Aggregated Semantic Matching for Short Text Entity Linking

Feng Nie[1]*, **Shuyan Zhou**[2]*, **Jing Liu**[3]*, **Jinpeng Wang**[4], **Chin-Yew Lin**[4], **Rong Pan**[1]*
[1]Sun Yat-Sen University [2]Harbin Institute of Technology [3]Baidu Inc. [4]Microsoft Research Asia
{fengniesysu, alexisxy0418}@gmail.com,
liujing46@baidu.com, {jinpwa, cyl}@microsoft.com, panr@sysu.edu.cn

Abstract

The task of entity linking aims to identify concepts mentioned in a text fragments and link them to a reference knowledge base. Entity linking in long text has been well studied in previous work. However, short text entity linking is more challenging since the texts are noisy and less coherent. To better utilize the local information provided in short texts, we propose a novel neural network framework, Aggregated Semantic Matching (ASM), in which two different aspects of semantic information between the local context and the candidate entity are captured via representation-based and interaction-based neural semantic matching models, and then two matching signals work jointly for disambiguation with a rank aggregation mechanism. Our evaluation shows that the proposed model outperforms the state-of-the-arts on public tweet datasets.

1 Introduction

The task of entity linking aims to link a mention that appears in a piece of text to an entry (i.e. entity) in a knowledge base. For example, as shown in Table 1, given a mention *Trump* in a tweet, it should be linked to the entity *Donald Trump*[1] in Wikipedia. Recent research has shown that entity linking can help better understand the text of a document (Schuhmacher and Ponzetto, 2014) and benefits several tasks, including named entity recognition (Luo et al.) and information retrieval (Xiong et al., 2017b). The research of entity linking mainly considers two types of documents: long text (e.g. news articles and web documents) and short text (e.g. tweets). In this paper, we focus on short text, particularly tweet entity linking.

Tweet
The vile #Trump humanity raises its gentle face in Canada ... chapeau to #Trudeau
Candidates
Donald Trump, Trump (card games), ...

Table 1: An illustration of short text entity linking, with mention *Trump* underlined.

One of the major challenges in entity linking task is ambiguity, where an entity mention could denote to multiple entities in a knowledge base. As shown in Table 1, the mention *Trump* can refer to U.S. president *Donald Trump* and also the card name *Trump (card games)*. Many of recent approaches for long text entity linking take the advantage of global context which captures the coherence among the mapped entities for a set of related mentions in a single document (Cucerzan, 2007; Han et al., 2011; Globerson et al., 2016; Heinzerling et al., 2017). However, short texts like tweets are often concise and less coherent, which lack the necessary information for the global methods. In the NEEL dataset (Weller et al., 2016), there are only 3.4 mentions in each tweet on average. Several studies (Liu et al., 2013; Huang et al., 2014) investigate collective tweet entity linking by pre-collecting and considering multiple tweets simultaneously. However, multiple texts are not always available for collection and the process is time-consuming. Thus, we argue that an efficient entity disambiguation which requires only a single short text (e.g., a tweet) and can well utilize local contexts is better suited in real word applications.

In this paper, we investigate entity disambiguation in a setting where only local information is available. Recent neural approaches have shown their superiority in capturing rich semantic sim-

*Correspondence author is Rong Pan. This work was done when the first and second author were interns and the third author was an employee at Microsoft Research Asia.

[1]https://en.wikipedia.org/wiki/Donald_Trump

Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018), pages 476–485
Brussels, Belgium, October 31 - November 1, 2018.

ilarities from mention contexts and entity contents. Sun et al. (2015); Francis-Landau et al. (2016) proposed using convolutional neural networks (CNN) with Siamese (symmetric) architecture to capture the similarity between texts. These approaches can be viewed as **representation**-focused semantic matching models. The representation-focused model first builds a representation for a single text (e.g., a context or an entity description) with a neural network, and then conducts matching between the abstract representation of two pieces of text. Even though such models capture distinguishable information from both mention and entity side, some concrete matching signals are lost (e.g., exact match), since the matching between two texts happens after their individual abstract representations have been obtained. To enhance the representation-focused models, inspired by recent advances in information retrieval (Lu and Li, 2013; Guo et al., 2016; Xiong et al., 2017a), we propose using **interaction**-focused approach to capture the concrete matching signals. The interaction-focused method tries to build local interactions (e.g., cosine similarity) between two pieces of text, and then uses neural networks to learn the final matching score based on the local interactions.

The representation- and interaction-focused approach capture abstract- and concrete-level matching signal respectively, they would be complement each other if designed appropriately. One straightforward way to combine multiple semantic matching signals is to apply a linear regression layer to learn a static weight for each matching signal(Francis-Landau et al., 2016). However, we observe that the importance of different signals can be different case by case. For example, as shown in Table 1, the context word *Canada* is the most important word for the disambiguation of *Trudeau*. In this case, the concrete-level matching signal is required. While for the tweet "*#StarWars #theForceAwakens #StarWarsForceAwakens @StarWars*", *@StarWars* is linked to the entity *Star Wars*[2]. In this case, the whole tweet describes the same topic "Star Wars", thus the abstract-level semantics matching signal is helpful. To address this issue, we propose using a rank aggregation method to dynamically combine multiple semantic matching signals for disambiguation.

In summary, we focus on entity disambiguation by leveraging only the local information. Specifically, we propose using both representation-focused model and interaction-focused model for semantic matching and view them as complementary to each other. To overcome the issue of the static weights in linear regression, we apply rank aggregation to combine multiple semantic matching signals captured by two neural models on multiple text pairs. We conduct extensive experiments to examine the effectiveness of our proposed approach, ASM, on both NEEL dataset and MSR tweet entity linking (MSR-TEL for short) dataset.

[2]https://en.wikipedia.org/wiki/Star_Wars

2 Background

2.1 Notations

Given a tweet t, it contains a set of identified queries $Q = (q_1, ..., q_n)$. Each query q in a tweet t consists of m and ctx, where m denotes an entity mention and ctx denotes the context of the mention, i.e., a piece of text surrounding m in the tweet t. An entity is an unambiguous page (e.g., Donald Trump) in a referent Knowledge Base (KB). Each entity e consists of ttl and $desc$, where ttl denotes the title of e and $desc$ denotes the description of e (e.g., the article defining e).

2.2 An Overview of the Linking System

Typically, an entity linking system consists of three components: mention detection, candidate generation and entity disambiguation. In this section, we will briefly presents the existing solutions for the first two components. In next section, we will introduce our proposed aggregated semantic matching for entity disambiguation.

2.2.1 Mention Detection

Given a tweet t with a sequence of words $w_1, ..., w_n$, our goal is to identify the possible entity mentions in the tweet t. Specifically, every word w_i in tweet t requires a label to indicate that whether it is an entity mention word or not. Therefore, we view it as a traditional named entity recognition (NER) problem and use BIO tagging schema. Given the tweet t, we aim to assign labels $y = (y_1, ..., y_n)$ for each word in the tweet t.

$$y_i = \begin{cases} B & w_i \text{ is a begin word of a mention,} \\ I & w_i \text{ is a non-begin word of a mention,} \\ O & w_i \text{ is not a mention word.} \end{cases}$$

In our implementation, we apply an LSTM-CRF based NER tagging model which automatically

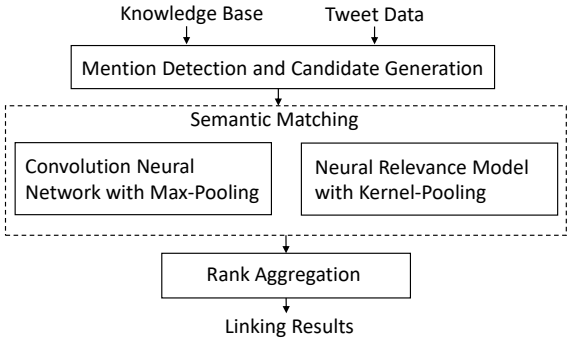


Figure 1: An overview of aggregated semantic matching for entity disambiguation.

learns contextual features for sequence tagging via recurrent neural networks (Lample et al., 2016).

2.2.2 Candidate Generation

Given a mention m, we use several heuristic rules to generate candidate entities similar to (Bunescu and Pasca, 2006; Huang et al., 2014; Sun et al., 2015). Specifically, given a mention m, we retrieve an entity as a candidate from KB, if it matches one of the following conditions: (a) the entity title exactly matches the mention, (b) the anchor text of the entity exactly matches the mention, (c) the title of the entity's redirected page exactly matches the mention Additionally, we add a special candidate **NIL** for each mention, which refers to a new entity out of KB. Given a mention, multiple candidates can be retrieved. Hence, we need to do entity disambiguation.

3 Aggregated Semantic Matching Model

We investigate entity disambiguation using only local information provided in short texts in this paper. Here, the local information includes a mention and its context in a tweet. Similar to (Francis-Landau et al., 2016), given a query q and an entity e, we consider semantic matching on the four text pairs for disambiguation: (1) the similarity $sim(m, ttl)$ between the mention and entity title, (2) the similarity $sim(m, desc)$ between the mention and entity description, (3) the similarity $sim(ctx, desc)$ between the context and entity description, (4) the similarity $sim(ctx, ttl)$ between the context and entity description. Fig. 1 illustrates an overview of our proposed Aggregated Semantic Matching for entity disambiguation. First, we use a representation-focused model and an interaction-focused neural model for semantic matching on four text pairs. Then, we introduce a pairwise rank aggregation to combine multiple semantic matching signals captured by the two neural models on four text pairs.

3.1 Semantic Matching

Formally, given two texts T_1 and T_2, the semantic similarity of the two texts is measured as a score produced by a matching function based on the representation of each text:

$$match(T_1, T_2) = F(\Phi(T_1), \Phi(T_2)) \quad (1)$$

where Φ is a function to learn the text representation, and F is the matching function based on the interaction between the representations.

Existing neural semantic matching models can be categorized into two types: (a) the representation-focused model which takes a complex representation learning function and uses a relatively simple matching function, (b) the interaction-focused model which usually takes a simple representation learning function and uses a complex matching function. In the remaining of this section, we will present the details of a representation-focused model (M-CNN) and an interaction-focused model (K-NRM). We will also discuss the advantages of these two models in the entity linking task.

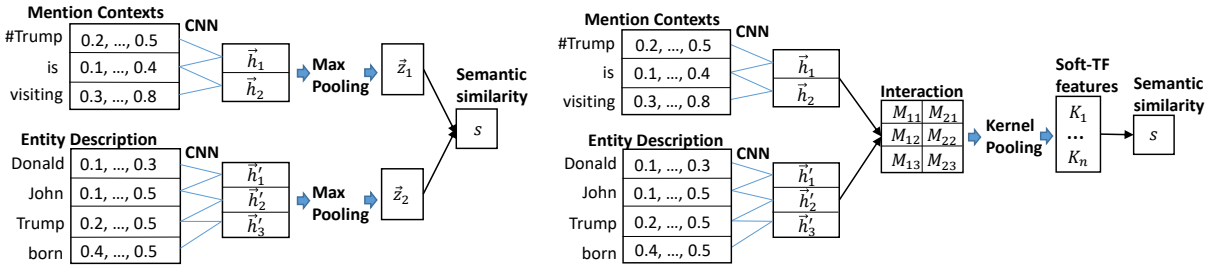
3.1.1 Convolution Neural Matching with Max Pooling (M-CNN)

Given two pieces of text $T_1 = \{w_1^1, ..., w_n^1\}$ and $T_2 = \{w_1^2, ..., w_m^2\}$, M-CNN aims to learn compositional and abstract representations (Φ) for T_1 and T_2 using a convolution neural network with a max pooling layer(Francis-Landau et al., 2016).

Figure 2a illustrates the architecture of M-CNN model. Given a sequence of words $w_1, ..., w_n$, we embed each word into a d dimensional vector, which yields a set of word vectors $v_1, ..., v_n$. We then map those word vectors into a fixed-size vector using a convolution network with a filter bank $M \in \mathbb{R}^{u \times d}$, where window size is l and u is the number of filters. The convolution feature matrix $H \in \mathbb{R}^{k \times (n-l+1)}$ is obtained by concatenating the convolution outputs $\overrightarrow{h}_i$:

$$\begin{aligned} \overrightarrow{h_j} &= max\{0, Mv_{j:(j+l)}\} \\ H &= [\overrightarrow{h}_1, ..., \overrightarrow{h}_{n-l+1}] \end{aligned} \quad (2)$$

where $v_{j:j+l}$ is a concatenation of the given word vectors and the max is element-wise. In this way, we extract word-level n-gram features of T_1 and



(a) M-CNN Model (b) K-NRM Model

Figure 2: The Architecture of models.

T_2 respectively. To capture the distinguishable information of T_1 and T_2, a max-pooling layer is applied and yields a fixed-length vector $\overrightarrow{z_1}$ and $\overrightarrow{z_2}$ for T_1 and T_2. The semantic similarity between T_1 and T_2 is measured using a cosine similarity $match(T_1, T_2) = cosine(\overrightarrow{z_1}, \overrightarrow{z_2})$.

In summary, M-CNN extracts distinguishable information representing the overall semantics (i.e. representations) of a string text by using a convolution neural network with max-pooling. However, the concrete matching signals (e.g., exact match) are lost, as the matching happens after their individual representation. We therefore introduce an interaction-focused model to better capture the concrete matching in the next section.

3.1.2 Neural Relevance Model with Kernel Pooling (K-NRM)

As shown in Fig. 2b, K-NRM captures the local interactions between T_1 and T_2 , and then uses a kernel-pooling layer (Xiong et al., 2017a) to softly count the frequencies of the local patterns. The final matching score is conducted based on the patterns. Therefore, the concrete matching information is captured.

Different from M-CNN, K-NRM builds the local interactions between T_1 and T_2 based on the word-level n-gram feature matrix calculated in Eq. 2. Formally, we construct a translation matrix M, where each element in M is the cosine similarity between an n-gram feature vector $\overrightarrow{h}^q_i$ in T_1 and an n-gram feature vector $\overrightarrow{h}^e_j$ in T_2, calculated as $M_{ij} = cosine(\overrightarrow{h}^q_i, \overrightarrow{h}^e_j)$.

Then, a scoring feature vector $\phi(M)$ is generated by a kernel-pooling technique.

$$\phi(M) = \sum_{i=1}^{n-l+1} \sqrt{\overrightarrow{K}(M_i)}$$
$$\overrightarrow{K}(M_i) = \{K_1(M_i), ..., K_K(M_i)\} \quad (3)$$

where $\overrightarrow{K}(M_i)$ applies K kernels to the i-th row of the translation matrix, and generates a $K-$dimensional scoring feature vector for the i-th n-gram feature in the query. The sqrt-sum of the scoring feature vectors of all n-gram features in query forms the scoring feature vector ϕ for the whole query, where the sqrt reduces the range of the value in each kernel vector. Note that the effect of $\overrightarrow{K}$ depends on the kernel used. We use the RBF kernel in this paper.

$$K_k(M_i) = \sum_j exp(\frac{-(M_{ij} - \mu_k)^2}{2\sigma^2}) \quad (4)$$

The RBF kernel K_k calculates how pairwise similarities between n-gram feature vectors are distributed around its mean μ_k: the more similarities closed to its mean μ_k, the higher the output value is. The kernel functions act as 'soft-TF' bins, where μ defines the similarity level that 'soft-TF' focuses on and σ defines the range of its 'soft-TF' count. Then the semantic similarity is captured with a linear layer $match(T_1, T_2) = w^T\phi(M)+b$, where $\phi(M)$ is the scoring feature vector.

In summary, K-NRM captures the concrete matching signals based on word-level n-gram feature interactions between T_1 and T_2. In contrast, M-CNN captures the compositional and abstract meaning of a whole text. Thus, we produce the semantic matching signals using both models to capture different aspect of semantics that are useful for entity linking.

3.2 Normalization Scoring Layer

We compute 4 types of semantic similarities between the query q and the candidate entity e (e.g., $sim(m, tit)$, $sim(m, desc)$, $sim(ctx, tit)$, $sim(ctx, desc)$) with the above two semantic matching models. We obtain 8 semantic matching signals, denoted as $f_1(q, e), ..., f_8(q, e)$ in to-

tal. The normalized ranking score for each semantic matching signals $f_i(q, e)$ is calculated as

$$s_i(q, e, f) = \frac{exp(f_i(q, e))}{\sum_{e'} exp(f_i(q, e'))} \quad (5)$$

where e' stands for any of the candidate entities for the given mention m. We then produce 8 semantic matching scores for each candidate entity of m, denoted as $S_{q,e} = \{s_1, ..., s_8\}$.

3.3 Rank Aggregation

Given a query q, we obtain multiple semantic matching signals for each entity candidate after the last step. To take advantage of different semantic matching models on different text pairs, a straightforward approach is using a linear regression layer to combine multiple semantic matching signals (Francis-Landau et al., 2016). The linear combination learns a static weight for each matching signal. However, as we pointed out previously, the importance of different signals varies for different queries. In some cases, the abstract-level signals are important. While the concrete-level signals are more important in other cases. To address this issue, we introduce a pairwise rank aggregation method to aggregate multiple semantic matching signals.

In the area of information retrieval, rank aggregation is combining rankings from multiple retrieval systems and producing a better new ranking (Carterette and Petkova, 2006). In our problem, given a query q, we have one ranking of the entity candidates for each semantic matching signal. We aim to find the final ranking by aggregating multiple rankings. Specifically, given a ranking of entities for one semantic matching signal, $e_1 \succ e_2 \succ e_3 \ldots$, where $i \succ j$ means entity i is ranked above j, we extract all entity pairs (e_i, e_j) from the ranking and assume that if $e_i \succ e_j$, then e_i is preferred to e_j. We union all pairwise preferences generated from multiple rankings as a single set, from which the final ranking is learned. In this paper, we apply TrueSkill (Herbrich et al., 2006) which is a Bayesian skill rating model. We present a two-layer version of TrueSkill with no-draw.

TrueSkill assumes that the practical performance of each player in a game follows a normal distribution $N(\mu, \sigma^2)$, where μ means the skill level of the player and σ stands for the uncertainty of the estimated skill level. Basically, TrueSkill learns the skill levels of players by leveraging Bayes' theorem. Given the current estimated skill levels of two players (prior probability) and the outcome of a new game between them (likelihood), TrueSkill model updates its estimation of player skill levels (posterior probability). TrueSkill updates the skill level μ and the uncertainty σ intuitively: (a) if the outcome of a new competition is expected, i.e., the player with higher skill level wins the game, it will cause small updates in skill level μ and uncertainty σ; (b) if the outcome of a new competition is unexpected, i.e., the player with lower skill level wins the game, it will cause large updates in skill level μ and uncertainty σ. According to these intuitions, the equations to update the skill level μ and uncertainty σ are as follows:

$$\begin{aligned}
\mu_{winner} &= \mu_{winner} + \frac{\sigma^2_{winner}}{c} * v(\frac{t}{c}, \frac{\varepsilon}{c}) \\
\mu_{loser} &= \mu_{loser} - \frac{\sigma^2_{loser}}{c} * v(\frac{t}{c}, \frac{\varepsilon}{c}) \\
\sigma^2_{winner} &= \sigma^2_{winner} * [1 - \frac{\sigma^2_{winner}}{c^2} * w(\frac{t}{c}, \frac{\varepsilon}{c})] \\
\sigma^2_{loser} &= \sigma^2_{loser} * [1 - \frac{\sigma^2_{loser}}{c^2} * w(\frac{t}{c}, \frac{\varepsilon}{c})]
\end{aligned} \quad (6)$$

where $t = \mu_{winner} - \mu_{loser}$ and $c^2 = 2\beta^2 + \sigma^2_{winner} + \sigma^2_{loser}$. Here, ε is a parameter representing the probability of a draw in one game, and $v(t, \varepsilon)$ and $w(t, \varepsilon)$ are weighting factors for skill level μ and standard deviation σ respectively. β is a parameter representing the range of skills. In this paper, we set the initial values of the skill level μ and the standard deviation σ of each player the same as the default values used in (Herbrich et al., 2006). We use $\mu - 3\beta$ to rank entities following (Herbrich et al., 2006).

4 Experiments

In this section, we describe our experimental results on tweet entity linking. Particularly, we investigate the difference between two semantic matching models and the effectiveness of jointly combining these two semantic matching signals.

4.1 Datasets & Evaluation Metric

In our experiments, we evaluate our proposed model ASM on the following two datasets.

NEEL Weller et al. (2016)**.** We use the dataset of Named Entity Extraction & Linking Challenge 2016. The training dataset consists of 6,025 tweets and includes 6,374 non-NIL queries and 2,291

NIL queries. The validation dataset consists of 100 tweets and includes 253 non-NIL queries and 85 NIL queries. The testing dataset consists of 300 tweets and includes 738 non-NIL queries and 284 NIL queries.

MSR-TEL Guo et al. (2013)[3]. This dataset consists of 428 tweets and 770 non-NIL queries. Since the NEEL test dataset has distribution bias problem, we add MSR-TEL as another dataset for the evaluation. In the NEEL testing dataset, 384 out of 1022 queries refer to three entities: 'Donald Trump', 'Star Wars' and 'Star Wars (The Force Awakens)'.

In this paper, we use accuracy as the major evaluation metric for entity disambiguation. Formally, we denote N as the number of queries and M as the number of correctly linked mentions given the gold mention (the top-ranked entity is the golden entity), $accuracy = \frac{M}{N}$. Besides, we use precision, recall and F1 measure to evaluate the end-to-end system. Formally, we denote $N^{'}$ as the number of mentions identified by a system and $M^{'}$ as the correctly linked mentions. Thus, $precision = \frac{M^{'}}{N^{'}}$, $recall = \frac{M^{'}}{N}$ and $F1 = 2 * \frac{precision * recall}{precision + recall}$.

4.2 Data Preprocessing

Tweet data All tweets are normalized in the following way. First, we use the Twitter-aware tokenizer in NLTK[4] to tokenize words in a tweet. We convert each hyperlink in tweets to a special token *URL*. Since hashtags usually does not contain any space between words, we use a web service[5] to break hastags into tokens (e.g., the service will break '#TheForceAwakens' into 'the force awakens') by following (Guo et al., 2013). Regarding to usernames (@) in tweets, we replace them with their screen name (e.g., the screen name of the user '@jimmyfallon' is 'jimmy fallon').

Wikipedia data We use the Wikipedia Dump on December 2015 as the reference knowledge base. Since the most important information of an entity is usually at the beginning of its Wikipedia article, we utilize only the first 200 words in the article as its entity description. We use the default English word tokenizer in NLTK to do the tokenization for each Wikipedia article.

Word embedding We use the word2vec toolkit (Mikolov et al., 2013) to pre-train word embeddings on the whole English Wikipedia Dump. The dimensionality of the word embeddings is set to 400. Note that we do not update the word embeddings during training.

4.3 Experimental Setup

In our main experiment, we compare our proposed approaches with the following baselines: (a) The officially ranked 1st and 2nd systems in NEEL 2016 challenge. We denote these two systems as `Rank1` and `Rank2`. (b) `TagMe`. Ferragina and Scaiella (2010) is an end-to-end linking system, which jointly performs mention detection and entity disambiguation. It focuses on short texts, including tweets. (c) `Cucerzan`. (Cucerzan, 2007) is a supervised entity disambiguation system that won TAC KBP competition in 2010. (d) `M-CNN`. To the best of our knowledge, (Francis-Landau et al., 2016) is the state-of-the-art neural disambiguation model. (e) `Ensemble`. The rank aggregated combination of two `M-CNN` models with different random seeds.

To fairly compare with the baselines of `Cucerzan` and `M-CNN`, we use the same mention detection and candidate generation for them and our approaches. We train an LSTM-CRF based tagger (Lample et al., 2016) for mention detection by using the NEEL training dataset. The precision, recall, and F1 of mention detection on NEEL testing dataset are 96.1%, 89.2%, 92.6% respectively. The precision, recall, and F1 of mention detection on MSR-TEL dataset are 80.3% 83.8% and 82% respectively. As we described in the previous section, we use the heuristic rules for candidate generation. The recall of candidate generation on NEEL and MSR-TEL is 88.7% and 92.5%.

When training our model, we use the stochastic gradient descent algorithm and the AdaDelta optimizer (Zeiler, 2012). The gradients are computed via back-propagation. The dimensionality of the hidden units in convolution neural network is set to 300. All the parameters are initialized with a uniform distribution $U(-0.01, 0.01)$. Since there is NIL entity in the dataset, we tune a NIL threshold for the prediction of NIL entities according to the validation dataset.

4.4 Main Results

The end-to-end performance of various approaches on the two datasets is shown in Table 2. Since there are no publicly available codes of

[3]Guo et al. (2013) only used a subset of this dataset for evaluation. Instead, we test on the full dataset.

[4]Natural Language Toolkit. http://www.nltk.org

[5]http://web-ngram.research.microsoft.com/info/break.html

Methods	NEEL			MSR-TEL[6]		
	Precision	Recall	F1	Precision	Recall	F1
Rank 1	-	-	50.1	-	-	-
Rank 2	-	-	39.6	-	-	-
TagMe	25.3	62.9	36.2	14.5	**69.2**	23.8
Cucerzan	65.4	57.9	61.4	62.6	63.3	62.9
M-CNN	69.5	64.9	67.1	61.6	62.3	62.1
+pre-train	69.7	65.1	67.3	64.5	65.2	64.8
Ensemble	69.7	65.1	67.3	63.5	64.2	63.8
+pre-train	70.2	65.5	67.8	64.9	65.6	65.2
ASM	70.6	65.9	68.2	64.2	64.9	64.5
+pre-train	**72.2**	**67.4**	**69.7**	**66.2**	66.9	**66.5**

Table 2: End-to-end performance of the systems on the two datasets

Methods	NEEL	MSR-TEL
Cucerzan	65.4	75.5
M-CNN	72.8	74.7
+pre-train	72.9	77.6
Ensemble	72.9	76.4
+pre-train	73.5	78.1
ASM	73.9	77.4
+pre-train	**75.5**	**79.4**

Table 3: The accuracy of entity disambiguation with golden mentions on the two datasets.

	(m, ttl)	(ctx, desc)	All Pairs
M-CNN	64.8	66.7	72.8
K-NRM	64.1	66.8	72.7
ASM	**65.1**	**69.7**	**73.9**

Table 4: The performance of two semantic matching models and their combinations on NEEL dataset.

Rank1 and Rank2, we give only the F1 scores of these two systems on NEEL dataset according to Weller et al. (2016). Note that the baseline systems Rank1, Rank2 and TagMe use different mention detection.

The systems of Rank1, Rank2, TagMe and Cucerzan are feature engineering based approaches. The systems of M-CNN and ASM are neural based approaches. From Table 2, we can observe that neural based approaches are superior to the feature engineering based approaches. Table 2 also shows that ASM outperforms the neural based method M-CNN. Our proposed method ASM also shows improvements over Ensemble, which indicates the necessity of combining representation- and interaction-focused models in entity disambiguation.

Moreover, we pre-train both M-CNN, Ensemble and ASM by using 0.5 million anchors in Wikipedia, and fine-tune the model parameters using non-NIL queries in NEEL training dataset. From Table 2, we can observe that the performance of neural models will be improved by using pre-training. The results in Table 2 show that our proposed ASM is still superior to M-CNN and Ensemble in the setting of pre-training.

Since entity disambiguation is our focus, we also give the disambiguation accuracy of different approaches by using the golden mentions in Table 3. Similarly, we observe that our proposed ASM outperforms baseline systems.

4.5 Model Analysis

In this section, we discuss several key observations based on the experimental results, and we mainly report the entity disambiguation *accuracy* when given the golden mentions.

4.5.1 Effect of Different Semantic Matching Methods

We empirically analyze the difference between the two semantic matching models (M-CNN and K-NRM) and show the benefits when combing the semantic matching signals from these two models.

[6]Note that the performance of all systems on MSR-TEL dataset might be under estimated, since not all mentions in each tweet were manually annotated. For example, a correctly identified mention given by a system, which was not manually annotated, will be judged as wrong. But we still give the comparisons of different approaches on MSR-TEL dataset.

	M-CNN win	M-CNN loss
K-NRM win	58.3%	6.3%
K-NRM loss	5.8%	29.6%

Table 5: The win-loss analysis of M-CNN and K-NRM on the pair (ctx, desc).

Query:	the vile #Trump humanity raises its gentle face in Canada ... chapeau to #**Trudeau**,URL
M-CNN:	Kevin Trudeau
K-NRM:	Justin Trudeau
Query:	RT @ MingNa : What is my plan to avoid spoiler about #theForceAwakens ? No Internet except to post my @**StarWars**
M-CNN:	Star Wars
K-NRM:	Comparison of Star Trek and Star Wars

Table 6: The top-1 results of M-CNN and K-NRM using (ctx,desc) pair for two queries. Mention is in **bold** and the golden answer is underlined.

We first compare the performance of two semantic matching models over the two text pairs: (a) *(m, ttl)* and (b) *(ctx, desc)*. These two pairs presents two extreme of the information used in the systems: *(m, ttl)* consumes the minimum amount of information from a query and an entity, while *(ctx, desc)* consumes the maximum amount of information from a query and an entity. From the first two columns in Table 4, we can observe that M-CNN performs comparably with K-NRM on the two text pairs. ASM that combines the two models obtains performance gains on the two individual text pairs. The third column in Table 4 also shows that ASM gives performance gains when using all text pairs. This indicates that M-CNN and K-NRM capture complementary information for entity disambiguation.

Moreover, we observe that the performance gains are different on the two pairs *(m, ttl)* and *(ctx, desc)*. The gain on *(ctx, desc)* is relatively larger. This indicates that M-CNN and K-NRM capture more different information when the text is long. Additionally, we show the win-loss analysis of the two semantic matching model for non-NIL queries on *(ctx, desc)* in Table 5. The 12.1% (=6.3% + 5.8%) difference between these two models confirms the necessity of combination.

Method	Without Pre-Train		With Pre-Train	
	NEEL	MSR-TEL	NEEL	MSR-TEL
Linear	73.1	75.7	73.8	78.1
ASM	**73.9**	**77.4**	**75.5**	**79.4**

Table 7: Comparison of rank aggregation and linear combination on two datasets.

To further investigate the difference between the two semantic matching models on short text, we did case study. Table 6 gives two examples. In the first example, the correct answer is 'Justin Trudeau' which contains the words of 'Canada' and 'trump' in its entity description. However, M-CNN fails to capture this concrete matching information, since the concrete information of text might be lost after the convolution layer and max-pooling layer. In contrast, K-NRM builds the n-gram level local interactions between texts, and thus successfully captures the concrete matching information (e.g. exact match) that results in a correct linking result. In the second example, both candidate entities 'Star Wars' and 'Comparison of Star Trek and Star Wars' contains the phrase 'Star Wars' for multiple times in their entity descriptions. In this case, K-NRM fails to distinguish the correct entity 'Star Wars' from the wrong entity 'Comparision of Star Trek and Star Wars', because it relies too much on the soft-TF information for matching. However, the soft-TF information in the descriptions of the two entities is similar. In contrast, M-CNN captures the whole meaning of the text and links the mention to the correct entity. A detailed analysis of n-grams extracted from the M-CNN is provided in the Appendix.

4.6 Effect of Rank Aggregation

Table 4 shows that the combination of multiple semantic matching signals yields the best performance. Table 7 compares two different combination of M-CNN and K-NRM models, the result shows that the rank aggregation method outperforms the linear combination. The rank aggregation method dynamically summarizes win-loss results for each signal and generates the final overall ranking by considering all win-loss results. The improvement of our method over the linear combination confirms that the importance of different semantic signals varies for different queries, and our method is more suitable for combining multiple semantic signals.

5 Related Work

Existing entity linking methods can roughly fall into two categories. Early work focus on local approaches, which identifies one mention each time, and each mention is disambiguated separately using hand-crafted features (Bunescu and Pasca, 2006; Ji and Grishman, 2008; Milne and Witten, 2008; Zheng et al., 2010). While recent work on entity linking has largely focus on global methods, which takes the mentions in the document as inputs and find their corresponding entities simultaneously by considering the coherency of entity assignments within a document. (Cucerzan, 2007; Hoffart et al., 2011; Globerson et al., 2016; Ganea and Hofmann, 2017).

Global models can tap into highly discriminative semantic signals (e.g. coreference and entity relatedness) that are unavailable to local methods, and have significantly outperformed the local approach on standard datasets(Globerson et al., 2016). However, global approaches are difficult to apply in domains where only short and noisy text is available (e.g. tweets). Many techniques have been proposed to short texts including tweets. Liu et al. (2013) and Huang et al. (2014) investigate the collective tweet entity linking by considering multiple tweets simultaneously. Meij et al. (2012) and Guo et al. (2013) perform joint detection and disambiguation of mentions for tweet entity linking using feature based learning methods.

Recently, some neural network methods have been applied to entity linking to model the local contextual information. He et al. (2013) investigate Stacked Denoising Auto-encoders to learn entity representation. Sun et al. (2015); Francis-Landau et al. (2016) apply convolutional neural networks for entity linking. Eshel et al. (2017) use recurrent neural networks to model the mention contexts. Nie et al. (2018) uses a co-attention mechanism to select informative contexts and entity description for entity disambiguation. However, none of these methods consider combining representation- and interaction-focused semantic matching methods to capture the semantic similarity for entity linking, and use rank aggregation method to combine multiple semantic signals.

6 Conclusion

We propose an aggregated semantic matching framework, ASM, for short text entity linking. The combination of the representation-focused semantic matching method and the interaction-focused semantic matching method capture both compositional and concrete matching signals (e.g. exact match). Moreover, the pairwise rank aggregation is applied to better combine multiple semantic signals. We have shown the effectiveness of ASM over two datasets through comprehensive experiments. In the future, we will try our model for long text entity linking.

7 Acknowledgement

We thank the anonymous reviewers for their helpful comments. We also thank Jin-Ge Yao, Zhirui Zhang, Shuangzhi Wu and Yin Lin for helpful conversations and comments on the work.

References

R Bunescu and M Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *EACL*, Trento, Italy.

Ben Carterette and Desislava Petkova. 2006. Learning a ranking from pairwise preferences. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 629–630. ACM.

S Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, volume 2007.

Yotam Eshel, Noam Cohen, and Kira Radinsky. 2017. Named entity disambiguation for noisy text. In *CoNLL*, volume 2017.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of CIKM 2010*, pages 1625–1628.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *In Proceedings of NAACL-HLT 2016*, pages 1256–1261.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of ACL 2016*.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of CIKM 2016*, pages 55–64.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To link or not to link? a study on end-to-end tweet entity linking. In *NAACL-HLT 2013*.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *Proceeding of the SIGIR 2011*, pages 765–774.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *Proceedings of ACL 2013*.

Benjamin Heinzerling, Michael Strube, and Chin-Yew Lin. 2017. Trust, but verify better entity linking through automatic verification. EACL.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. Trueskill$^{\text{tm}}$: A bayesian skill rating system. In *In Proceedings of NIPS 2006.*, pages 569–576.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of EMNLP 2011*.

Hongzhao Huang, Yunbo Cao, Xiaojiang Huang, Heng Ji, and Chin-Yew Lin. 2014. Collective tweet wikification based on semi-supervised graph regularization. In *Proceedings of ACL 2014*.

Heng Ji and Ralf Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL 2008*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *In Proceedings of NAACL-HLT 2016*, pages 260–270.

Xiaohua Liu, Yitong Li, Haocheng Wu, Ming Zhou, Furu Wei, and Yi Lu. 2013. Entity linking for tweets. In *ACL (1)*, pages 1304–1311.

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *In Proceedings of NIPS 2006.*, pages 1367–1375.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*.

Edgar Meij, Wouter Weerkamp, and Maarten de Rijke. 2012. Adding semantics to microblog posts. In *Proceedings of the WSDM, 2012*, pages 563–572.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

David N. Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of CIKM 2008*.

Feng Nie, Yunbo Cao, Jinpeng Wang, Chin-Yew Lin, and Rong Pan. 2018. Mention and entity description co-attention for entity disambiguation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Michael Schuhmacher and Simone Paolo Ponzetto. 2014. Knowledge-based graph document modeling. In *Proceedings of CIKM 2014*, pages 543–552.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of IJCAI 2015*.

Katrin Weller, Aba-Sah Dadzie, and Danica Radovanovic. 2016. Making sense of microposts (#microposts2016) social sciences track. In *Proceedings of the 6th Workshop on 'Making Sense of Microposts' co-located with the 25th International World Wide Web Conference (WWW 2016), Montréal, Canada, April 11, 2016.*, pages 29–32.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017a. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of SIGIR 2017*, pages 55–64.

Chenyan Xiong, Russell Power, and Jamie Callan. 2017b. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1271–1279.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method.

Zhicheng Zheng, Fangtao Li, Minlie Huang, and Xiaoyan Zhu. 2010. Learning to link entities with knowledge base. In *NAACL-HLT 2010*.