition function.

For all $\mathbf{Z}_i\subset\mathbf{Z}$, there exists $\mathbf{X}_i\subset\mathbf{X}$ such that $\mathbf{Z}_i=\mathscr{T}(\mathbf{X}_i)$. The restriction of $\mathscr{R}$ to $\mathbf{Z}_i$ can be represented as $\mathscr{R}\restriction_{\mathbf{Z}_i=\mathscr{T}(\mathbf{X}_i)}=\{(z,y):(z,y)\in\mathscr{R}\wedge\exists(x_1,\cdots,x_m)((x_1,\cdots,x_m)\in\mathbf{X}_i\wedge z=\mathscr{T}(x_1,\cdots,x_m))\}$. Given $\mathbf{Z}=\mathscr{T}(\mathbf{X})$, both $\mathscr{R}(\mathbf{Z})$ and $\mathscr{T}(\mathbf{X})$ can be compounded into $(\mathscr{R}\circ\mathscr{T})(\mathbf{X})$. If $\mathbf{X}_i\subset\mathbf{X}$, then $\mathscr{R}\restriction_{\mathscr{T}(\mathbf{X}_i)}\subset(\mathscr{R}\circ\mathscr{T})(\mathbf{X})$.

[1] f_c is used to denote features.

Given a visible feature set $\mathbf{X}_i$, by Equations (4) and (6), the probability of y given $\mathbf{X}_i$ is computed as

$$p(y|\mathbf{X}_i) = \frac{1}{Z}\exp\left(\sum_{\substack{x_1,\cdots,x_m \\ \in \mathbf{X}_i}} \lambda_c f_c\left(\mathscr{R}\restriction_{\mathscr{T}(x_1,\cdots,x_m)}\right)\right) \tag{7}$$

Estimation of λ_c utilizes the *Kullback-Leibler* divergence defined as

$$D(p\|q) = \sum_{\omega\in\Omega} p(\omega)\lg\frac{p(\omega)}{q(\omega)} \tag{8}$$

where p can be given by training samples with empirical distribution $\tilde{p}$. We can choose q_* under the following restriction:

$$\arg\min_{q_*\in\bigtriangleup} D(\tilde{p}\|q_*) \tag{9}$$

where $\bigtriangleup$ is the simplex on Ω with the Gibbs distribution $p(y|\mathbf{X}_i)$ under the maximum entropy principle.

Inference of q_* is tractable only for a restricted set of models. In NLP, most models are actually intractable. Various methods, such as *generalized iterative scaling* (GIS), *improved iterative scaling* (IIS) and *limited-memory quasi-Newton* (L-BFGS), can be used to solve the learning problem. In our experiment, L-BFGS is used. It can be treated as a black-box optimization procedure [43].

F. FEATURE SELECTION

The feature operations can generate a large number of raw features. Many feature operations are defined manually, and may influenced by subjective judgments. Because considering a large number of features will require large memory and computational resources and the results will suffer from degradation caused by noise, feature selection will help improve performance.

Let $\mathcal{F}_C$ be an atomic feature set, represented as

$$\mathcal{F}_C \subset \{f : \Omega \to \{0,1\}\} \tag{10}$$

In likelihood-driven methods, the gain of selecting $f_c \in \mathcal{F}_C$ can be computed by *Kullback-Leibler* divergence:

$$G(\lambda_c, f_c) = D(\tilde{p}\|q) - D(\tilde{p}\|q_{\lambda_c f_c}) \tag{11}$$

$G(\lambda_c, f_c)$ is the improvement observed if feature f_c is added with weight λ_c. Above, $\tilde{p}$ is the empirical distribution of training samples. The term $q_{\lambda_c f_c}$ is the model after f_c has been added. Weight λ_c is computed by Equation (9). Feature f_c that maximizes the gain can be determined by

$$\arg\max_{f_c\in\mathcal{F}_C} G(\lambda_c, f_c) \tag{12}$$

Considering Equation (12), Berger *et al.* [40] adopted an incremental approach to constructing a feature set from $\mathcal{F}_C$. In that approach, atomic feature set $\mathcal{F}_C$ is given in advance, a feature selection method is used, and no features are induced.

Della Pietra *et al.* [41] and McCallum [43] discussed a feature induction method, generating combined features from the atomic feature set by

$$\mathcal{F}_{C_{t+1}} = \{f_c | f_c = \langle f_i, f_j\rangle; i \neq j; f_i, f_j \in \mathcal{F}_{C_t}\} \tag{13}$$

In other words, combined features are induced by combining two or more atomic features. However, in this method atomic features being combined are selected randomly, and no structural characteristics of the sentence are considered. This approach may lead to overfitting. Furthermore, because the resulting features are generated by randomly combining atomic features, the output is uninterpretable.

As discussed in Sections III-C.1 and III-C.2, in our SSM model, feature calculus provides a guided feature induction process. In what follows, we discuss how feature selection is used to delete redundant or inconsistent feature operations.

The restriction of $\mathscr{R}$ to $\mathbf{Z}$ is referred as $\mathscr{R}\restriction_{\mathbf{Z}}$. Feature selection is formalized as the problem of finding a subset of $\mathbf{Z}$, referred to as $\hat{\mathbf{Z}}$, such that $\mathscr{R}\restriction_{\hat{\mathbf{Z}}}$ is used to replace $\mathscr{R}\restriction_{\mathbf{Z}}$. This can be done by Equation (12).

After $\hat{\mathbf{Z}}$ has been selected, for each $\hat{\mathbf{Z}} \subset \mathbf{Z}$, there exists $\hat{\mathbf{X}}$ such that $\hat{\mathbf{X}} \subset \mathbf{X}$ and $\hat{\mathbf{Z}} = \mathscr{T}(\hat{\mathbf{X}})$. Let $\hat{\mathbf{X}}$ be the *support* of $\hat{\mathbf{Z}} = \mathscr{T}(\mathbf{X})$, denoted by $\hat{\mathbf{X}} = supp(\mathscr{T})$. $\hat{\mathbf{X}}$ is the smallest subset of $\mathbf{X}$ such that if $\exists\mathbf{X}' \subset \mathbf{X}$, $\mathscr{T}(\mathbf{X}') = \mathscr{T}(\hat{\mathbf{X}})$, then $\hat{\mathbf{X}} \subset \mathbf{X}'$. $\mathscr{R}\restriction_{\mathscr{T}(\hat{\mathbf{X}})}$ is represented as

$$\mathscr{R}\restriction_{\mathscr{T}(\hat{\mathbf{X}})} = \{(z,y) : \exists(x_i,\cdots,x_m)((x_i,\cdots,x_m) \in \hat{\mathbf{X}} \wedge z = \mathscr{T}(x_i,\cdots,x_m)), y \in \mathbf{Y}\}$$

To select features in SSM, it is better to find $\hat{\mathbf{X}}$ such that $\hat{\mathbf{Z}} = \mathscr{T}(\hat{\mathbf{X}})$), which reduces $\mathbf{X}$ and $\mathbf{Z}$ into $\hat{\mathbf{X}}$ and $\hat{\mathbf{Z}}$, respectively. Let $\bar{\mathbf{Z}} = \mathbf{Z} - \hat{\mathbf{Z}} = \{z : z \notin \hat{\mathbf{Z}}, z \in \mathbf{Z}\}$. Based on feature operations ($\to$, $\leftrightarrow$, $\mathscr{F}$ and $\mathscr{P}$), we define four sets as

$$\hat{\mathbf{KB}}' = \{u : \exists x_i(x_i \in \mathbf{X} \wedge z_j \in \bar{\mathbf{Z}} \wedge u = x_i \to z_j)\}$$
$$\hat{\mathbf{KB}}'' = \{u : \exists x_i(x_i \in \mathbf{X} \wedge z_j \in \bar{\mathbf{Z}} \wedge u = x_i \leftrightarrow z_j)\}$$
$$\hat{\mathbf{F}} = \{u : \exists(\mathrm{F}_i, x_s, \cdots, x_t)((x_s,\cdots,x_t) \in \mathbf{X} \wedge \mathrm{F}_i(x_s,\cdots,x_t) \in \bar{\mathbf{Z}} \wedge u = \mathrm{F}_i)\}$$
$$\hat{\mathbf{P}} = \{u : u = \mathrm{P}_i, \exists(x_s,\cdots,x_t)((x_s,\cdots,x_t) \in \mathbf{X} \wedge \mathbf{KB} \models \mathrm{P}_i(x_s,\cdots,x_t) \wedge \mathrm{P}_i \in \bar{\mathbf{Z}})\}$$

where elements of $\hat{\mathbf{F}}$ and $\hat{\mathbf{P}}$ are function and predicate constants. Elements of $\hat{\mathbf{KB}}'$ and $\hat{\mathbf{KB}}''$ are predicates.

