Joint Recognition and Linking of Fine-Grained Locations from Tweets

Zongcheng Ji Aixin Sun Gao Cong Jialong Han
School of Computer Engineering, Nanyang Technological University, Singapore
{jizongcheng, jialonghan}@gmail.com, {axsun, gaocong}@ntu.edu.sg

ABSTRACT

Many users casually reveal their locations such as restaurants, landmarks, and shops in their tweets. Recognizing such fine-grained locations from tweets and then linking the location mentions to well-defined location profiles (*e.g.,* with formal name, detailed address, and geo-coordinates etc.) offer a tremendous opportunity for many applications. Different from existing solutions which perform location recognition and linking as two sub-tasks sequentially in a pipeline setting, in this paper, we propose a novel joint framework to perform location recognition and location linking simultaneously in a joint search space. We formulate this end-to-end location linking problem as a structured prediction problem and propose a beam-search based algorithm. Based on the concept of multi-view learning, we further enable the algorithm to learn from unlabeled data to alleviate the dearth of labeled data. Extensive experiments are conducted to recognize locations mentioned in tweets and link them to location profiles in Foursquare. Experimental results show that the proposed joint learning algorithm outperforms the state-of-the-art solutions, and learning from unlabeled data improves both the recognition and linking accuracy.

Keywords

Twitter; Tweet; POI; Location Recognition; Location Linking; Structured Prediction; Beam Search; Multi-view Learning

1. INTRODUCTION

Through tweets (*i.e.,* short messages up to 140 characters), users share their opinions, emotions, daily life activities and other information. In many tweets, users casually or implicitly reveal their locations at fine-grained granularity, such as restaurants, landmark buildings, and schools, to name a few. *Recognizing mentions of fine-grained locations, and more importantly, linking these mentions to well-defined locations* (*e.g.,* with formal name and geo-coordinates) offer a tremendous opportunity for many applications. Examples include user profiling, precise location-based services, and even disaster management [24, 28, 33].

Recognizing fine-grained location mentions and linking these mentions to well-defined locations are both challenging tasks. Here a fine-grained location is a *focused geographic entity* such as district, area, street, road or a *specific point location* such as hotels, landmarks, schools, shopping centers, and restaurants etc., similar to the definition of point-of-interest (POI) [24, 28, 37].[1] Due to the informal writing of tweets, locations in tweets are often mentioned in casual manner with incomplete name, nickname, acronym, or even self-defined short forms with ambiguity. For example, *mac* may refer to McDonald's chain restaurant or product from Apple. The word *popular* is often used by Singapore users to refer to the *Popular Bookstore* in Singapore. Even if we successfully recognize *mac* in a tweet as a location mention, linking this mention to a well-defined location (*i.e.,* a specific McDonald's branch with geo-coordinates in this case) remains challenging, because there could be many branches in a small geographical region. On the other hand, before location linking is possible, a collection of well-defined locations is essential in this task.

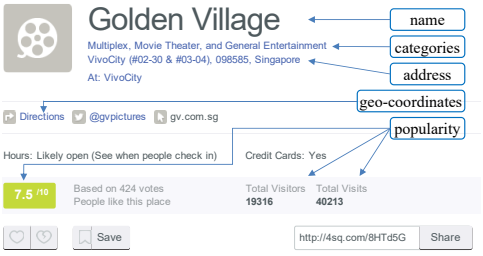


Figure 1: A location profile with attributes from Foursquare.

In this study, we utilized Foursquare[2] to get a large collection of well-defined locations. Foursquare offers a profile page for each location through crowdsourcing and check-ins. A *Location Profile* (LP) contains name, categories, address, geo-coordinates and ratings etc (see Figure 1). Note that, for the POIs with the same name, for example the chain movie theaters Golden Village in Singapore, distinct location profiles are created in Foursquare each with its own addresses and geo-coordinates. Figure 1 shows the location profile of Golden Village located in VivoCity, a popular shopping mall in Singapore.[3] To summarize, our task is ***end-to-end location linking***, *i.e.,* to recognize location mentions in tweets, and link mentions to well-defined locations, in the form of location profiles derived from Foursquare.

At first glance, our problem can be treated as a specific case of traditional end-to-end entity linking problem, because in our case only location entities are considered but not other types of entities like

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
http://dx.doi.org/10.1145/2872427.2883067.

[1] We use "fine-grained location" (or simply location) with "POI" interchangeably.
[2] `https://foursquare.com/about/`
[3] Golden Village: `http://4sq.com/8HTd5G` VivoCity: `http://4sq.com/6nH2dl`

persons and organizations. Existing solutions [10, 14, 34, 35] proposed for end-to-end entity linking often address the problem with two separate steps in a pipeline: first to recognize entity mentions and then to link the recognized entities. One major limitation of this pipelined framework is the error propagation from entity recognition to entity linking, and there is no feedback from the linking step to the recognition step. One type of errors is the recognition system may mistakenly predict two location mentions as a single mention. For example, it may predict "Golden Village VivoCity" as a single location mention, instead of two location mentions: "Golden Village" (the movie theater shown in Figure 1) and "VivoCity" (the shopping mall). In fact, there is no linkable location profile for "Golden Village VivoCity", but linkable location profiles for "Golden Village" and "VivoCity". Another kind of errors occurs if the recognition system leaves off one or more words from a multi-word mention, causing the linking system to link to no or a wrong location profile. The main challenge to fixing the two types of errors is to get feedback from entity linking to guide entity recognition. Although many studies attempt to jointly perform entity recognition and entity linking [16, 18, 30, 32, 43], we are aware of only one work [43] on fixing the aforementioned errors. Nevertheless, the solution in [43] still relies on the pipelined framework (see Section 2 for more details).

In this paper, we propose a novel joint model named **JoRL** (**Jo**int **R**ecognition and **L**inking), based on structured perceptron with inexact search [8, 19]. Our joint model is a single model, performing location mention recognition and linking *simultaneously*, which is significantly different from the pipelined approach where the two sub-tasks are carried out sequentially. In addition, the joint model makes it easy to introduce arbitrary global features to fix the aforementioned errors.

To train JoRL, sufficient labeled data is crucial. However, data labeling is tedious and time-consuming. Thus, it is worth considering semi-supervised learning to utilize unlabeled data. As mentioned earlier, our proposed joint model is based on structured perceptron [8, 19], which is an online algorithm and generates models that output merely a prediction with no additional confidence measures. Thus, the traditional bootstrapping methods (based on selecting unlabeled data whose predicted output has a high confidence score) cannot be directly applied. In this work, we employ the concept of multi-view learning [12], and implement the *consensus maximization principle* [4], *i.e.,* it minimizes the error on the labeled data and maximizes the agreement on the unlabeled data. With this principle, we do not need to know the confidence score of each predicted output, but to make consensus on the outputs of multiple independent hypotheses. Finally, we develop this principle into a semi-supervised structured perceptron algorithm, which generates much more accurate models with the help of unlabeled data. We summarize our contributions as follows.

- We propose and define an end-to-end location linking task, which consists of fine-grained location recognition from tweets and then linking the location mentions to well-defined location profiles collected from Foursquare.
- We propose a novel joint model, JoRL, which is based on structured perceptron with inexact search, to perform location recognition and linking simultaneously.
- We propose a semi-supervised structured perceptron algorithm, based on the concept of multi-view learning, to alleviate the dearth of labeled data.
- We conduct extensively experiments on tweets for fine-grained location recognition and linking to a collection of location profiles from Foursquare. Experiment results demonstrate the effectiveness of the proposed joint model.

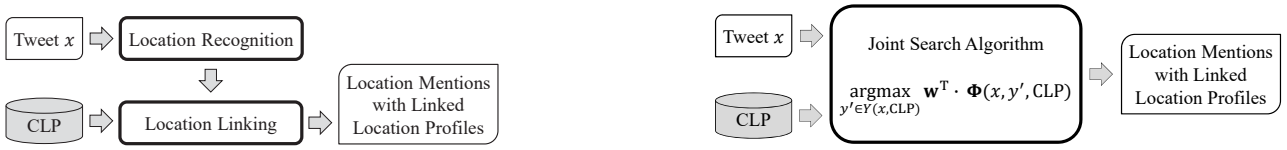
2. RELATED WORK

Location Extraction from Tweets. Extracting locations from tweets has attracted much attention recently [20, 24, 28, 36, 44]. By retraining off-the-shelf NER tools using annotated tweets, two studies show that the performance of extracting fine-grained location mentions remains inferior [28, 44]. Li and Sun [24] propose to build a POI inventory from Foursquare check-in tweets, and develop a time-aware POI tagger. The tagger identifies fine-grained location mentions and achieves state-of-the-art performance. All above work focuses on addressing the *geo/non-geo ambiguity*, which aims to predict whether a mention is a location [2]. The *geo/geo ambiguity* (*i.e.,* different locations share the same name), is not resolved. The two most related studies resolving geo/geo ambiguity using heuristic rules, are at coarse level of granularity [20, 36]. Different from these work, we aim to resolve the two levels of ambiguities simultaneously at a fine-grained level of granularity.

