SRCB Entity Discovery and Linking (EDL) and Event Nugget Systems for TAC 2017

Shanshan Jiang, Yihan Li, Tianyi Qin, Qian Meng, Bin Dong

Ricoh Software Research Center (Beijing) Co.,Ltd.

Room 2801, 28th Floor, Tengda Plaza, No.168, Xiwai Street, Haidian District, Beijing, China

{shanshan.jiang, yihan.li, tianyi.qin, qian.meng, bin.dong}@srcb.ricoh.com

Abstract

The SRCB team participated in Entity Discovery and Linking task and Event Nugget Detection task in TAC Knowledge Base Population (KBP) 2017. The EDL system includes English and Chinese entity discovery, candidate generation, entity linking and NIL cluster. The Event Nugget Detection system identifies event nugget mentions and puts them into co-reference chains. We develop event nugget system based in English and Chinese.

1 Introduction

In this paper, we describe the SRCB team's participation in TAC KBP 2017 event nugget track and entity discovery and linking task.

We participate in event nugget detection and coreference for both English and Chinese task. For English task, we combines bidirectional LSTM models and Conditional Random Filed Models to detect event type. Feature based svm model was used to classify event realis. And we employ maximum entropy based classifier and sieve approach to detect event coreference. For Chinese task, svm model was employ to detect event type and realis.

We also participate in entity discovery and linking task in both English and Chinese. We use a Bi-directional LSTM model for entity discovery. And we use a retrieve-based method to realize the candidate generation. The entity linking problem is treated as a binary classification problem. Finally, we use a rule-based method to go the clustering.

The paper is organized as follows: Section 2 describes our method in event nugget task. Section 3 describes the entity discovery and linking system. Section 4 concludes the paper.

2 Event Detection and Coreference

The SRCB team participates in the event nugget task of TAC-KBP 2017. The task we participate consists of two subtasks: event nugget detection and event nugget coreference. The event nugget detection task aims to identify the selected event types and subtypes taken from Rich ERE annotation guidelines. Besides, the task is also required to identify three REALIS values for event mentions. The event coreference task aims to identify the coreference links between event mention instances within a document. The event nugget task consists of three languages (English, Chinese and Spanish). The SRCB team submitted systems for both English and Chinese.

For English tasks, we experiment with combination of two genre of models for nugget extractions: a neural network based event extraction system and a traditional Conditional Random Field (CRF) based event extraction system. For event coreference systems, we experiment with two approaches: maximum entropy based classifier and sieve approach. In addition, we train a SVM classifier for realis detection. The systems run in a pipelined version of the three stages.

For Chinese tasks, we extract features and train SVM model to measure the confidence of an instance to trigger an event. Realis of candidate triggers are also classified by SVM model. Event coreference is solved by rule based method.

2.1 Event Detection and Coreference for English

2.1.1 Event Type Detection Model

We trained an ensemble model that combine with neural network model and Conditional Random Field (CRF) model. In our initial experiments, we found that neural network model can result in more trigger candidates than CRF model, while CRF can detect trigger candidates with higher precision than neural network model. Thus, an ensemble model combined with neural network model and CRF model is supposed to yield better results than each individual model.

2.1.1.1 Neural Network Model

Recurrent neural networks (RNN) can capture historical contextual information based on memory cells, and thus they are often used to processing sequential data. To solve the long-term dependency problem, Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) structure is proposed. The LSTM maintain input gate, forget gate, update gate and output gate to maintain historical information and current information. The memory cell is elaborately designed to delivery message between memories. Previous researches (Chiu and Nichols, 2015; Zeng et al., 2016) have successfully employed RNN network to solve sequence labelling problem. In this paper, we followed the previous work (Graves et al., 2013) and adopted bidirectional LSTM (BiLSTM) networks. The BiLSTM summaries both past and future contexts for a given time, and thus more sentence-level information can be used for better prediction. Based on BiLSTM, we employed the following architecture for trigger identification and classification. Figure 1 list our architecture.

