Learning User Latent Attributes on Social Media

Binxuan Huang

Institute for Software Research
Carnegie Mellon University

This thesis proposal is submitted for the degree of
Doctor of Philosophy

School of Computer Science August 2019

Table of contents

Chapter 1

Introduction

In recent years, there is a growing interest in using social media to understand social phenomena. Researchers have demonstrated many important applications of user activity understanding in online social media, such as presidential election prediction [82], earthquake early detection [74], and disaster management [15].

Commonly, a social media site is mixed with users with various attributes, in terms of gender, location, political affiliation, social roles, and etc. Different types of users may have different motivations, different opinions towards certain topics, different resources at their disposal, different behaviors in events. If researchers want to understand what is happening on a social media site, it is important to know where a post comes from, who wrote this post, and which party the author belongs to.

In this thesis, the goal is to predict users' latent attributes such as their locations, social identities, and political orientations. Many times, these attributes are not explicitly provided by users. For example, there are no ways for users to specify their ideologies on Twitter. Even though there is a location field in a Twitter user's profile, it is often empty or filled with noisy information unrelated with any geographical location [35]. However, many times online users would unavoidably provide indicative clues which are helpful for identifying their attributes in their posts. For example, from a post "food in cmu is delicious :)", we can infer that this user is probably a student living in Pittsburgh.

Thanks to the massive text data on social media, we can learn rich knowledge from text to predict users' attributes. In the meanwhile, text data from social media often comes with a significant amount of meta data. Take Twitter for example, a single tweet object contains one short text with multiple meta fields like posting time, tweet language, user's personal description, and etc. How to efficiently handle text data combined with meta information still needs to be considered. Furthermore, data from social networks also contains rich connection information, eg. mentioning, following. It is still a challenge task to combine text, meta data, user network together for user attributes prediction.

In this thesis, I will approach user attributes prediction at three levels — single post, user timeline, graph-level classification. First, I will present a global location prediction system that uses one single tweet as input to learn one user's location. It utilizes location-related features in a tweet, such as text and user profile metadata. Second, I plan to extend the tweet-level prediction system to user-level, which combines multiple posts in one user's timeline. I will demonstrate the effectiveness of this model on the task of user social identity classification. An improved user-level hierarchical location prediction model will also be presented. In these described models, I mainly focus on learning user attributes from users themselves. In the next step, I will consider social graph as additional information to improve performance. Users connected in a social network often show similarities in certain aspects, which is a well-known phenomenon called social homophily [49]. Last, to reduce the computation cost for learning each individual attribute separately, I intend to propose a multi-task learning system that learns all attributes jointly.

Unlike previous work, my approaches use neural network to learn rich text representations and combine various feature fields. With the graph-level classification framework, the performance will be improved a lot. Though I mainly use tweet data to predict user attributes like location, social identity, the methodology can be easily extended to other platforms like Facebook and Weibo, as well as other characteristics, eg. gender and age.

1.1 Background

Web user attributes prediction or user profiling has long been studied in the literature. Typically, researchers first define several categories to describe users. Then they use machine learning methods to classify users into these predefined attribute categories. Early work first represents a research paper by normalized term frequency, then these term frequency vectors are feed into a k-Nearest Neighbour classifier to classify topics of these papers[50]. Thus a user's research interests can be inferred by using cumulative paper topics. Similarly, Godoy et al. also transform webpages into term vectors, but they use hierarchical clustering instead of classification to group users based on their web interests [26].

Later, various feature engineering based methods are proposed to improve the performance of user attributes prediction. Following these term frequency based methods, Shmueli-Scheuer et al. further apply feature selection method to choose the best feature combinations [76]. Rao et al. create several features like tweeting frequency and number of followers in addition to the term features for Twitter user demographic prediction [69].

Similarly, Pennacchiotti add topic features from an LDA model and sentiment features from a sentiment lexicon for Twitter user classification [56]. Great improvement has been achieved with these hand-crafted features.

In recent years, deep neural network based methods show great learning capacity for various machine learning tasks, eg. computer recognition [34], language modeling [21], text classification [20]. Different from previous research, I intend to use deep neural networks to get better representative features without heavy feature engineering. The methods proposed in this thesis can also be easily generalized to other knowledge domains. Although some previous work use network features to improve their performance [56, 2, 69], most of them only touch some simple network heuristics like number of followers, n-popular friends. These methods may not fully utilize the network structure information. In this work, I will incorporate graph neural networks that propagate features from network neighbourhoods to make a better decision.

1.2 Datasets

1.2.1 Location Datasets

For tweet-level geolocation prediction, we collect geotagged tweets from the whole world between January 7, 2017 and Febuary 1, 2017. For each user appeared in this collection, we randomly select one tweet for each city that one user has visited before. There are 3,321,194 users and 4,645,692 tweets in total.

Table 1.1 Summaries about the tweet-level location prediction dataset. Numbers in brackets are standard deviation.

# of tweets	# of users	# of timezones	# of lang.	# of countries (or regions)	Tweets per country	# of cities	Tweets per city
4645692	3321194	417	103	243	19118.0 (99697.1)	3709	1252.5(4184.5)

For user-level location prediction, we adopt three commonly used benchmark datasets. They are Twitter-US, Twitter-World, and WNUT.

Twitter-US is a dataset compiled by Roller et al. [72]. It contains 429K training users, 10K development users, and 10K test users in North America. The ground truth location of each user is set to the first geotag of this user in the dataset.

Twitter-World is a Twitter dataset covering the whole world, with 1,367K training users, 10K development users, and 10K test users [30]. The ground truth location for each user is the center of the closest city to the first geotag of this user. Only English tweets are included in this dataset, which makes it more challenging for a global-level location prediction task.

We downloaded these two datasets from website [1]. Each user in these two datasets is represented by the concatenation of their tweets, followed by the geo-coordinates. We queried Twitter's API to add user metadata information to these two datasets in February 2019. We only get metadata for about 53% and 67% users in Twitter-US and Twitter-World respectively. Because of Twitter's privacy policy change, we could not get the time zone information anymore at the time of collection.

WNUT is released in the 2nd Workshop on Noisy User-generated Text [33]. The original user-level dataset consists of 1 million training users, 10K users in development set and test set each. Because of Twitter's data sharing policy, only tweet ids of training and development data are provided. We have to query Twitter's API to reconstruct the training and development dataset. We finished our data collection around August 2017. About 25% training and development users' data cannot be accessed at that time. The full anonymized test data is downloaded from the workshop website [2].

Table 1.2 shows a brief summary of these three user-level location prediction datasets we use here.

Table 1.2 A brief summary of our user-level location prediction datasets. For each dataset, we report the number of users, number of users with metadata, number of tweets, and average number of tweets per user. We collected metadata for 53% and 67% of users in Twitter-US and Twitter-World. Time zone information was not available when we collected meta data for these two datasets. About 25% of training and development users' data was inaccessible when we collected WNUT in 2017.

	Twitter-US			Twitter-World			WNUT		
	Train	Dev.	Test	Train	Dev.	Test	Train	Dev.	Test
# users	429K	10K	10K	1.37M	10K	10K	742K	7.46K	10K
# users with meta	228K	5.32K	5.34K	917K	6.50K	6.48K	742K	7.46K	10K
# tweets	36.4M	861K	831K	11.2M	488K	315K	8.97M	90.3K	99.7K
# tweets per user	84.60	86.14	83.12	8.16	48.83	31.59	12.09	12.10	9.97

1.2.2 Identity Datasets

We build two datasets from Twitter — public figure dataset, identity dataset. In our first public figure dataset, we use Twitter's verification as a proxy for public figures, and these

[1] https://github.com/afshinrahimi/geomdn

[2] https://noisy-text.github.io/2016/geo-shared-task.html

Table 1.3 A brief summary of our two identity datasets.

	Public figure		Identity						
	Verified	Unverified	Media	Reporter	Celebrity	Government	Company	Sport	Normal
Train	152368	365749	1140	614	876	844	879	870	6623
Dev.	1452	3548	52	23	38	40	35	43	269
Test	2926	7074	97	39	75	81	66	74	568

verified accounts include users in music, government, sports, business, and etc[3]. We sampled 156746 verified accounts and 376371 unverified accounts through Twitter's sample stream data [4]. Then we collected their most recent 20 tweets from Twitter's API in November 2018. We randomly choose 5000 users as a development set and 10000 users as a test set. A brief summary of this dataset is shown in Table 1.3.

In addition, we introduce another human labeled identity dataset for fine-grained identity classification, which contains seven identity classes: "news media", "news reporter", "government official", "celebrity", "company", "sport", and "normal people". For each identity, we manually labelled thousands of Twitter users and collected their most recent 20 tweets for classification in November 2018. For the normal Twitter users, we randomly sampled them from the Twitter sample stream.

1.2.3 Ideology datasets

I am developing one large-scale political orientation dataset for training neural network. We will first identify several popular political figures' Twitter accounts, then crawl followers of these accounts. Those followers who only follow one party will be used as noisy-labeled data.

We will also use another human labeled introduced by Preoctiuc et al.[60]. It consists 3,938 users who self-report their political ideology on a seven point scale: Very conservative (1), Conservative (2), Moderately conservative (3), Moderate (4), Moderately liberal (5), Liberal (6), Very liberal (7).

1.3 Summary of Proposed Work and Thesis Timeline

This work will be composed by seven chapters.

Chapter 1 introduces this thesis and provide some background information.

[3]https://help.twitter.com/en/managing-your-account/about-twitter-verified-accounts

[4]https://developer.twitter.com/en/docs/tweets/sample-realtime/overview/GET_statuse_sample.html

Events	Date
Chapter 2: Tweet-level Location Prediction	Done
Chapter 3: User-level Identity Classification	Done
Chapter 4: Hierarchical User Location Prediction	Done
Propose	Aug. 22, 2019
Chapter 5: Graph-level Attributes Prediction	Jan. 11, 2020
Chapter 6: User Attributes Prediction via Multi-task Learning	May 18, 2020
Final thesis	June 30, 2020
Defense	July 15, 2020

Table 1.4 Timeline for proposed work

Chapter 2 presents one tweet-level location prediction system that can operate on the global-level with multi-lingual support. It combines heterogeneous feature sources in a single model and yields better performance than one previous state-of-the-art stacking method.

Chapter 3 and 4 are two user-level prediction models. One is for user-level identity classification, another is for hierarchical location prediction. These models extract features from various text sources and learn correlations between these features.

In chapter 5, I will explore ways to combine social networks into our prediction models. Specifically, I plan to use previous user-level architecture to extract user representations and build a graph attention network on top these user representation features.

In chapter 6, I plan to predict all these attributes together via multi-task learning.

Chapter 7 summarize this thesis and provide some discussion.

Table 1.4 is the timeline for the porposed work.

Chapter 2

Tweet-level Location Prediction (Completed)

2.1 Introduction

Recently, there is growing interest in using social media to understand social phenomena. For example, researchers have shown that analyzing social media reveals important geospatial patterns for keywords related to presidential elections[82]. People can use Twitter as a sensor to detect earthquakes in real-time[74]. Recent research also has demonstrated that Twitter data provides real-time assessments of flu activity[1].

Using Twitter's API[1], a keyword search can be done and we can easily get tweet streams from across the world containing keywords of interest. However, we cannot conduct a fine-grained analysis in a specific region using such a keyword-based search method. Alternatively, using the same API tweets with geo-information can be collected via a bounding box. Since less than 1% of tweets are tagged with geo-coordinates[28], using this location-based search means we will lose the majority of the data. If we can correctly locate those ungeotagged tweets returned from a keyword search stream, that would enable us to study users in a specific region with far more information.