Other studies attempt to predict the locations of Twitter users [6, 7, 13, 17, 21, 31] or tweets [5, 15, 23, 27, 40]. Most solutions estimate a Twitter user's or a tweet's location at coarse level of granularity, ranging from country, state, to city levels. Here, we do not predict locations for Twitter users or tweets, but aim to extract and link the fine-grained location mentions from tweet content.

Entity Linking (EL) for Tweets. Tweet entity linking aims to link the named entities detected from tweets with their mapping entries in a knowledge base like Wikipedia. Some studies [29, 42] assume that the named entities are pre-detected, and focus on developing techniques to improve the linking accuracy. Other studies [16, 18, 32] perform entity mention detection and entity linking jointly. However, all these studies generate candidate mentions and their mapping entities based on a dictionary constructed from Wikipedia, and then predicts whether a pair of mention-entity is a true mapping. They evaluate the performance solely on the final linking accuracy, and do not perform entity type classification. Moreover, these methods cannot identify the entity mentions that do not have a match in the dictionary. In this paper, we target on linking the location mentions in a tweet with their corresponding location profiles. We extract all possible location mentions in a tweet, even if the surface form of a location mention does not exist in the dictionary and is unlinkable to any location profile.

Joint NER and EL. Recent research [30, 43] shows that jointly performing NER and EL on formal text mutually enhances both tasks. Sil and Yates [43] re-rank a set of candidate mentions and entities, which are over-generated according to a pipelined method (*i.e.,* first recognize then link entities). The success of the re-ranking model lies in the dependency between mention boundary decisions and entity linking decisions. However, their method post-processes the output of a pipelined method, and entity type classification cannot be performed in their re-ranking model. Luo et al. [30] develop a joint model for NER and EL based on semi-CRF [39]. The success of their joint model lies in the consistency between the entity type of a mention and the category of the mapping entity. Their method only addresses local features; capturing long-distance and cross-task dependencies through global features will make their model very complex. As there is only one entity type (*i.e.,* location) in our task, the consistency between entity type and category becomes less meaningful. Thus, their method cannot be directly adopted. In this work, we propose a novel joint model based on inexact search, which makes it possible to introduce arbitrary global features to capture various dependencies. Our joint model performs mention boundary detection, type classification, and location linking simultaneously.



(a) Traditional Pipeline Architecture

(b) Joint Recognition and Linking (JoRL) Framework

Figure 2: Overview of the pipeline architecture and the JoRL framework for end-to-end location linking.

Structured Perceptron with Inexact Search. This technique has demonstrated its effectiveness in many NLP tasks [8, 19]. Recently, structured perceptron with inexact search has been applied to jointly perform extraction of entity mentions, relations, and events from formal text [25, 26]. However, due to the noise and informal writing nature of tweets, our end-to-end location linking problem faces different challenges. More importantly, we further extend this technique to be applied in a semi-supervised learning setting, which is fundamentally different from the techniques [25, 26].

Multi-view Learning. Multi-view learning algorithms, such as co-training [3], address the problem of semi-supervised learning, where a learning algorithm has access to limited amount of labeled data and (typically much larger) unlabeled data. The co-training algorithm learns two initially independent hypotheses which bootstrap each other. Dasgupta et al. [11] and Abney [1] give PAC bounds on the error of co-training in terms of the disagreement rate of hypotheses on unlabeled data in two independent views. This justifies the direct minimization of the disagreement [4]. The most relevant to our work is the use of a multi-view hidden markov perceptron to minimize the error on the labeled data and the disagreement on the unlabeled data [4]. Our work is significantly different from the work [4] in that: their decoding algorithm is based on exact inference, while ours is based on inexact search, which makes the loss function and update rules fundamentally different.

3. PROBLEM STATEMENT

We first define the end-to-end location linking problem, and then introduce a solution based on the traditional pipeline architecture (see Figure 2(a)). The joint model will be presented in Section 4.

3.1 Problem Definition

Given a tweet x published by a user from a predefined geographical region, and a Collection of Location Profiles (CLP) which consists of location profiles of POIs in the same geographical region, the task of ***end-to-end location linking*** is to identify all location mentions $M = \{m_1, m_2, ..., m_{|M|}\}$ in x and to link each identified mention m_i to the corresponding location profile e_i in CLP, $m_i \rightarrow e_i$. If there is no mapping location profile in CLP for m_i, then $m_i \rightarrow NIL$, where NIL denotes that m_i is unlinkable.

Recall that we use the two terms POI and location (or fine-grained location) interchangeably, referring to a *focused geographic entity* such as district, area, street, road or a *specific point location* such as hotels, schools, shopping centers, and restaurants.

3.2 The Pipeline Architecture

A straightforward solution is to first recognize location mentions and then to link the locations, shown in Figure 2(a). Next, we present a solution with state-of-the-art methods for the two sub-tasks.

3.2.1 Location Recognition

Location recognition aims to detect the boundaries of all possible mentions and then determine whether a candidate mention is a true location. The main challenge here is to resolve the *geo/non-geo ambiguity* (*e.g.,* whether a mention *mac* is a McDonald's restaurant or an Apple product) [2]. Li and Sun [24] propose to use a POI inventory and several types of features to train a linear-chain Conditional Random Fields (CRF) model [22] to extract POIs with their temporal awareness label from tweets, which has gained state-of-the-art performance. We use the method [24] to address the location recognition sub-task. Specifically, the location recognition subtask is casted as a sequential token tagging task, and the BILOU scheme [38] is applied. Figure 3(a) shows example BILOU tags (y) for the tokens in an input tweet (x). The features used for the CRF model are the token-based features listed in Table 2.

Table 1: Location profile mapping dictionary D. The subscript number is the index of the mapping location profile in CLP.

key (Surface Form)	*value* (Mapping Location Profile [Index])
Golden Village	`Golden Village`$_{[5278]}$ `Golden Village`$_{[5279]}$ ...
VivoCity	`VivoCity`$_{[2515]}$
Vivo City	`VivoCity`$_{[2515]}$
Cable Car	`Cable Car`$_{[3746]}$

3.2.2 Location Linking

Location linking aims to link a detected location mention m_i to its mapping location profile e_i from CLP. If there is no mapping location profiles in CLP, then $m_i \rightarrow NIL$. The main challenge here is to resolve the *geo/geo ambiguity* (*e.g.,* a location mention of *mac* refers to which particular McDonald's restaurant) [2].

Location Profile Mapping Dictionary. To link a location mention m_i, the set of candidate location profiles R^i that may be referred by m_i needs to be retrieved. Among $e_i \in R^i$, the best mapping profile is then linked by m_i. To this end, we build a location profile mapping dictionary D, as shown in Table 1.

The $D.key$ column of the mapping dictionary is a list of surface forms that a location mention may appear in, and $D.value$ column is the set of location profiles that can be mapped by each key. Note that, the same location profile (*e.g.,* `VivoCity`$_{[2515]}$) may be mapped by multiple surface forms (*e.g., VivoCity* and *Vivo City*), and the same surface form may have mappings to many location profiles. For example, *Golden Village* may be mapped to one of the 12 movie theaters located at different places.

Location Profile Linking. If there is more than one candidate profile for m_i, *i.e.,* $|R^i| > 1$, we define a linking quality measure $LQ(r^i_j)$ for each $r^i_j \in R^i$ to find the most likely link for m_i:

$$LQ(r^i_j) = \omega^T * f(r^i_j) = \omega^T_{local} * f_{local}(r^i_j) + \gamma * Coh(r^i_j). \quad (1)$$

In this equation, $f_{local}(r^i_j)$ is the vector representation of local features derived from location profile r^i_j, and ω_{local} is the weight vector. Based on the intuition that multiple locations mentioned in a single tweet are often geographically close to each other, we define a geographical coherence $Coh(r^i_j)$ for each candidate location profile r^i_j (to be formally defined shortly). The geographical coherence is

x	watching	Blackhat	at	Golden	Village	VivoCity	excited
y	O	O	O	B-POI	L-POI	U-POI	O

(a) Example BILOU scheme output y for location recognition from tweet x.

x	watching	Blackhat	at	Golden Village	VivoCity	excited
y	$S(1,1,O)$	$S(2,2,O)$	$S(3,3,O)$	$S(4,5,POI)$ $L(4,5,5278)$	$S(6,6,POI)$ $L(6,6,2515)$	$S(7,7,O)$

(b) Example output structure representation y for an example input tweet x.

Figure 3: Example BILOU scheme and output structure representation y for an example input tweet x.

also known as global feature, and γ is its weight. Both local and global features are summarized in the rows named "location profile-related features" in Table 2.

We then denote $\omega = \langle \omega_{local}, \gamma \rangle$ as the weight vector of all the features $f(r_j^i) = \langle f_{local}(r_j^i), Coh(r_j^i) \rangle$. In order to learn the weight vector ω automatically, we use a max-margin technique [41, 42] based on labeled data.