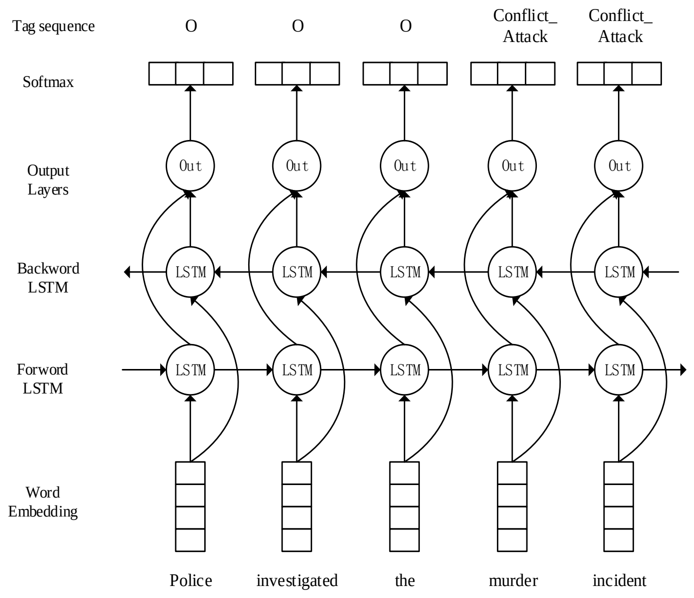

Figure 1. The main architecture of our BiLSTM model

As can be from Figure 1, our model read input sentence word by word. Words are converted to vectors by word embedding. Then, one bidirectional LSTM layer is constructed. A forward LSTM network computes the hidden state $\overrightarrow{h_t}$ of the past context of the sentence at word w_t, while a backward LSTM network reads the same sentence in reverse and outputs $\overleftarrow{h_t}$ given the future context. In our implementation, we concatenate these two vectors to form the hidden state of a BiLSTM network, i.e. $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. Then h_t is fed into a softmax layer to produce the log probabilities of each label for w_t. We treat trigger labelling task as a sequence labelling problem. If a word is an event trigger with type *type*, the label is *type*. Continuous words with the same type label are regarded as the same event with the specified type value. If a word is not an event trigger, the label is O. In this way, our labeling scheme can deal with multi-word triggers.

As there are 8 types and 18 subtypes defined for evaluation in 2017, and subtypes are unique to each other, thus once subtype is determined for one trigger, its type is also determined. In our labeling scheme, event types are set to predefined 18 subtypes. Since the number of each subtype in training data vary significantly, it is often hard for model to label subtypes which are with few instances in training data. As such, we adopt over-sampling technique to increase the numbers of subtypes with few instances.

2.1.1.2 Conditional Random Field Models

A linear chain CRF is employed to extract event mention span and type. The reason is that CRFs has strong reasoning ability, and be able to use complicated and overlapping and non-independent features for training and reasoning, to make full use of context information as features, can also add any other external features, so that the model can be obtained the information is very rich.

CRF, unlike LSTM and other models, can consider the long-term context information, it considers more the whole sentence of the local characteristics of the linear weighted combination (scanning the whole sentence through the feature template). The key point is that the CRF model is p (y | x, w), note that y and x are all sequences, it is somewhat like list wise, the optimization is a sequence of y = (y1, y2, ..., yn)， rather than the y_t of a certain moment, that is, find the highest probability sequence y = (y1, y2, …, yn) makes p(y1, y2, …, yn| x, w) is the highest, it calculates a joint probability, the optimization is the whole sequence (final goal), not the optimal splicing of each moment, at this point, CRF is better than LSTM. The accuracy of the result is higher than that of LSTM.

CRF Features include token, lemma, stemming, POS tag, dependency type, grammar, NER nearby, token position in sentence, sentence position in document, trigger word dictionary, and WordNet[1] index. Text processing employs Stanford CoreNLP[2] tookit.

2.1.1.3 Ensemble Model

Based on the above described neural network model and CRF model, we combine the outputs of those models to produce more reliable results.

First, we train an ensemble neural network model consisting of 10 BiLSTM models. The training data is split into 10 parts. For each BiLSTM model, 9 of 10 parts data are used as training data, and the remaining part data is used as validation data. Then, voting strategy is adopted to combine outputs of the 10 BiLSTM models. The final ensemble model is represented as en_BiLSTM.

Then, we combine outputs of en_BiLSTM and CRF models as final outputs. We employ the following strategy to combine outputs.

