Extracting N-ary Cross-sentence Relations using Constrained Subsequence Kernel

Sachin Pawar[1,2], Pushpak Bhattacharyya[2], and Girish K. Palshikar[1]

[1] TCS Research & Innovation, Pune, India-411013
[2] Indian Institute of Technology Bombay, Mumbai, India-400076
sachin7.p@tcs.com, pb@cse.iitb.ac.in, gk.palshikar@tcs.com

Abstract. Most of the past work in relation extraction deals with relations occurring within a sentence and having only two entity arguments. We propose a new formulation of the relation extraction task where the relations are more general than intra-sentence relations in the sense that they may span multiple sentences and may have more than two arguments. Moreover, the relations are more specific than corpus-level relations in the sense that their scope is limited only within a document and not valid globally throughout the corpus. We propose a novel sequence representation to characterize instances of such relations. We then explore various classifiers whose features are derived from this sequence representation. For SVM classifier, we design a Constrained Subsequence Kernel which is a variant of Generalized Subsequence Kernel. We evaluate our approach on three datasets across two domains: biomedical and general domain.

Keywords: Relation Extraction · N-ary Relations · Cross-sentence Relations · Subsequence Kernel

1 Introduction

The task of traditional relation extraction (RE) deals with identifying whether any pre-defined *semantic relation* holds between a pair of *entity mentions* in a given sentence [24, 12, 14]. In this paper, we propose a new formulation of the traditional relation extraction task, where the relations are more general than ACE-like intra-sentence relations [7] in the sense that they may span multiple sentences and may have more than two entity arguments (N-ary). In addition, the relations are more specific than Freebase-like relations [2] in the sense that their scope is limited only within a document and not valid globally throughout the corpus. Hence, distant supervision based approaches [11, 18] involving corpus-level relations will not be applicable. E.g., *Drug2AE* is such a relation between a *Drug* and an *AdverseEvent* it causes. Scope of the *Drug2AE* relation is limited only within a particular document (case report of a patient), it may not be valid in another document (case report of another patient) even if it contains mentions of the exactly same *Drug* and *AdverseEvent*. Hence, our proposed formulation is useful to retrieve only the relevant documents which actually *express* a relation among the entities; and not just *contain* the entities. Figure 1 shows where our proposed formulation of relation extraction lies within the spectrum of traditional formulations.

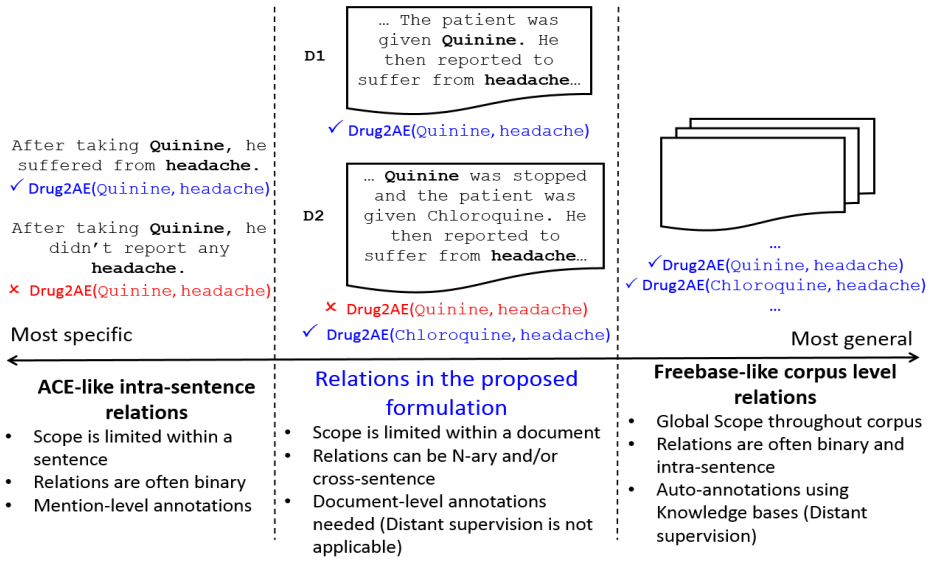

Fig. 1. Spectrum of relation extraction formulations.

Table 1. Example of the *Succession* relation and its instances

Relation type: *Succession*; **Signature:** (*ORG*, *POST*, *PER*, *PER*); **Meaning:** *Succession*$(D, org, post, per_1, per_2)$ holds if the document D reports that in the organization org, the person per_1 is succeeded by per_2 for the post $post$
Following relation instances are *similar* and are part of a single *group* (see Table 3 for the corresponding document *news_1.txt*):
⟨*news_1.txt*, `AB Volvo`, `chairman`, `Pehr G. Gyllenhammar`, `Bert-Olof Svanholm`⟩
⟨*news_1.txt*, `Volvo`, `chairman`, `Pehr G. Gyllenhammar`, `Bert-Olof Svanholm`⟩
⟨*news_1.txt*, `AB Volvo`, `chairman`, `Pehr G. Gyllenhammar`, `Mr. Svanholm`⟩
⟨*news_1.txt*, `Volvo`, `chairman`, `Pehr G. Gyllenhammar`, `Mr. Svanholm`⟩

In the proposed formulation, each relation type is associated with a *signature* which is a sequence of entity types of its arguments. We define a *candidate relation instance* to be a tuple of a document name followed by an ordered sequence of entities which are type compatible with the relation's signature. If the relation holds for the argument entities within the document, then the candidate relation instance is referred as a *relation instance* (Example in Table 1). In this new formulation, a new challenge emerges: each entity argument of a candidate relation instance may be mentioned multiple times throughout the document either as it is or in the form of its *aliases*; e.g. `AB Volvo`, `Volvo`; or `Bert-Olof Svanholm`, `Mr. Svanholm`. Two candidate relation instances are said to be *similar* if their corresponding argument entities are identical or aliases of each other. Thus, the candidate relation instances in a document can be partitioned into

Table 2. Example of annotations $\langle D_1, LE_1, LR_1 \rangle$

Target Relation: *Interact*; **Signature:** (*Drug*, *Gene*, *Mutation*); *Interact* (D_i, d, g, m) holds if the document D_i mentions that the drug d treats the mutation m in the gene g
Entity annotations LE_1 for document D_1: ⟨gefitinib:*Drug*⟩,⟨erlotinib:*Drug*⟩,⟨EGFR:*Gene*⟩, ⟨T790M:*Mutation*⟩,⟨A750P:*Mutation*⟩, ⟨L858R:*Mutation*⟩
Relation annotations LR_1 for document D_1: ⟨erlotinib,EGFR,L858R⟩,⟨erlotinib,EGFR,T790M⟩

groups of similar instances. Thus, all the relation instances in Table 1 are similar and therefore part of the same group.

Our task in this paper is to identify which of the candidate relation instances *truly* represent a relation type of interest. To characterize any candidate relation instance, we propose a novel *sequence representation*. This representation is designed in such a way that it will be same for all *similar* candidate relation instances in a *group*. We explore various classifiers whose features are derived from this sequence representation. We also propose a new kernel function "Constrained Subsequence Kernel" (CSK) which is designed to compute number of common subsequences of interest between any two sequence representations. Specific contributions of this work are: 1) a new formulation of the N-ary and cross-sentence relation extraction task, 2) a novel sequence representation for characterizing the candidate relation instances, and 3) a specifically designed subsequence kernel.

2 Problem Definition

Inputs: Target relation type R, its signature, a set of test documents $\{\langle D'_1, LE'_1 \rangle, \cdots \langle D'_L, LE'_L \rangle\}$ where LE'_i is the list of entities in document D'_i along with their types
Output: For each test document D'_i, all the candidate relation instances for R are classified to check whether they are *true* relation instances to produce a list LR'_i of such instances for R.
Training regime: To learn relation extraction model for R, we need a set of annotated training documents $\{\langle D_1, LE_1, LR_1 \rangle, \cdots \langle D_K, LE_K, LR_K \rangle\}$, where LE_i is the list of entities in the document D_i along with their types and LR_i is the list of relation instances of type R which hold within D_i. It is enough to annotate any one representative relation instance from each *group* of similar instances. We refer to this annotation scheme as *RIGD* (at least one **R**elation **I**nstance per **G**roup per **D**ocument) level as against the traditional mention-level annotation (Table 2).
Scope and Assumptions: Multiple target relation types are handled by providing separate annotations (LR_i's) and learning separate extraction models for each relation type. Identification of entities and their types is out of scope of this paper. For each test document D'_i, we assume availability of the LE'_i's. We do not assume availability of alias information. For identifying aliases, we use high-precision domain-specific rules; details of these rules are included in the Appendix.

Table 3. Example of a news article *news_1.txt* where entities of interest are highlighted

An extraordinary shareholders meeting of **AB Volvo** in Gothenburg, Sweden, elected **Bert-Olof Svanholm chairman** of the Swedish automotive group, in line with an earlier proposal. **Mr. Svanholm** is **president** of **ABB Asea Brown Boveri Ltd.**, an engineering concern jointly owned by **Asea AB of Sweden** and **BBC Brown Boveri AG** of Switzerland. **He** succeeds **Pehr G. Gyllenhammar**,who resigned in December after the collapse of a plan to merge **Volvo's** vehicle operations with those of French partner **Renault SA.**

Table 4. Examples of sequence representations. We use ; for separating tokens in a sequence

Sequence representation of T_1=⟨D=*news_1.txt*, E_1=AB Volvo, E_2=chairman, E_3=Pehr G. Gyllenhammar, E_4=Bert-Olof Svanholm⟩:
extraordinary; shareholders; meeting; of; E_1; in; gothenburg; sweden; elected; E_4; E_2; of; Swedish; automotive; group; in; line; with; earlier; proposal; ***SB***; E_4; OE_{POST}; of; OE_{ORG}; engineering; concern; jointly; owned; by; OE_{ORG}; OE_{ORG}; of; switzerland; ***SB***; E_4; succeeds; E_3; resigned; in; december; after; collapse; of; plan; to; merge; E_1; vehicle; operations; with; of; french; partner; OE_{ORG}
Sequence representation of T_2=⟨D=*news_1.txt*, E_1=ABB Asea Brown Boveri Ltd., E_2=president, E_3=Bert-Olof Svanholm, E_4=Pehr G. Gyllenhammar⟩:
extraordinary; shareholders; meeting; of; OE_{ORG}; in; gothenburg; sweden; elected; E_3; OE_{POST}; of; swedish; automotive; group; in; line; with; earlier; proposal; ***SB***; E_3; E_2; of; E_1; engineering; concern; jointly; owned; by; OE_{ORG}; OE_{ORG}; of; switzerland; ***SB***; E_3; succeeds; E_4; resigned; in; december; after; collapse; of; plan; to; merge; OE_{ORG}; vehicle; operations; with; of; french; partner; OE_{ORG}

Evaluation: We evaluate at *RIGD*-level where a true positive is counted for each *group* of predicted relation instances, if any of the relation instance from the group is listed in gold-annotations (LR_i). A false positive is counted for each group of predicted relation instances, if none of the relation instances from the group are listed in gold-annotations. Also, a false negative is counted for each group of relation instances listed in gold-annotations, if none of the relation instances in the group appear as the predicted relation instances.