The geographical coherence $Coh(r_j^i)$ of a candidate location profile r_j^i is measured based on the geographical distance between the geo-coordinates of r_j^i and those of the (possible) mapping location profiles of the other location mentions in *the same* tweet.

$$Coh(r_j^i) = \frac{1}{|M|-1} \sum_{\substack{1 \le c \le |M| \\ c \ne i}} nDist(r_j^i, e_c) \quad (2)$$

$$nDist(r_j^i, e_c) = \max\left\{0, 1 - \frac{dist(r_j^i, e_c)}{dist_{\text{MAX}}}\right\} \quad (3)$$

$Coh(r_j^i)$ is only valid when there are multiple location mentions in a single tweet, *i.e.,* $|M| \ge 2$. In Equation 2, e_c is the mapping location profile for a mention $m_c \in M$ (where $c \ne i$), and $nDist(r_j^i, e_c)$ is normalized distance between r_j^i and e_c. Defined in Equation 3, $dist(r_j^i, e_c)$ is the haversine distance between the geo-coordinates of r_j^i and e_c, and $dist_{\text{MAX}}$ is the maximum haversine distance between two locations within the predefined geographical region.

During location linking, the mapping location profile e_c for a mention m_c is *unknown* and also needs to be assigned in this task. To solve this problem, we use an effective greedy algorithm, named *iterative substitution algorithm*, to jointly optimize the selection of mapping location profiles for all the location mentions in a single tweet. This algorithm has gained state-of-the-art performance for list linking [41] and tweet entity linking [42].

NIL Prediction. Because the mapping location profiles of some location mentions may not exist in CLP, we have to deal with the problem of predicting *unlinkable* location mentions. If the size of candidate location profiles R^i for location mention m_i is zero, we predict m_i as an unlinkable mention and return $m_i \to NIL$ undoubtedly. Otherwise, we will get $e_i = \arg\max_{r_j^i \in R^i} LQ(r_j^i)$. Here, we have to validate whether $m_i \to e_i$ holds. We adopt a simple and widely used method to learn a NIL-threshold τ to validate $m_i \to e_i$. That is, $m_i \to e_i$ holds if $LQ(e_i) > \tau$, and $m_i \to NIL$ otherwise.

4. JOINT RECOGNITION AND LINKING

The pipeline architecture has two limitations. First, it prohibits interactions between the two-subtasks. Errors from location recognition propagate to location linking and there is no feedback from the linking step to the recognition step. Second, it over-simplifies the whole task as a set of independent local classifiers without taking into account of cross-task dependencies. Next, we cast the whole task into a structured prediction problem, which performs the two sub-tasks jointly and overcomes aforementioned limitations.

4.1 Structured Prediction Formulation

We now introduce a new representation for the output of end-to-end location linking task.

Output Structure. Given an input tweet x, the output structure y consists of two types of nodes:

- **Segment Node** $S(l, r, t)$: A segment is a sequence of characters. l and r denote the left and right boundaries of a segment, $1 \le l \le r \le |x|$, $|x|$ is the length of tweet x; $t \in T$ is the type of the segment, $T = \{POI, NPOI, O\}$. $t = POI$ if the segment is a true location mention and $t = NPOI$ if the segment has ambiguity to be a location (*e.g., mac*) but is predicted not a location mention. Type O denotes that the segment is clearly not a location mention. Length of type O segment is always 1.
- **Linking Node** $L(l, r, e)$: This kind of node is produced only for POI type of segment. l and r denote the left and right boundaries of a segment, and e is the linked location profile of the segment. $e = NIL$ if the location is unlinkable.

Figure 3(b) shows the output structure representation y for an example input tweet x. The two nodes $S(4, 5, POI)$, $L(4, 5, 5278)$ denote the segment "Golden Village" is a location mention and links to the location profile with index ID 5278, respectively.

With the new structured representation, end-to-end location linking becomes a structured prediction problem, which is to predict the most probable output structure $\hat{y}$ for a given tweet x.

Structured Prediction. Let $x \in \mathcal{X}$ be an input tweet, CLP be a collection of location profiles, and $y' \in \mathcal{Y}(x, \text{CLP})$ be a candidate output structure, our goal is to predict the most probable output structure $\hat{y}$ for x. We use the following linear model to predict the most probable output structure $\hat{y}$ for x:

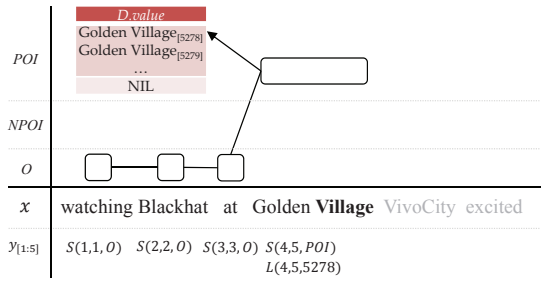
$$\hat{y} = \arg\max_{y' \in \mathcal{Y}(x, \text{CLP})} \mathbf{w}^{\mathrm{T}} \cdot \quad (x, y', \text{CLP}) \quad (4)$$

where (x, y', CLP) is the feature vector that characterizes the input tweet x together with a candidate output structure y', and $\mathbf{w}$ is the corresponding feature weights. When the context is clear, we will use (x, y') instead of (x, y', CLP) for short.

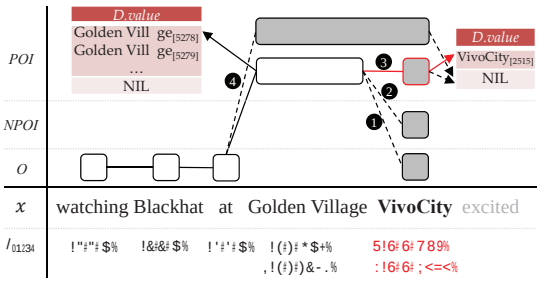
With the current problem definition, end-to-end location linking can be performed naturally in a joint search space simultaneously, shown in Figure 2(b). Next, we show that the proposed joint framework overcomes limitations of the pipeline architecture by enabling cross-task dependencies.

4.2 Joint Decoding Algorithm

Decoding (aka inference) procedure, the key step in both training and test, aims to search for the best output structure under the current model parameters. Unfortunately, it is intractable to perform the exact search in our JoRL framework. The reasons are two-fold: (1) Since the output structure contains mention boundary



(a) One possible partial output structure $y_{[1:5]}$ in the beam $B[5]$ after decoding at the token "Village".



(b) Decoding at the token "VivoCity". The grey rectangle with red boarder is the preferred partial output structure.

Figure 4: An example of decoding at the tokens "Village" and "VivoCity" for a given tweet x. The rectangles denote the segments with different types (*i.e.*, $\{POI, NPOI, O\}$), among which the gray ones are possible segments generated when taking the recognition action. The solid lines and arrows denote correct recognition and linking actions, respectively, while the dotted denote incorrect ones.

detection, segment type classification, and location profile linking at the same time, the search space becomes much more complex; and (2) We hope to make use of arbitrary global features from the entire output structure to capture long-distance and cross-task dependencies, which makes the inference even harder.

To address this problem, we employ *beam-search*, an instance of inexact search, to approximate Equation 4. Specifically, we propose a beam-search algorithm, similar to the multiple-beam algorithm [45], to incrementally expand partial output structures for an input tweet, to find the optimal output structure with the best score.

Let $\hat{d}_t$ be the maximum length of a segment with type t. The k-best partial output structures for x ending at the i-th token is:

$$B[i] = \underset{y_{[1:i]} \in \mathcal{Y}(x, \text{CLP}, i)}{\arg\,\text{top}^k} \mathbf{w}^{\mathrm{T}} \cdot \quad (x, y_{[1:i]}) \tag{5}$$

where $y_{[1:i]}$ denotes the partial output structure whose last segment ends at the i-th token, and $\mathcal{Y}(x, \text{CLP}, i)$ stands for the search space.

Our proposed joint decoding algorithm is shown in Algorithm 1. For each token index i, it maintains a beam $B[i]$ for the partial output structures whose last segments end at the i-th token. Two kinds of actions, corresponding to the two kinds of nodes, are taken during the decoding:

- **Recognition Action** (Lines 3-10): The algorithm enumerates all possible length (*i.e.*, $d = 1...\hat{d}_t$) of segments ending at the current token index i with various types, generating possible segment nodes $\{S(j, i, t)\}$. Each segment node $S(j, i, t)$ is then appended to each existing partial output structure $y_{[1:i-d]}$ from the previous beam $B[i-d]$ to form a new partial output structure $y_{[1:i]}$ (Line 8). Finally, the k-best partial output structures are recorded in the current beam $B[i]$.

- **Linking Action** (Lines 11-20): After the recognition action, the algorithm checks whether the type t of the last segment node $S(j, i, t)$ of each partial output structure $y_{[1:i]}$ is POI. If $t = POI$, then the algorithm enumerates all possible candidate location profiles, that could be linked to the mention $x_{[j,i]}$, together with a default NIL, generating possible linking nodes $\{L(j, i, e)\}$. Each linking node $L(j, i, e)$ is then appended to the current partial output structure $y_{[1:i]}$ to form a new partial output structure $y'_{[1:i]}$ (Line 16). If t is not POI, the current partial output structure $y_{[1:i]}$ remains the same (Line 19). Finally, all partial output structures after taking the linking action are re-ranked, and the new k-best partial output structures are recorded in the current beam $B[i]$.

Algorithm 1 Joint Decoding Algorithm based on Beam-Search

Input: a tweet $x = x_{[1...|x|]}$, and a CLP
k: beam size
T: types of a segment
$\hat{d}_t$: maximum length of a segment with type t, $t \in T$
Output: best output structure $\hat{y}$ for x

```
1:  initialize |x| empty beams B[1...|x|]
2:  for i ∈ 1...|x| do
      /* recognition action */
3:    buf ← ∅
4:    for t ∈ T do
5:      for d ∈ 1...d̂_t do
6:        j ← max(1, i − d + 1)
7:        for y_{[1:i−d]} ∈ B[i − d] do
8:          y_{[1:i]} ← Append(y_{[1:i−d]}, S(j, i, t))
9:          buf ← buf ∪ {y_{[1:i]}}
10:   B[i] ← k-best(buf)
      /* linking action */
11:   buf ← ∅
12:   for y_{[1:i]} ∈ B[i] do
13:     S(j, i, t) ← Last_Segment_Node(y_{[1:i]})
14:     if t is POI then
15:       for e ∈ Candidate_Entities(x_{[j:i]}, CLP) ∪ {NIL} do
16:         y'_{[1:i]} ← Append(y_{[1:i]}, L(j, i, e))
17:         buf ← buf ∪ {y'_{[1:i]}}
18:     else
19:       buf ← buf ∪ {y_{[1:i]}}
20:   B[i] ← k-best(buf)
21: return B[|x|][0]
```

There are $|x|$ steps to incrementally search for the optimal output structures, where $|x|$ is the length of the input tweet x, in number of tokens. The time complexity for the recognition action is at most $k \cdot |T| \cdot \hat{d}_t$ and for linking action is $k \cdot (|E| + 1)$, respectively. Thus, the overall time complexity of the joint decoding algorithm is $O(|x| \cdot k \cdot (|T| \cdot \hat{d}_t + |E|))$, where $|T|$ is the number of segment types, and $|E|$ is the num of candidate entities for a specific segment.

Figure 4 illustrates the decoding at the tokens "Village" and "VivoCity" by considering again the tweet shown in Figure 3(b). For simplicity, only a small search space is presented. Suppose we have already taken the two actions when decoding at the token "Village", and one possible partial output structure is illustrated in Figure 4(a). Now, we move on to token "VivoCity" (Figure 4(b)). First, the algorithm generates all possible segment nodes ending at the token "VivoCity". Four possible segment nodes are illustrated with gray rectangles in the figure. They are $S(6, 6, O)$, $S(6, 6, NPOI)$,

Algorithm 2 Structured Perceptron Algorithm with Beam-Search & Early-Update (Learning from labeled data)

Input: labeled data $\mathcal{D}^{\mathcal{L}} = \{x_i, y_i\}_{i=1}^{n}$
I: maximum iteration number
Output: model parameters $\mathbf{w}$

1: initialize $\mathbf{w} \leftarrow \mathbf{0}$
2: **for** $t \in 1...I$ **do**
3: **for** each $(x, y) \in \mathcal{D}^{\mathcal{L}}$ **do**
4: $(y', z) \leftarrow \mathsf{BeamSearch}(x, y, \mathbf{w})$
5: **if** $z \neq y'$ **then**
6: $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{\Phi}(x, y') - \mathbf{\Phi}(x, z)$
7: **return** $\mathbf{w}$

$S(6, 6, POI)$, and $S(4, 6, POI)$, respectively. Then, these segment nodes are appended to the preceding partial output structures in the beams of "Village" and "at", respectively. The figure shows four lines of this process, indicated by the circled numbers 1-4 in the figure. Next, the algorithm enumerates all possible candidate location profiles with a default NIL for the two segment nodes $S(6, 6, POI)$ and $S(4, 6, POI)$, generating the following three linking nodes $L(6, 6, 2515)$, $L(6, 6, \text{NIL})$ and $L(4, 6, \text{NIL})$. Then, the three linking nodes are linked by the two segment nodes with arrows as shown in the figure. Finally, $S(6, 6, POI)$ and $L(6, 6, 2515)$, other than $S(4, 6, POI)$ and $L(4, 6, \text{NIL})$, are preferred by the model because the indicative global feature for fixing under-segmentation error (see Section 4.5.2) takes effect.

4.3 Learning from Labeled Data

We now learn the model parameters w from labeled samples. The learning problem is formulated as an optimization problem: given a set of n labeled samples $\mathcal{D}^{\mathcal{L}} = \{x_i, y_i\}_{i=1}^{n}$, and a loss function $\ell(x_i, y_i, \hat{y}_i)$, our goal is to minimize the empirical risk, given by $\mathcal{R}^{\mathcal{L}}$:

$$\mathcal{R}^{\mathcal{L}} = \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i, \hat{y}_i) \tag{6}$$

$$\ell(x_i, y_i, \hat{y}_i) = h\big(\mathsf{w} \cdot \quad (x_i, y_i) - \mathsf{w} \cdot \quad (x_i, \hat{y}_i)\big) \tag{7}$$

In Equation 7, $h(z) = \max(0, -z)$ is defined as a slightly modified version of hinge loss. If the predicted $\hat{y}_i$ is different from the ground truth y_i, a loss is produced; otherwise there is no loss.

In our implementation, we apply structured perceptron [8], an extension of the standard perceptron for structured prediction, to estimate the model parameters w from labeled data. The perceptron learns w in an online fashion. For each labeled example (x_i, y_i), the algorithm uses Equation 4 to search for the best output structure $\hat{y}_i$ for x_i under the current model. If $\hat{y}_i$ is incorrect, then the parameters are updated as follows:

$$\mathsf{w} \leftarrow \mathsf{w} + \quad (x_i, y_i) - \quad (x_i, \hat{y}_i) \tag{8}$$

Huang et al. [19] proved the convergence of structured perceptron when inexact search is applied with violation-fixing update methods such as early-update [9]. Since we use beam-search in this work, we apply early-update method for model training, as shown in Algorithm 2. For each training example (x, y), the algorithm performs $\mathsf{BeamSearch}$, which is identical to the decoding algorithm in Algorithm 1 with the following one exception. If y', the prefix of the ground truth y, falls out of the beam after each execution of k-best function (Lines 10 and 20 in Algorithm 1), then y' and the top partial output structure z in the current beam are returned for updating parameters (Line 4 in Algorithm 2). Following [8], to avoid overfitting, we use averaged parameters after training to decode the test instances.

4.4 Learning from Labeled and Unlabeled Data

Supervised learning often requires a large number of labeled samples. It is worth considering semi-supervised learning with the help of unlabeled data, such that the trained model has a lower generalization error than its fully-supervised counterpart. However, structured perceptron [8, 19] is an online algorithm, which generates models that output merely a prediction with no additional confidence measure. The traditional bootstrapping methods, which are based on selecting unlabeled data whose predicted output has a high confidence score, cannot be directly applied.

In this work, we employ the concept of multi-view learning [12], and implement the *consensus maximization principle* [4]. The objective is to minimize the error on the labeled data and maximize the agreement on the unlabeled data from multiple models. In other words, we aim to make consensus on the outputs of multiple independent hypotheses. To this end, we propose a semi-supervised structured perceptron algorithm.

Given n labeled examples $\mathcal{D}^{\mathcal{L}} = \{x_i, y_i\}_{i=1}^{n}$, and m unlabeled examples $\mathcal{D}^{\mathcal{U}} = \{x_j\}_{j=n+1}^{n+m}$, in the multi-view learning scenario, we decompose the feature map (x, y) into multiple feature maps $^{v}(x, y)$, each being a representation for one view $v \in V$. We assume that each feature map is sufficient for learning and predicting the output structure $\hat{y}^v$ for an input x, irrespective of whether it is labeled or unlabeled.

$$\hat{y}^v = \underset{y' \in \mathcal{Y}(x, \text{CLP})}{\arg\max} \ \mathsf{w}^{v\mathrm{T}} \cdot \quad ^{v}(x, y')$$

According to the consensus maximization principle, we aim to minimize the error on the labeled data and the disagreement on the unlabeled data. Thus, our goal is to minimize the empirical risk, given by $\mathcal{R}^{\mathcal{L}+\mathcal{U}}$:

$$\mathcal{R}^{\mathcal{L}+\mathcal{U}} = \frac{1}{n|V|} \sum_{v=1}^{|V|} \sum_{i=1}^{n} \ell_1(x_i, y_i, \hat{y}_i^v) + \frac{\lambda}{m\binom{V}{2}} \sum_{\substack{u,v=1 \\ u \neq v}}^{|V|} \sum_{j=n+1}^{n+m} \ell_2(x_j, \hat{y}_j^u, \hat{y}_j^v) \tag{9}$$

In $\mathcal{R}^{\mathcal{L}+\mathcal{U}}$, λ is a trade-off parameter to determine the overall influence of the disagreement on the unlabeled data. $\ell_1(x_i, y_i, \hat{y}_i^v)$ is the loss on a labeled example (x_i, y_i) on one view representation $v \in V$, which is similar to Equation 7 except that the feature map and weights are based on view v.

$$\ell_1(x_i, y_i, \hat{y}_i^v) = h\big(\mathsf{w}^v \cdot \quad ^{v}(x_i, y_i) - \mathsf{w}^v \cdot \quad ^{v}(x_i, \hat{y}_i^v)\big)$$

If the predicted $\hat{y}_i^v$ is the same as the ground truth y_i, then there is no loss; otherwise a loss is produced and the parameters are updated as follows:

$$\mathsf{w}^v \leftarrow \mathsf{w}^v + \quad ^{v}(x_i, y_i) - \quad ^{v}(x_i, \hat{y}_i^v). \tag{10}$$

In $\mathcal{R}^{\mathcal{L}+\mathcal{U}}$, $\ell_2(x_j, \hat{y}_j^u, \hat{y}_j^v)$ is the loss on an unlabeled example x_j, which measures the pairwise disagreement between any two independent views $u \in V$ and $v \in V$ $(u \neq v)$.

$$\ell_2(x_j, \hat{y}_j^u, \hat{y}_j^v) = h\left(\frac{1}{2}\Big(\frac{\mathsf{w}^u \cdot \quad ^{u}(x_j, \hat{y}_j^u) - \mathsf{w}^u \cdot \quad ^{u}(x_j, \hat{y}_j^v)}{\mathsf{w}^u \cdot \quad ^{u}(x_j, \hat{y}_j^u)}\Big)^2 + \frac{1}{2}\Big(\frac{\mathsf{w}^v \cdot \quad ^{v}(x_j, \hat{y}_j^v) - \mathsf{w}^v \cdot \quad ^{v}(x_j, \hat{y}_j^u)}{\mathsf{w}^v \cdot \quad ^{v}(x_j, \hat{y}_j^v)}\Big)^2\right)$$

Specifically, for each unlabeled example x_j, the algorithm use $\mathsf{BeamSearch}$ to search for the best output structure $\hat{y}_j^u$ and $\hat{y}_j^v$ for views u and v under the current models w^u and w^v, respectively.

Algorithm 3 Structured Perceptron Algorithm with Beam-Search & Early-Update (Learning from labeled and unlabeled data)

Input: labeled data $\mathcal{D}^{\mathcal{L}} = \{x_i, y_i\}_{i=1}^{n}$
unlabeled data $\mathcal{D}^{\mathcal{U}} = \{x_j\}_{j=n+1}^{n+m}$
I: maximum iteration number
V: set of independent views, $|V| \geq 2$
λ: trade-off parameter
Output: model parameters $\mathbf{w}^v, v \in V$

1: initialize $\mathbf{w}^v \leftarrow \mathbf{0}, v \in V$
2: **for** $t \in 1...I$ **do**
/*Learning from labeled data */
3: **for** each $(x, y) \in \mathcal{D}^{\mathcal{L}}$ **do**
4: **for** each view $v \in V$ **do**
5: $(y', z^v) \leftarrow \text{BeamSearch}(x, y, \mathbf{w}^v)$
6: **if** $z^v \neq y'$ **then**
7: $\mathbf{w}^v \leftarrow \mathbf{w}^v + \mathbf{\Phi}^v(x, y') - \mathbf{\Phi}^v(x, z^v)$
/*Learning from unlabeled data */
8: **for** each $x \in \mathcal{D}^{\mathcal{U}}$ **do**
9: **for** each two independent views u and v **do**
10: $\hat{y}^u \leftarrow \text{BeamSearch}(x, \mathbf{w}^u)$
11: $\hat{y}^v \leftarrow \text{BeamSearch}(x, \mathbf{w}^v)$
12: **if** $\hat{y}^u \neq \hat{y}^v$ **then**
13: update $\mathbf{w}^u$, $\mathbf{w}^v$ according to Eq. 11 and 12
14: **return** $\mathbf{w}^v, v \in V$

If $\hat{y}_j^u$ is the same as $\hat{y}_j^v$, then there is no loss; otherwise a loss is produced and the parameters are updated as follows:

$$\mathbf{w}^u \leftarrow \mathbf{w}^u + \lambda \frac{n}{m} \eta_1^u \quad {}^u(x_j, \hat{y}_j^u) + \lambda \frac{n}{m} \eta_2^u \quad {}^u(x_j, \hat{y}_j^v) \qquad (11)$$

$$\mathbf{w}^v \leftarrow \mathbf{w}^v + \lambda \frac{n}{m} \eta_1^v \quad {}^v(x_j, \hat{y}_j^v) + \lambda \frac{n}{m} \eta_2^v \quad {}^v(x_j, \hat{y}_j^u) \qquad (12)$$

where

$$\eta_1^u = \frac{(f_2 - f_1) f_2}{f_1^3} \qquad \eta_2^u = \frac{f_1 - f_2}{f_2^2}$$

$$\eta_1^v = \frac{(g_1 - g_2) g_1}{g_2^3} \qquad \eta_2^v = \frac{g_2 - g_1}{g_2^2}$$

$$f_1 = \mathbf{w}^u \cdot \quad {}^u(x_j, \hat{y}_j^u) \qquad g_1 = \mathbf{w}^v \cdot \quad {}^v(x_j, \hat{y}_j^u)$$

$$f_2 = \mathbf{w}^u \cdot \quad {}^u(x_j, \hat{y}_j^v) \qquad g_2 = \mathbf{w}^v \cdot \quad {}^v(x_j, \hat{y}_j^v)$$

The algorithm for learning from labeled and unlabeled data is shown in Algorithm 3. Lines 3-7 are used to learn feature weights from the labeled data which are similar to Algorithm 2. Lines 8-13 are specific for fine-tuning of the feature weights according to the disagreement on the unlabeled examples. Finally, the k-best partial output structures for x ending at the i-th token is calculated as:

$$B[i] = \underset{y_{[1:i]} \in \mathcal{Y}(x, \text{CLP}, i)}{\arg \text{top}^k} \sum_{v=1}^{|V|} \mathbf{w}^{v\text{T}} \cdot \quad {}^v(x, y_{[1:i]}) \qquad (13)$$

In our experiments, we set $|V| = 2$, which is commonly used in many studies on multi-view learning. To avoid overfitting, we also use averaged parameters after training to decode test instances [8].

4.5 Features

All the features are listed in Table 2. The table also shows how the features are split to two independent views for multi-view learning.

4.5.1 Local Features

All the features derived from individual segments are known as local features. Specifically, local features include token-based and segment-based features for location recognition, and location profile-related and NIL-related features for location linking.

Token-based Features: Although the output structure is defined based on segments, we can still derive token-based features from the tokens contained in the segments. Specifically, a segment node $S(l, r, t)$ is converted to BILOU tags for tokens within it. The token-based features are those that are used in the pipeline architecture.

Segment-based Features: This set of features captures the properties of a segment, like the segment itself and surrounding words.

Location Profile-related Features: These features are directly derived from a location profile including its popularity, whether it is the only candidate location profile for a surface name and surface matching.

NIL-related Features: The features are indicators for NIL prediction. For example, whether there are location profiles whose name match a segment's surface form exactly.

4.5.2 Global Features

We develop three kinds of global features. The *geographical coherence* feature introduced in Section 3.2.2 is one of them. It captures long-distance dependencies within a task. The other two kinds of features, by capturing cross-task dependencies, overcome the two limitations in pipeline architecture discussed in Section 1. Note that, these two kinds of features cannot be directly used in any step of the pipeline because 1) these features are segment-based while the features used in the CRF model are token-based; 2) these features encode the feedback information from the linking step which cannot be obtained in the recognition step of the pipeline; 3) these features are not designed for linking purpose, but to give feedback to the recognition step.

Fixing Under-segmentation Error: An under-segmentation error occurs if the system wrongly recognizes two mentions that happen to appear near one another to a single location mention. To address this, we encode two *binary* features.

The first feature is to penalize the output structure with under-segmentation error. The feature is triggered (and set to 1) if the current segment (*e.g.,* "Golden Village VivoCity") is not in the dictionary, but its two sub-segments (*e.g.,* "Golden Village" and "VivoCity") are both in the dictionary. The second feature is to reward the output structure with the right segmentation and linking. The feature value is triggered (and set to 1) if a segment appears in the address attribute of another segment's mapping location profile within the same tweet. Suppose we have already linked the segment "Golden Village" to the location profile shown in Figure 1 (this movie theater is located inside VivoCity), the next segment "VivoCity" appears in the address attribute of the linked "Golden Village". Then the current output structure is rewarded by triggering this feature.

Fixing Mis-segmentation Error: A mis-segmentation error occurs if the system leaves off one or more words from a multi-word location mention. We also use two *binary* features to address this.