1) If an event trigger with type Type1 identified by en_BiLSTM is not in the outputs of CRF, add the event trigger to the final outputs;
2) If there is an event trigger with type Type1 identified by en_BiLSTM, and there is also the same event trigger with type Type2 identified by CRF, drop the event trigger with type Type1 and add the event trigger with type Type2 to the

[1] http://wordnet.princeton.edu/

[2] https://stanfordnlp.github.io/CoreNLP/

final outputs, regardless whether Type2 is the same with Type1 or not.

3) For the event triggers that are identified by CRF while are not identified by en_BiLSTM, add all these ones to the final outputs.

2.1.1.4 Realis Model

We build svm model to classify realis value for event trigger. The features includes:

1) Trigger: It indicates which trigger words it is.
2) Ner: NER tags for the current word and nearby words.
3) Pos: POS tags for the current word and nearby words.
4) Tense: Whether the word is ended with "ed" or not.
5) Synset: WordNet synsets for the current word.

The trained svm model assign a REALIS value for each identified trigger. The REALIS value include ACTUAL, OTHER and GENERIC.

2.1.2 Event Nugget Coreference

2.1.2.1 Sieve based method

We employ a 2-pass sieve method to cluster event nuggets into hoppers. The method was proved significantly efficient by UTD team @KBP2016 (Lu and Ng, 2016). Each pair of event nuggets which have the same subtype goes through the sieves to be classified whether to be coreferent or not. In each sieve, a 1-NN classifier is employed.

In the 1st sieve, given a test pair, training pairs that meet the following conditions are considered as neighbors: training pair's subtype is the same as test pair's; training pair's lemmatized nuggets are the same as test pair's; the sentence distance between two nuggets of training pair is in the range that of test pair plus or minus *m*, wherein *m* is a parameter. Jaccard distance of lemmatized sentences is used to measure the similarity between a training pair and a test pair.

Those test pairs which are determined not coreferent by the 1st sieve go to the 2nd sieve. In the 2nd sieve, which is not that strict as the 1st sieve, nuggets of test pair have the same lemma; nuggets of neighbor training pair have the same lemma, but not necessarily identical with test pair's; training pair's subtype is the same as test pair's. The same measure with the 1st sieve is utilized to calculate distance between test pair and training pair.

Test pairs of the same subtype which are coreferent with each other form a graph, in which vertices are nuggets and edges are coreferent relationship. Then each connected sub-graph is a hopper.

2.1.2.2 ME based method

The Maximum Entropy (ME) model is employed to extract event coreference. The methods used are as follows, each two mention span is divided into a pair, and classify each pair. Finally, the same class will be linked together. Features for ME models include lemma, stemming, sentence position in document, POS tag, dependency type, grammar, NER, NER nearby and the same WordNet.

2.1.3 Experiments

2.1.3.1 Setup

For development, we use LDC2017E02 (2014 and 2015), LDC2015E29, and LDC2015E68 as training data, and LDC2017E02 (2016) as testing data. For evaluation on 2017 data, we further include LDC2016E31 and LDC2017E02 (2016) as training data.

To properly deal with English words, 50,000 most frequent words are considered respectively. The English tokens and Arabic numerals in sentences are ignored since there is only a small fraction of these tokens. We insert starting and ending tokens for every sentence. All the words that are not in the vocabulary are labeled by a special token "UNK".

In our experiments, word embeddings are pre-trained using Wikipedia English[3] data. And during the process of model training, word embeddings are

[3] https://en.wikipedia.org/wiki/Main_Page

allowed to be tuned by the neural models. Word embeddings and recurrent layers are set to 512 dimensions. We used SGD to optimize all parameters and mini batch size is set to 256. The initial learning rate is 0.5, moving average decay 0.9 once the current loss is highest in the last 5 updates. Most parameters are initialized by randomly sampling from a uniform distribution between -0.1 and 0.1. All our models were trained on a NVIDIA GeForce GTX 1080 GPU. The training stage of each model took about 1.5 hours.

2.1.3.2 Results and Analysis

SRCB submitted three runs to the EN evaluation this year (called srcb1, srcb2 and srcb3). We used the ensemble model in all three runs for event nugget detection. We tried different combination of parameters for neural network.

The best performance of the three runs on the 2017 official evaluation data for English are listed in Table 1. All the scores are computed using the official scorer this year. In the results, we found that the sieve based approach showed improvement over ME based method.