With this motivation, we are aiming to study the problem of inferring a tweet's location. Specifically, we are trying to predict on a tweet by tweet basis, which country and which city it comes from. Most of the previous studies rely on rich user information(tweeting history and/or social ties), which is time-consuming to collect because of the Twitter API's speed limit. Thus those methods could not be directly applied to Twitter streams. In this paper, we study a global location prediction system working on each single tweet. One data

[1]https://dev.twitter.com/docs

sample is one tweet JSON object returned by Twitter's streaming API. Our system utilizes location-related features in a tweet, such as text and user profile meta-data. We summarize useful features that can provide information for location prediction in Table **??**.

Table 2.1 Feature table for tweet location prediction.

Feature	Type
Tweet content	Free text
User personal description	Free text
User name	Free text
User profile location	Free text
Tweet language(TL)	Categorical
User language(UL)	Categorical
Timezone(TZ)	Categorical
Posting time(PT)	UTC timestamp

Recent research has shown that using bag-of-words and classical machine learning algorithms such as Naive Bayes can provide us a text-based location classifier with good accuracy[32]. Different from previous research, we intend to use the convolutional neural network(CNN) to boost prediction power. Inspired by the success of convolutional neural network in text classification[44], we are going to use CNN to extract location related features from texts and train a classifier that combines high-level text feature representations with these categorical features. To benchmark our method, we compared our approach with a stacking-based method. Experimental results demonstrate that our approach achieves 92.1% accuracy on country-level prediction and 52.8% accuracy on city-level prediction, which greatly outperforms our baseline methods on both tasks.

2.2 Related Work

There is increasing interest in inferring Twitter user's location, which is largely driven by the lack of sufficient geo-tagged data[28]. In many situations, it is important to know where a tweet came from in order to use the information in the tweet to effect a good social outcome. Key examples include: disaster relief[47], earthquake detection[24], and predicting flu trends[1].

A majority of previous works either focus on a local region e.g. United States[16], Sweden[7], or using rich user information like a certain number of tweets for each user[16], user's social relationship[43, 62]. Different from these works, this paper works on worldwide tweet location prediction. We only utilized features in one single tweet without any external information. Thus this method could be easily applied to real-time Twitter stream.

For fine-grained location prediction, there are several types of location representation methods existing in literature. One typical method is to divide earth into small grids and try to predict which cell one tweet comes from. Wing and Baldridge introduced a grid-based representation with fixed latitude and longitude[86]. Based on the similarity measured by Kullback-Leibler divergence, they assign each ungeotagged tweet to the cell with most similar tweets. Because cells in urban area tend to contain far more tweets than the ones in rural areas, the target classes are rather imbalanced[31]. To overcome this, Roller et al. further proposed an adaptive grid representation using K-D tree partition[72]. Another type of representation is topic region. Hong et al. proposed a topic model to discover the latent topic words for different regions[38]. Such parametric generative model requires a fixed number of regions. However, the granularity of topic regions is hard to control and will potentially vary over time[32].

The representation we choose is city-based representation considering most tweets come from urban area. One early work proposed by Cheng et al. using a probabilistic framework to estimate Twitter user's city-level location based on the content of tweets[16]. Their framework tries to identify local words with probability distribution smoothing. However, such method needs a certain number of tweets(100) for each user to get a good estimation. Han et al. proposed a stacking-based approach to predict user's city[31]. They combine tweet text and meta-data in user profile with stacking[87]. Specifically, they train a multinomial naive Bayes base classifier on tweet text, profile location, timezone. Then they train a meta-classifier over the base classifiers. More recently, Han et al. further did extensive experiments to show that using feature selection method, such as information gain ratio[64] could greatly improve the classification performance.

2.3 Tweet Location Prediction

2.4 Location prediction

In this section, we will introduce our location prediction approach. We first briefly describe the useful features in a tweet JSON object. After that, we will further explain how we utilize these features in our prediction model.

2.4.1 Feature Set

We have listed all useful information we want to utilize in Table **??**. Tweet content, user personal description, user name and profile location are four text fields that we will use.

Twitter users often reveal their home location in their profile location and personal description. However, location indicating words are often mixed with informal tweet text(e.g. chitown for Chicago). It is unrealistic to use a gazetteer to find these words. In this work, we choose to apply CNN on these four text fields to extract high-level representations.

In addition to these four text fields. there are another three categorical features: tweet language, user language,and timezone. Tweet language is automatically determined by Twitter's language detection tool. User language and timezone are selected by the user in his/her profile. These three categorical features are particularly useful for distinguishing users at the country-level.

The last feature is UTC posting time. Using posting timestamp as a discriminative feature is motivated by the fact that people in a region are more active on Twitter at certain times during the day. For example, while people in United Kingdom start to be active at 9:00 am in UTC time, most of the people in United States are still asleep. We transform the posting time in UTC timestamp into discrete time slots. Specifically, we divide 24 hours into 144 time slots each with a length of 10 minutes. Thus each tweet will have a discrete time slot number in the range of 144, which can be viewed as a categorical feature. In Figure 2.1, we plotted the probability distribution of an user posting tweets in each time slot in three different countries. As expected, there is a big variance between these three countries.

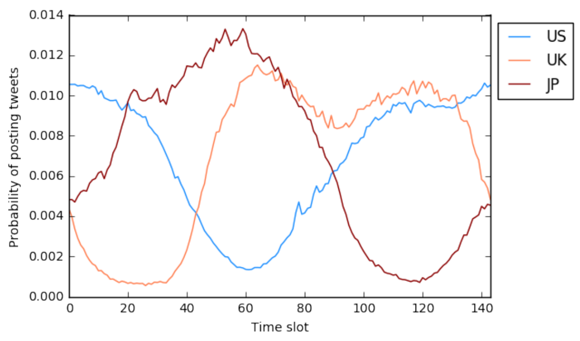

Fig. 2.1 The probability of an user posting a tweet in different time slot in three different countries: United States, United Kingdom, Japan.

2.4.2 Our Approach

Our approach is based on the convolutional neural network for sentence classification proposed by Kim[44]. Different from traditional bag-of-words method, such convolutional neural networks take the word order into consideration. Our model architecture is shown in Figure 4.1. We use this CNN architecture to extract high-level features from four text fields in a tweet. Let $x_i^t \in R^k$ be the k-dimensional word vector corresponding to the *i*-th word in the text t, where $t \in \{$tweet content, user description, profile location, user name$\}$. As a result, one text of length n can be represented as a matrix

$$X^t_{1:n} = x^t_1 \oplus x^t_2 \oplus .... \oplus x^t_n \tag{2.1}$$

where $\oplus$ is concatenation operator. In the convolutional layer, we apply each filter $w \in R^{hk}$ to all the word vector matrices, where h is the window size and k is the length of a word vector. For example, applying filter w to a window of word vectors $x^t_{i:i+h-1}$, we generated $c^t_i = f(w \cdot x^t_{i:i+h-1} + b)$. Here $b \in R$ is a bias term and we choose $f(x)$ as a non-linear ReLU function $max(x, 0)$. Sliding the filter window from the beginning of a word matrix till the end, we generated a feature vector $c^t = [c^t_1, c^t_2, ..., c^t_{n-h+1}]$ for each text t. If we have m filters in the convolutional layer, then we can produce m feature vectors for each text field and $4m$ vectors in total.

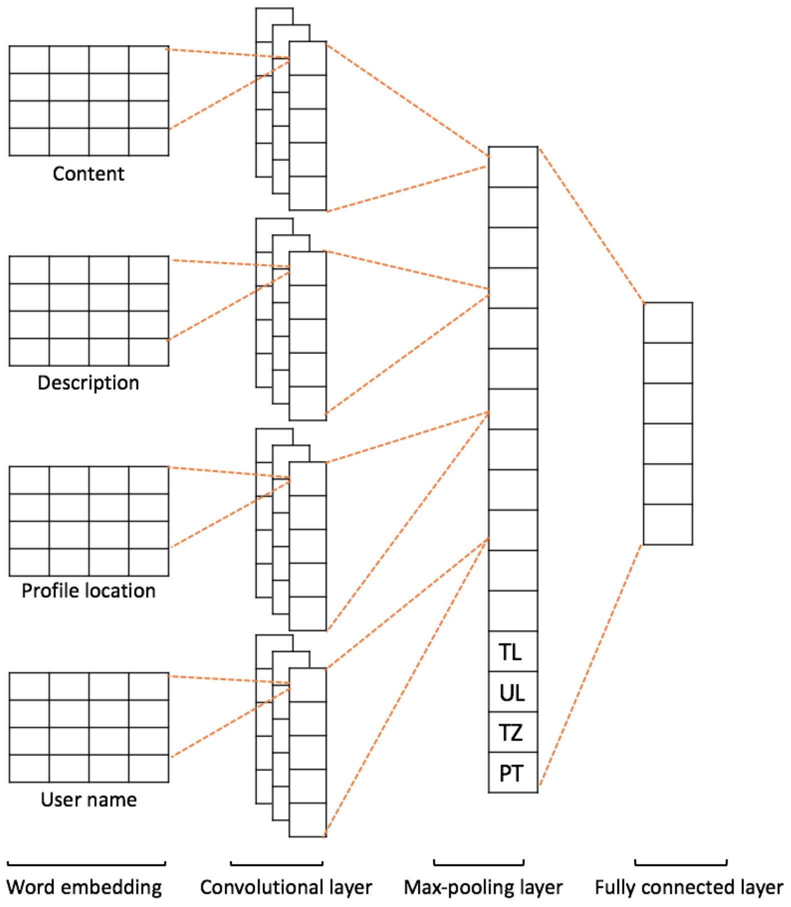

Fig. 2.2 Architecture of our tweet location prediction model.

In the max-pooling layer, we apply a pooling operation over each feature vector generated in the convolutional layer. Each pooling operation takes a feature vector as input and outputs the maximum value $\hat{c}^t = max(c^t)$. $\hat{c}^t$ can be viewed as the most representative feature generated by a filter on text t. Hence we finally got a long vector $\theta \in R^{4m}$ after the max-pooling layer. To avoid the co-adaptation of hidden units, we apply dropout on the max-pooling layer that randomly set elements in θ to zero in the training phase. After that, we append four categorical features tweet language(TL), user language(UL), timezone(TZ) and posting time(PT) with one-hot encoding at the end of θ and get $\hat{\theta}$. In the last fully connected layer, we use a softmax function over this long vector $\hat{\theta}$ to generate the probability distribution over locations. Specifically, the probability of one tweet coming from location l_i is

$$P(l_i|\hat{\theta}) = \frac{exp(\beta_i^T \hat{\theta})}{\sum_{j=1}^{L} exp(\beta_j^T \hat{\theta})} \tag{2.2}$$

where L is the number of locations and β_i are parameters in softmax layer. The output predicted location is just the location with highest probability.

The minimization objective in the training phase is the categorical cross-entropy loss. The parameters to be estimated include word vectors, weight vectors w for each filters, the weight vectors β in softmax layer, and all the bias terms. The optimization is performed using mini-batch stochastic gradient descent and back-propagation[73].

2.5 Experiments

2.5.1 Evaluation Measures

Following previous work of tweet geolocation prediction[31], we used four evaluation measures listed below. One thing to note is that when we calculated the error distance we used distance between predicted city and the true coordinates in the tweet rather than the center of assigned closest city.