The first feature is to penalize the output structure with mis-segmentation error. If the following word (*e.g.,* "Bar") of an already linked segment (*e.g.,* "Cable Car") matches a category in its linked location profile, then this feature is set to 1. Here, we consider the mention of "Cable Car Bar" in a sentence is the complete location mention of the location, although the name of the matching location profile is indeed "Cable Car". The second feature is to reward the output structure with the right segmentation and linking. This feature is set to 1 if the current segment has been successfully linked to a location profile such that the segment contains the location profile's name and one of its categories. For example, "Starbucks Café" successfully linked to a location profile named "Starbucks" with category "Café" is rewarded by triggering this feature.

Table 2: Summary of the features used in this work.

Local Features for Location Recognition		**View**
Token-based Features	The word itself and its lowercased form The word shape: all capitalized, is-capitalized, all-numerics, alphanumeric Word prefixes and suffixes, from 1 to 3 characters Bag-of-words of the context window up to 5 words POS tags of the preceding word, the current word, and the following word Brown clustering features based on the 4th, 8th, and 12th bits of the path[4] Pre-label of words with the POI Inventory based on BILOU schema	1
Segment-based Features	The segment itself and its lowercased form The segment shape: whether all the words are all-capitalized, initial-capitalized Bag-of-words in a segment's contextual window of size 2 POS tags of the current segment's surrounding two words Whether the segment appears in the POI Inventory	2
Local Features for Location Linking		**View**
Location Profile-related Features	Popularity: #visits, #visitors, rating Whether this is the only candidate location profile for a surface form Whether there are other words that appear in both the tweet and a location profile's address attribute	1
	Surface Matching: Jaccard similarity between a mention and a location profile's name attribute	2
NIL-related Features	Whether there is no candidate location profile from CLP Whether there is any location profile's name attribute that exactly matches the segment's surface form	1
	Min/max/average scores for the features about popularity and surface matching	2
Global Features for Location Linking (capturing long-distance dependences within a task)		**View**
Location Profile-related Features	Geographical Coherence: Averaged distance from the other mapping location profiles' geo-locations	2
Global Features for Location Recognition & Linking (capturing cross-task dependences)		**View**
Fixing Under-segmentation Error	Penalize the output structure with under-segmentation error Reward the output structure with the right segmentation and linking	1
Fixing Mis-segmentation Error	Penalize the output structure with under-segmentation error Reward the output structure with the right segmentation and linking	2

Table 3: Example check-in tweets and linking location profiles

Check-in Tweets [Foursquare link masked]	Mapping Location Profiles
I'm at **Golden Village** (Singapore) [link]	`Golden Village`$_{[5278]}$
I'm at **Vivo City** (Singapore) [link]	`VivoCity`$_{[2515]}$
Ice Age 4 (@ **Golden Village**) [link]	`Golden Village`$_{[5279]}$
quiet day at work. (@ **Cable Car**) [link]	`Cable Car`$_{[3746]}$

Table 4: Summary of the dataset.

#location profiles in CLP	22,414
#keys in location profile mapping dictionary	24,750
#$\langle key, value \rangle$ mappings in dictionary	63,091
#average mappings for each key in dictionary	2.55
#entries in POI inventory	27,386
#labeled tweets for evaluation	3,611
#candidate location mentions	5,707
#location mentions that are NPOI	4,165
#location mentions that are POI	1,542
#location mentions that are linkable POI	543
#location mentions that are unlinkable POI	999

5. EXPERIMENTS

5.1 Data Preparation

Check-in Tweets. We collect 326,853 tweets that are associated with Foursquare check-ins for constructing location profiles, mapping dictionary, and POI inventory (to be discussed shortly). These check-in tweets were posted by Foursquare apps to Twitter by users in Singapore. A check-in tweet usually contains formal or informal location names. Table 3 shows two kinds of common check-in tweets. The first two simply report users' current locations, while the latter two report users' activities at the locations. Observe that check-in tweets are relatively well formatted, and the location names can be reliably extracted by applying handcrafted regular expressions. Their mapping location profiles from Foursquare can be crawled following the urls in the tweets, see the second column in Table 3. Because of the large number of check-ins, the POI coverage for a given geographical region is broad and is in a fine-grained scale. These tweets are not used for evaluating the methods.

Collection of Location Profiles (CLP). The CLP is constructed via crawling short urls in the check-in tweets. From the check-in tweets, we get 28,134 unique location profiles. As the location profiles in Foursquare are collected via crowdsourcing, the collected location profiles are noisy. We filter the location profiles with fewer than 5 visitors and the ones under category "Home (private)". We also filter the location profiles that have non-Latin characters in its name attribute. After cleaning, our CLP contains 22,414 location profiles.

Location Profile Mapping Dictionary. The mapping dictionary is used to generate candidate location profiles for a location mention surface form (see Table 1). We construct this dictionary in two steps. First, the name attribute of each location profile is added to $D.key$, which maps to the index of the location profile (*i.e.,* $D.value$). Second, recall that a check-in tweet often contains informal POI names, we extract these POI names from check-in tweets and add these names to $D.key$ for the corresponding location profiles linked by the urls in check-in tweets. For example, the POI name "Vivo City" in the second check-in tweet in Table 3 is added to the surface form of `VivoCity`$_{[2515]}$ (note the different spellings).

As the result, we have a location profile mapping dictionary with 24,750 keys and 63,091 $\langle key, value \rangle$ mappings. Recall that, the same surface form may link to multiple location profiles and one location profile may have multiple surface forms. On average, each surface form has 2.55 mapping location profiles.

POI Inventory. In Twitter, people often mention locations with abbreviations or partial names. A POI inventory is an augmented version of the surface forms (*i.e.,* the keys) in the mapping dictio-

[4]We use the POS tagger and Brown clusters provided in TwitterNLP, which is available at `https://github.com/aritter/twitter_nlp`, to extract POS related features and Brown clustering features.

nary. It consists of not only the surface forms, but also the partial names derived from these surface forms. For example, partial name "Vivo", derived from surface form "Vivo City", is included in the POI inventory. POI inventory was proposed in [24] and used as a "noisy version of gazetteer" to *pre-label* candidate location mentions in tweets. It has shown to be an important resource in fine-grained location recognition. Following the method in [24], we derived a POI inventory with 27,386 entries from the location surface forms.

Labeled Tweets for Evaluation. The tweets with ground truth annotations for end-to-end location linking evaluation are created as follows. We randomly sample 100 users from the 53,836 Singapore Twitter users, who specifies Singapore in the location field of her Twitter profile. Then, we use the Twitter API to collect at most 2,000 tweets from each sampled user. The following preprocessing are applied: remove all retweets and tweets with fewer than two words, remove stop words, mask urls by *LINK*, mask @username by *USER*, remove # from hashtags (*i.e.,* hashtags are treated as common words). In total, we have 100,058 processed tweets.

Each tweet is then matched against POI inventory with case-insensitive leftmost longest match. After the matching process, there are 24,129 tweets that each contains at least one candidate POI mention. From these tweets, 4,012 tweets are randomly sampled for manual annotation to create the ground truth dataset.

For each candidate POI mention in the sampled tweets, human annotators are asked to assign one of the following 4 labels:

- $NPOI$. This label denotes the candidate mention is not a true location mention.
- Index of the mapping location profile in CLP. This implies that the candidate mention is a true POI.
- NIL. This label denotes that the candidate mention is a true POI, but has no mapping location profile in our CLP.
- $Unknown$. This label denotes that the annotator cannot determine whether the mention is a true POI or not.

To facilitate the annotation process, for each tweet to be labeled, the previous and the following two tweets posted by the same user are provided. These five tweets and their timestamps together provide the context for the annotation. The annotators are also encouraged to utilize a map interface to quickly locate the correct location profiles, particularly the names that may link to multiple locations. The following are three example tweets with their assigned labels.

1. My bag smells like [**mac**]$_{\text{NPOI}}$
2. *USER* [**kfc**]$_{1052}$ at [**Iot 1**]$_{18835}$ hahahaha
3. At [**Teddy & Me cafe**]$_{\text{NIL}}$! It 's so cute !!! :) *LINK*

After annotation on the sampled 4,012 tweets, 169 tweets are filtered out for containing words mostly in other language than English. In the remaining 3,843 tweets, there are 232 tweets within each the only candidate POI mention is labeled as *Unknown*. Finally, we obtain 3,611 labeled tweets as the evaluation set. A summary of this labeled dataset is shown in Table 4. In these 3,611 tweets, there are 5,707 candidate POI mentions which involve 1,516 distinct candidate POI names. About 27% or 1,542 candidate POI mentions are true POIs. Among them, 543 are linkable and are labeled with the indexes of the mapping location profiles. The remaining 999 POI mentions are unlinkable, and are labeled with NIL.

We randomly split the 3,611 labeled tweets into three parts: 211 tweets for development, 2,500 tweets for testing, and the remaining 900 tweets for training. To evaluate the performance with different sizes of labeled data, we use different number of labeled tweets in training: 100, 300, 500, 700 and 900. To involve unlabeled data for multi-view learning, we randomly sample 1,000, 5,000, 10,000, 15,000 tweets from the remaining unlabeled tweets that has at least one candidate POI mention.