Table 1: Our best performance on the 2017 official evaluation data for English

	Micro Average			Macro Average		
Attributes	Prec	Rec	F1	Prec	Rec	F1
plain	68.04	66.53	67.27	68.07	68.04	68.06
mention_type	56.83	55.57	56.19	57.02	56.82	56.92
realis_status	47.95	46.89	47.42	48.77	48.73	48.75
mention_type+realis_status	39.69	38.81	39.24	40.47	40.17	40.32
Overall Average CoNLL score	35.33					

2.2 Event Detection and Coreference for Chinese

We extract triggers from training data and expand candidate trigger words by HIT synonym dictionary and custom news corpus, and filter candidate triggers by a TF-IDF like method. After finding all the instances of candidate triggers in data, we extract features and train a 2-categories SVM model to determine the confidence of a candidate trigger to trigger an event. Realis of candidate triggers are classified by SVM while coreference between triggers are evaluated by rule methods.

2.2.1 Event Nugget Detection Model

2.2.1.1 Event Triggers and Trigger Expand

Events are expressed as trigger words in text data. A type of event has its unique set of trigger words, and trigger words can define the type of an event as its descriptor. We can extract and keep a correspondence table between trigger words and event types by locating trigger words in training data.

Event triggers and their corresponding event types are annotated in training data of ACE 2005 and TAC KBP 2016. We can extract a correspondence table of event type and its triggers:

<TriggerWord,EventType>

In practice, there exists three main problems for extracting correspondence table: (1) Correspondence table can’t express bisection relations between trigger words and event types due to the polysemy phenomenon. For example, “离开”(which means leave in English) can trigger both Movement-TransportPerson events and Personnel-EndPosition events. Polysemy is more common in single character words in Chinese. (2) Trigger words in training data is too few to cover all the words that can trigger events in Chinese, causing a low recall rate. (3) Presence of a trigger word in data may probably not trigger its corresponding event, causing a low precision rate.

For the first problem, we designed a TF-IDF like method to determine the confidence of a "trigger-event type" relation, filter "trigger-event type" relations with lower confidence value, and keep the bisection between triggers and event types:

$$Confidence = ETF * EIDF$$

$$ETF = \frac{freq(E\text{-}w)}{freq(E)}$$

$$EIDF = \log_2\left(\frac{N}{EDF}\right)$$

For the second problem, we believe words with the same semantics can trigger the same type of event. If one word is trigger word for a type of event, its synonyms can also trigger the same kind of event. After filtering the corresponding table by TF-IDF like method, we introduce semantic network resources to expand the corresponding table. With additional synonym of trigger words, the "trigger-event type" corresponding table can cover most trigger words in Chinese, leading to the solution to the second problem.

"Synonym dictionary of Harbin Institute of technology" offers a dictionary with 5 levels of coding of major category, middle category, minor category, word group and atomic word group. For example:

Hc27A01= 就职 到职 到任 上任 走马上任 下车 下车伊始 就任 赴任 走马赴任 新任

On basis of the synonym dictionary, we can expand new trigger words with a unique event type according to trigger words in training data. In practice, we iterate the top-4 bits of tags (Hc27 for Hc27A01) in synonym dictionary, if two or more words under a tag can trigger the same type of event in training data, then we add all the words under that tag to the correspondence table, as trigger words of the event type. For example, if both "就职" and "到任" are trigger words of event type "Personnel-EndPosition" in training data, we add all the words with tag "Hc27" to the correspondence table.

A large scale of trigger words will cause the third problem: presence of a trigger word in data may probably not trigger its corresponding event. To solve this problem, we build a two-categories SVM to determine the confidence of a candidate trigger word to trigger an event. We name each presence of words in correspondence table in training data a candidate trigger. We extract context features of candidate features to determine whether it triggers an event.

2.2.1.2 SVM Features

SVM model is trained for determine whether a candidate trigger is a real one, it needs features of candidate triggers. In the event nugget detection task, we focus on sentences with candidate triggers and extract POS, semantics, named entities, syntax, dependency and statistic features of candidate triggers on its corresponding context.