Consider $\mathbf{KB}$, $\mathbf{F}$ and $\mathbf{P}$ being used and subtract $\hat{\mathbf{KB}}' \cup \hat{\mathbf{KB}}''$, $\hat{\mathbf{F}}$ and $\hat{\mathbf{P}}$, respectively. The mapping $\mathscr{T}$ from $\mathbf{X}$ to $\mathbf{Z}$ can be redefined as $\hat{\mathscr{T}}$. Because $\hat{\mathscr{T}}(\mathbf{X}) = \hat{\mathbf{Z}}$ and $\hat{\mathbf{Z}} \subset \mathbf{Z}$, it holds that $\mathscr{T}$ is an *extension* of $\hat{\mathscr{T}}$, $dom(\hat{\mathscr{T}}) \subset dom(\mathscr{T})$ and $\hat{\mathscr{T}} = \mathscr{T}$ for all $x \in dom(\hat{\mathscr{T}})$. Therefore, we can use any feature operations, and subsequently filter them if necessary. For example, in Section III-C.2 the constraint on $\mathscr{P}$, $\mathbf{KB} \models \mathrm{P}_i^n$, can be replaced by $\mathrm{P}_i^n = true$, which simplifies the problem of inference and enables more available features.

$\hat{\mathbf{Z}}$ is a subset of $\mathbf{Z}$. To select features from $\mathbf{Z}$, a feature (or several features) in $\mathbf{Z}$ that satisfies a predefined evaluation function (e.g., the function in Equation (12)) is chosen each

time. One reasonable stopping criterion entails evaluation on test data withheld from evaluation data. Determining the size of $\hat{\mathbf{Z}}$ based on $\mathbf{Z}$ is also an open issue. In this field, many methods have been proposed; examples include wrappers and variable ranking [31].

IV. CASE STUDY 1: RELATION RECOGNITION

The task of relation recognition entails recognizing relationships between two named entities. In relation recognition, the arguments (two named entities) are supposed to be known and provided by the annotated corpus. This task is often approached as a classification problem. Methods for relation recognition can be roughly divided into feature-based methods (e.g., Kambhatla [11]) and kernel-based methods (e.g., Zhang *et al.* [44]). One important characteristic of relation recognition is that two named entities constitute a structured relation mention. Based on the SSM approach and using the annotated named entities, we can segment a sentence precisely and group features into different sets. Therefore, better performance is expected to result from using feature calculus. In this section, relation recognition is performed to show the flexibility of SSM in capturing structural information of sentences.

A. EXPERIMENTAL SETTING

The ACE 2005 corpus[2] is used in our experiments. It contains documents collected from broadcasts, newswires and online blogs. The corpus is annotated with three languages: Chinese, English and Arabic. After documents with incorrect annotations have been filtered out, the remainder contains 628 Chinese documents. The corpus defines 6 relation types and 18 relation subtypes. There are 9,244 Chinese relation mentions. They are manually annotated, and referred to as positive instances.

The result of performing the task should be a recognition of relationships between any two pairs of named entities in a sentence. If two named entities have no predefined relationship, they are considered a negative instance. The methods of Chen *et al.* [12] and Kambhatla [11] are used to generate negative instances for training a classifier. As a result, a large number of relation instances are generated. There are 93,283 Chinese negative instances in total. Finally, there are 7 relation types to be used in the evaluation (all negative instances are labeled ''Negative'').

We use the traditional metrics (precision, recall, and F-score) to evaluate performance. F-score is computed as

$$\frac{2 \times (Precision \times Recall)}{Precision + Recall} \tag{14}$$

Fivefold cross-validation is used for evaluation. All relation instances are randomly partitioned into five groups. Performance metrics are reported for 6 positive relation types. The results of five runs are averaged; this approach is also known as ''macro average'' and entails the following calculation:

$$B_{macro} = \frac{1}{q}\sum_{i=1}^{q} B_i \tag{15}$$

where q represents the number of relation types, and B corresponds to P, R or F, respectively. In our experiments, the resulting performance is referred to as ''Total''.

B. FEATURE CALCULUS

Let $\mathbf{R}$ represent a relation set, and x_{e1} and x_{e2} denote two named entities in a sentence. If a predefined relation type exists between x_{e1} and x_{e2}, it is represented as $(x_{e1}, x_{e2}) \in \mathbf{R}$ (or, more concisely, $x_{e1}\mathbf{R}x_{e2}$). Based on $\mathbf{R}$, we can define two entity sets: $\mathbf{X}_{E1} = \{x : \exists x_{e2}(x\mathbf{R}x_{e2})\}$ and $\mathbf{X}_{E2} = \{x : \exists x_{e1}(x_{e1}\mathbf{R}x)\}$. The total entity set is represented as $\mathbf{X}_{E*} = \mathbf{X}_{E1} \cup \mathbf{X}_{E2}$. An element of $\mathbf{X}_{E*}$ is denoted by x_{e*}.

To illustrate the method of generating combined features, we consider the original feature space $\mathbf{X}$ given by

$$\mathbf{X} = \{\mathbf{X}_{E1}, \mathbf{X}_{E2}\}$$

In what follows, based on $\mathbf{X}$, new feature sets can be induced step-by-step.

Two functions $RightPosOf(x_{e*})$ and $LeftPosOf(x_{e*})$ are defined to obtain POS tags of x_{e*}. They correspond to words located on both sides of two entity mentions. There are four feature sets in total, denoted by[3] $\mathbf{Z}_{RP1}$, $\mathbf{Z}_{LP1}$, $\mathbf{Z}_{RP2}$ and $\mathbf{Z}_{LP2}$, where $\mathbf{Z}_{RP1} = RightPosOf(\mathbf{X}_{E1})$ represents all right POS tags of $\mathbf{X}_{E1}$. All of these features are extracted by a POS tagger[4].

Another three functions $TypeOf(x_{e*})$, $SubTypeOf(x_{e*})$ and $HeadOf(x_{e*})$ are defined to obtain types, subtypes and heads of x_{e*}. As a result, six feature sets are extracted: $\mathbf{Z}_{T1}$, $\mathbf{Z}_{T2}$, $\mathbf{Z}_{S1}$, $\mathbf{Z}_{S2}$, $\mathbf{Z}_{H1}$ and $\mathbf{Z}_{H2}$ (e.g., $\mathbf{Z}_{T1} = TypeOf(\mathbf{X}_{E1})$). These features are manually annotated in the ACE corpus.

The position structure of x_{e1} and x_{e2} is extracted by function $PositionOf(x_{e1}, x_{e1})$. Because two entity mentions can be nested, they may have four coarse structures; e.g., x_{e1} may be nested in x_{e2} or x_{e1} may be in front of x_{e2}.

Given the functions discussed above, based on the original feature space $\mathbf{X}$, eight feature sets can be generated as follows:

$$\begin{aligned}
\mathbf{Z}_1 &= \{z : z = RightPosOf(x_{e1})||TypeOf(x_{e1})\}\\
\mathbf{Z}_2 &= \{z : z = LeftPosOf(x_{e1})||TypeOf(x_{e1}))\}\\
\mathbf{Z}_3 &= \{z : z = RightPosOf(x_{e2})||TypeOf(x_{e2}))\}\\
\mathbf{Z}_4 &= \{z : z = LeftPosOf(x_{e2})||TypeOf(x_{e2})\}\\
\mathbf{Z}_5 &= \{z : z = TypeOf(x_{e1})||TypeOf(x_{e2}))\}\\
\mathbf{Z}_6 &= \{z : z = SubTypeOf(x_{e1})||SubTypeOf(x_{e2})\}\\
\mathbf{Z}_7 &= \{z : z = HeadOf(x_{e1}) * HeadOf(x_{e2})\}\\
\mathbf{Z}_8 &= \{z : z = PositionBetween(x_{e1}, x_{e2})\}
\end{aligned}$$

[2]https://www.ldc.upenn.edu/collaborations/past-projects/ace

[3]Subscripts R, M and L stand for right, left and middle, and indices 1 and 2 correspond to the first and the second named entities. ''Bin'', ''POS'', ''Type'', ''Subtype'' and ''Head'' are denoted by ''B'', ''P'', ''T'', ''S'' and ''H'', respectively.

[4]http://ictclas.org/

In relation recognition, utilizing two entity mentions, a sentence can be segmented into five bins (or fewer, including two entity mentions). Each bin can be used to extract n-gram features independently. The bins are represented as

$$\mathbf{Z}_{bin} = \{\mathbf{Z}_{BL}, \mathbf{Z}_{BM}, \mathbf{Z}_{BR}, \mathbf{Z}_{B1}, \mathbf{Z}_{B2}\}$$

Because many n-gram features are fragmented and noisy, they can be filtered by predicate $\mathrm{P}_{\mathrm{word}}(z)$ that determines whether an n-gram feature is a word. This process can be represented as

$$\mathbf{Z}_{\mathrm{word}} = \{z : \mathrm{P}_{\mathrm{word}}(z), z \in \mathbf{Z}_{bin}\}$$

It can also be formalized as $\mathbf{KB}_{\mathrm{lex}} \models \mathbf{Z}_{bin} \rightarrow \mathbf{Z}_{\mathrm{word}}$ (or $\mathbf{Z}_{\mathrm{word}} = \rightarrow \mathbf{Z}_{bin}$). For Chinese relation recognition, $\mathbf{Z}_{\mathrm{word}}$ contains every word in a sentence. The resulting features are the same as Omni-word features proposed by Chen *et al.* [12].

Then, the transformed feature set for the task of relation recognition is given by

$$\begin{aligned}\mathbf{Z}_{\mathrm{SSM}} = \bigcup \mathbf{Z}_i|_{1\le i\le 8} &\cup \mathbf{Z}_{\mathrm{word}} \cup \mathbf{Z}_{RP1} \cup \mathbf{Z}_{LP1} \\ &\cup \mathbf{Z}_{RP2} \cup \mathbf{Z}_{LP2} \cup \mathbf{Z}_{T1} \cup \mathbf{Z}_{T2} \cup \mathbf{Z}_{S1} \\ &\cup \mathbf{Z}_{S2} \cup \mathbf{Z}_{H1} \cup \mathbf{Z}_{H2}\end{aligned}$$

To show the flexibility of feature calculus, we present an example of generating sophisticated features. Let $\mathbf{KB}_{\mathrm{syn}}$ and $\mathbf{KB}_{\mathrm{mor}}$ be a synonym set and a morpheme set. $\mathbf{KB}_{\mathrm{syn}} \models x \leftrightarrow z$ represents that x is a synonym of z. $\mathbf{KB}_{\mathrm{mor}} \models x \rightarrow z$ means that z is an implied morpheme of x. The implication and equivalence operations are defined as

$$\begin{aligned}\mathscr{T}_{\rightarrow}(\mathbf{X}_i) &= \{z : \exists x_i (x_i \in \mathbf{X}_i \wedge \mathbf{KB}_{\mathrm{mor}} \models x_i \rightarrow z)\} \\ \mathscr{T}_{\leftrightarrow}(\mathbf{X}_i) &= \{z : \exists x_i (x_i \in \mathbf{X}_i \wedge \mathbf{KB}_{\mathrm{syn}} \models x_i \leftrightarrow z)\}\end{aligned}$$

Based on feature operations $\mathscr{T}_{\rightarrow}(\mathbf{X}_i)$ and $\mathscr{T}_{\leftrightarrow}(\mathbf{X}_i)$, a composite mapping can be defined as $(\mathscr{T}_{\rightarrow} \circ \mathscr{T}_{\leftrightarrow})(\mathbf{X}_i)$. It returns a set of synonyms.

C. CAPTURING STRUCTURAL INFORMATION

In this experiment, we consider two models proposed by Che *et al.* [45] and Zhang *et al.* [46] for comparison. They are also feature-based models. Features proposed by the cited studies are formalized by SSM as follows:

$$\begin{aligned}\mathbf{Z}_1 &= \{z : z = \mathit{TypeOf}(x_{e1})\} \cup \{z : z = \mathit{TypeOf}(x_{e2})\} \\ \mathbf{Z}_2 &= \{z : z = \mathit{SubTypeOf}(x_{e1})\} \cup \{z : z = \mathit{SubTypeOf}(x_{e2})\} \\ \mathbf{Z}_3 &= \{z : z = \mathit{PositionBetween}(x_{e1}, x_{e2})\} \\ \mathbf{Z}_4 &= \{z : z = \mathit{TheLeftTwoWordsOf}(x_{e1})\} \cup \{z : z = \mathit{TheLeftTwoWordsOf}(x_{e2})\} \\ \mathbf{Z}_5 &= \{z : z = \mathit{TheRightTwoWordsOf}(x_{e1})\} \cup \{z : z = \mathit{TheRightTwoWordsOf}(x_{e2})\} \\ \mathbf{Z}_6 &= \{\mathit{TheLeftTwoPosOf}(x_{e1})\} \cup \{z : z = \mathit{TheLeftTwoPosOf}(x_{e2})\} \\ \mathbf{Z}_7 &= \{z : z = \mathit{TheRightTwoPosOf}(x_{e1})\} \cup \{z : z = \mathit{TheRightTwoPosOf}(x_{e2})\} \\ \mathbf{Z}_8 &= \{z : z = \mathit{TheUni\text{-}InternalNgramOf}(x_{e1}, x_{e2})\} \\ \mathbf{Z}_9 &= \{z : z = \mathit{TheBi\text{-}InternalNgramOf}(x_{e1}, x_{e2})\} \\ \mathbf{Z}_{10} &= \{z : z = \mathit{TheExternalNgramOf}(x_{e1}, x_{e2})\}\end{aligned}$$

In what follows, $\mathbf{Z}_{\mathrm{chen}}$ and $\mathbf{Z}_{\mathrm{zhang}}$ denote models proposed by Chen *et al.* [45] and Zhang *et al.* [46], respectively. The features used by these models are as follows:

$$\begin{aligned}\mathbf{Z}_{\mathrm{chen}} &= \mathbf{Z}_1 \cup \mathbf{Z}_2 \cup \mathbf{Z}_3 \cup \mathbf{Z}_4 \cup \mathbf{Z}_5 \cup \mathbf{Z}_6 \cup \mathbf{Z}_7 \\ \mathbf{Z}_{\mathrm{zhang}} &= \mathbf{Z}_1 \cup \mathbf{Z}_2 \cup \mathbf{Z}_3 \cup \mathbf{Z}_8 \cup \mathbf{Z}_9 \cup \mathbf{Z}_{10}\end{aligned}$$

To illustrate the performance of using features without structural information, we generate bag-of-words features as

$$\mathbf{Z}_{\mathrm{Atom}} = \bigcup \mathit{Atomizing}(z_i)|_{z_i \in \mathbf{Z}_{\mathrm{SSM}}}$$

Using neural network-based modes for entity relation recognition is currently very popular. Such models can extract high-order semantic features from raw inputs automatically. For neural network-based models, position embedding is a widely used method of capturing structural information of sentences (e.g., Zeng *et al.* [47], Santos *et al.* [48], Wang *et al.* [27] and Huang *et al.* [28]). For every word in a sentence, position embedding is generated by mapping its distances from two named entities into a vector. Each position embedding is concatenated with the respective word's embedding, and is subsequently used as input to a deep neural network.

For a comparison with a neural network-based method, we consider a CNN+Attention neural network model in our experiment. It is referred to as $\mathbf{Z}_{\mathrm{NN}}$. It contains an embedding layer, a convolutional layer, a max-pooling layer, an attention layer, a fully connected layer and a softmax layer. Word embeddings are initialized by the BERT approach [24], where position embeddings were encoded. In the neural network-based model, we divide the same data into training data, development data and testing data in proportions of 6:2:2.

The result is given in Table 4. The upper part of the table ("Positive Ins.") shows the performance of using only positive relation instances. The lower part of the table ("Pos., Neg. Ins.") illustrates the performance of using both positive and negative instances.

As shown in Table 4, if only positive instances are used, $\mathbf{Z}_{\mathrm{SSM}}$, $\mathbf{Z}_{\mathrm{chen}}$ and $\mathbf{Z}_{\mathrm{zhang}}$ perform similarly and attain values above 92%. $\mathbf{Z}_{\mathrm{SSM}}$ outperforms $\mathbf{Z}_{\mathrm{chen}}$ and $\mathbf{Z}_{\mathrm{zhang}}$ but shows only a slight improvement. $\mathbf{Z}_{\mathrm{Atom}}$ only uses atomic features and exhibits lower performance. The $\mathbf{Z}_{\mathrm{NN}}$ model exhibits the worst performance. The reason is that the number of positive instances is small, and is not sufficient for training a neural network. Furthermore, $\mathbf{Z}_{\mathrm{NN}}$ only uses lexical features of sentences and does not use features such as entity types, POS tags, heads, etc. In relation recognition, these features are annotated manually and are very informative in this task.

In relation recognition, two named entities that have no predefined relationship are considered a negative instance.

TABLE 4. Capturing structural information.

Data		$\mathbf{Z}_{\text{Chen}}$			$\mathbf{Z}_{\text{Zhang}}$			$\mathbf{Z}_{\text{Atom}}$			$\mathbf{Z}_{\text{NN}}$			$\mathbf{Z}_{\text{SSM}}$		
	Type	**P**(%)	**R**(%)	**F**(%)	**P**(%)	**R**(%)	**F**(%)	**P**(%)	**R**(%)	**F**(%)	**P**(%)	**R**(%)	**F**(%)	**P**(%)	**R**(%)	**F**(%)
Positive Ins.	PHYS	86.88	86.55	86.72	86.29	87.37	86.82	82.14	84.67	83.39	70.38	62.03	65.95	86.99	88.25	87.62
	PAR-WHO	91.06	92.91	91.98	90.49	92.64	91.56	88.78	90.58	89.67	77.29	84.35	80.67	91.57	94.23	92.88
	PER-SOC	98.34	98.49	98.41	98.50	99.24	98.87	96.12	89.75	92.83	85.27	74.32	79.42	98.94	98.79	98.86
	ORG-AFF	94.29	93.42	93.85	96.59	95.21	95.90	94.07	94.85	94.45	93.35	75.47	83.46	96.04	94.98	95.51
	ART	94.32	92.23	93.26	95.36	91.28	93.27	90.23	84.94	87.51	88.17	57.75	69.79	96.45	90.49	93.37
	GEN-AFF	88.30	87.98	88.14	89.54	88.56	89.05	86.65	85.20	85.92	90.94	70.12	79.18	89.95	88.77	89.35
	Total	92.20	91.93	92.06	92.80	92.38	92.59	89.67	88.33	88.99	86.29	74.84	79.67	93.32	92.58	92.95
Pos., Neg. Ins.	PHYS	68.91	59.73	63.99	72.65	65.26	68.76	79.87	74.05	76.85	74.53	80.34	77.32	85.27	82.91	84.07
	PAR-WHO	89.45	87.72	88.57	79.81	77.99	78.89	89.31	88.64	88.97	84.45	85.00	84.72	92.61	91.59	92.10
	PER-SOC	90.19	70.63	79.22	75.40	56.32	64.48	91.13	77.40	83.71	91.35	64.19	75.40	99.01	90.51	94.57
	ORG-AFF	91.37	84.15	87.61	85.35	85.07	85.21	93.49	90.45	91.94	83.08	90.65	86.70	95.78	93.93	94.85
	ART	83.52	70.68	76.56	80.87	70.36	75.25	88.42	73.85	80.48	77.54	75.35	76.43	93.95	86.21	89.91
	GEN-AFF	84.02	83.93	83.98	83.28	82.93	83.11	86.92	82.72	84.77	83.17	79.76	81.43	89.92	88.55	89.23
	Total	84.57	76.14	80.14	79.56	72.99	76.13	88.19	81.19	84.54	82.35	79.22	80.33	92.76	88.95	90.81

Note that many relation types are asymmetric. According to the approach to generating negative instances, for every entity pair (e.g., [A,B]), if it is a positive instance, there exists a negative instance (e.g, [B,A]). Because such instances are located in the same sentence, they have the same context. Therefore, negative instances have a powerful impact on relation recognition. The results in the lower part of Table 4 show the influence of negative instances.

The addition of negative instances considerably reduces the performance of $\mathbf{Z}_{\text{chen}}$ and $\mathbf{Z}_{\text{zhang}}$. However, $\mathbf{Z}_{\text{SSM}}$ still offers a robust performance. This result indicates that SSM has a strong ability to capture structural information of sentences.

One interesting finding is that for the $\mathbf{Z}_{\text{NN}}$ model, the addition of negative instances results in no apparent decline of performance. Instead, it even exceeds the performance with only positive instances. The reason is that a large number of negative instances are useful in learning better word representations needed for the task. Because many manually annotated features (e.g., entity types, and heads) are still unused, the $\mathbf{Z}_{\text{NN}}$ model may be potentially extended in our future research.

D. USING EXTERNAL KNOWLEDGE

Compared with Chinese, English has a richer morphology. In this section, the ACE 2005 English corpus is used to demonstrate the ability of SSM to use external knowledge. This corpus contains 506 English documents annotated with 6,583 English positive relation instances. We use the approach discussed in Section IV-A to generate 86,777 negative instances. Using the same settings, the model proposed by Kambhatla [11] is considered for comparison, where the used features are as follows:

$$\mathbf{Z}_1 = \{z : z = \mathit{TypeOf}(x_{e1})\} \cup \{z : z = \mathit{TypeOf}(x_{e2})\}$$
$$\mathbf{Z}_2 = \{z : z = \mathit{SubTypeOf}(x_{e1})\} \cup \{z : z = \mathit{SubTypeOf}(x_{e2})\}$$
$$\mathbf{Z}_3 = \{z : z = \mathit{MentionLevelOf}(x_{e1})\} \cup \{z : z = \mathit{MentionLevelOf}(x_{e2})\}$$
$$\mathbf{Z}_4 = \{z : z = \mathit{WordsInMention}(x_{e1})\} \cup \{z : z = \mathit{WordsInMention}(x_{e2})\}$$
$$\mathbf{Z}_5 = \{z : z = \mathit{WordsBetween}(x_{e1}, x_{e2})\}$$
$$\mathbf{Z}_6 = \{z : z = \mathit{PositionBetween}(x_{e1}, x_{e2})\}$$
$$\mathbf{Z}_7 = \{z : z = \mathit{ParsingDependencyOf}(x_{e1}) \cup \{z : z = \mathit{ParsingDependencyOf}(x_{e2})$$

In $\mathbf{Z}_7$, to obtain the parsing dependency information of x_{e1} and x_{e2}, the Stanford parser[5] is used to parse every relation mention. For example, 'NNP_E1_dep' means that the word to the left of the first entity ('E1') is a noun phrase ('NNP'). This comparison model is referred to as $\mathbf{Z}_{\text{Kambhatla}}$. It contains the following features:

$$\mathbf{Z}_{\text{Kambhatla}} = \bigcup \mathbf{Z}_i|_{1 \le i \le 7}$$

Another two external knowledge resources are WordNet [49] and a word root list that contains 986 roots. They are represented as

$$\mathbf{Z}_{\text{wn}} = \{z : \exists z_i \in \mathbf{Z}_{bin}, z \text{ is a synonym for } z_i.\}$$
$$\mathbf{Z}_{\text{root}} = \{z : \exists z_i \in \mathbf{Z}_{bin}, z \text{ is a root of } z_i.\}$$

In this experiment, features are added gradually to show the influence of external knowledge. To illustrate the ability of SSM to use external knowledge, in the lower part of Table 5 ("Atomic Features"), all features in $\mathbf{Z}_{\text{SSM}}$, $\mathbf{Z}_{\text{SSM}} \cup \mathbf{Z}_{\text{wn}}$ and $\mathbf{Z}_{\text{SSM}} \cup \mathbf{Z}_{\text{root}}$ are used as atomic and are also added gradually.

In Table 5, the results shown for $\mathbf{Z}_{\text{Kambhatla}}$ represent the performance of using features proposed in Kambhatla [11]. These results are used as our baseline. Because $\mathbf{Z}_{\text{Kambhatla}}$ is mainly used for comparison, in the lower part of Table 5, this model's change into atomic features is not considered.

$\mathbf{Z}_{\text{SSM}}$ is the model using manipulated feature sets. Because manipulated features can effectively capture the structural information in a sentence, performance improves considerably. In the lower part of Table 5 that shows results obtained if $\mathbf{Z}_{\text{SSM}}$ is used in the form of atomic features, its performance

[5]http://nlp.stanford.edu/software/lex-parser.shtml

TABLE 5. Using external knowledge.

		$\mathbf{Z}_{\text{Kambhatla}}$			$\mathbf{Z}_{\text{SSM}}$			$\mathbf{Z}_{\text{SSM}} \cup \mathbf{Z}_{\text{wn}}$			$\mathbf{Z}_{\text{SSM}} \cup \mathbf{Z}_{\text{root}}$		
	Type	**P**(%)	**R**(%)	**F**(%)	**P**(%)	**R**(%)	**F**(%)	**P**(%)	**R**(%)	**F**(%)	**P**(%)	**R**(%)	**F**(%)
Grouped Features	PHYS	58.87	38.14	46.29	68.15	54.93	60.83	66.19	58.19	61.93	73.54	64.99	69.00
	PAR-WHO	62.33	60.71	61.51	83.72	81.55	82.62	85.23	81.15	83.14	86.73	83.44	85.06
	PER-SOC	76.04	51.16	61.16	84.60	73.13	78.45	84.80	76.31	80.33	93.98	87.79	90.78
	ORG-AFF	72.86	69.21	70.99	83.77	80.30	82.00	85.25	80.89	83.01	91.47	86.06	88.68
	ART	67.97	48.22	56.41	79.84	65.53	71.98	79.10	65.53	71.68	88.86	75.27	81.50
	GEN-AFF	50.84	41.55	45.73	60.39	59.55	59.97	63.79	56.37	59.85	67.49	63.85	65.62
	Total	64.82	51.50	57.39	76.74	69.16	72.76	77.39	69.74	73.37	83.68	76.90	80.14
Atomic Features	PHYS	×	×	×	61.45	40.82	49.05	59.17	45.91	51.71	66.51	47.22	55.23
	PAR-WHO	×	×	×	79.36	74.37	76.78	76.64	76.57	76.60	77.51	74.57	76.01
	PER-SOC	×	×	×	79.93	62.75	70.31	83.19	71.30	76.79	82.50	61.05	70.17
	ORG-AFF	×	×	×	79.67	73.41	76.41	81.33	74.32	77.67	80.22	72.28	76.04
	ART	×	×	×	62.31	44.97	52.24	67.99	50.23	57.77	72.48	50.07	59.23
	GEN-AFF	×	×	×	59.47	43.90	50.51	60.00	44.87	51.34	58.43	38.36	46.32
	Total	×	×	×	70.36	56.70	62.80	71.38	60.53	65.51	72.94	57.26	64.16

decreases significantly. The reason is the same as that for the results in Table 4.

In the $\mathbf{Z}_{\text{SSM}} \cup \mathbf{Z}_{\text{wn}}$ model, morphology functions of WordNet are used to generate the top three frequent hypernyms and synonyms of synsets in $\mathbf{Z}_{BL}$, $\mathbf{Z}_{BM}$ and $\mathbf{Z}_{BR}$. This model's performance improves consistently with expectations.

In the $\mathbf{Z}_{\text{SSM}} \cup \mathbf{Z}_{\text{root}}$ model, for every word in five bins ($\mathbf{Z}_{BL}$, $\mathbf{Z}_{BM}$, $\mathbf{Z}_{BR}$, $\mathbf{Z}_{B1}$ and $\mathbf{Z}_{B2}$), if a root is observed, it is added to the bin's information. Because roots of words are informative, this approach also improves performance considerably.

As the results show, if $\mathbf{Z}_{\text{wn}}$ and $\mathbf{Z}_{\text{root}}$ are added as atomic features, performance does not change significantly. This finding indicates that using external knowledge can lead to good performance only if it is modeled appropriately. The advantage of SSM is that it provides a systematic and novel approach to manipulating features in a sentence.

V. CASE STUDY 2: NAMED ENTITY RECOGNITION

In this case study, the task of named entity recognition is considered to show the methodology of SSM.

An entity is defined as an object or a set of objects. An entity mention (or, more concisely, a mention) is an occurrence of a named entity in a sentence. The main challenge of named entity recognition is the feature sparsity problem whereby only a few words can be used to perform the task. It currently remains a challenging task for many languages, especially Chinese, because of the lack of delimiting words and capitalization. Furthermore, in Chinese almost every single character can be a monosyllabic morpheme or a word [50]. As a result, it is difficult to distinguish between the linguistic roles of characters.

Named entity recognition is often modeled as a sequence tagging problem. Given a sequence of characters, a sequence model returns a maximized label sequence. Each tag indicates whether the role of a word is in the *beginning*, *inside*, or *outside* of a named entity (the 'B-I-O' encoding). The task of named entity recognition can be formalized as follows.

Let T_i, x_i be random variables over the label set and tagging units. The value of T_i is one of {B, I, O}. Sequences $T = T_1, \cdots, T_n$ and $S = x_1, \cdots, x_n$ are variable sequences. Given a sentence $(S = x_1, \cdots, x_n)$, the objective of named entity recognition is to determine a label sequence $T_1, \cdots, T_n$ that solves the problem of $\arg\max P(T_1, \cdots, T_n | x_1, \cdots, x_n)$.

The sequence model can effectively find entity mentions with flattened structure in a sentence. The main problem is that nested entity mentions cannot be appropriately identified [32]. To solve this nesting problem, several strategies have been proposed: the *outermost* model, the *innermost* model, the *layering* model and the *cascading* model. In the *outermost* model, if entity mentions are nested, the model selects the outermost mentions. The *innermost* model always selects the innermost mention. The layering model and the cascading model will be discussed in Section V-B in detail.

Based on the boundary assembling method proposed by Chen *et al.* [32], the task of recognizing a named entity can be divided into three steps. First, boundaries of named entities are detected. Because boundaries are unambiguous and are independent of other NLP tasks, boundary recognition has high performance. Second, recognized boundaries are assembled into named entity candidates. Third, another classifier is used to obtain the final decision. Experiments have shown that the boundary assembling method can effectively recognize nested named entities.

In this case study, the boundary assembling method is used to demonstrate the application of SSM methodology to named entity recognition. The ACE 2005 Chinese corpus is used. There are 33,932 entity mentions in total, where 24,731 are outermost mentions and 25,766 are innermost mentions.

A. FEATURE CALCULUS

To implement feature calculus for named entity recognition, entity mention boundaries are selected as linguistic units for feature grouping. The *beginning* boundary set of entity mentions is referred to as $\mathbf{X}_B$, and the *last* boundary set of entity mentions is referred to as $\mathbf{X}_L$. Therefore, the original feature space is represented as

$$\mathbf{X} = \{\mathbf{X}_B, \mathbf{X}_L\}$$

Based on feature set $\mathbf{X}$, the task of named entity recognition is formalized as follows: for all $x_i^B \in \mathbf{X}_B$, there exists at least one $x_j^L \in \mathbf{X}_L$ such that characters from x_i^B to x_j^L represent an entity mention, referred to as $Mention(x_i^B, x_j^L)$ (or, more concisely, $Mention(x_i, x_j)$).

The difficulty is that it is not possible to know both $\mathbf{X}_B$ and $\mathbf{X}_L$ in advance, and they should be detected first. For convenience, sequence $x_1\ x_2 \cdots x_{i-1}x_i x_{i+1} \cdots x_{j-1}x_j x_{j+1} \cdots x_n$, where each x_i is a character (or a word) indicating the *beginning* or the *last* boundary of a named entity, is considered as an example of extraction of the corresponding features. Possible features used to recognize $\mathbf{X}_B$ and $\mathbf{X}_L$ are as follows:

$$\begin{aligned}
\mathbf{Z}_1 &= \{z : z = UnigramOf(x_i)\} \\
&= \{x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}\} \\
\mathbf{Z}_2 &= \{z : z = BigramOf(x_i)\} = \{x_{i-1}x_i, x_i x_{i+1}\} \\
\mathbf{Z}_3 &= \{z : z = IsSurname(x_i)\} \\
\mathbf{Z}_4 &= \{z : z = IsLocation(x_i)\} \\
\mathbf{Z}_5 &= \{z : z = IsPronoun(x_i)\} \\
\mathbf{Z}_6 &= \{z : z = IsLeftBoundary(x_i)\} \\
\mathbf{Z}_7 &= \{z : z = IsRightBoundary(x_i)\}
\end{aligned}$$

$\mathbf{Z}_1$ and $\mathbf{Z}_2$ are character-based (character unigram and bigram) features. $\mathbf{Z}_3$, $\mathbf{Z}_4$ and $\mathbf{Z}_5$ are gazetteer-based features with Boolean values $\{true, false\}$. $\mathbf{Z}_6$ and $\mathbf{Z}_7$ are segmentation-based features that also have Boolean values. The maximum matching (MM) method is used to extract word segmentation features. $\mathbf{Z}_3$, $\mathbf{Z}_4$ and $\mathbf{Z}_5$ contain 539 surnames, 31,726 location names and 125 pronouns, respectively.

Li *et al.* [51] have reported the best performance on the ACE 2005 Chinese corpus. The researchers use features from $\mathbf{Z}_1$ to $\mathbf{Z}_7$. These features are called the *basic features* and denoted by $\mathbf{Z}_{\text{basic}}$. This method is used for comparison.

$$\mathbf{Z}_{\text{basic}} = \bigcup \mathbf{Z}_i|_{1 \le i \le 7}$$

Sequence methods have the shortcoming that they cannot use nonlocal features appropriately [10]. In the boundary assembling method, after entity boundaries have been detected, boundary pairs (e.g., x_i and x_j) can be used to extract features such as the characters to the left of x_i, the characters to the right of x_j and characters from x_i to x_j. The features are as follows:

$$\begin{aligned}
\mathbf{Z}_8 &= \{z : z = Left3gramOf(x_i, x_j)\} \\
&= \{x_{i-3}x_{i-2}x_{i-1}, x_{i-2}x_{i-1}, x_{i-1}\} \\
\mathbf{Z}_9 &= \{z : z = Right3gramOf(x_i, x_j)\} \\
&= \{x_{j+1}, x_{j+1}x_{j+2}, x_{j+1}x_{j+2}x_{j+3}\} \\
\mathbf{Z}_{10} &= \{z : z = Inner3gramOf(x_i, x_j)\} \\
&= \{x_k x_{k+1}x_{k+2}|_{i \le k \le j-2}, x_k x_{k+1}|_{i \le k \le j-1}, x_k|_{i \le k \le j}\} \\
\mathbf{Z}_{11} &= \{z : z = Left1gramOf(x_i, x_j)||Right1gramOf(x_i, x_j)\} \\
&= \{x_{i-1}||x_{j+1}\} \\
\mathbf{Z}_{12} &= \{z : z = Left2gramOf(x_i, x_j)||Right2gramOf(x_i, x_j)\} \\
&= \{x_{i-2}x_{i-1}||x_{j+1}x_{j+2}\} \\
\mathbf{Z}_{13} &= \{z : z = WordsBetween(x_i, x_j)\} \\
\mathbf{Z}_{14} &= \{z : z = WordsLeft(x_i, x_j)\} \\
\mathbf{Z}_{15} &= \{z : z = WordsRight(x_i, x_j)\}
\end{aligned}$$

Features from $\mathbf{Z}_8$ to $\mathbf{Z}_{15}$ are named *extended features* and denoted by $\mathbf{Z}_{\text{extend}}$. Because extraction of these features depends on both *beginning* and *last* boundaries having been recognized, they are considered nonlocal features.

$$\mathbf{Z}_{\text{extend}} = \bigcup \mathbf{Z}_i|_{8 \le i \le 15}$$

The $\mathbf{Z}_{\text{basic}}$ feature set is widely used in named entity recognition. However, $\mathbf{Z}_{\text{extend}}$ has rarely been discussed.

In related studies, all features are used the same way. In other words, the differences between them are not considered. In practice, we observe that various features many behave differently. In general, $\mathbf{Z}_1$, $\mathbf{Z}_2$, $\mathbf{Z}_8$ and $\mathbf{Z}_9$ are local features. They are denoted by $\mathbf{Z}_{\text{local}}$. To extract features $\mathbf{Z}_{10}$ to $\mathbf{Z}_{15}$, a named entity candidate must be determined first. Therefore, we treat them as non-local features, denoted by $\mathbf{Z}_{\text{non-local}}$. As to features $\mathbf{Z}_3$ to $\mathbf{Z}_7$, it is difficult to determine whether they are local or non-local.

B. UTILIZING MORE FEATURES

To perform a comparison with nested approaches, we use methods discussed by Alex *et al.* [52] who focus on recognizing nested entity mentions. Two techniques are implemented for comparison: *cascading* and *layering*.

The *cascading* model uses a classifier for a type of named entity. For the same entity type, if nested mentions occur within,the model always collects innermost mentions. In the *layering* model, all nested entities are divided into layers (an outermost layer and an innermost layer). Each layer is processed by a single classifier. We use the CRF toolkit and the $\mathbf{Z}_{\text{basic}}$ feature set to implement these models.

Because the deep neural network is widely used to recognize named entities, we also implement a Bi-LSTM-CRF for comparison. Given a sentence, every character is mapped into a 300 dimensional vector by a lookup table, which is initialized with a random process.[6] Then, a bidirectional long short-term memory (Bi-LSTM) layer is implemented, which transforms a word embedding into a 128 × 2 dimensional vector. Afterwards, two dense layers are implemented, which map the dimension of vectors from 256 to 128 and 128 to 2 respectively. Finally, the CRF layer output a maximized label sequence. The "Adam" optimizer is used. Learning rate, weight decay rate and batch size are set as 0.00005, 0.01 and 30 respectively. A dropout regularization with value 0.5 is set to avoid the over-fitting problem. Because the Bi-LSTM-CRF

[6] In this place, we do not use the BERT approach, because word embedding has a huge influence on the performance of recognizing named entities. With word embedding pre-trained from external resources, it's hard to compare their abilities to utilize features from a sentence.

TABLE 6. Comparing performance with other state-of-art methods.

		Cascading			Layering			Bi-LSTM-CRF			SSM		
Type	**Count**	**P** (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)
FAC	1,688	63.20	21.26	31.82	56.90	28.08	37.60	41.96	38.21	40.00	57.82	36.14	44.48
GPE	8,627	84.03	63.40	72.27	76.40	66.85	71.31	63.80	63.80	63.80	80.03	74.71	77.28
LOC	1,546	65.47	32.01	43.00	61.02	34.73	44.27	47.01	35.03	40.15	58.28	45.86	51.71
ORG	6,636	72.77	48.22	58.00	68.11	48.71	56.80	62.35	57.07	59.60	71.25	56.72	63.16
PER	14,393	75.91	55.04	63.81	72.84	61.44	66.66	70.36	66.92	68.59	74.68	63.86	68.85
VEH	665	56.50	30.67	39.76	63.69	35.33	45.45	50.89	47.11	48.39	60.15	36.54	45.46
WEA	377	59.14	25.72	35.85	63.64	28.11	39.00	47.46	32.18	38.36	61.79	34.75	44.48
Det	33,932	74.59	49.39	59.43	74.23	61.53	67.28	70.19	69.75	69.97	76.88	68.70	72.56
Total	33,932	76.52	51.80	61.80	71.93	56.57	63.33	64.72	60.68	62.63	73.98	62.16	67.56

is a sequence model, if named entities are nested, we only collect the outmost entity.

In the SSM method, we first use the maximum entropy toolkit and $\mathbf{Z}_{\text{local}} = \{\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_8, \mathbf{Z}_9\}$ to extract possible entity boundaries ($\{\mathbf{X}_B, \mathbf{X}_L\}$). Afterwards, for each boundary $x_j \in \mathbf{X}_L$, we retrieve the left *Last* boundary that crosses a *Beginning* boundary. Between the two *Last* boundaries, the top *two Beginning* boundaries with higher probabilities (e.g., x_j, $x_{j'}$) are combined with x_i, resulting in, e.g., *Mention*(x_i, x_j), *Mention*$(x_i, x_{j'})$. After all mention candidates have been collected, we use both $\mathbf{Z}_{\text{basic}}$ and $\mathbf{Z}_{\text{extend}}$ feature sets to recognize them with the maximum entropy classifier. The result is shown in Table 6. The row labeled ''Det'' denotes the performance of finding entity mentions. The row labeled ''Total'' corresponds to the task of finding entity mentions and recognizing the type of each mention.

As Table 6 shows, compared with the *cascading* and *layering* methods, an impressive improvement is attained for ''ORG'' (organization). The reason for this type is that organization names often have a long range and exhibit a significant nesting problem. In this condition, structural information is very important for recognition. On the other hand, because of a small number of annotated instances, the compared methods exhibit lower performance for entity types ''FAC'' (facility), ''LOC'' (location), ''VEH'' (vehicle) and ''WEA'' (weapon). Overall, the SSM model outperforms the compared methods for all entity types.

The *cascading* model received the worst performance. Because it recognizes every entity type by an independent classifier, it can not take full advantage of annotated named entities. Furthermore, some nested mentions have the same type, they can not be identified by the *cascading* method. The *layering* model gets better performance. Because many nested named entities have two layers, they can be identified by the *layering* model. Comparing the *layering* model with the Bi-LSTM-CRF model, the former has better performance in recognizing named entities. The reason is that the Bi-LSTM-CRF model is a sequence model, which can not recognize nested named entities. However, in entity mention detection, the Bi-LSTM-CRF model outperforms the *layering* model, even if it only collects the outmost entities. It is due to the fact that neural network can automatically learn better abstract features from row inputs when the annotated entity mention is sufficient enough.

The result shows that the SSM method outperforms the *cascading*, *layering* and Bi-LSTM-CRF methods in terms of the F-score by 3% in entity mention detection and by 4% in entity mention recognition. The reason for SSM's outperformance is that boundaries of named entities are unambiguous and depend on no single NLP task. Furthermore, assembling boundaries can generate nested named entities. The presented example has shown that the task of named entity recognition can be formalized within the framework of SSM. Feature grouping and feature calculus can be used to manipulate features to capture structural information and semantic information of sentences.

VI. CONCLUSION

In this paper, we propose using SSM, a technique developed using set theory, for information extraction. This method's ability to effectively capture structural information by using external knowledge and more features is demonstrated. This framework can be extended to support more information extraction tasks (e.g., event extraction, and coreference resolution). It can also be extended to ensure the Markov dependency. Furthermore, we hope that in future research it can be combined with neural networks, a topic that has been extensively explored in recent years.

REFERENCES

[1] A. McCallum, ''Information extraction: Distilling structured data from unstructured text,'' *Queue*, vol. 3, no. 9, pp. 48–57, 2005.
[2] R. Grishman, ''Information extraction: Capabilities and challenges,'' Tech. Rep., 2012.
[3] Q. Le and T. Mikolov, ''Distributed representations of sentences and documents,'' in *Proc. ICML*, 2014, pp. 1–9.
[4] X. Deng, Y. Li, J. Weng, and J. Zhang, ''Feature selection for text classification: A review,'' *Multimedia Tools Appl.*, vol. 78, no. 3, pp. 3797–3816, 2019.
[5] Y. Chen, Q. Zheng, and P. Chen, ''A set space model for feature calculus,'' *IEEE Intell. Syst.*, vol. 32, no. 5, pp. 36–42, Sep./Oct. 2017.
[6] Y. Ren and D. Li, ''Fast and robust wrapper method for N -gram feature template induction in structured prediction,'' *IEEE Access*, vol. 5, pp. 19897–19908, 2017.
[7] W. Zeng, X. Zhao, J. Tang, and H. Shang, ''Collective list-only entity linking: A graph-based approach,'' *IEEE Access*, vol. 6, pp. 16035–16045, 2018.
[8] M. Sahami and T. D. Heilman, ''A Web-based Kernel function for measuring the similarity of short text snippets,'' in *Proc. 15th Int. Conf. World Wide Web*, May 2006, pp. 377–386.

[9] S. Zhao and R. Grishman, "Extracting relations with integrated information using Kernel methods," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics*, 2005, pp. 419–426.
[10] L. Ratinov and D. Roth, "Design challenges and misconceptions in named entity recognition," in *Proc. 13th Conf. Comput. Natural Lang. Learn.*, Jun. 2009, pp. 147–155.
[11] N. Kambhatla, "Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations," in *Proc. Interact. Poster Demonstration Sessions*, Jul. 2004, p. 22.
[12] Y. Chen, Q. Zheng, and W. Zhang, "Omni-word feature and soft constraint for Chinese relation extraction," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, 2014, pp. 572–581.
[13] Y. Chen, Q. Zheng, and P. Chen, "Feature assembly method for extracting relations in Chinese," *Artif. Intell.*, vol. 228, pp. 179–194, Nov. 2015.
[14] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[15] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proc. 4th Int. Joint Conf. Natural Lang. Process.*, 2009, pp. 1003–1011.
[16] A. Moro, H. Li, S. Krause, F. Xu, R. Navigli, and H. Uszkoreit, "Semantic rule filtering for Web-scale relation extraction," in *Proc. Int. Semantic Web Conf.*, 2013, pp. 347–362.
[17] K. Hacioglu, B. Douglas, and Y. Chen, "Detection of entity mentions occurring in English and Chinese text," in *Proc. Conf. Hum. Lang. Technol. Empirical Methods Natural Lang. Process.*, 2005, pp. 379–386.
[18] F. Ali, D. Kwak, P. Khan, S. H. A. Ei-Sappagh, S. M. R. Islam, and D. Park, "Merged ontology and SVM-based information extraction and recommendation system for social robots," *IEEE Access*, vol. 5, pp. 12364–12379, 2017.
[19] Y. Wu, J. Zhao, B. Xu, and H. Yu, "Chinese named entity recognition based on multiple features," in *Proc. Conf. Hum. Lang. Technol. Empirical Methods Natural Lang. Process.*, 2005, pp. 427–434.
[20] B. Chen, Z. Hao, X. Cai, R. Cai, W. Wen, J. Zhu, and G. Xie, "Embedding logic rules into recurrent neural networks," *IEEE Access*, vol. 7, pp. 14938–14946, 2019.
[21] M. Wang, W. Che, and C. D. Manning, "Joint word alignment and bilingual named entity recognition using dual decomposition," in *Proc. 51st Annu. Meeting Assoc. Comput. Linguistics*, 2013, pp. 1073–1082.
[22] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.
[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*. [Online]. Available: https://arxiv.org/abs/1810.04805
[25] X. Luo, W. Zhou, W. Wang, Y. Zhu, and J. Deng, "Attention-based relation extraction with bidirectional gated recurrent unit and highway network in the analysis of geological data," *IEEE Access*, vol. 6, pp. 5705–5715, 2018.
[26] M. N. A. Ali, G. Tan, and A. Hussain, "Boosting Arabic named-entity recognition with multi-attention layer," *IEEE Access*, vol. 7, pp. 46575–46582, 2019.
[27] L. Wang, Z. Cao, G. de Melo, and Z. Liu, "Relation classification via multi-level attention CNNS," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2016, pp. 1298–1307.
[28] X. Huang, "Attention-based convolutional neural network for semantic relation extraction," in *Proc. 26th Int. Conf. Comput. Linguistics*, 2016, pp. 2526–2536.
[29] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," 2015, *arXiv:1503.00075*. [Online]. Available: https://arxiv.org/abs/1503.00075
[30] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," 2014, *arXiv:1404.2188*. [Online]. Available: https://arxiv.org/abs/1404.2188
[31] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Jan. 2003.
[32] Y. Chen, Q. Zheng, and P. Chen, "A boundary assembling method for Chinese entity-mention recognition," *IEEE Intell. Syst.*, vol. 30, no. 6, pp. 50–58, Nov./Dec. 2015.
[33] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel, "The automatic content extraction (ACE) program-tasks, data, and evaluation," in *Proc. LREC*, vol. 4, 2004, pp. 837–840.
[34] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, 1988.
[35] G. Salton and C.-S. Yang, "On the specification of term values in automatic indexing," *J. Document.*, vol. 29, no. 4, pp. 351–372, 1973.
[36] Z. GuoDong, S. Jian, Z. Jie, and Z. Min, "Exploring various knowledge in relation extraction," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics*, 2005, pp. 427–434.
[37] S. Muggleton and L. De Raedt, "Inductive logic programming: Theory and methods," *J. Logic Program.*, vols. 19–20, pp. 629–679, May/Jul. 1994.
[38] L. Dehaspe, "Maximum entropy modeling with clausal constraints," in *Proc. Int. Conf. Inductive Logic Program.*, 1997, pp. 109–124.
[39] C. M. Cumby and D. Roth, "Learning with feature description logics," in *Proc. Int. Conf. Inductive Logic Program.* Berlin, Germany: Springer, 2003, pp. 32–47.
[40] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra, "A maximum entropy approach to natural language processing," *Comput. Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.
[41] S. Della Pietra, V. Della Pietra, and J. Lafferty, "Inducing features of random fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 4, pp. 380–393, Apr. 1997.
[42] S. Guiasu and A. Shenitzer, "The principle of maximum entropy," *Math. Intell.*, vol. 7, no. 1, pp. 42–48, 1985.
[43] A. McCallum, "Efficiently inducing features of conditional random fields," in *Proc. 19th Conf. Uncertainty Artif. Intell.*, 2003, pp. 403–410.
[44] M. Zhang, J. Zhang, J. Su, and G. Zhou, "A composite kernel to extract relations between entities with both flat and structured features," in *Proc. 21st Int. Conf. Comput. Linguistics 44th Annu. Meeting Assoc. Comput. Linguistics*, 2006, pp. 825–832.
[45] W. Che, T. Liu, and S. Li, "Automatic entity relation extraction," *J. Chin. Inf. Process.*, vol. 19, no. 2, pp. 1–6, 2005.
[46] P. Zhang, W. Li, Y. Hou, and D. Song, "Developing position structure-based framework for chinese entity relation extraction," *ACM Trans. Asian Lang. Inf. Process.*, vol. 10, no. 3, p. 14, 2011.
[47] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, "Relation classification via convolutional deep neural network," in *Proc. COLING*, 2014, pp. 2335–2344.
[48] C. N. dos Santos, B. Xiang, and B. Zhou, "Classifying relations by ranking with convolutional neural networks," 2015, *arXiv:1504.06580*. [Online]. Available: https://arxiv.org/abs/1504.06580
[49] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[50] K.-J. Chen and M.-H. Bai, "Unknown word detection for Chinese by a corpus-based learning method," *Int. J. Comput. Linguistics Chin.*, vol. 3, no. 1, pp. 27–44, Feb. 1998.
[51] W. Li, D. Qian, Q. Lu, and C. Yuan, "Detecting, categorizing and clustering entity mentions in Chinese text," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2007, pp. 647–654.
[52] B. Alex, B. Haddow, and C. Grover, "Recognising nested named entities in biomedical text," in *Proc. Workshop BioNLP Biol., Transl., Clin. Lang. Process.*, 2007, pp. 65–72.

YANPING CHEN is currently an Associate Professor with the College of Computer Science and Technology, Guizhou University, Guiyang. His research interests include artificial intelligence and natural language processing.

GUORONG WANG is currently a Graduate Student with the College of Computer Science and Technology, Guizhou University, Guiyang. Her research interest includes natural language processing.

QINGHUA ZHENG is currently a Professor with the Department of Computer Science and Technology, Xi'an Jiaotong University. His research interests include multimedia distance education and computer network security.

RUIZHANG HUANG is currently an Associate Professor with the College of Computer Science and Technology, Guizhou University, Guiyang. Her research interests include information retrieval and text mining.

YONGBIN QIN is currently a Professor with the College of Computer Science and Technology, Guizhou University, Guiyang. His research interests include big data processing, cloud computing, and text mining.

PING CHEN received the Ph.D. degree in information technology from George Mason University. He is currently an Associate Professor of computer science and the Director of the Artificial Intelligence Laboratory, University of Massachusetts Boston, Boston. His research interests include data mining and computational semantics.