5.2 Parameter Setting and Evaluation Metric

In training and test, we set $dist_{\text{MAX}} = 22km$ in Equation 3. The NIL threshold $\tau = 0.01$ is learned with the development set. The maximum length $\hat{d}_t$ of a segment with type t is collected from all the labeled data, and we set $\hat{d}_{POI} = 8$ and $\hat{d}_{NPOI} = 4$. The beam size is learned with the development set. Larger beam size than 3 leads to marginal increase in performance but much longer decoding time, similar to the findings in [26]. As a trade-off, we set the beam size to be 3 throughout the experiments. The trade-off parameter λ is set to 0.01 (see Equation 9).

For both location recognition and linking, we adopt three widely used metrics for evaluation: Precision (Pr), Recall (Re) and F_1. To have single number to measure the performance on the development set, we use *harmonic mean* of F_1 for location recognition and F_1 for linking on the development set. Figure 5(a) shows the learning curves on the development set for JoRL$_{\text{L}}$ (joint model learning from labeled data) and JoRL$_{\text{LU}}$ (joint model learning from both labeled and unlabeled data). This plot is with 500 labeled tweets and 5,000 unlabeled tweets. Both curves converge after 10 iterations. Similar observations hold for other settings. Therefore, we set the number of training iterations to be 10 throughout our experiments.

5.3 Experimental Results

In the experiments, we compare the following three methods:

PiRL: This baseline method is based on the pipeline architecture. We implement the state-of-the-art methods for both location recognition [24] and location linking [42] (see Section 3.2).

JoRL$_{\text{L}}$: This method is based on the proposed joint framework for location recognition and linking (JoRL), learning from labeled data (see Section 4.3). JoRL$_{\text{L}}$ uses all features listed in Table 2.

JoRL$_{\text{LU}}$: This method is also based on JoRL. It utilizes multi-view learning and learns from both labeled and unlabeled data (see Section 4.4). JoRL$_{\text{LU}}$ uses all features in Table 2, in 2 views.

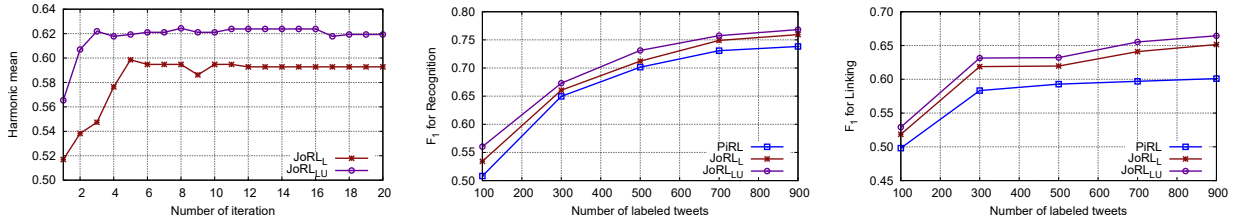
Overall Comparison. Figures 5(b) and 5(c) show the overall results of the three methods PiRL, JoRL$_{\text{L}}$, and JoRL$_{\text{LU}}$.

First, in terms of F_1, JoRL$_{\text{L}}$ consistently outperforms PiRL on both location recognition and linking. Depending on the size of labeled data, the improvement over PiRL is around 4.1-8.4% and 1.5-5.2% respectively on the two sub-tasks. Second, when involving unlabeled data, JoRL$_{\text{LU}}$ also consistently outperforms PiRL on both sub-tasks by around 6.2-10.5% and 3.6-10.3%, respectively. Here, we use 5,000 unlabeled tweets in this comparison, and we will study the impact of the size of unlabeled data later. Last, JoRL$_{\text{LU}}$ outperforms JoRL$_{\text{L}}$ on both sub-tasks and the improvements are around 2% and 1.1-4.9%, respectively.

In short, both joint models JoRL$_{\text{L}}$ and JoRL$_{\text{LU}}$, outperform the pipelined method PiRL consistently. One reason is that the proposed joint framework utilizes global features, which not only prevent the recognition errors, but also improve the linking accuracy. The learning from unlabeled data, further improves the model accuracy.

Impact of Global Features. Table 5 compares 5 methods with and without global features, where (l) stands for local features and (l+g) for local and global features. In this comparison, the number of labeled and unlabeled tweets are 500 and 5,000, respectively (other settings show similar results). We make the following observations:

In evaluating location recognition, when the joint model only uses local features, both JoRL$_{\text{L}}$ (l) and JoRL$_{\text{LU}}$ (l) achieve better Re and F_1 but poorer Pr, compared to PiRL. The reason is that the



(a) Learning curves on development set (b) Location recognition (c) Location linking

Figure 5: Learning curve on development set for JoRL$_L$ and JoRL$_{LU}$, and overall comparison of methods PiRL, JoRL$_L$, and JoRL$_{LU}$.

Table 5: Impact of Global Features.

Method	Location Recognition			Location Linking		
	Pr	Re	F_1	Pr	Re	F_1
PiRL	0.8498	0.5973	0.7015	0.6905	0.5194	0.5928
JoRL$_L$ (l)	0.8473	0.6087	0.7084	0.7763	0.5075	0.6137
JoRL$_L$ (l+g)	0.8561	0.6097	0.7122	0.7880	0.5104	0.6196
JoRL$_{LU}$ (l)	0.7975	0.6646	0.7250	0.7291	0.5463	0.6246
JoRL$_{LU}$ (l+g)	0.8190	0.6605	0.7312	0.7500	0.5463	0.6321

Table 6: Performance on NIL/NPOI Prediction.

Methods	NPOI Prediction			NIL Prediction		
	Pr	Re	F_1	Pr	Re	F_1
PiRL	0.9075	0.9769	0.9409	**0.7752**	0.5246	0.6257
JoRL$_L$	0.9084	**0.9836**	0.9445	0.7346	0.5483	0.6279
JoRL$_{LU}$	**0.9306**	0.9601	**0.9451**	0.7047	**0.5975**	**0.6467**

joint model is based on beam search, which enumerates all possible segments during the decoding in the joint search space. Thus more locations are recognized but with poorer precision. When we introduce global features, both JoRL$_L$ (l+g) and JoRL$_{LU}$ (l+g) outperform PiRL on all the three metrics Pr, Re and F_1. In addition, both methods achieve better F_1 than their counterparts without using global features. This demonstrates the effectiveness of global features in fixing the recognition errors.

For location linking, global features bring improvements on all three metrics to both methods JoRL$_L$ and JoRL$_{LU}$. Both methods outperform PiRL and the best F_1 is achieved by JoRL$_{LU}$ (l+g). This set of results indicates that the global features are also very helpful to improve the linking accuracy.

Impact of Unlabeled Data Size. Figure 6 illustrates the impact of involving different size of unlabeled data. Observe that the joint model with multi-view learning outperforms the fully-supervised counterpart. This observation indicates that involving unlabeled data leads to more accurate models. The performance reaches its best when 5,000 unlabeled tweets are involved. Too many unlabeled tweets do not necessarily lead to better results.

Performance on NPOI/NIL Prediction. We now report the results on $NPOI$ and NIL prediction, listed in Table 6. For $NPOI$ prediction, all methods achieve excellent F_1 scores, above 0.94. The joint models outperform the pipelined approach by a small margin. For NIL prediction, the joint models outperform the pipelined approach in terms of Re and F_1, but deliver poorer Pr.

5.4 Error Analysis

Although the proposed joint model achieves better performance than state-of-the-art methods, there are still some errors that cannot be well addressed by the current model. As a case study, we show three examples in Table 7. We discuss the possible reasons for errors and aim to address these issues in our future work.

The first example shows an error propagation issue. The current model fails to recognize the right mention of "St . Ignatius Church"; subsequently it is impossible to link. The location profile-related features do not come to a rescue because the name of location profile

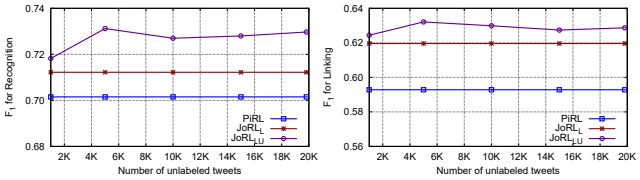


Figure 6: Impact of Unlabeled Data Size.

Table 7: Three kinds of failed examples with current model.

#	Failed example with ground truth annotation (next line)
1	a fundraising for Philippines OSLP at [**St**]$_{\text{NIL}}$. Ignatius [**Church**]$_{\text{NPOI}}$ a fundraising for Philippines OSLP at [**St . Ignatius Church**]$_{17519}$
2	*USER* [**kfc**]$_{1052}$ at [**lot 1**]$_{\text{NIL}}$ hahahaha *USER* [**kfc**]$_{1052}$ at [**lot 1**]$_{18835}$ hahahaha
3	It has been a long time since I had one for one at [**starbucks**]$_{2102}$ (: It has been a long time since I had one for one at [**starbucks**]$_{2071}$ (:

[17519] is "Church Of St. Ignatius". The difference in word order causes the failure in location mention recognition.

In the second example, "kfc" is correctly recognized and linked. Our model recognizes "lot 1" but fails to link it. The linking fails because of the significant difference between "lot 1" and "Lot One Shoppers' Mall", the name of the correct location profile [18835]. Interestingly, the phrase *lot 1* appears both in the tweet and the address attribute of the location profile [1052], which helps to decide the correct linking of "kfc". This is possible due to the third feature in location profile-related features in Table 2.

The last example has correct recognition, but incorrect linking. This error occurs due to the lack of context for correct linking. The system then chooses the one with highest popularity, which is different from the ground truth label. However, the human annotators are provided with four more contextual tweets for the right linking.

6. CONCLUSION

In this paper, we propose a novel joint model to recognize informal location mentions from tweet content and link the recognized locations to well-defined location profiles. The main advantage of the proposed model is that the joint model allows global features to alleviate the error propagation problem occurred in the traditional pipelined architecture. Based on the concept of multi-view learning, we further propose a semi-supervised learning algorithm to alleviate the dearth of labeled data. Experiments conducted on a labeled tweet collection and location profiles from Foursquare demonstrate the effectiveness of our proposed JoRL model.

Acknowledgments. This research was supported by Singapore Ministry of Education Academic Research Fund Tier 2 ARC30/12 and MOE2014-T2-2-066. The authors want to thank Xutao Li, Kaiqi Zhao and Quan Yuan for their helpful discussions and the anonymous reviewers for their valuable comments.

7. REFERENCES

[1] S. Abney. Bootstrapping. In *ACL*, pages 360–367, 2002.

[2] E. Amitay, N. Har'El, R. Sivan, and A. Soffer. Web-a-where: Geotagging Web Content. In *SIGIR*, pages 273–280, 2004.

[3] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, pages 92–100, 1998.

[4] U. Brefeld, C. Büscher, and T. Scheffer. Multi-view discriminative sequential learning. In *ECML*, pages 60–71. 2005.

[5] B. Cao, F. Chen, D. Joshi, and P. S. Yu. Inferring Crowd-Sourced Venues for Tweets. In *IEEE Big Data*, 2015.

[6] S. Chandra, L. Khan, and F. B. Muhaya. Estimating Twitter User Location Using Social Interactions–A Content Based Approach. In *IEEE SocialCom*, pages 838–843, 2011.

[7] H.-w. Chang, D. Lee, M. Eltaher, and J. Lee. @Phillies Tweeting from Philly? Predicting Twitter User Locations with Spatial Word Usage. In *ASONAM*, pages 111–118, 2012.

[8] M. Collins. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *ACL*, pages 1–8, 2002.

[9] M. Collins and B. Roark. Incremental Parsing with the Perceptron Algorithm. In *ACL*, 2004.

[10] S. Cucerzan. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *EMNLP-CoNLL*, pages 708–716, 2007.

[11] S. Dasgupta, M. L. Littman, and D. McAllester. PAC Generalization Bounds for Co-training. *NIPS*, pages 375–382, 2002.

[12] V. R. de Sa. Learning Classification with Unlabeled Data. In *NIPS*, pages 112–119, 1994.

[13] J. Eisenstein, B. O'Connor, N. A. Smith, and E. P. Xing. A Latent Variable Model for Geographic Lexical Variation. In *EMNLP*, pages 1277–1287, 2010.

[14] P. Ferragina and U. Scaiella. Tagme: On-the-fly Annotation of Short Text Fragments (by wikipedia entities. In *CIKM*, pages 1625–1628, 2010.

[15] D. Flatow, M. Naaman, K. E. Xie, Y. Volkovich, and Y. Kanza. On the Accuracy of Hyper-local Geotagging of Social Media Content. In *WSDM*, pages 127–136, 2015.

[16] S. Guo, M.-W. Chang, and E. Kiciman. To Link or Not to Link? A Study on End-to-End Tweet Entity Linking. In *NAACL*, 2013.

[17] B. Hecht, L. Hong, B. Suh, and E. H. Chi. Tweets from Justin Bieber's Heart: The Dynamics of the Location Field in User Profiles. In *SIGCHI*, pages 237–246, 2011.

[18] H. Huang, Y. Cao, X. Huang, H. Ji, and C.-Y. Lin. Collective Tweet Wikification based on Semi-supervised Graph Regularization. In *ACL*, pages 380–390, 2014.

[19] L. Huang, S. Fayong, and Y. Guo. Structured Perceptron with Inexact Search. In *NAACL*, pages 142–151, 2012.

[20] D. Inkpen, J. Liu, A. Farzindar, F. Kazemi, and D. Ghazi. Detecting and Disambiguating Locations Mentioned in Twitter Messages. In *Computational Linguistics and Intelligent Text Processing*, pages 321–332. 2015.

[21] S. Kinsella, V. Murdock, and N. O'Hare. "I'M Eating a Sandwich in Glasgow": Modeling Locations with Tweets. In *SMUC*, pages 61–68, 2011.

[22] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, pages 282–289, 2001.

[23] K. Lee, R. K. Ganti, M. Srivatsa, and L. Liu. When Twitter Meets Foursquare: Tweet Location Prediction Using Foursquare. In *MOBIQUITOUS*, pages 198–207, 2014.

[24] C. Li and A. Sun. Fine-grained Location Extraction from Tweets with Temporal Awareness. In *SIGIR*, pages 43–52, 2014.

[25] Q. Li and H. Ji. Incremental Joint Extraction of Entity Mentions and Relations. In *ACL*, pages 402–412, 2014.

[26] Q. Li, H. Ji, and L. Huang. Joint Event Extraction via Structured Prediction with Global Features. In *ACL*, pages 73–82, 2013.

[27] W. Li, P. Serdyukov, A. P. de Vries, C. Eickhoff, and M. Larson. The Where in the Tweet. In *CIKM*, pages 2473–2476, 2011.

[28] J. Lingad, S. Karimi, and J. Yin. Location Extraction from Disaster-related Microblogs. In *WWW*, pages 1017–1020, 2013.

[29] X. Liu, Y. Li, H. Wu, M. Zhou, F. Wei, and Y. Lu. Entity Linking for Tweets. In *ACL*, pages 1304–1311, 2013.

[30] G. Luo, X. Huang, C.-y. Lin, and Z. Nie. Joint Named Entity Recognition and Disambiguation. In *EMNLP*, pages 879–888, 2015.

[31] J. Mahmud, J. Nichols, and C. Drews. Home Location Identification of Twitter Users. *ACM Transactions on TIST*, pages 47:1—-47:21, 2014.

[32] E. Meij, W. Weerkamp, and M. de Rijke. Adding Semantics to Microblog Posts. In *WSDM*, pages 563–572, 2012.

[33] S. E. Middleton, L. Middleton, and S. Modafferi. Real-time Crisis Mapping of Natural Disasters using Social Media. *IEEE Intelligent Systems*, pages 9–17, 2014.

[34] R. Mihalcea and A. Csomai. Wikify!: Linking Documents to Encyclopedic Knowledge. In *CIKM*, pages 233–242, 2007.

[35] D. Milne and I. H. Witten. Learning to Link with Wikipedia. In *CIKM*, pages 509–518, 2008.

[36] S. Paradesi. Geotagging Tweets Using Their Content. In *Twenty-Fourth International FLAIRS Conference*, 2011.

[37] A. Rae, V. Murdock, A. Popescu, and H. Bouchard. Mining the Web for Points of Interest. In *SIGIR*, pages 711–720, 2012.

[38] L. Ratinov and D. Roth. Design Challenges and Misconceptions in Named Entity Recognition. In *CoNLL*, pages 147–155, 2009.

[39] S. Sarawagi and W. W. Cohen. Semi-markov Conditional Random Fields for Information Extraction. In *NIPS*, pages 1185–1192, 2004.

[40] A. Schulz, A. Hadjakos, H. Paulheim, J. Nachtwey, and M. Max. A Multi-Indicator Approach for Geolocalization of Tweets. In *ICWSM*, pages 573–582, 2013.

[41] W. Shen, J. Wang, P. Luo, and M. Wang. LIEGE: Link Entities in Web Lists with Knowledge Base. In *SIGKDD*, pages 1424–1432, 2012.

[42] W. Shen, J. Wang, P. Luo, and M. Wang. Linking Named Entities in Tweets with Knowledge Base via User Interest Modeling. In *SIGKDD*, pages 68–76, 2013.

[43] A. Sil and A. Yates. Re-ranking for Joint Named-Entity Recognition and Linking. In *CIKM*, pages 2369–2374, 2013.

[44] J. Yin, S. Karimi, and J. Lingad. Pinpointing Locational Focus in Microblogs. In *ADCS*, pages 66:66—-66:72, 2014.

[45] Y. Zhang and S. Clark. Joint Word Segmentation and POS Tagging using a Single Perceptron. In *ACL*, pages 888–896, 2008.