Candidate triggers and their corresponding event types may have hidden relationship in the same sentence. Triggers of the same event type may present repeatedly in the same sentence, and some event types have special relationships. For example, triggers of event type "Conflict-Attack" and event type "Life-Die" are more likely to co-present in the same sentence due to the fact that they have a cause and effect relationship. Therefore, besides POS, semantics, named entities, syntax, dependency and statistic features, we extract event types triggered by other candidate triggers in the same sentence with a candidate trigger as its context trigger features. The following table shows our features.

Table 2: Features for svm model

Feature	**Description**
pos0	POS of the candidate trigger
pos-5.....pos5	POS of words in candidate trigger's context
ner-5……ner5	Named entities in candidate trigger's context
index_article, index_sentence	Index of candidate trigger in article and sentence
length	Candidate trigger's characters
sem1...sem5	5-level codes of candidate trigger in "Synonym dictionary of Harbin Institute of technology"
conA0,conB0	Corresponding event type of candidate trigger
conA-3,conB-3......conA3,conB3	Context trigger type and subtype features
dep-2,dep-1,dep1,dep2	Dependency features between candidate features and other words in sentence.

2.2.1.3 REALIS detection

Similar to event nugget detection, we build a SVM model to perform the REALIS classification, where the features we use are listed in Table.

2.2.2 Event Coreference Model

Similar to the event coreference model in English event coreference task, except we lower the confidence of single character words in Chinese.

2.2.3 Experiments

We trained our SVM model on Chinese data of ACE 2005 and TAC 2016 in event nugget detection and REALIS detection. Since there exists 38 event types in ACE 2005 rather than 18 event types in TAC 2017, we re-annotated the event type tags in ACE 2005.

The approach and training datasets described above are valid for all three runs that we have submitted. Below we describe the run specific settings for each submission and report the official submission scores.

Submission 1. Training data includes modified ACE 2005 and TAC 2017, c=1.0, SVM model for REALIS detection.

Submission 2. Training data includes modified ACE 2005 and TAC 2017, c=4.0, SVM model for REALIS detection. Submission 2 is our best official submission.

Table 3: Results of Submission 2

	Micro Average			Macro Average		
Attributes	Prec	Rec	F1	Prec	Rec	F1
plain	47.48	46.76	47.12	47.05	47.86	47.45
mention_type	42.47	41.82	42.14	42.28	42.99	42.63
realis_status	33.80	33.29	33.54	33.07	33.70	33.38
mention_type+realis_status	30.66	30.19	30.42	30.08	30.74	30.40
Overall Average CoNLL score	24.15					

Submission 3. Training data includes modified ACE 2005 and TAC 2017, c=4.0, rule-based model for REALIS detection.

3 Tri-lingual Entity Discovery and Linking

3.1 System Architecture

3.1.1 Preprocessing

In the conventional approaches, entity discovery is the first step of the EDL system. Entity discovery is a supervised sequence labeling problem, and the preprocessing should satisfy specific tasks. In our system, the xml tags, text content between "<quote>" tags, page URL and several non-text symbols in the raw data are considered as noisy.

On Chinese data set, because the raw data are mainly written by simplified Chinese, all of the traditional Chinese characters need to normalize to simplified Chinese characters. In addition, there are many full-shaped alphabetic characters, numbers and symbols carried the same information as half-shaped ones, the normal way is convert them to half-shaped.

3.1.2 Mention Recognition

In this competition, besides the 5-class category (PER, LOC, ORG, GPE, FAC), a 2-class classification (NOM, NAM) was also required. It was straight forward to training two models separately, however, we hired only one Bi-directional LSTM to model all these information. The 5-class category and 2-class category shared the same weights within the Bi-directional LSTM, and they had separate Softmax layer to perform classification of different types.

The Mention Recognition was treated as sequence labeling problem (Collobert et al., 2011; Lample et al., 2016). The target was to narrow the gap between the predicted sequence and the labeled sequence. Cross-Entropy was used to measure the difference, and the Adam algorithm was applied for optimizing. Additional to the methods above, we also used pre-trained word-embedding, character-level features (Chiu and Nichols, 2015; Santos and Guimaraes, 2015), capital features, CRF-Layer (Huang et al, 2015) to boost the basic model.