- Acc: The percentage of correct location predictions.
- Acc@Top5: The percentage of true location in our top 5 predictions.
- Acc@161: The percentage of predicted city which are within a 161km(100 mile) radius of the true coordinates in the original tweet to capture near-misses. This measure is only tested on city-level prediction.

- Median: The median distance from the predicted city to the true coordinates in the original tweet. This measure is only tested on city-level prediction.

2.5.2 Baseline Method

We compared our approach with one commonly used ensemble method in previous research works [31, 32]. We implemented an ensemble classifier based on stacking[87] with 5-fold cross validation. The training of stacking consists of two steps. First, five multinomial naive Bayes base classifiers are trained on different types of data(tweet content, user description, profile location, user name and the remaining categorical features). The outputs from the base classifiers are used to train a multinomial naive Bayes classifier in the second layer. We call such method STACKING in this paper. Same as [32], we also use information gain ratio to do feature selection on text tokens. We call STACKING with feature selection STACKING+.

2.5.3 Hyperparameters and Training

We used a tweet-specific tokenizer provided by NLTK to tokenize text fields. We built our dictionary based on the words that appeared in text, user description, and profile location. To reduce low-utility words and noise, we removed all words that had a word frequency less than 10. For our proposed approach, we used filter windows(h) of 3,4,5 with 128 feature vectors each, a dropout rate of 0.5 and batch size of 1024. We initialize word vectors using word2vec vectors trained on 100 billion words from Google News. The vectors have dimensionality of 300 and were trained using the continuous bag-of-words architecture[51]. For those words that are not included in word2vec, we initialized them randomly. We also performed early stopping based on the accuracy over the development set. Training was done through stochastic gradient descent using Adam update rule with learning rate 10^{-3}[45]. For our baseline models, we applied additive smoothing with $\alpha = 10^{-2}$, which is selected on the development set. For STACKING+ method, we first ranked these words by their information gain ratio value, then selected the top n% words as our vocabulary. The tuning of n is based on accuracy over the development set. We selected n as 40%, 55% for city-level prediction and country-level prediction respectively.

2.5.4 Results

The comparison results between our approach and the baseline methods are listed in Table 2.2. Our approach achieves 92.1% accuracy and 52.8% accuracy on country-level and city-level location prediction respectively. Our approach is consistently better than the previous model

Table 2.2 Country-level and city-level location prediction results.

	Country		City			
	Acc	Acc@Top5	Acc	Acc@161	Acc@Top5	Median
STACKING	0.868	0.947	0.389	0.573	0.595	77.5 km
STACKING+	0.871	0.950	0.439	0.616	0.629	47.2 km
Our approach	**0.921**	**0.972**	**0.528**	**0.692**	**0.711**	**28.0** km

on the country-level location prediction task as shown in Table 2.2. It greatly outperforms our baseline methods over all the measures, especially on the city-level prediction task. It could assign more than half of the test tweets to the correct city and gain more than 20% relative improvement over the accuracy of the STACKING+ method.

Our approach performs better for countries with a large number of tweets. In Figure 2.3, we plotted the precision and recall value for each country as a scatter chart. The dot size is proportional to the number of tweets that come from that country. Turkey appears to be the country with highest precision and recall. These results suggest that our approach works better with more data samples.

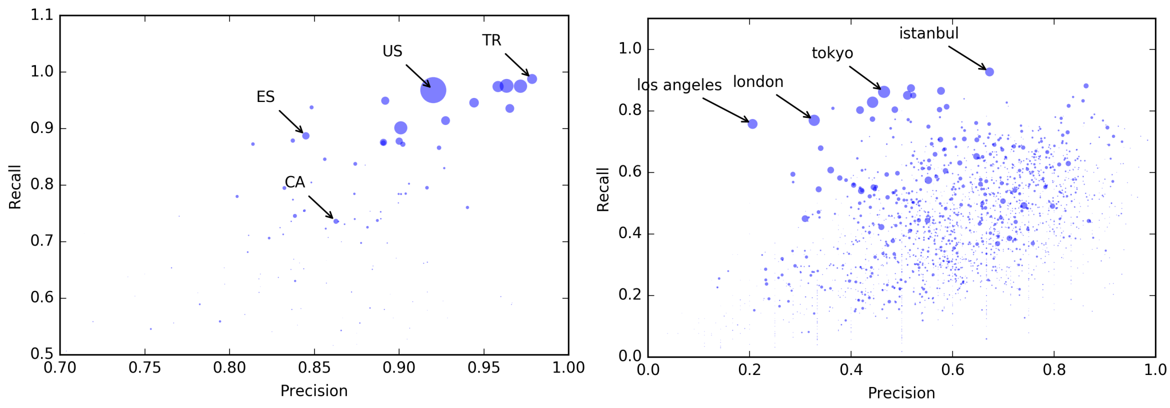

Fig. 2.3 Two scatter graphs that show the performances for each country and cities. The x-axis is precision, y-axis is recall. Each dot represents a country/city. The dot size is proportional to the number of tweets that comes from the correponding location. Some tiny invisible country outside of the scope are not shown in the figure.

The same graph is also plotted for city prediction in Figure 2.3. Because of the skewness of our data and the difficulty of city-level prediction, our classifier tends to generate labels towards big cities, which leads to high recall and low precision for cities like Los Angeles.

Chapter 3

User-level Social Identity Classification (Completed)

3.1 Introduction

An identity is a characterization of the role an individual takes on. It is often described as the social context specific personality of an individual actor or a group of people [4]. Identities can be things like jobs (e.g. “lawyer”, “teacher”), gender (man, woman), or a distinguishing characteristic (e.g. “a shy boy”, “a kind man”). People with different identities tend to exhibit different behaviors in the social space [12]. In this paper, we use identity to refer to the roles individuals or groups play in society.

Specifically on social media platforms, there are many different kinds of actors using social media, e.g., people, organizations, and bots. Each type of actors has different motivations, different resources at their disposal, and may be under different internal policies or constraints on when they can use social media, how they can represent themselves, and what they can communicate. If we want to understand who is controlling the conversation and whom is being impacted, it is important to know what types of actors are doing what.

To date, for Twitter, most research has separated types of actors largely based on whether the accounts are verified by Twitter or not [36], or whether they are bots or not [17]. However, a variety of different types of actors may be verified - e.g., news agencies, entertainment or sports team, celebrities, and politicians. Similarly, bots can vary - e.g., news bots and non-news bots. If we could classify the identities of actors on Twitter, we could gain an improved understanding of who was doing the influencing and who was being influenced [14]. This would lead to improved accuracy in measuring the impact of marketing and influence campaigns.

In this paper, the primary goal is to classify Twitter users based on their identities on social media. First, we introduce two datasets for Twitter user identity classification. One is automatically collected from Twitter aiming at identifying public figures on social media. Another is a human labeled dataset for more fine-grained Twitter user identity classification, which includes identities like government officials, news reporters, etc. Second, we present a hierarchical self-attention neural network for Twitter user identity classification. In our experiments, we show our method achieves excellent results when compared to many strong classification baselines. Last but not least, we propose a transfer learning scheme for fine-grained user identity classification which boosts our model's performance a lot.

3.2 Related Work

Sociologists have long been interested in the usage of identities across various social contexts [79]. As summarized in [78], three relatively distinct usages of *identity* exist in the literature. Some use identity to refer to the culture of a people [11]. Some use it to refer to common identification with a social category [80]. While others use identity to refer to the role a person plays in highly differentiated contemporary societies. In this paper, we use the third meaning. Our goal for identity classification is to separate actors with different roles in online social media.

Identity is the way that individuals and collectives are distinguished in their relations with others [41]. Certain difficulties still exist for categorizing people into different groups based on their identities. Recasens et al. [70] argue that identity should be considered to be varying in granularity and a categorical understanding would limit us in a fixed scope. While much work could be done along this line, at this time we adopt a coarse-grained labeling procedure, that only looks at major identities in the social media space.

Twitter, a popular online news and social networking site, is also a site that affords interactive identity presentation to unknown audiences. As pointed out by Robinson [71], individuals form new cyber identities on the internet, which are not necessarily the way they would be perceived offline. A customized identity classifier is needed for online social media like Twitter.

A lot of research has tried to categorize Twitter users based on certain criteria, like gender [9], location [39], and political orientation [19]. Another similar research topic is bot detection [17], where the goal is to identify automated user accounts from normal Twitter accounts. Differing from them, our work tries to categorize Twitter users based on users' social identity or social roles. Similarly, Priante et al. [61] also study identity classification on Twitter. However, their approach is purely based on profile description, while we combine user self-description and tweets together. Additionally, we demonstrate that tweets are more helpful for identity classification than personal descriptions in our experiments.

In fact, learning Twitter users' identities can benefit other related tasks. Twitter is a social media where individual user accounts and organization accounts co-exist. Many user classification methods may not work on these organization accounts, e.g., gender classification. Another example is bot detection. In reality, accounts of news agencies and celebrities often look like bots [18], because these accounts often employ automated services or teams (so called cyborgs), and they also share features with certain classes of bots; e.g., they may be followed more than they follow. Being able to classify actors' roles on Twitter would improve our ability to automatically differentiate pure bots from celebrity accounts.

3.3 Method

In this section, we describe details of our hierarchical self-attention neural networks. The overall architecture is shown in Figure 3.1. Our model first maps each word into a low dimension word embedding space, then it uses a Bidirectional Long Short-Term Memory (Bi-LSTM) network [37] to extract context specific semantic representations for words. Using several layers of multi-head attention neural networks, it generates a final classification feature vector. In the following parts, we elaborate these components in details.

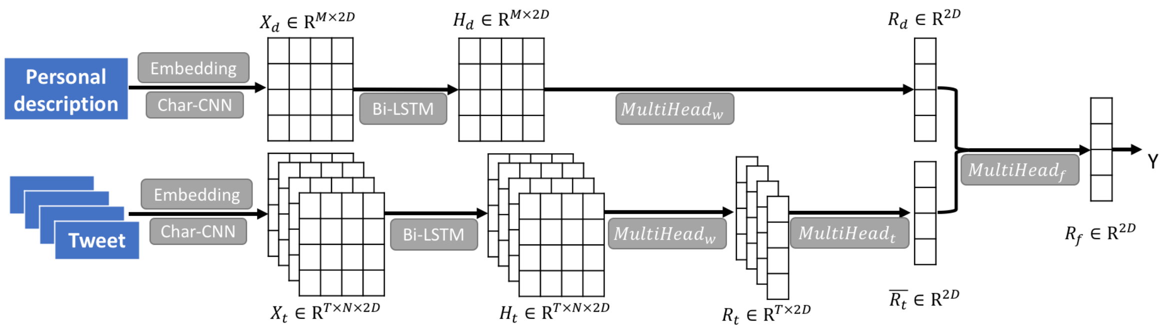
$X_d \in R^{M \times 2D}$
$H_d \in R^{M \times 2D}$
$R_d \in R^{2D}$
Personal description
Embedding
Char-CNN
Bi-LSTM
$MultiHead_w$
$MultiHead_f$
Y
Tweet
Embedding
Char-CNN
Bi-LSTM
$MultiHead_w$
$MultiHead_t$
$R_f \in R^{2D}$
$X_t \in R^{T \times N \times 2D}$
$H_t \in R^{T \times N \times 2D}$
$R_t \in R^{T \times 2D}$
$\overline{R_t} \in R^{2D}$

Fig. 3.1 The architecture of hierarchical self-attention neural networks for identity classification.

3.3.1 Word Embedding

Our model first maps each word in description and tweets into a word embedding space $\in R^{V \times D}$ by a table lookup operation, where V is the vocabulary size, and D is the embedding dimension.

Because of the noisy nature of tweet text, we further use a character-level convolutional neural network to generate character-level word embeddings, which are helpful for dealing with out of vocabulary tokens. More specifically, for each character c_i in a word $w = (c_1, ..., c_k)$, we first map it into a character embedding space and get $v_{c_i} \in R^d$. Then a convolutional neural network is applied to generate features from characters [44]. For a character window $v_{c_i:c_{i+h-1}} \in R^{h \times d}$, a feature θ_i is generated by $\theta_i = f(w \cdot v_{c_i:c_{i+h-1}} + b)$ where $w \in R^{h \times d}$ and b are a convolution filter and a bias term respectively, $f(\cdot)$ is a non-linear function *relu*. Sliding the filter from the beginning of the character embedding matrix till the end, we get a feature vector $\theta = [\theta_1, \theta_2, ..., \theta_{k-h+1}]$. Then, we apply max pooling over this vector to get the most representative feature. With D such convolutional filters, we get the character-level word embedding for word w.

The final vector representation $v_w \in R^{2D}$ for word w is just the concatenation of its general word embedding vector and character-level word embedding vector. Given one description with M tokens and T tweets each with N tokens, we get two embedding matrices $X_d \in R^{M \times 2D}$ and $X_t \in R^{T \times N \times 2D}$ for description and tweets respectively.

3.3.2 Bi-LSTM

After get the embedding matrices for tweets and description, we use a bidirectional LSTM to extract context specific features from each text. At each time step, one forward LSTM takes the current word vector v_{w_i} and the previous hidden state $\overrightarrow{h_{w_{i-1}}}$ to generate the hidden state for word w_i. Another backward LSTM generates another sequence of hidden states in the reversed direction.

$$\begin{aligned} \overrightarrow{h_{w_i}} &= \overrightarrow{LSTM}(v_{w_i}, \overrightarrow{h_{w_{i-1}}}) \\ \overleftarrow{h_{w_i}} &= \overleftarrow{LSTM}(v_{w_i}, \overleftarrow{h_{w_{i+1}}}) \end{aligned} \tag{3.1}$$

The final hidden state $h_{w_i} \in R^{2D}$ for word w_i is the concatenation of $\overrightarrow{h_{w_i}}$ and $\overleftarrow{h_{w_i}}$ as $h_{w_i} = [\overrightarrow{h_{w_i}}, \overleftarrow{h_{w_i}}]$. With T tweets and one description, we get two hidden state matrices $H_t \in R^{T \times N \times 2D}$ and $H_d \in R^{M \times 2D}$.

3.3.3 Attention

Following the Bi-LSTM layer, we use a word-level multi-head attention layer to find important words in a text [83].

Specifically, a multi-head attention is computed as follows:

$$MultiHead(H_d) = Concat(head_1, ..., head_h)W^O$$
$$head_i = softmax(\frac{H_d W_i^Q \cdot (H_d W_i^K)^T}{\sqrt{d_k}})H_d W_i^V$$

where $d_k = 2D/h$, W_i^Q, W_i^K, $W_i^V \in R^{2D \times d_k}$, and $W^O \in R^{hd_k \times 2D}$ are projection parameters for query, key, value, and output respectively.

Take a user description for example. Given the hidden state matrix H_d of the description, each head first projects H_d into three subspaces — query $H_d W_i^Q$, key $H_d W_i^K$, and value $H_d W_i^V$. The matrix product between key and query after softmax normalization is the self-attention, which indicates important parts in the value matrix. The multiplication of self-attention and value matrix is the output of this attention head. The final output of multi-head attention is the concatenation of h such heads after projection by W^O.

After this word-level attention layer, we apply a row-wise average pooling to get a high-level representation vector for description.

$$R_d = row\_avg(MultiHead_w(H_d)) \in R^{2D} \tag{3.2}$$

Similarly, we can get T representation vectors from T tweets using the same word-level attention, which forms $R_t \in R^{T \times 2D}$.

Further, a tweet-level multi-head attention layer computes the final tweets representation vector $\bar{R}_t$ as follows:

$$\bar{R}_t = row\_avg(MultiHead_t(R_t)) \in R^{2D} \tag{3.3}$$

In practise, we also tried using an additional Bi-LSTM layer to model the sequence of tweets, but we did not observe any significant performance gain.

Given the description representation R_d and tweets representation $\bar{R}_t$, a field attention generates the final classification feature vector

$$R_f = row\_avg(MultiHead_f([R_d; \bar{R}_t])) \tag{3.4}$$

where $[R_d; \bar{R}_t] \in R^{2 \times 2D}$ means concatenating by row.

3.3.4 Final Classification

Finally, the probability for each identity is computed by a softmax function:

$$P = softmax(WR_f + b) \tag{3.5}$$

where $W \in R^{|C| \times 2D}$ is the projection parameter, $b \in R^{|C|}$ is the bias term, and C is the set of identity classes. We minimize the cross-entropy loss function to train our model.

3.4 Experiments

3.4.1 Hyperparameter Setting

In our experiments, we initialize the general word embeddings with released 300-dimensional Glove vectors[1] [57]. For words not appearing in Glove vocabulary, we randomly initialize them from a uniform distribution $U(-0.25, 0.25)$. The 100-dimensional character embeddings are initialized with a uniform distribution $U(-1.0, 1.0)$. These embeddings are adapted during training. We use filter windows of size 3,4,5 with 100 feature maps each. The state dimension D of LSTM is chosen as 300. For all the multi-head attention layers, we choose the number of heads as 6. We apply dropout [77] on the input of Bi-LSTM layer and also the output of the softmax function in these attention layers. The dropout rate is chosen as 0.5. The batch size is 32. We use Adam update rule [45] to optimize our model. The initial learning rate is 10^{-4} and it drops to 10^{-5} at the last 1/3 epochs. We train our model 10 epochs, and every 100 steps we evaluate our method on development set and save the model with the best result. All these hyperparameters are tuned on the development set of identity dataset.

3.4.2 Baselines

MNB: Multinomial Naive Bayes classifier with unigrams and bigrams. The term features are weighted by their TF-IDF scores. Additive smoothing parameter is set as 10^{-4} via a grid search on the development set of identity dataset.
SVM: Support Vector Machine classifier with unigrams and linear kernel. The term features are weighted by their TF-IDF scores. Penalty parameter is set as 100 via a grid search on the development set of identity dataset.
CNN: Convolutional Neural Networks [44] with filter window size 3,4,5 and 100 feature

[1] https://nlp.stanford.edu/projects/glove/

maps each. Initial learning rate is 10^{-3} and drops to 10^{-4} at the last 1/3 epochs.
Bi-LSTM: Bidirectional-LSTM model with 300 hidden states in each direction. The average of output at each step is used for the final classification.
Bi-LSTM-ATT: Bidirectional-LSTM model enhanced with self-attention. We use multi-head attention with 6 heads.
fastText [42]: we set word embedding size as 300, use unigram, and train it 10 epochs with initial learning 1.0.
For methods above, we combine personal description and tweets into a whole document for each user.

3.4.3 Results

Table 3.1 Comparisons between our methods and baselines on identity classification.

		Public Figure		Identity	
		Accuracy	Macro-F1	Accuracy	Macro-F1
Baselines	MNB	81.81	82.79	82.9	75.91
	SVM	90.60	88.59	85.9	80.19
	fastText	90.93	89.01	85.7	80.01
	CNN	91.45	89.85	85.9	81.24
	Bi-LSTM	93.10	91.84	86.5	84.25
	Bi-LSTM-ATT	93.23	91.94	87.3	83.35
Ablated Models	w/o attentions	93.78	92.45	87.0	83.26
	w/o charcnn	93.47	92.23	89.0	85.39
	w/o description	92.39	90.90	86.7	81.56
	w/o tweets	91.62	89.77	84.2	78.41
	Full Model	**94.21**	**93.07**	**89.5**	**86.09**
	Full Model-transfer			**91.6**	**88.63**

In Table 3.1, we show comparison results between our model and baselines. Generally, LSTM based methods work the best among all these baseline approaches. SVM has comparable performance to these neural network based methods on the identity dataset, but falls behind on the larger public figure dataset.

Our method outperforms these baselines on both datasets, especially for the more challenging fine-grained identity classification task. Our model can successfully identify public figures with accuracy 94.21% and classify identity with accuracy 89.5%. Compared to a strong baseline Bi-LSTM-ATT, our model achieves a 2.2% increase in accuracy, which shows that our model with structured input has better classification capability.

Table 3.2 The effectiveness of different levels of attentions tested on the identity dataset.

	Accuracy	Macro-F1
Full Model	89.5	86.09
w/o word attention	88.8	84.41
w/o field attention	88.5	85.24
w/o tweet attention	88.5	84.6
w/o all attention	87.0	83.26

We further performed ablation studies to analyze the contribution of each model component, where we removed attention modules, character-level word embeddings, tweet texts, and user description one by one at a time. As shown in Table 4.2, attention modules make a great contribution to the final classification performance, especially for the more fine-grained task. We present the performance breakdown for each attention module in Table 3.2. Each level of attention effectively improve the performance of our model. Recognizing important words, tweets, and feature fields at different levels is helpful for learning classification representations. According to Table 4.2, the character-level convolutional layer is also helpful for capturing some character-level patterns.

We also examined the impact of two different text fields: personal description and tweets. Indeed, we found that what users tweeted about is more important than what they described themselves. On both datasets, users' tweets provide more discriminative power than users' personal descriptions.

3.4.4 Transfer Learning for Fine-grained Identity Classification

In reality, it is expensive to get a large-scale human labeled dataset for training a fine-grained identity classifier. However, a well-known drawback of neural network based methods is that they require a lot of data for training. Recently, learning from massive data and transferring learned knowledge to other tasks attracts a lot of attention [59, 22]. Since it is relatively easier to get a coarse-grained identity dataset to classify those public figures, we explore how to use this coarse-grained public figure dataset to help the training of fine-grained identity classifier.

Specifically, we first pretrain a binary classifier on the public figure dataset and save the best trained model on its development set. To make a fair comparison, we excluded all the users appearing in identity dataset from the public figure dataset when we built our datasets. Then we initialize the fine-grained identity classifier with this pretrained model except for the final classification layer. After such initialization step, we first train the final classification layer for 3 epochs with learning rate 0.01, and then train our full identity classification model

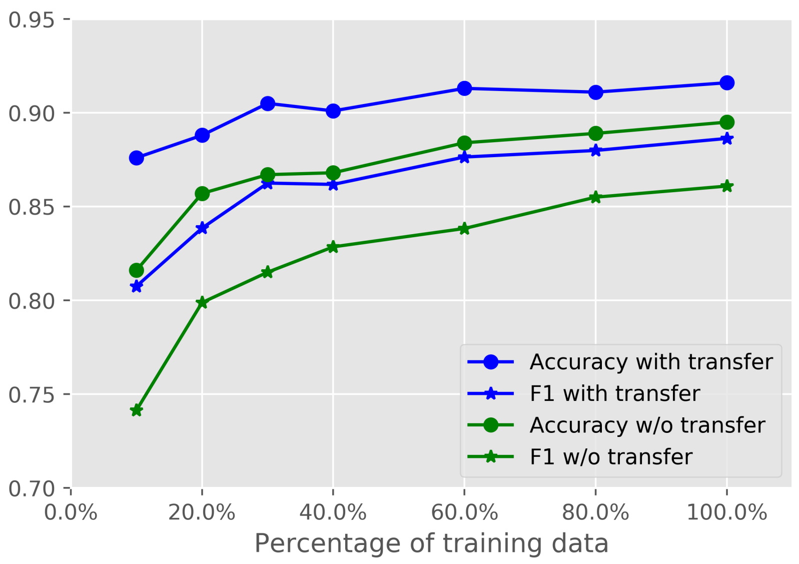

Fig. 3.2 Performance comparison between our model with transfer learning and without. We train our model on various amounts of training data.

with the same procedure as before. We observe a big performance boost when we apply such pretraining as shown in Table 4.2. The classification accuracy for the fine-grained task increases by 2.1% with transfer learning.

We further examined the performance of our model with pretraining using various amounts of training data. As shown in Figure 3.2, our pretrained model reaches a comparable performance only with 20%-30% labeled training data when compared to the model trained on full identity dataset without pretraining. Using only 20% of training data, we can get accuracy 0.888 and F1 0.839. If we increase the data size to 30% of the training data, the accuracy and F1 will increase to 0.905 and 0.863 respectively. Such pretraining makes great improvements over fine-grained identity classification especially when we lack labeled training data.

Chapter 4

Hierarchical User Location Prediction (Completed)

4.1 Introduction

Accurate estimation of user location is an important factor for many online services, such as recommendation systems [63], event detection [74], and disaster management [13]. Though internet service providers can directly obtain users' location information from some explicit metadata like IP address and GPS signal, such private information is not available for third-party contributors. With this motivation, researchers have developed location prediction systems for various platforms, such as Wikipedia [55], Facebook [6], and Twitter [30].

In the case of Twitter, due to the sparsity of geotagged tweets [27] and the unreliability of user self-declared home location in profile [35], there is a growing body of research trying to determine users' locations automatically. Various methods have been proposed for this purpose. They can be roughly divided into three categories. The first type consists of tweet text-based methods, where the word distribution is used to estimate geolocations of users [72, 86]. In the second type, methods combining metadata features such as time zone, profile description are developed to improve performance [31]. Network-based methods form the last type. Several studies have shown that incorporating friends' information is very useful for this task [53, 25]. Empirically, models enhanced with network information works better than the other two types, but they do not scale well to larger datasets [65].

In recent years, neural network based prediction methods have shown great success on this Twitter user geolocation prediction task [66, 53]. However, these neural network based methods largely ignore the hierarchical structure among locations (eg. country versus city), which have been shown very useful in previous study [48, 85]. In recent work, Huang and Carley [39] also demonstrate that country-level location prediction is much easier than city-level location prediction. It is natural to ask whether we can incorporate the hierarchical structure among locations into a neural network and use the coarse-grained location prediction to guide the fine-grained prediction.

In this paper, we present a hierarchical location prediction neural network (HLPNN) for user geolocation on Twitter. Our model combines text features, metadata features (personal description, profile location, name, user language, time zone), and network features together, and learns two classification representations for country-level and city-level predictions respectively. It first computes the country-level prediction, which is further used to guide the city-level prediction. Our model is flexible in accommodating different feature combinations, and it achieves state-of-the-art results under various feature settings.

4.2 Related Work

As a popular user profiling task, location inference has been widely studied in the literature. Though there are some potential privacy concerns, accurate user geolocation is a key factor for many important applications such as earthquake detection [24], and disaster management [13]. Because of insufficient geotagged data on Twitter [27], there is a growing interest in predicting Twitter users' locations.

Early work tried to identify users' locations by mapping their IP addresses to physical locations [10]. However, such private information is only accessible to internet service providers. There is no easy way for a third-party to find Twitter users' IP addresses. Later, various text-based location prediction systems were proposed. Bilhaut et al. [8] utilize a geographical gazetteer as an external lexicon and present a rule-based geographical references recognizer. Amitay et al. [3] extracted location-related information listed in a gazetteer from web content to identify geographical regions of webpages. However, as shown in [7], performances of gazetteer-based methods are hindered by the noisy and informal nature of tweets.

Moving beyond methods replying on external knowledge sources (eg. IP and gazetteers), many machine learning based methods have recently been applied to location prediction. Typically, researchers first represent locations as earth grids [86, 72], regions [54], or cities [31]. Then location classifiers are built to categorize users into different locations. Han et al. [30] first utilized feature selection methods to find location indicative words, then they used multinomial naive Bayes and logistic regression classifiers to find correct locations. Han et al. [31] further present a stacking based method that combines tweet text and metadata together. Along with these classification methods, some approaches also try to learn topic regions automatically by topic modeling, but these do not scale well to the magnitude of social media [38, 88].

Recently, deep neural network based methods are becoming popular for location prediction [52]. Huang and Carley [39] integrate text and user profile metadata into a single model using convolutional neural networks, and their experiments show superior performance over stacked naive Bayes classifiers. Miura et al. [53], Ebrahimi et al. [25] incorporate user network connection information into their neural models, where they use network embeddings to represent users in a social network. Rahimi et al. [67] also uses text and network feature together, but their approach is based on graph convolutional neural networks.

Similar to our method, some research has tried to predict user location hierarchically [48, 85]. Mahmud et al. [48] develop a two-level hierarchical location classifier which first predicts a coarse-grained location (country, time zone), and then predicts the city label within the corresponding coarse region. Wing and Baldridge [85] build a hierarchical tree of earth grids. The probability of a final fine-grained location can be computed recursively from the root node to the leaf node. Both methods have to train one classifier separately for each parent node, which is quite time-consuming for training deep neural network based methods. Additionally, certain coarse-grained locations may not have enough data samples to train a local neural classifier alone. Our hierarchical location prediction neural network overcomes these issues and only needs to be trained once.

4.3 Method

There are seven features we want to utilize in our model — tweet text, personal description, profile location, name, user language, time zone, and mention network. The first four features are text fields where users can write anything they want. User language and time zone are two categorical features that are selected by users in their profiles. The mention network is constructed directly from mentions in tweets.

We propose a hierarchical prediction framework that combines all seven features together for user location prediction. Our model first predicts the home country of a Twitter user, then uses the country-level prediction to guide the city-level prediction.

The overall architecture of our hierarchical location prediction model is shown in Figure 4.1. It first maps four text features into a word embedding space. A bidirectional LSTM (Bi-LSTM) neural network [37] is used to extract location-specific features from these text embedding vectors. Following Bi-LSTM, we use a word-level attention layer to generate representation vectors for these text fields. Combining all the text representations, a user language embedding, a timezone embedding, and a network embedding, we apply several

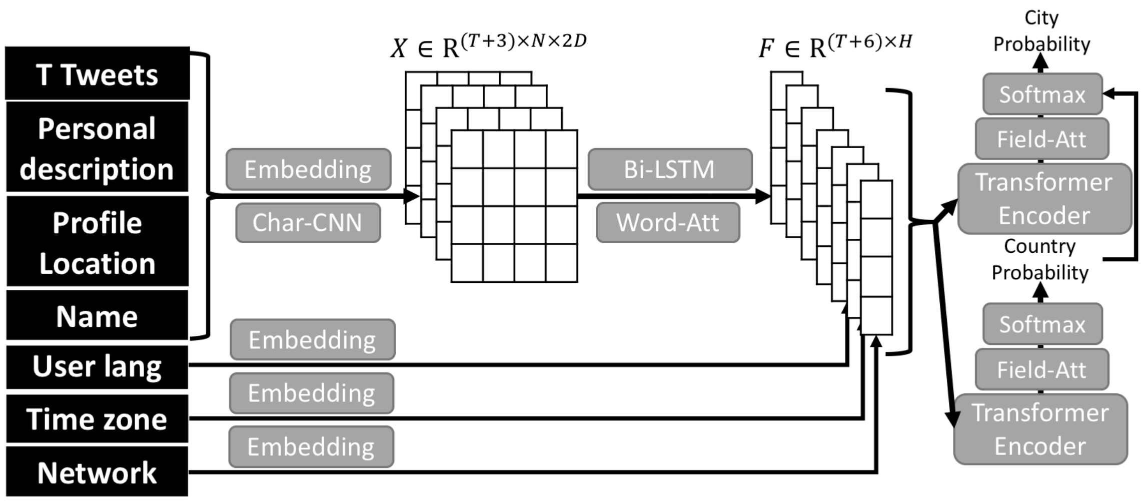

Fig. 4.1 The architecture of our hierarchical location prediction neural network.

layers of transformer encoders [83] to learn the correlation among all the feature fields. The probability for each country is computed after a field-level attention layer. Finally, we use the country probability as a constraint for the city-level location prediction. We elaborate details of our model in following sections.

4.3.1 Word Embedding

Assume one user has T tweets, there are $T+3$ text fields for this user including personal description, profile location, and name. We first map each word in these $T+3$ text fields into a low dimensional embedding space. The embedding vector for word w is computed as $x_w = [E(w), CNN_c(w)]$, where $[,]$ denotes vector concatenation. $E(w)$ is the word-level embedding retrieved directly from an Embedding matrix $E \in R^{V\times D}$ by a lookup operation, where V is the vocabulary size, and D is the word-level embedding dimension. $CNN_c(w)$ is a character-level word embedding that is generated from a character-level convolutional layer. Using character-level word embeddings is helpful for dealing with out-of-vocabulary tokens and overcoming the noisy nature of tweet text.

The character-level word embedding generation process is as follows. For a character c_i in the word $w = (c_1, ..., c_k)$, we map it into a character embedding space and get a vector $v_{c_i} \in R^d$. In the convolutional layer, each filter $u \in R^{l_c\times d}$ generates a feature vector $\theta = [\theta_1, \theta_2, ..., \theta_{k-l_c+1}] \in R^{k-l_c+1}$, where $\theta_i = relu(u \circ v_{c_i:c_{i+l_c-1}} + b)$. b is a bias term, and "$\circ$" denotes element-wise inner product between u and character window $v_{c_i:c_{i+l_c-1}} \in R^{l_c\times d}$. After this convolutional operation, we use a max-pooling operation to select the most representative feature $\hat{\theta} = max(\theta)$. With D such filters, we get the character-level word embedding $CNN_c(w) \in R^D$.

4.3.2 Text Representation

After the word embedding layer, every word in these $T+3$ texts are transformed into a $2D$ dimension vector. Given a text with word sequence $(w_1, ..., w_N)$, we get a word embedding matrix $X \in R^{N \times 2D}$ from the embedding layer. We then apply a Bi-LSTM neural network to extract high-level semantic representations from text embedding matrices.

At every time step i, a forward LSTM takes the word embedding X_i of word w_i and previous state $\overrightarrow{h_{i-1}}$ as inputs, and generates the current hidden state $\overrightarrow{h}_i$. A backward LSTM reads the text from w_N to w_1 and generates another state sequence. The hidden state $h_i \in R^{2D}$ for word w_i is the concatenation of $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$. Concatenating all the hidden states, we get a semantic matrix $H \in R^{N \times 2D}$

$$\begin{aligned}
\overrightarrow{h_i} &= \overrightarrow{LSTM}(X_i, \overrightarrow{h_{i-1}}) \\
\overleftarrow{h_i} &= \overleftarrow{LSTM}(X_i, \overleftarrow{h_{i+1}}) \\
h_i &= [\overrightarrow{h_i}, \overleftarrow{h_i}]
\end{aligned} \tag{4.1}$$

Because not all words in a text contribute equally towards location prediction, we further use a multi-head attention layer [83] to generate a representation vector $f \in R^{2D}$ for each text. There are h attention heads that allow the model to attend to important information from different representation subspaces. Each head computes a text representation as a weighted average of these word hidden states. The computation steps in a multi-head attention layer are as follows.

$$\begin{aligned}
f &= MultiHead(q, H) = [head_1, ..., head_h]W^O \\
head_i(q, H) &= softmax(\frac{qW_i^Q \cdot (HW_i^K)^T}{\sqrt{d_k}})HW_i^V
\end{aligned}$$

where $q \in R^{2d}$ is an attention context vector learned during training, $W_i^Q, W_i^K, W_i^V \in R^{2D \times d_k}$, and $W^O \in R^{2D \times 2D}$ are projection parameters, $d_k = 2D/h$. An attention head $head_i$ first projects the attention context q and the semantic matrix H into query and key subspaces by W_i^Q, W_i^K respectively. The matrix product between query qW_i^Q and key HW_i^K after softmax normalization is an attention weight that indicates important words among the projected value vectors HW_i^V. Concatenating h heads together, we get one representation vector $f \in R^{2D}$ after projection by W^O for each text field.

4.3.3 Feature Fusion

For two categorical features, we assign an embedding vector with dimension $2D$ for each time zone and language. These embedding vectors are learned during training. We pretrain network embeddings for users involved in the mention network using LINE [81]. Network embeddings are fixed during training. We get a feature matrix $F \in R^{(T+6)\times 2D}$ by concatenating text representations of $T+3$ text fields, two embedding vectors of categorical features, and one network embedding vector.

We further use several layers of transformer encoders [83] to learn the correlation between different feature fields. Each layer consists of a multi-head self-attention network and a feed-forward network (FFN). One transformer encoder layer first uses input feature to attend important information in the feature itself by a multi-head attention sub-layer. Then a linear transformation sub-layer FFN is applied to each position identically. Same as [83], we employ residual connection [34] and layer normalization [5] around each of the two sub-layers. The output F_1 of the first transformer encoder layer is generated as follows.

$$F' = LayerNorm(MultiHead(F,F) + F)$$
$$F_1 = LayerNorm(FFN(F') + F')$$

where $FFN(F') = max(0, F'W_1 + b_1)W_2 + b2$, $W_1 \in R^{2D\times D_{ff}}$, and $W_2 \in R^{D_{ff}\times 2D}$.

Since there is no position information in the transformer encoder layer, our model cannot distinguish between different types of features, eg. tweet text and personal description. To overcome this issue, we add feature type embeddings to the input representations F. There are seven features in total. Each of them has a learned feature type embedding with dimension $2D$ so that one feature type embedding and the representation of the corresponding feature can be summed.

Because the input and the output of transformer encoder have the same dimension, we stack L layers of transformer encoders to learn representations for country-level prediction and city-level prediction respectively. These two sets of encoders share the same input F, but generate different representations F^L_{co} and F^L_{ci} for country and city predictions.

The final classification features for country-level and city-level location predictions are the row-wise weighted average of F_{co} and F_{ci}. Similar to the word-level attention, we use a field-level multi-head attention layer to select important features from $T+6$ vectors and fuse them into a single vector.

$$F_{co} = MultiHead(q_{co}, F^L_{co})$$
$$F_{ci} = MultiHead(q_{ci}, F^L_{ci})$$

where $q_{co}, q_{ci} \in R^{2D}$ are two attention context vectors.

4.3.4 Hierarchical Location Prediction

The final probability for each country is computed by a softmax function

$$P_{co} = softmax(W_{co}F_{co} + b_{co})$$

where $W_{co} \in R^{M_{co} \times 2D}$ is a linear projection parameter, $b_{co} \in R^{M_{co}}$ is a bias term, and M_{co} is the number of countries.

After we get the probability for each country, we further use it to constrain the city-level prediction

$$P_{ci} = softmax(W_{ci}F_{ci} + b_{ci} + \lambda P_{co}Bias)$$

where $W_{ci} \in R^{M_{ci} \times 2D}$ is a linear projection parameter, $b_{ci} \in R^{M_{ci}}$ is a bias term, and M_{ci} is the number of cities. $Bias \in R^{M_{co} \times M_{ci}}$ is the country-city correlation matrix. If city j belongs to country i, then $Bias_{ij}$ is 0, otherwise -1. λ is a penalty term learned during training. The larger of λ, the stronger of the country constraint. In practise, we also experimented with letting the model learn the country-city correlation matrix during training, which yields similar performance.

We minimize the sum of two cross-entropy losses for country-level prediction and city-level prediction.

$$loss = -(Y_{ci} \cdot logP_{ci} + \alpha Y_{co} \cdot logP_{co})$$

where Y_{ci} and Y_{co} are one-hot encodings of city and country labels. α is the weight to control the importance of country-level supervision signal. Since a large α would potentially interfere with the training process of city-level prediction, we just set it as 1 in our experiments. Tuning this parameter on each dataset may further improve the performance.

4.4 Experiment Settings

4.4.1 Text Preprocessing & Network Construction

For all the text fields, we first convert them into lower case, then use a tweet-specific tokenizer from NLTK[1] to tokenize them. To keep a reasonable vocabulary size, we only keep tokens with frequencies greater than 10 times in our word vocabulary. Our character vocabulary includes characters that appear more than 5 times in the training corpus.

We construct user networks from mentions in tweets. For WNUT, we keep users satisfying one of the following conditions in the mention network: 1. users in the original dataset 2.

[1]https://www.nltk.org/api/nltk.tokenize.html

users who are mentioned by two different users in the dataset. For Twitter-US and Twitter-World, following previous work [67], a uni-directional edge is set if two users in our dataset directly mentioned each other, or they co-mentioned another user. We also remove celebrities who are mentioned by more than 10 different users in the dataset.

4.4.2 Evaluation Metrics

We evaluate our method using four commonly used metrics listed below.
Accuracy: The percentage of correctly predicted home cities.
Acc@161: The percentage of predicted cities which are within a 161 km (100 miles) radius of true locations to capture near-misses.
Median: The median distance measured in kilometer from the predicted city to the true location coordinates.
Mean: The mean value of error distances in predictions.

4.4.3 Hyperparameter Settings

In our experiments, we initialize word embeddings with released 300-dimensional Glove vectors [57]. For words not appearing in Glove vocabulary, we randomly initialize them from a uniform distribution U(-0.25, 0.25). We choose the character embedding dimension as 50. The character embeddings are randomly initialized from a uniform distribution U(-1.0,1.0), as well as the timezone embeddings and language embeddings. These embeddings are all learned during training.

Network embeddings are trained using LINE [81] with parameters of dimension 600, initial learning rate 0.025, order 2, negative sample size 5, and training sample size 10000M. Network embeddings are fixed during training. For users not appearing in the mention network, we set their network embedding vectors as 0.

A brief summary of hyperparameter settings of our model is shown in Table 4.1. The initial learning rate is 10^{-4}. If the validation accuracy on the development set does not increase after an epoch, we decrease the learning rate to 10^{-5} and train the model for additional 3 epochs. Empirically, training terminates within 10 epochs. Penalty λ is initialized as 1.0 and is adapted during training. We apply dropout on the input of Bi-LSTM layer and the output of two sub-layers in transformer encoders with dropout rate 0.3 and 0.1 respectively. We use the Adam update rule [45] to optimize our model. Gradients are clipped between -1 and 1. The maximum numbers of tweets per user for training and evaluating on Twitter-US are 100 and 200 respectively. We only tuned our model, learning rate, and dropout rate on the development set of WNUT.

Table 4.1 Hyperparameter settings of our hierarchical user location prediction model.

	Twitter-US	Twitter-World	WNUT
Batch size	32	64	64
Initial learning rate	10^{-4}	10^{-4}	10^{-4}
D: Word embedding dimension	300	300	300
d: Char. embedding dimension	50	50	50
l_c: filter sizes in Char. CNN	3,4,5	3,4,5	3,4,5
Filter number for each size	100	100	100
h: number of heads	10	10	10
L: layers of transformer encoder	3	3	3
λ: initial penalty term	1	1	1
α: weight for country supervision	1	1	1
D_{ff}: inner dimension of FFN	2400	2400	2400
Max number of tweets per user	100	50	20

4.4.4 Baseline Comparisons

In our experiments, we evaluate our model under four different feature settings: Text, Text+Meta, Text+Network, Text+Meta+Network. HLPNN-Text is our model only using tweet text as input. HLPNN-Meta is the model that combines text and metadata (description, location, name, user language, time zone). HLPNN-Net is the model that combines text and mention network. HLPNN is our full model that uses text, metadata, and mention network for Twitter user geolocation.

We present comparisons between our model and previous work in Table 4.2. As shown in the table, our model outperforms these baselines across three datasets under various feature settings.

Only using text feature from tweets, our model HLPNN-Text works the best among all these text-based location prediction systems and wins by a large margin. It not only improves prediction accuracy but also greatly reduces mean error distance. Compared with a strong neural model equipped with local dialects [66], it increases Acc@161 by an absolute value 4% and reduces mean error distance by about 400 kilometers on the challenging

Twitter-World dataset, without using any external knowledge. Its mean error distance on Twitter-World is even comparable to some methods using network feature [23].

With text and metadata, HLPNN-Meta correctly predicts locations of 57.2% users in WNUT dataset, which is even better than these location prediction systems that use text, metadata, and network. Note that Miura et al. [53] used 279K users added with metadata in their experiments on Twitter-US, while we use all 449K users for training and evaluation, and only 53% of them have metadata, which makes it difficult to make a fair comparison.

Adding network feature further improves our model's performances. It achieves state-of-the-art results combining all features on these three datasets. Even though unifying network information is not the focus of this paper, our model still outperforms or has comparable results to some well-designed network-based location prediction systems like [67]. On Twitter-US dataset, our model variant HLPNN-Net achieves a 4.6% increase in Acc@161 against previous state-of-the-art methods [23] and [67]. The prediction accuracy of HLPNN-Net on WNUT dataset is similar to [53], but with a noticeable lower mean error distance.

Table 4.2 Comparisons between our method and baselines. We report results under four different feature settings: Text, Text+Metadata, Text+Network, Text+Metadata+Network. "-" signifies that no results were published for the given dataset, "*" denotes that results are cited from [66]. Note that Miura et al. [53] only used 279K users added with metadata in their experiments of Twitter-US.

	Twitter-US			Twitter-World			WNUT			
	Acc@161↑	Median↓	Mean↓	Acc@161↑	Median↓	Mean↓	Accuracy↑	Acc@161↑	Median↓	Mean↓
Text										
Wing and Baldridge [85]	49.2	170.5	703.6	32.7	490.0	1714.6	-	-	-	-
Rahimi et al. [68]*	50	159	686	32	530	1724	-	-	-	-
Miura et al. [53]-TEXT	55.6	110.5	585.1	-	-	-	35.4	50.3	155.8	1592.6
Rahimi et al. [66]	55	91	581	36	373	1417	-	-	-	-
HLPNN-Text	**57.1**	**89.92**	**516.6**	**40.1**	**299.1**	**1048.1**	**37.3**	**52.9**	**109.3**	**1289.4**
Text+Meta										
Miura et al. [53]-META	**67.2**	**46.8**	**356.3**	-	-	-	54.7	70.2	0	825.8
HLPNN-Meta	61.1	64.3	454.8	**56.4**	**86.2**	**762.1**	**57.2**	**73.1**	**0**	**572.5**
Text+Net										
Rahimi et al. [65]*	60	78	529	53	111	1403	-	-	-	-
Rahimi et al. [66]	61	77	515	53	104	1280	-	-	-	-
Miura et al. [53]-UNET	61.5	65	481.5	-	-	-	**38.1**	**53.3**	**99.9**	1498.6
Do et al. [23]	66.2	45	433	53.3	118	1044	-	-	-	-
Rahimi et al. [67]-MLP-TXT+NET	66	56	420	58	**53**	1030	-	-	-	-
Rahimi et al. [67]-GCN	62	71	485	54	108	1130	-	-	-	-
HLPNN-Net	**70.8**	**31.6**	**361.5**	**58.9**	59.9	**827.6**	37.8	**53.3**	105.26	**1297.7**
Text+Meta+Net										
Miura et al. [52]	-	-	-	-	-	-	47.6	-	16.1	1122.3
Jayasinghe et al. [40]	-	-	-	-	-	-	52.6	-	21.7	1928.8
Miura et al. [53]	70.1	41.9	335.7	-	-	-	56.4	71.9	**0**	780.5
HLPNN	**72.7**	**28.2**	**323.1**	**68.4**	**6.20**	**610.0**	**57.6**	**73.4**	**0**	**538.8**

Chapter 5

Graph-level Attributes Prediction

5.1 Introduction

In previous chapters, I have shown various methods for user attributes prediction. Most of them only use user's local features without any network information. The only exception is the hierarchical location prediction neural network where we use network embedding to provide network structure information to the model. However, a drawback of these type of embedding methods is they cannot handle newly incoming users. Test users should be available during training so that we can build a network to train embeddings like LINE [81]. Besides, such a model does not utilize features from neighbourhood friends, which may contain useful information for prediction.

In this chapter, I will explore using graph neural networks (GNN) to incorporate graph connections with inductive training. Graph neural networks are deep learning-based methods that operate on graphs. At each layer, GNNs aggregate information from neighbourhoods and generate hidden states for each node. Because GNNs do not require a fixed graph, we can easily apply them to new graphs on the fly, which is suitable for the inductive setting.

Most of the previous work either focus on classification only using network information [58], or combining network information with processed features [29]. However, in this work, users in a network are associated with raw text features, which cannot be utilized directly in a graph neural network. In this chapter, I will explore how to combine my previous user-level learning architecture with graph neural networks.

There are two main components in my proposed method. One is user-level feature extractor, which is just my previous user-level learning module. Another is a graph neural network that propagate features from users' neighbourhoods. Generally, the learning process can be written as

$$x_i = F(tweets_i, description_i) \quad (5.1)$$
$$h_i = GNN(x_i, x_{[nei_i]}) \quad (5.2)$$
$$y_i = MLP(x_i, h_i) \quad (5.3)$$

where F represents a user-level feature learning function, $x_{[nei_i]}$ are the features of neighbour users of user i. $MLP(\cdot)$ is a multilayer perception network.

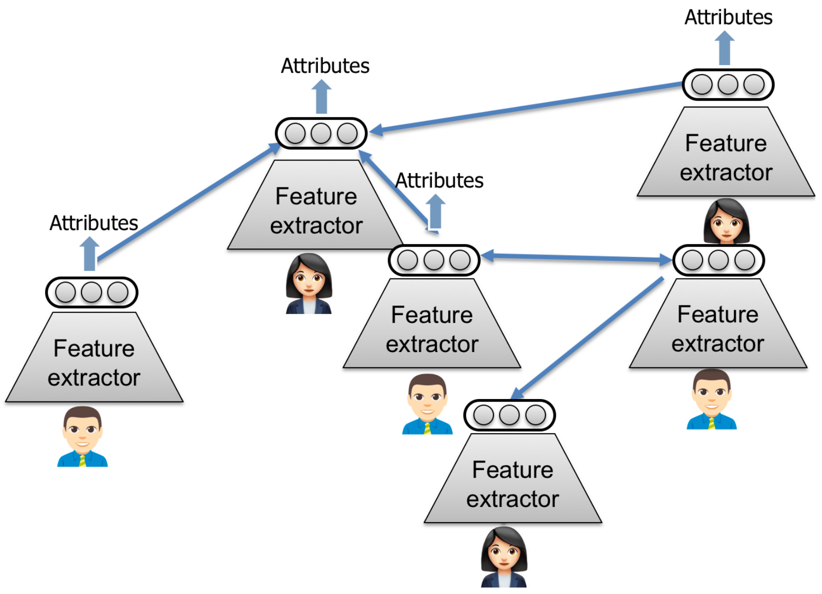

Fig. 5.1 An illustration of the graph-level attributes learning process.

An illustration of the learning process is shown in Fig. 5.1, where users' local features are first computed by a feature extractor function, then these features are propagated on the graph via a graph neural network.

In addition, such feature propagation procedure may not capture global characteristics of nodes in a social media. For example, users with millions of followers may not share much common properties with their followers. Incorporating classic network metrics like degree centrality, betweenness centrality may be a possible way to alleviate this issue.

5.2 Related Work

The concept of graph neural networks was first introduced in [75]. Given a graph with adjacent matrix $A \in R^{N\times N}$, the representation h_i for a node i is updated as follows:

$$h_i = f(x_i, x_{e[i]}, h_{n[i]}, x_{n_i}) \tag{5.4}$$

$$o_i = g(h_i, x_i) \tag{5.5}$$

where x_i, $x_{e[i]}$, $h_{n[i]}$, x_{n_i} are features of node i, features of its edges, the states, and the features of its neighbourhood. Function f is a contraction map and are shared across layers. The final representation h_i for node i is a fixed point of f. Combining h_i and x_i, it outputs label o_i for node i. In general, this process can be viewed as features propagation from neighbourhood.

There are several GNN variants exist in the literature. Kipf and Welling introduce a simplified spectral approach called graph convolutional neural networks (GCN) [46]. They use one-step neighbourhood to update the state of a central node as:

$$H^{l+1} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^l\Theta^l \tag{5.6}$$

where $\hat{A} = A + I$, $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$, $H^l \in R^{N\times C_l}$ is the stacked states for all nodes at layer l, H^0 is stacked node features X, $\Theta^l \in R^{C_l\times C_{l+1}}$ is a filter parameter. C_l is the dimension of hidden states at layer l.

Another popular variant is the graph attention network (GAT) [84]. Again, a node's state is updated by aggregating its neighbourhood's states. GAT adopted one widely used multi-head attention method in natural language processing to learn important nodes in neighbourhood [83]. Using K attention heads, GAT update states by

$$h_i^{l+1} =_{k=1}^{K} \sigma\Big(\sum_{j\in n[i]} \alpha_{ij}^{lk} W^{lk} h_j^l\Big) \tag{5.7}$$

$$\alpha_{ij}^{lk} = \frac{exp(LeakyReLU(a_k^{l\,T}[W^{lk}h_i^l||W^{lk}h_j^l]))}{\sum_{u\in n[i]} exp(LeakyReLU(a_k^{l\,T}[W^{lk}h_i^l||W^{lk}h_u^l]))} \tag{5.8}$$

where represents vector concatenation, α_{ij}^{lk} is the attention coefficient of node i to its neighbour j in attention head k at layer l. $W^{lk} \in R^{\frac{C_{l+1}}{K}\times C_l}$ is a linear transformation for input states. σ denotes a sigmoid function. $a_k^l \in R^{\frac{2C_{l+1}}{K}}$ is an attention context vector learned during training.

In practise, researchers have observed that deeper GNN models could not improve performance and even perform worse, which is partially due to more layers would also propagate noisy information from expanded neighborhood [46]. A common option is using a residual connection as shown in Eq. 5.9, which adds states from lower layer directly to higher layer and avoids the local features getting vanished in higher layers.

$$H^{l+1} = GNN(H^l, A; \Theta^l) + H^l \tag{5.9}$$

where Θ_l is parameter of GNN at layer l.

Chapter 6

Attributes Prediction via Multi-task Learning

6.1 Introduction

In this last chapter, the goal is to predict users' attributes via multi-task learning. In our previous setting, only one target attribute is learned during training, while in this chapter I would like to predict these attributes all at once. An obvious advantage of such multi-task learning is predicting all attributes at once can greatly reduce the prediction time consumption. Secondly, since several tasks may correlate with each other, learning one attribute may benefit another, eg. location versus political orientation.

6.1.1 Model 1

A simple idea to achieve such multi-task learning is that we can share inputs and hidden representations for various tasks. Then different output layers are applied to get various attributes prediction results. However, the importance of different input features may be different for various attributes. Possibly we can use attention mechanism to select the best feature combinations for various tasks as shown in Fig. 6.1.

6.1.2 Model 2

Another possible direction is that we can learn universal representations with unsupervised learning on graph [29]. So that such representations can be used for various tasks.

Following previous work [81, 29], in the unsupervised setting, we can learn user representations by network modeling. Specifically, given a user v_i with raw features

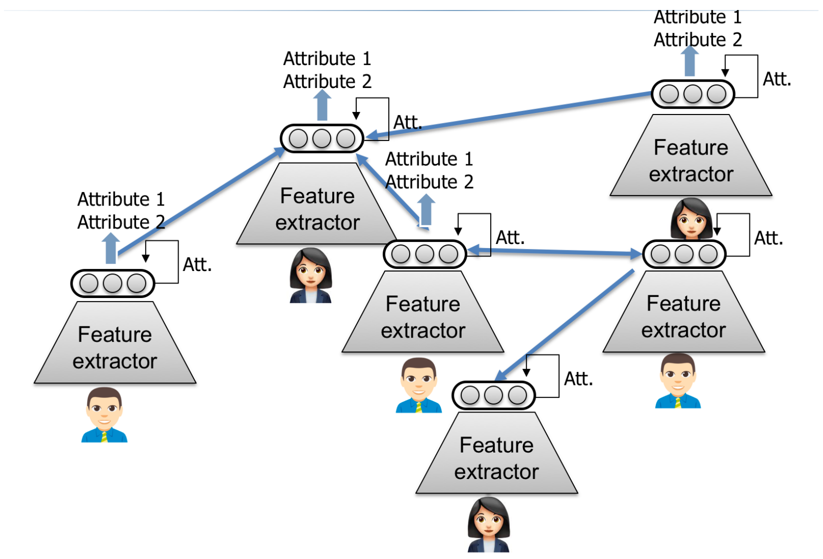

Fig. 6.1 An illustration of the attention enhanced multi-task learning.

$tweets_i, description_i$, the goal is to optimize the probability of observing a context user v_j:

$$x_i = F(tweets_i, description_i) \tag{6.1}$$

$$h_i = GNN(x_i, x_{[nei_i]}) \tag{6.2}$$

$$p(v_j|v_i) = \frac{exp(h_j^T h_i)}{\sum_{k=1}^{N} exp(h_k^T h_i)} \tag{6.3}$$

where context user v_j is generated by a random walk starting from user v_i.

Optimizing the conditional probability in Eq. 6.3 for all context node pairs implies that nodes in proximity should have similar hidden states. In practise, optimizing Eq. 6.3 is computationally expensive, since there are N nodes involved in the denominator. So we use negative sampling [51] to approximate it and the objective becomes:

$$log\sigma(h_j^T h_i) + \sum_{k=1}^{K} E_{v_k \sim P_N(v)}[log\sigma(-h_k^T h_i)] \tag{6.4}$$

The task turns into distinguishing the context node v_j from K randomly sampled negative nodes.

After we get universal representations of users, we can directly use them to train a simple linear classifier for attributes prediction.

References

[1] Achrekar, H., Gandhe, A., Lazarus, R., Yu, S.-H., and Liu, B. (2011). Predicting flu trends using twitter data. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 702–707. IEEE.

[2] Al Zamal, F., Liu, W., and Ruths, D. (2012). Homophily and latent attribute inference: Inferring latent attributes of twitter users from neighbors. In *Sixth International AAAI Conference on Weblogs and Social Media*.

[3] Amitay, E., Har'El, N., Sivan, R., and Soffer, A. (2004). Web-a-where: geotagging web content. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 273–280. ACM.

[4] Ashforth, B. E. and Mael, F. (1989). Social identity theory and the organization. *Academy of management review*, 14(1):20–39.

[5] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[6] Backstrom, L., Sun, E., and Marlow, C. (2010). Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th international conference on World wide web*, pages 61–70. ACM.

[7] Berggren, M., Karlgren, J., Östling, R., and Parkvall, M. (2016). Inferring the location of authors from words in their texts. *arXiv preprint arXiv:1612.06671*.

[8] Bilhaut, F., Charnois, T., Enjalbert, P., and Mathet, Y. (2003). Geographic reference analysis for geographic document querying. In *Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references-Volume 1*, pages 55–62. Association for Computational Linguistics.

[9] Burger, J. D., Henderson, J., Kim, G., and Zarrella, G. (2011). Discriminating gender on twitter. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1301–1309. Association for Computational Linguistics.

[10] Buyukokkten, O., Cho, J., Garcia-Molina, H., Gravano, L., and Shivakumar, N. (1999). Exploiting geographical location information of web pages.

[11] Calhoun, C. J. (1994). Social theory and the politics of identity.

[12] Callero, P. L. (1985). Role-identity salience. *Social psychology quarterly*, pages 203–215.

[13] Carley, K. M., Malik, M., Landwehr, P. M., Pfeffer, J., and Kowalchuck, M. (2016). Crowd sourcing disaster management: the complex nature of twitter usage in padang indonesia. *Safety science*, 90:48–61.

[14] Cha, M., Haddadi, H., Benevenuto, F., Gummadi, P. K., et al. (2010). Measuring user influence in twitter: The million follower fallacy. *Icwsm*, 10(10-17):30.

[15] Chatfield, A. T. and Brajawidagda, U. (2013). Twitter early tsunami warning system: A case study in indonesia's natural disaster management. In *2013 46th Hawaii International Conference on System Sciences*, pages 2050–2060. IEEE.

[16] Cheng, Z., Caverlee, J., and Lee, K. (2010). You are where you tweet: a content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 759–768. ACM.

[17] Chu, Z., Gianvecchio, S., Wang, H., and Jajodia, S. (2010). Who is tweeting on twitter: human, bot, or cyborg? In *Proceedings of the 26th annual computer security applications conference*, pages 21–30. ACM.

[18] Chu, Z., Gianvecchio, S., Wang, H., and Jajodia, S. (2012). Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing*, 9(6):811–824.

[19] Colleoni, E., Rozza, A., and Arvidsson, A. (2014). Echo chamber or public sphere? predicting political orientation and measuring political homophily in twitter using big data. *Journal of Communication*, 64(2):317–332.

[20] Conneau, A., Schwenk, H., Barrault, L., and Lecun, Y. (2016). Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

[21] Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org.

[22] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[23] Do, T. H., Nguyen, D. M., Tsiligianni, E., Cornelis, B., and Deligiannis, N. (2017). Multiview deep learning for predicting twitter users' location. *arXiv preprint arXiv:1712.08091*.

[24] Earle, P. S., Bowden, D. C., and Guy, M. (2012). Twitter earthquake detection: earthquake monitoring in a social world. *Annals of Geophysics*, 54(6).

[25] Ebrahimi, M., ShafieiBavani, E., Wong, R., and Chen, F. (2018). A unified neural network model for geolocating twitter users. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 42–53.

[26] Godoy, D. and Amandi, A. (2005). User profiling for web page filtering. *IEEE Internet computing*, 9(4):56–64.

[27] Graham, M., Hale, S. A., and Gaffney, D. (2014). Where in the world are you? geolocation and language identification in twitter. *The Professional Geographer*, 66(4):568–578.

[28] Hale, S., Gaffney, D., and Graham, M. (2012). Where in the world are you? geolocation and language identification in twitter. *Proceedings of ICWSM*, 12:518–521.

[29] Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

[30] Han, B., Cook, P., and Baldwin, T. (2012). Geolocation prediction in social media data by finding location indicative words. *Proceedings of COLING 2012*, pages 1045–1062.

[31] Han, B., Cook, P., and Baldwin, T. (2013). A stacking-based approach to twitter user geolocation prediction. In *ACL (Conference System Demonstrations)*, pages 7–12.

[32] Han, B., Cook, P., and Baldwin, T. (2014). Text-based twitter user geolocation prediction. *Journal of Artificial Intelligence Research*, 49:451–500.

[33] Han, B., Rahimi, A., Derczynski, L., and Baldwin, T. (2016). Twitter geolocation prediction shared task of the 2016 workshop on noisy user-generated text. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 213–217.

[34] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[35] Hecht, B., Hong, L., Suh, B., and Chi, E. H. (2011). Tweets from justin bieber's heart: the dynamics of the location field in user profiles. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 237–246. ACM.

[36] Hentschel, M., Alonso, O., Counts, S., and Kandylas, V. (2014). Finding users we trust: Scaling up verified twitter users using their communication patterns. In *ICWSM*.

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

[38] Hong, L., Ahmed, A., Gurumurthy, S., Smola, A. J., and Tsioutsiouliklis, K. (2012). Discovering geographical topics in the twitter stream. In *Proceedings of the 21st international conference on World Wide Web*, pages 769–778. ACM.

[39] Huang, B. and Carley, K. M. (2017). On predicting geolocation of tweets using convolutional neural networks. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, pages 281–291. Springer.

[40] Jayasinghe, G., Jin, B., Mchugh, J., Robinson, B., and Wan, S. (2016). Csiro data61 at the wnut geo shared task. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 218–226.

[41] Jenkins, R. (2014). *Social identity*. Routledge.

[42] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

[43] Jurgens, D. (2013). That's what friends are for: Inferring location in online social media platforms based on social relationships. *ICWSM*, 13:273–282.

[44] Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

[45] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[46] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[47] Landwehr, P. M. and Carley, K. M. (2014). Social media in disaster relief. In *Data mining and knowledge discovery for big data*, pages 225–257. Springer.

[48] Mahmud, J., Nichols, J., and Drews, C. (2012). Where is this tweet from? inferring home locations of twitter users. In *Sixth International AAAI Conference on Weblogs and Social Media*.

[49] McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444.

[50] Middleton, S. E., Shadbolt, N. R., and De Roure, D. C. (2004). Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88.

[51] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[52] Miura, Y., Taniguchi, M., Taniguchi, T., and Ohkuma, T. (2016). A simple scalable neural networks based model for geolocation prediction in twitter. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 235–239.

[53] Miura, Y., Taniguchi, M., Taniguchi, T., and Ohkuma, T. (2017). Unifying text, metadata, and user network representations with a neural network for geolocation prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1260–1272.

[54] Miyazaki, T., Rahimi, A., Cohn, T., and Baldwin, T. (2018). Twitter geolocation using knowledge-based methods. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 7–16.

[55] Overell, S. E. (2009). *Geographic information retrieval: Classification, disambiguation and modelling*. PhD thesis, Citeseer.

[56] Pennacchiotti, M. and Popescu, A.-M. (2011). A machine learning approach to twitter user classification. In *Fifth International AAAI Conference on Weblogs and Social Media*.

[57] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[58] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

[59] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

[60] Preoţiuc-Pietro, D., Liu, Y., Hopkins, D., and Ungar, L. (2017). Beyond binary labels: political ideology prediction of twitter users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 729–740.

[61] Priante, A., Hiemstra, D., van den Broek, T., Saeed, A., Ehrenhard, M., and Need, A. (2016). # whoami in 160 characters? classifying social identities based on twitter profile descriptions. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 55–65.

[62] Qian, Y., Tang, J., Yang, Z., Huang, B., Wei, W., and Carley, K. M. (2017). A probabilistic framework for location inference from social media. *arXiv preprint arXiv:1702.07281*.

[63] Quercia, D., Lathia, N., Calabrese, F., Di Lorenzo, G., and Crowcroft, J. (2010). Recommending social events from mobile phone location data. In *2010 IEEE international conference on data mining*, pages 971–976. IEEE.

[64] Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

[65] Rahimi, A., Cohn, T., and Baldwin, T. (2015a). Twitter user geolocation using a unified text and network prediction model. *arXiv preprint arXiv:1506.08259*.

[66] Rahimi, A., Cohn, T., and Baldwin, T. (2017). A neural model for user geolocation and lexical dialectology. *arXiv preprint arXiv:1704.04008*.

[67] Rahimi, A., Cohn, T., and Baldwin, T. (2018). Semi-supervised user geolocation via graph convolutional networks. *arXiv preprint arXiv:1804.08049*.

[68] Rahimi, A., Vu, D., Cohn, T., and Baldwin, T. (2015b). Exploiting text and network context for geolocation of social media users. *arXiv preprint arXiv:1506.04803*.

[69] Rao, D., Yarowsky, D., Shreevats, A., and Gupta, M. (2010). Classifying latent user attributes in twitter. In *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, pages 37–44. ACM.

[70] Recasens, M., Hovy, E., and Martí, M. A. (2011). Identity, non-identity, and near-identity: Addressing the complexity of coreference. *Lingua*, 121(6):1138–1152.

[71] Robinson, L. (2007). The cyberself: the self-ing project goes online, symbolic interaction in the digital age. *New Media & Society*, 9(1):93–110.

[72] Roller, S., Speriosu, M., Rallapalli, S., Wing, B., and Baldridge, J. (2012). Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1500–1510. Association for Computational Linguistics.

[73] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[74] Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM.

[75] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

[76] Shmueli-Scheuer, M., Roitman, H., Carmel, D., Mass, Y., and Konopnicki, D. (2010). Extracting user profiles from large scale data. In *Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud*, page 4. ACM.

[77] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[78] Stryker, S. and Burke, P. J. (2000). The past, present, and future of an identity theory. *Social psychology quarterly*, pages 284–297.

[79] Tajfel, H. (1974). Social identity and intergroup behaviour. *Information (International Social Science Council)*, 13(2):65–93.

[80] Tajfel, H. (1982). *Social identity and intergroup relations*. Cambridge University Press.

[81] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

[82] Tsou, M.-H., Yang, J.-A., Lusher, D., Han, S., Spitzberg, B., Gawron, J. M., Gupta, D., and An, L. (2013). Mapping social activities and concepts with social media (twitter) and web search engines (yahoo and bing): a case study in 2012 us presidential election. *Cartography and Geographic Information Science*, 40(4):337–348.

[83] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[84] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

[85] Wing, B. and Baldridge, J. (2014). Hierarchical discriminative classification for text-based geolocation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 336–348.

[86] Wing, B. P. and Baldridge, J. (2011). Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 955–964. Association for Computational Linguistics.

[87] Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.

[88] Zhang, Y., Wei, W., Huang, B., Carley, K. M., and Zhang, Y. (2017). Rate: Overcoming noise and sparsity of textual features in real-time location estimation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2423–2426. ACM.