3 Proposed Approach

We propose a novel **sequence representation** for a candidate relation instance, which captures its important characteristics. An N-ary candidate relation instance is a $(N+1)$ tuple containing document name D and N entity arguments–$E_1, E_2, \cdots E_N$. Here, $E_1, \cdots E_N$ may not be of N *distinct* entity types; E_i just represents the i^{th} entity argument of the relation. Sequence of entity types of these N entity arguments

($ET_1, ET_2, \cdots ET_N$) is the *signature* of the relation type. E.g., the relation *Succession* (Table 1) holds between 4 entity arguments of entity types: ET_1= *ORG*, ET_2= *POST* and ET_3=ET_4= *PER*. An example instance of this relation is represented as the following 5-tuple (Table 3): T_1= $\langle D$= *news_1.txt*, E_1= `AB Volvo`, E_2= `chairman`, E_3= `Pehr G. Gyllenhammar`, E_4= `Bert-Olof Svanholm`$\rangle$. There are other candidate relation instances for which the relation does not hold but they just conform to its signature; e.g., T_2= $\langle D$= *news_1.txt*, E_1= `ABB Asea Brown Boveri Ltd.`, E_2= `president`, E_3= `Bert-Olof Svanholm`, E_4= `Pehr G. Gyllenhammar`$\rangle$. Hence, T_1 is a positive instance for the relation type *Succession* whereas T_2 is a negative instance.

Span and Minimal Span: We define *span* of a candidate relation instance T as the sequence of sentences in the document D covering all the mentions of its argument entities (including aliases). The sequence starts with the earliest mention of any entity argument (or its alias) involved in T and stretches up to the latest mention. For each argument pair of T, minimum number of sentences separating corresponding entity mentions is computed. We define *minimal span* of a candidate relation instance as the maximum of minimum number of separating sentences across all argument pairs. We use *minimal span* to filter out candidate relation instances having values more than some threshold. E.g., T_1 has span of 3 sentences and the minimal span of 2 (Table 3) corresponding to the argument pair of E_2 and E_3 which are 2 sentences apart and this is the maximum separation across all argument pairs.

3.1 Constructing Sequence Representations

We propose to characterize any candidate relation instance T of relation type R in the form of **a sequence of *tokens* of certain types**:

• E_i: Mentions of the i^{th} entity argument (and its aliases) of T within the span of T

• *SB*: Sentence boundaries of the sentences in the *span* of T

• OE_{ET_j}: Mentions of *other* entities than the argument entities (and their aliases) which occur within the span of T and which are of type ET_j. Tokens of this type encode important discourse information by capturing mentions of other entities of type ET_j.

• Words: All the words (excluding stop words) occurring within the span of the instance T.

In order to construct the sequence representation of a candidate relation instance, these tokens are arranged sequentially from the beginning of the *span* till the end. The tokens are arranged in the same order as they occur in the document. In other words, these tokens are place-holders in the sequence representation, for each important piece of information. Table 4 shows sequence representations for our example instances T_1 and T_2. It can be observed that for T_1, all the mentions of the entity argument `Bert-Olof Svanholm` including its aliases (`Mr. Svanholm` and `He`)[3] are captured using the token E_4. Also, `president` is represented using the token OE_{POST}, which is an entity of type *POST* but is not an argument entity of T_1 .

[3] In principle, coreferences of entity arguments can be used instead of just aliases when we add tokens of the form E_i and OE_{ET_j} in the sequence representation. But coreference resolution is itself a difficult problem and is not accurate enough in practice for Biomedical domain documents. Hence, we use coreferences only for general domain dataset.

Table 5. A few example subsequences of sequence representations of T_1 and T_2 (Table 4)

Subsequences not satisfying Constraint 1: (meeting; OE_{ORG}; elected; E_3), (E_4; OE_{POST}; of)
Subsequences not satisfying Constraint 2: (E_1; E_2), (E_3; E_3), (E_2; E_4)
Subsequences satisfying both the constraints: (meeting; E_1; elected; E_4), (E_4; succeeds; E_3), (elected; E_4; E_2; of; group), (E_2; of; E_1; *SB*; E_3)

Generalizing the sequence representation: We create clusters of the frequently occurring words in a domain. By considering cosine similarity among the word embeddings, we apply hierarchical clustering with complete linkage. E.g., in Biomedical domain, following is an example word cluster {radiotherapy, chemotherapy, adjuvant, immunotherapy}. These word clusters are used to generalize the sequence representation by replacing a single word token with a set of 2 tokens containing word itself along with its cluster ID. Hence, the sequence representation for T_1 would now be: {c12, extraordinary}; {c43, shareholders}; {c47, meeting}; of; E_1; in;$\cdots$

3.2 Constrained Subsequence Kernel (CSK)

Generalized Subsequence Kernel (GSK) proposed by Mooney and Bunescu [13] computes the number of common subsequences of length n shared by two *generalized* sequences, in polynomial time. The sequences which share more such common subsequences, get a higher similarity score. Moreover, the subsequences are weighted by their sparseness in the original sequences, i.e. subsequences which are not contiguous and spread over a greater length in the sequences will get lower weights (inversely proportional to length of their spread in those sequences). In a *generalized* sequence, any token in a sequence can be generalized to a set of values; e.g., we use word cluster IDs as the generalizations for actual words.

We propose a variant of the generalized subsequence kernel, namely *Constrained Subsequence Kernel* (CSK). As the name suggests, CSK differs from the original kernel GSK by constraining the subsequences to consider. Our goal is to design a kernel function such that it will compute a high similarity score among the two candidate relation instances if both of them are *true* relation instances. The similarity should be lower if one of them is a true relation instance and other is not. Hence, the intuition is that the common subsequences (to consider during kernel computation) should contain at least two distinct tokens of the type E_i. Because presence of at least two of these tokens in a subsequence, ensures that the subsequence captures interaction among at least two entity arguments. Also, the common subsequences are constrained to have length of at least 3. This ensures that any common subsequence will contain at least one token other than the two tokens corresponding to entity arguments. Thus, the following 2 constraints are considered (see Table 5).

Constraint 1: A subsequence should contain **at least two distinct tokens from** $E_1, E_2, \cdots, E_N$

Constraint 2: A subsequence should contain **at least three tokens**

3.3 Formal Definition of CSK

Let $CSK(s, t, n, \lambda, a, b)$ be the constrained subsequence kernel which computes number of λ-weighted common subsequences of length n shared by the sequences s and t such that each of these common subsequences contains particular tokens a and b. Here, a and b are considered to incorporate the *Constraint 1*; and the *Constraint 2* is trivially satisfied by considering CSK with $n \geq 3$. Also, λ is a number between 0 and 1. Each common subsequence is weighted by λ^l where l is the sum of lengths of the subsequence's spread in s and t. Let $\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4$ and Σ_5 be disjoint spaces representing various types of tokens used in the sequence representation: (i) $\Sigma_1 = \{E_1, E_2, \cdots E_N\}$, for N-ary relation type, E_i is a place-holder for mentions of the i^{th} entity argument of the relation type, (ii) $\Sigma_2 = \{SB\}$, the "sentence break" token, (iii) $\Sigma_3 = \{OE_{ET_1}, \cdots OE_{ET_M}\}$, M is the no. of distinct entity types in the relation's signature, (iv) Σ_4=Set of words, (v) Σ_5=Set of word cluster IDs

Sequence representation for any candidate relation instance belongs to Σ^*, where $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \{\Sigma_4 \times \Sigma_5\}$. Each sparse subsequence of such sequence representations then belongs to Σ'^* where $\Sigma' = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4 \cup \Sigma_5$. We design an efficient recursive formulation for computing $CSK(s, t, n, \lambda, a, b)$. This formulation is derived on the similar lines as that of the generalized subsequence kernel [13] and hence the same notations ($K_n(s,t)$, $K'_n(s,t)$ and $K''_n(s,t)$) are used with similar meaning. However, we introduce additional auxiliary functions (see Table 6) for taking the *Constraint 1* into consideration for CSK computation. As per its definition, $K'_n(s,t)$ adds length from beginning of a common subsequence to the end of the sequences s and t. We define $aK'_n(s,t)$ to be a similar function with the only difference that it considers only those common subsequences which contain the token a. Similarly, the function $aK''_n(s,t)$ is defined analogous to $K''_n(s,t)$. The other auxiliary functions are defined analogously. The detailed computation steps are depicted in the Table 7. Here, the function $c(x, y)$ computes the number of common tokens between the sets x and y. Also, as a and b would always be from $\{E_1, E_2, \cdots E_N\}$ and there are no generalizations defined for these tokens, the equality conditions ($x == a$ and $x == b$) will be satisfied only when singleton tokens are involved. This simplifies the computation for additional auxiliary functions. Recursive computations of K' and K'' are same as the generalized subsequence kernel. Table 7 shows recursive updates for other functions (like aK'', bK'', abK'', aK', bK' and abK') by taking into consideration the first constraint. These values are computed incrementally from $i = 0$ to n and finally used to compute the value of $abK_n(s,t)$ i.e. $CSK(s, t, n, \lambda, a, b)$.

Table 6. Additional auxiliary functions defined analogous to K'_n and K''_n

Auxiliary Functions	**For counting common subsequences which...**
aK'_n, aK''_n	contain token a
bK'_n, bK''_n	contain token b
abK'_n, abK''_n	contain both the tokens a and b

Table 7. Recursive and efficient computation of Constrained Subsequence Kernel (CSK)

Recursive computation for $abK_n(s,t) = CSK(s,t,n,\lambda,a,b)$:

$K'_0(s,t) = 1$, for all s,t
$K''_i(sx,ty) = \lambda K''_i(sx,t) + \lambda^2 K'_{i-1}(s,t)c(x,y)$
If $x == y$ **and** $x == a$ **then**,
  $aK''_i(sx,ty) = \lambda aK''_i(sx,t) + \lambda^2 K'_{i-1}(s,t)$
  If $i > 1$ **then**,
    $bK''_i(sx,ty) = \lambda bK''_i(sx,t) + \lambda^2 bK'_{i-1}(s,t)$
    $abK''_i(sx,ty) = \lambda abK''_i(sx,t) + \lambda^2 bK'_{i-1}(s,t)$
Else If $x == y$ **and** $x == b$ **then**,
  $bK''_i(sx,ty) = \lambda bK''_i(sx,t) + \lambda^2 K'_{i-1}(s,t)$
  If $i > 1$ **then**,
    $aK''_i(sx,ty) = \lambda aK''_i(sx,t) + \lambda^2 aK'_{i-1}(s,t)$
    $abK''_i(sx,ty) = \lambda abK''_i(sx,t) + \lambda^2 aK'_{i-1}(s,t)$
Else If $c(x,y) > 0$ **then**,
  If $i > 1$ **then**,
    $aK''_i(sx,ty) = \lambda aK''_i(sx,t) + \lambda^2 aK'_{i-1}(s,t)c(x,y)$
    $bK''_i(sx,ty) = \lambda bK''_i(sx,t) + \lambda^2 bK'_{i-1}(s,t)c(x,y)$
    $abK''_i(sx,ty) = \lambda abK''_i(sx,t) + \lambda^2 abK'_{i-1}(s,t)c(x,y)$
Else
  $aK''_i(sx,ty) = \lambda aK''_i(sx,t)$**;** $bK''_i(sx,ty) = \lambda bK''_i(sx,t)$
  $abK''_i(sx,ty) = \lambda abK''_i(sx,t)$

$K'_i(sx,t) = \lambda K'_i(s,t) + K''_i(sx,t)$**;** $aK'_i(sx,t) = \lambda aK'_i(s,t) + aK''_i(sx,t)$
$bK'_i(sx,t) = \lambda bK'_i(s,t) + bK''_i(sx,t)$**;** $abK'_i(sx,t) = \lambda abK'_i(s,t) + abK''_i(sx,t)$

$Sum := 0$
For $j = 1$ to $|t|$
  If $x = t[j]$ **and** $x = a$ **then**, $Sum := Sum + \lambda^2 bK'_{n-1}(s, t[1:j-1])$
  Else If $x = t[j]$ **and** $x = b$ **then**, $Sum := Sum + \lambda^2 aK'_{n-1}(s, t[1:j-1])$
  Else If $c(x,t[j]) > 0$ **then**, $Sum := Sum + \lambda^2 abK'_{n-1}(s, t[1:j-1])c(x,t[j])$
$abK_n(sx,t) = abK_n(s,t) + Sum$

3.4 Classifying Candidate Relation Instances

For each training document D_i (Table 2), initially candidate relation instances are generated using all possible combinations of entities in LE_i which conform to the signature of the target relation type. Candidate relation instances having *minimal span* more than some threshold are filtered out as they are unlikely to be true relation instances. Out of the remaining candidate relation instances, the ones which are *similar* to any of the instance in LR_i are treated as positive instances for a *binary* classifier and the remaining as negative instances. During testing, candidate relation instances are generated for a document D'_i in a similar way by using LE'_i and the signature of the target relation type. We explored 3 classifiers whose features are derived from the proposed sequence representation, either explicitly (MaxEnt) or implicitly (SVM with CSK & LSTM).

Maximum Entropy (MaxEnt) Classifier [1]: The features are explicitly engineered from the sequence representation of any candidate relation instance (Table 8).

Table 8. MaxEnt features for a candidate relation instance T and its sequence repr. $Seq(T)$

Feature	Description
$TupleSpan$	Integer-valued feature indicating the minimal span of T in terms of number of sentences
$SentDiff_{ij}$ $SameLine_{ij}$	Integer-valued features for each pair of E_i & E_j indicating minimum number of sentences separating them in $Seq(T)$; or indicating whether they occur in a single sentence in the span of T
$E_iE_jOE_{ET_k}$	Boolean feature for each triplet of E_i, E_j and OE_{ET_k} which is true if OE_{ET_k} occurs between E_i and E_j in $Seq(T)$; and ET_k is entity type of either E_i or E_j. These features capture key discourse information about mentions of other entities occurring between the mentions of argument entities of T.
$E_iE_jNoOE_{ET_k}$	Boolean feature for each triplet of E_i, E_j and OE_{ET_k} which is true if no OE_{ET_k} occurs between E_i and E_j in $Seq(T)$; and ET_k is entity type of either E_i or E_j
Word / Cluster	Each word occurring in $Seq(T)$ and its cluster ID are boolean features because some of these words may be key lexical cue-words for the relation type

LSTM [8]**-based Classifier**: We define an embedded representation for each unique token appearing in sequence representations. This representation is a concatenation of two vectors. For word tokens, the first vector is initialized using pre-trained word embeddings and the second vector is set to all zeros. For other tokens (e.g. E_i, OE_{ET_j}, *SB*), the first vector is set to all zeros whereas the second vector contains one-hot representation for all the distinct non-word tokens. Sequence of these tokens is then passed through an LSTM layer and the output of the final step is connected through a hidden layer to a softmax layer representing the two class labels.

Support Vector Machines with CSK: Our principal approach is SVM [5] with the CSK kernel. Let R be an N-ary relation type and s, t be the sequence representations of any two candidate relation instances. Let CSK_N^λ be the overall kernel across all entity arguments of R. It is computed and normalized as follows:

$$CSK_N^\lambda(s,t,n) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} CSK(s,t,n,\lambda,E_i,E_j)$$

$$NCSK_N^\lambda(s,t,n) = \frac{CSK_N^\lambda(s,t,n)}{\sqrt{CSK_N^\lambda(s,s,n) \cdot CSK_N^\lambda(t,t,n)}}$$

We combine kernel functions for various subsequence lengths (i.e. n) to get final kernel function:

$$CSK_{Final}^\lambda(s,t) = \frac{\sum_{k=3}^{N'} 2^{N'-k} \cdot NCSK_N^\lambda(s,t,k)}{\sum_{k=3}^{N'} 2^{N'-k}}$$

N' is a parameter controlling the number of different subsequence lengths. E.g., if $N' = 5$ then subsequences of lengths $3, 4$ and 5 are considered.

Table 9. Mention-level and RIGD-level evaluation results for the binary *Lives_in* relation (Bacteria Biotope)

Level	Fold	MaxEnt			LSTM			SVM with CSK			[10]		
		P	R	F	P	R	F	P	R	F	P	R	F
Mention	train_dev	61.2	55.8	58.4	62.8	54.9	58.5	69.0	57.3	**62.6**	58.2	61.0	59.6
	dev_train	46.7	55.9	50.9	37.8	47.7	42.2	60.1	52.3	**55.9**	46.9	55.2	50.7
	Average	54.0	55.9	54.7	50.3	51.3	50.4	64.6	54.8	**59.3**	52.6	58.1	55.2
RIGD	train_dev	53.8	50.9	52.3	54.2	46.7	50.2	63.2	55.2	**58.9**	_	_	_
	dev_train	43.5	59.5	50.3	35.4	52.4	42.3	56.2	54.2	**55.2**	_	_	_
	Average	48.7	55.2	51.3	44.8	49.6	46.3	59.7	54.7	**57.1**	_	_	_

4 Experimental Analysis

We evaluate our approach on 2 datasets from Biomedical domain and 1 general domain dataset. For the proposed new formulation of the N-ary cross-sentence relation extraction task, there are no readily available public datasets. We converted[4] annotations of some public datasets used for other tasks, to the *RIGD*-level annotations.

4.1 Datasets

Bacteria Biotope: Bacteria Biotope task [6] was held as a part of BioNLP 2016 shared task. It has annotations for *Lives_In* event having two entity arguments– *Bacteria* and the location where it was found (either *Habitat* or *Geographical*). We mapped this event to a binary, cross-sentence relation *Lives_In*, and converted the mention-level annotations to *RIGD* level. We used simple rules to identify aliases of the bacteria names. e.g. `Salmonella Typhimurium`↔`S. Typhimurium` and `salmonellae`↔`salmonella`. We used *train* and *dev* partitions of the dataset[5] to carry out 2-fold cross-validation (Table 9), similar to VERSE [10] which was the best performing system for this task.

Drug-Gene-Mutation: Peng et al. [15] released a dataset[6] where known interactions among *Drug*, *Gene* and *Mutation* were captured as a traditional mention-level ternary cross-sentence relation *Interact*. We converted these annotations for the *Interact* relation to *RIGD*-level. As we do not have access to complete documents, pseudo-documents were created for each section of consecutive sentences used in the dataset. As the original annotations were obtained through distant supervision; instead of using all non-positive candidate relation instances as negative instances, we only used the explicitly annotated negative relation instances. In order to compare the performance with the approach of Peng et al. [15], we computed the "Accuracy" metric at mention-level along with other metrics at *RIGD*-level (Table 10).

MUC-6: We used the *train* and *test* datasets for management succession event from MUC-6 [20] and converted the annotations into a 4-ary relation *Succession* (Table 1).

[4] Conversion and evaluation scripts for all the datasets will be made public if the paper is accepted.

[5] `http://2016.bionlp-st.org/tasks/bb2`

[6] `http://hanover.azurewebsites.net`

The event arguments in the original dataset are mapped to entity arguments for the relation. 32 documents each from train and test partitions contained the complete *Succession* relation and were used for our evaluation (Table 11). The original dataset labelled "gold" aliases for entities; these were used for converting the annotations. But our algorithm does not use gold aliases during testing; rather we use simple high-precision domain-specific rules (e.g. `Bert-Olof Svanholm` and `Mr. Svanholm` are identified as aliases).

Table 10. Mention-level accuracy & RIGD-level (P,R,F) for the ternary *Interact* relation (Drug-Gene-Mutation). The results by Quirk and Poon [17] are mentioned as reported by Peng el al. [15]

Approach	P	R	F	Accuracy
Maxent	73.2	65.2	69.0	76.0
LSTM	72.9	67.4	70.0	77.5
SVM with CSK	75.5	67.1	71.1	78.8
Quirk and Poon[17]	-	-	-	77.7
Peng et al.[15]	-	-	-	82.4
Song et al.[19]	-	-	-	**83.2**

Table 11. RIGD-level (P,R,F) for the 4-ary *Succession* relation (MUC-6)

Approach	P	R	F
Maxent	31.8	46.1	37.6
LSTM	20.0	28.9	25.2
SVM with CSK	73.3	28.9	**41.5**

Dataset Statistics are shown in Table 12.
Implementation Details: Minimal spans used for filtering candidate relation instances were as follows (#sentences): *Succession*-2, *Lives_In*-4, *Interact*-2. For all the experiments, the CSK parameters were set as follows: $N' = 4$, $\lambda = 0.9$. We used libsvm [3] and keras [4] for SVM and LSTM implementations and our own implementation for MaxEnt. C parameter for SVM was set to 1 for all experiments and instance weights were used (in all classifiers) so as to ensure that sum of weights of positive instances is same as that of negative instances. 100-dim GloVe [16] word vectors (pre-trained on Wikipedia) were used for finding word clusters as well as for the LSTM-based classifier.

4.2 Analysis of Results and Errors

Results: The SVM with CSK outperformed MaxEnt and LSTM-based classifiers on all the 3 datasets (Tables 9, 10 and 11). For Bacteria-Biotope dataset, it outperformed the previous work, VERSE [10]. For Drug-Gene-Mutation dataset, it outperformed one out of three previous works and for MUC-6, there is no comparable previous result. Also, the SVM with CSK was observed to have higher precision than recall. This is because

each possible subsequence (satisfying the constraints) of the tokens leads to a separate dimension in the transformed space using CSK. Thus, feature-space representation using CSK is sparse and it leads to higher precision and lower recall for SVM, with limited training data. There is scope for improving the recall by better generalizing the sequence representations.

Table 12. Dataset statistics. Except for the *Interact* relation where negative instances are explicitly annotated, for other relations negative instances are automatically generated.

Relation	**#Documents**	**Mention-level Instances**		**RIGD-level Instances**	
		#pos	#neg	#pos	#neg
Lives_In (Bacteria-Biotope)	107	814	2506	392	1517
Interact (Drug-Gene-Mutation)	3652	3407	3564	2187	3182
Succession,Train (MUC-6)	36	377	43940	66	8344
Succession,Test (MUC-6)	36	474	13734	76	2874

Ablation Analysis: We perform ablation analysis to evaluate contributions of key design elements: constraints and word clusters. For all the datasets, the reported results (Tables 9, 10 and 11) are with word clusters and using the constraints. Table 13 shows the effect of discarding each of these design elements. Constraints in CSK were observed to be beneficial for all the datasets, whereas word clusters were useful only for *Lives_In* relation.

Table 13. Ablation Analysis (all the numbers are at RIGD-level)

Setting	*Lives_In* (Bacteria-Biotope)			*Interact* (Drug-Gene-Mutation)			*Succession* (MUC-6)		
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
M: SVM with CSK	59.7	54.7	57.1	75.5	67.1	71.1	73.3	28.9	41.5
M without Constraint 1 (SVM with GSK)	74.6	12.7	21.6	73.1	57.3	64.2	100.0	7.9	14.6
M without word clusters	54.7	54.5	54.5	75.3	67.7	71.3	70.6	31.6	43.6

Error Analysis: We analyzed poorer performance for the *Succession* relation and observed that two major reasons are: *Class Imbalance* and presence of two arguments with the same entity type *PER*. As it is a 4-ary relation, number of possible candidate relation instances is high and very few of them actually represent the relation; resulting in *Class Imbalance*. For T_1 (Table 4), if last two arguments (E_3 and E_4) are swapped, we get almost identical sequence representation with just E_3 and E_4 swapping their positions. This new instance (with swapped E_3 and E_4) is a negative instance for the *Succession* relation unlike T_1. It is challenging for the classifiers to distinguish between these nearly identical sequence representations of opposite classes, with limited training data.

We analyzed poorer performance for the *Interact* relation as compared to the state-of-the-art and observed that a major reason was absence of "gold" entities

information in the original dataset which only annotated entities which are part of annotated relation instances and not *all* entities. Tokens of the type OE_{ET_j} in our sequence representation depend on information of mentions of all entities; hence the sequence representation could not characterize the relation instance completely. Also, the annotation labels obtained through distant supervision are not perfect. E.g., we get a false positive for predicting $\langle$`ipilimumab`, `BRAF`, `V600E`$\rangle$ but it is a *true* relation instance as per the following sentence in the document: `Rapid improvement of therapeutic responses using combined vemurafenib plus ipilimumab therapy for BRAF V600E mutation positive melanoma is expected.`

Although, the LSTM-based classifier and the SVM with CSK both use the same sequence representation, the LSTM-based classifier performs poorly in comparison. Through the constraint on subsequences, the SVM with CSK harnesses the knowledge that the tokens of type E_i are more important. Whereas the LSTM-based classifier does not explicitly harness this knowledge and needs further exploration.

5 Related Work

Our formulation of the relation extraction task differs from the past work on the *distant supervision* based relation extraction [11, 18, 9, 22] and *slot filling* tasks [21] in terms of scope of the relations. Rather than extracting corpus-level facts / relations, our approach focuses on determining whether a relation holds for a tuple of entities within scope of a particular document. Also, these approaches extract only binary relations which are expressed in single sentences. To the best of our knowledge, the problem of cross-sentence relation extraction was first addressed by Swampillai and Stevenson [23]. They proposed to introduce a dependency link between the root nodes of parse trees containing the given pair of entities. They developed features based on the shortest path connecting the pair of entities in the new "fused" tree. Recently, Quirk and Poon [17] proposed a new approach for cross-sentence relation extraction using distant supervision. They proposed a graph representation which incorporates both standard dependencies and discourse relations. In a document graph, each node is labeled with its lexical item, lemma and POS tag. Edges are added between adjacent words as well as between words connected with dependencies. Inter-sentential edges are added in 3 cases: i) edge between root nodes of adjacent sentences, ii) discourse relations and iii) co-references. Features are then extracted from multiple paths in this graph and a binary logistic regression classifier is trained using these features.

Peng et al. [15] proposed a general framework for N-ary cross-sentence relation extraction, based on graph LSTMs. They used the same document graph as proposed by Quirk and Poon [17] as a backbone for their graph LSTM. The word embeddings of input text are provided to the input layer. Next layer is formed by the graph LSTM which learns a contextual representation for each word. For the entities in a relation instance, their contextual representations are concatenated and provided as the input to the relation classifier. For training graph LSTMs using backpropagation, the document graph needs to be partitioned into 2 directed acyclic graphs (DAGs). Song et al. [19] proposed graph-state LSTMs which do not need such partitioning and use the original graph. They used a parallel state to model each word, where state values are enriched recur-

rently via message passing. In contrast to these mention-level N-ary and cross-sentence relation extraction approaches, our proposed formulation captures a broader view of any relation instance in a single representation; by incorporating multiple mentions of the entity arguments and their aliases in a document.

6 Conclusion and Future Work

We proposed a new formulation of the relation extraction task, where the relations are more general than intra-sentence relations in the sense that they may span multiple sentences (**cross-sentence**) and may have more than two arguments (**N-ary**). Also, these relations are more specific than corpus-level relations because their scope is only limited within a document. A novel approach as well as new schemes for annotation and evaluation were proposed for this proposed formulation. We designed a sequence representation for characterizing instances of such relations and explored various classifiers whose features are derived from this sequence representation. We also designed the Constrained Subsequence Kernel for the SVM classifier. We evaluated our approach on three datasets across two domains. In future, we plan to explore various directions identified in the results analysis section.

References

1. Berger, A.L., Pietra, V.J.D., Pietra, S.A.D.: A maximum entropy approach to natural language processing. Computational linguistics **22**(1), 39–71 (1996)
2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 1247–1250. AcM (2008)
3. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**, 27:1–27:27 (2011), software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
4. Chollet, F., et al.: Keras. `https://github.com/keras-team/keras` (2015)
5. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**(3), 273–297 (1995)
6. Delėger, L., Bossy, R., Chaix, E., Ba, M., Ferrė, A., Bessieres, P., Nėdellec, C.: Overview of the bacteria biotope task at bionlp shared task 2016. In: Proceedings of the 4th BioNLP Shared Task Workshop. pp. 12–22 (2016), `http://2016.bionlp-st.org/tasks/bb2`
7. Doddington, G.R., Mitchell, A., Przybocki, M.A., Ramshaw, L.A., Strassel, S., Weischedel, R.M.: The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation. In: LREC. vol. 2, p. 1 (2004)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
9. Hoffmann, R., Zhang, C., Ling, X., Zettlemoyer, L., Weld, D.S.: Knowledge-based weak supervision for information extraction of overlapping relations. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. pp. 541–550. Association for Computational Linguistics (2011)
10. Lever, J., Jones, S.J.: Verse: Event and relation extraction in the bionlp 2016 shared task. In: Proceedings of the 4th BioNLP Shared Task Workshop. pp. 42–49 (2016)

11. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2. pp. 1003–1011. Association for Computational Linguistics (2009)
12. Miwa, M., Bansal, M.: End-to-end relation extraction using lstms on sequences and tree structures. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 1105–1116 (2016)
13. Mooney, R.J., Bunescu, R.C.: Subsequence kernels for relation extraction. In: Advances in neural information processing systems. pp. 171–178 (2005)
14. Pawar, S., Palshikar, G.K., Bhattacharyya, P.: Relation extraction : A survey. arXiv preprint arXiv:1712.05191 (2017)
15. Peng, N., Poon, H., Quirk, C., Toutanova, K., Yih, W.t.: Cross-sentence n-ary relation extraction with graph lstms. Transactions of the Association for Computational Linguistics **5**, 101–115 (2017)
16. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
17. Quirk, C., Poon, H.: Distant supervision for relation extraction beyond the sentence boundary. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. pp. 1171–1182 (2017)
18. Riedel, S., Yao, L., McCallum, A.: Modeling relations and their mentions without labeled text. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 148–163. Springer (2010)
19. Song, L., Zhang, Y., Wang, Z., Gildea, D.: N-ary relation extraction using graph-state lstm. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 2226–2235 (2018)
20. Sundheim, B.M.: Overview of results of the muc-6 evaluation. In: Proceedings of a workshop on held at Vienna, Virginia: May 6-8, 1996. pp. 423–442. Association for Computational Linguistics (1996)
21. Surdeanu, M.: Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In: TAC (2013)
22. Surdeanu, M., Tibshirani, J., Nallapati, R., Manning, C.D.: Multi-instance multi-label learning for relation extraction. In: Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning. pp. 455–465. Association for Computational Linguistics (2012)
23. Swampillai, K., Stevenson, M.: Extracting relations within and across sentences. In: RANLP. pp. 25–32 (2011)
24. Zhou, G., Su, J., Zhang, J., Zhang, M.: Exploring various knowledge in relation extraction. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05). pp. 427–434. Association for Computational Linguistics, Ann Arbor, Michigan (June 2005). https://doi.org/10.3115/1219840.1219893, `http://www.aclweb.org/anthology/P05-1053`

Appendix

Alias Detection Rules

We use a few high-precision rules for identifying aliases. The information of aliases is needed at several stages of our approach:

• Identifying positive / negative instances for training the classifiers: Alias information is needed to check whether any candidate relation instance is similar to any relation instance listed in gold-annotations (i.e. in LR_i). Only when it is not similar to any of the relation instances in LR_i, it is treated as a negative instance for classifiers.
• Construction of sequence representations for candidate relation instances: Alias information is needed for adding tokens of the type E_i and OE_{ET_j} at appropriate positions in sequence representations, as described in Section 3.2.
• Evaluation: Alias information is also needed for evaluation as described in Section 2.

Following are the details of the high-precision rules used for identifying aliases.

Bacteria Biotope dataset: Two entity mentions are aliases of each other if:

- The two entity mentions are exactly same
- If the first word of one entity mention is a short-form of the first word of another entity mention and rest all the words in these mentions are same. Here, short-form is just the first character followed by "." (e.g. `Salmonella Typhimurium` ↔ `S. Typhimurium`)
- One entity mention is prefix of another and its length is more than half of the length of the other entity mention

The above rules are used while constructing sequence representations as well as while evaluation.

Drug-Gene-Mutation dataset: Two entity mentions are aliases of each other only if one of the entity mentions is prefix of another. This rule is used while constructing sequence representations as well as while evaluation.

MUC-6 dataset: Two entity mentions are aliases of each other if:

- The two entity mentions are exactly same
- Any one is the prefix of the other
- Special case: one starts with `Chief Executive` and another is `CEO`
- If one entity mention has the first word as `Mr.` or `Ms.` and its last word is the suffix of another mention
- Except for the case where both the entity mentions are of type *POST*: if one entity mention is suffix of another

The above rules are used while evaluation. In addition to the above rules, we also use Stanford CoreNLP coreferences for construction of sequence representations, i.e. for adding tokens of the type E_i and OE_{ET_j} at appropriate positions in sequence representations.