3.1.2.1 Pre-trained word-embedding

The Word-Embedding was initialized randomly or pre-trained using Gensim (Rehurek and Sojka, 2010). Comparisons were made between these two initialization methods. Unsurprisingly, the pre-trained Word-Embedding was far better. The dataset for training the Word Embedding came from Wiki. Redundant marks, short sentences and low frequency words were removed for cleaning. Stemming were performed. As the result, we got Word-Embedding of different dimensions for nearly 600,000 words.

3.1.2.2 Character-level feature

The Word-Embedding was aimed to capture semantic and syntax information while the character-level feature was targeting the morphological information of words (Chiu and Nichols, 2015; Santos and Guimaraes, 2015). Prefix and suffix were the two spot where the morphological information lies. Feature extracting windows were set for prefix and suffix separately. The size of these windows were fixed, so it possible to project the original information using a matrix into the feature space. The projection matrix was initialized randomly.

3.1.2.3 Capital feature

Capital features (Collobert et al., 2011) refer to three features: 1) If the letters within the current word were all capital letters; 2) If the letters within the current word were all lowercase; 3) If the first letter of the current word was uppercase, while the other letters were lowercase; These features were thought to be beneficial for entity names containing abbreviation. However, in our experiments, capital features could only make marginally improvement. A guess was that similar information had already discovered by the character-level feature detector.

3.1.2.4 CRF-Layer

BIO (Collobert et al., 2011) marking strategy was applied for labeling the entities. The beginning word within an entity name were marked as B (begin). Other words within an entity name were marked as I (inside). Word not within an entity name were marked as O (outside). The CRF-Layer were set to honor the BIO constraints where mark I could only appear after mark B, which could upgrade the accuracy of the system, but in our experiments, CRF-Layer could only benefit marginally. The reason for this phenomenon was that the Bi-directional LSTM was almost good enough to model the rules between B, I, O marks. It is very likely that no benefit will gain by adding a CRF-Layer to a full trained Bi-directional LSTM model.

3.1.3 Candidate Generation

In this paper, a retrieve-based system is proposed as the candidate generation module to generate candidates for each detected mention.

The input to this candidate generation step is a detected mention, the output from this step is a candidate list, which consists of a list of related Wikipedia entities possibly matching this mention.

Firstly, each mention is first expanded into a list of different queries based on some pre-defined rules. These queries are assumed to represent different ways to rename the same entities. For example, given a detected mention "Steve", we need to expand it to generate a list of different queries, which may include Steve Jobs, Steve Nash, etc.

Original query is expanded following the query expansion steps below:

- The original detected mention is added to the query list.
- If any longer mention in the same source document contains the original mention, all of these longer mentions are added to the query list. For a mention like "Steve", if another mention "Steve Nash" is found from the same document, then "Steve Nash" is added to the query list of mention "Steve".
- If a detected mention is NOM, the nearest NAM mention is added to the query list. For example. If the detected mention is "president" and the nearest NAM mention is "Barack Obama", then "Barack Obama" is added to the query list.

After the query list is ready, the candidate entity list is generated from the retrieve results in the name of Wikipedia entities. To improve the recall of the results, fuzzy search and partial matching is imported in this retrieve step. Finally, the top N records from fuzzy query retrieve and match query retrieve is combined as the list of candidate entities. If the candidate entity is a disambiguation page in Wikipedia, add the outlink of the page is added to the candidate entity list.

3.1.4 Entity Linking

3.1.4.1 Feature

In this paper, some well-established features are used in the entity linking.

1. Mention String Comparison: The mention string comparison between the entity mention and the candidate entity name is the most direct feature in entity linking.

- Whether the entity mention exactly matches the candidate entity name.
- Whether the candidate entity name starts with or ends with the entity mention.
- Whether the candidate entity name is the prefix of or postfix of the entity mention.
- Whether the entity mention is wholly contained in the candidate entity name, or vice-versa.
- Whether all of the letters of the entity mention are found in the same order in the candidate entity name.
- The ratio of same words to the shorter between the entity mention and the candidate entity name.

- The ratio of the recursively longest common subsequence to the shorter among the entity mention and the candidate entity name.

2. Entity Popularity: The popularity of the candidate entity on the mention is another very useful feature in entity linking. However, this prior probability is hard to calculate due to the lack of training data in target languages. A compromised entity popularity is used here. The entity popularity is defined as the normalized inlink number of candidate entity in the entity list.

3. Entity Type: This feature is to indicate whether the type of the entity mention (i.e., people, location, and organization) in text is consistent with the category of the candidate entity in the knowledge base. The entity type feature for each candidate entity is defined as the sum of the conditional probability of the category of the candidate entity in Wikipedia on its detected mention type.

4. Word Vector Feature: given the mention string w and the candidate entity e, this returns the similarity of the two corresponding word vectors. Similarity is calculated as cosine of the angle between two vectors. If the mention string contains multiple words, we sum their word vectors on each dimension.

5. Topic Model Feature: similar with word vector feature, given the mention string w and the candidate entity e, this returns the cosine similarity of the two corresponding topic distributions.

3.1.4.2 Linking as Regression

The correctness of candidate entities can be considered as labels for an entity-mention pair. For each pair, if the entity is the correct one, the label is 1. Otherwise, the label is 0. Then, this label can be predicted by a classification model, which can be learned with supervised model. In the entity linking step, the model need to select the "unique" correct candidate entity for each mention. So, we use a regression model to predict the label, and choose the candidate entity with the highest prediction result. To predict the final result, these features, explained in Section 3.3 are fed into a regular feedforward neural network, to compute a matching score.

For the NIL mention, we employ a NIL threshold to predict the unlinkable entity mention. If the score of the top candidate entity is smaller than the NIL threshold, then return NIL for the entity mention, predict the mention m as unlinkable. The NIL threshold is also learned from the training data.

3.1.4.3 NIL Clustering

We use simple rule based method to do the clustering, from the participating team in 2016, USTC NELSLIP, (Liu et al., 2016), to conduct NIL clustering. The method contains two rules: them:

- Different named NIL mentions are grouped into one cluster only if their mention strings are the same (case insensitive);
- The nominal NIL mention is always grouped to its nearest named mention with the same mention type.

3.2 Experiments

Data in LDC2017E03 were chosen as the developing dataset. To be specific, data of 2014 and 2015 were the training dataset, while data of 2016 were the validation dataset. Also, Wiki data were included for the word embedding and LDA training. Evaluation score in validation dataset is shown as below:

Table 4: score on EDL 2016 dataset

	P	R	F
Mention evaluation	0.85	0.625	0.721
Linking evaluation	0.674	0.569	0.617
Clustering evaluation	0.702	0.694	0.645

4 Conclusions

In this paper, we describe the submissions of SRCB in event nugget task and entity discovery and linking task.

In event nugget task, we built event trigger detection systems for both Chinese and English tasks, and evaluated the performance using TAC

2017 corpus. For English tasks, the ensemble model of BiLSTM and CRF achieves significantly better results than other models for event detection and event type classification. We also found that the sieve based approach perform better than ME based method for event coreference. For Chinese tasks, we primly used svm based method and achieved competitive results.

In entity discovery and linking task, our system mainly contains a mention recognition model based on BiLSTM, a retrieve-based candidate generation method, a binary classification entity linking, and a rule-based NIL clustering.

5 References

Jing Lu and Vincent Ng. 2016. UTD's Event Nugget Detection and Coreference System at KBP 2016.

Chiu, J. P., & Nichols, E. 2015. Named entity recognition with bidirectional LSTM-CNNs. arXiv preprint arXiv:1511.08308.

Zeng, Y., Luo, B., Feng, Y., & Zhao, D. 2016. WIP Event Detection System at TAC KBP 2016 Event Nugget Track.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. Neural computation 9(8):1735–1780.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 6645–6649.

Collobert R, Weston J, Bottou L, et al. Natural language processing (almost) from scratch[J]. Journal of Machine Learning Research, 2011, 12(Aug): 2493-2537.

Lample G, Ballesteros M, Subramanian S, et al. 2016. Neural architectures for named entity recognition[J]. arXiv preprint arXiv:1603.01360.

Santos C N, Guimaraes V. 2015. Boosting named entity recognition with neural character embeddings[J]. arXiv preprint arXiv:1505.05008.

Huang Z, Xu W, Yu K. 2015. Bidirectional LSTM-CRF models for sequence tagging[J]. arXiv preprint arXiv:1508.01991.

Rehurek R, Sojka P. 2010. Software framework for topic modelling with large corpora[C]//In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks.