Simple and Knowledge-intensive Generative Model for Named Entity Recognition

Chun-Kai Wang Bo-June (Paul) Hsu Ming-Wei Chang Emre Kıcıman
Microsoft Research
{chunkaiw,paulhsu,minchang,emrek}@microsoft.com

ABSTRACT

Almost all of the existing work on Named Entity Recognition (NER) consists of the following pipeline stages – part-of-speech tagging, segmentation, and named entity type classification. The requirement of hand-labeled training data on these stages makes it very expensive to extend to different domains and entity classes. Even with a large amount of hand-labeled data, existing techniques for NER on informal text, such as social media, perform poorly due to a lack of reliable capitalization, irregular sentence structure and a wide range of vocabulary.

In this paper, we address the lack of hand-labeled training data by taking advantage of weak super vision signals. We present our approach in two parts. First, we propose a novel generative model that combines the ideas from Hidden Markov Model (HMM) and n-gram language models into what we call an N-gram Language Markov Model (NLMM). Second, we utilize large-scale weak supervision signals from sources such as Wikipedia titles and the corresponding click counts to estimate parameters in NLMM. Our model is simple and can be implemented without the use of Expectation Maximization or other expensive iterative training techniques. Even with this simple model, our approach to NER on informal text outperforms existing systems trained on formal English and matches state-of-the-art NER systems trained on hand-labeled Twitter messages. Because our model does not require hand-labeled data, we can adapt our system to other domains and named entity classes very easily. We demonstrate the flexibility of our approach by successfully applying it to the different domain of extracting food dishes from restaurant reviews with very little extra work.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *language models*

General Terms

Algorithms, Experimentation, Measurement

Keywords

Entity recognition, Entity segmentation, Language modeling, Social media, Restaurant reviews

1. INTRODUCTION

In addition to already existing online informal text such as reviews and forum posts, the advent of microblogging services such as Twitter, Tumblr, and Facebook, together with the ability to publish from anywhere via smartphones, has lowered the barrier to content creation significantly in recent years. Coupled with the ease of reaching a broad audience via online social networks, casual or informal content creation has seen an explosion in popularity.

The microblogging service Twitter alone reports it publishes over 340M tweets per day[1]. The same factors that encourage prolific publication also encourage these Twitter messages, or tweets, to be more informal, noisier and difficult to interpret than previously studied corpora. For example, Ritter et al. [47], reports over 50 lexical variations for *tomorrow*. Other factors that confound interpretation of informal content include capitalization that signals emotional intensity instead of proper noun status, a higher frequency of misspellings introduced by limited keyboards on mobile devices[2], and the abbreviations and grammatical errors introduced to fit messages into the 140-character limit imposed by services such as Twitter.

Despite the brevity and difficulty of interpreting individual tweets, in large numbers tweets have already been shown useful for predicting movie box office revenue [3], political election polls [45,49], flu trends [1] and even stock market movements [6]. Whether it is movie titles, names of politicians, or company names, all of these tasks that currently use keyword matching can benefit from a generalized Named Entity Recognition (NER) system that can learn different entity classes. Informal content exists everywhere and will be increasingly important for us to understand. However, current NER systems such as Stanford NER that achieve F_1 scores of 0.87 on news articles [21], achieve a significantly lower F_1 score of 0.39 on tweets with a precision as low as 0.35.

Furthermore, there are other services just as popular as Twitter such as Facebook generating over 400M status updates per day[3], Tumblr with over 40M posts per day[4], Yelp with over 20M

[1] http://blog.twitter.com/2012/03/twitter-turns-six.html

[2] http://allthingsd.com/20110927/nearly-half-of-tweets-originate-from-mobile-says-twitter-engineering-head/

[3] http://zonetwork.net/there-are-300-thousand-status-updates-per-minute-on-facebook

[4] http://techcrunch.com/2011/09/26/tumblr-raises-85-million-round-from-richard-branson-vcs/

restaurant reviews[5] and a myriad of other smaller services with tremendous growth. Facebook, for example, does not impose a 140-character limit on their status updates. This means that it is important for the research community to develop a generalized NER system that can not only learn different entity classes to recognize, such as products and movies, but also adapt to different domains of informal content, such as Twitter and Facebook.

Many state-of-the-art NER systems have focused on methods that require hand-labeled training data. However, we would need a significantly larger corpus of training data to capture the same amount of representation in large amounts of informal content such as Twitter. With tweets like the following:

mi luv yuh long time Amazon

It is unlikely that manageable sizes of labeled training data will be sufficiently representative of the Twitter corpus and corpora from other domains of informal content. We, therefore, need an approach that can process large-scale corpora to train models that are representative of these domains of informal content without hand-labeled training data.

With this motivation, we present a new model and use large-scale weak supervision signals to replace the need of hand-labeled training data. This system matches state-of-the-art NER systems trained on a large corpus of hand-labeled Twitter data.

In this paper, we present the following contributions:

1. A generative model which we call N-gram Language Markov Model that explicitly models entity boundaries and combines ideas from Hidden Markov Model (HMM) and n-gram language models.
2. Use of Wikipedia titles and page view counts as weak supervision signals to replace hand-labeled training data. While many NER systems use Wikipedia as a resource to learn entity names, to the best of our knowledge, we are the first to use popularity data improving F_1 score by 78%.
3. A generalized NER system that can not only learn different entity classes but also adapt to different domains with very little extra work. We demonstrate this by adapting the NER system to recognize food dishes from restaurant reviews.

The rest of this paper will be structured as follows. In Section 2 we present related work. Section 3 presents details of our model. Section 4 describes the background language model. Section 5 formulates the foreground language model and in Section 6 we evaluate our NER system against existing baselines. To demonstrate the flexibility of our model in adapting to other entity classes and domains, we also evaluate our NER system for food recognition. We then discuss areas of future work in Section 7 and finally conclude in Section 8.

2. RELATED WORK

Thanks to popular conferences such as MUC, ACE, CoNLL and their easily accessible training sets, NER on formal text such as news articles has been a widely studied task in the past two decades [43]. Earlier techniques include rule-based approaches [32] and Hidden Markov Models (HMM) [4,42,52] while later approaches have favored Conditional Random Fields (CRF) proposed by Lafferty et al. [33] enabling state-of-the-art systems such as the Stanford NER to achieve F_1 scores of 0.87 [21] and over 0.90 from Ratinov and Roth [46].

Although NER on news articles have achieved near-human results, NER on other domains have been much less studied. Recent approaches have proposed domain adaptation [17] to alleviate the need of additional labeled data in the new domain for supervised learning systems [2,5,11,13,22,24,29,50]. However, even for seemingly identical domains such as Reuters and Wall Street Journal, Ciaramita and Altun [12] reported a significant degrade in F_1 score from 0.91 to 0.64 making transfer learning from the news domain to social media such as tweets rather hard.

Finin et al. [20] uses both Amazon Mechanical Turk and CrowdFlower to collect named entity annotations for Twitter. Liu et al. [37] present a semi-supervised approach for NER on Twitter combining CRF trained from human labeled tweets with k-nearest neighbor. Ritter et al. [47] use over 350k tokens of labeled news and Twitter data to rebuild all the components of a CRF-based NER system including a part-of-speech (POS) tagger, noun-phrase chunking, capitalization classifier and named entity segmentation while using LabeledLDA and Freebase dictionaries for distant supervision.

The motivation of this paper is to create a generalized NER system that can not only easily learn different entity classes but also easily adapt to different domains by replacing hand-labeled training data with weak supervision signals. In contrast to prior art, we present a novel generative model combining the ideas from HMM and n-gram language models that uses weak supervision signals and does not rely on capitalization or POS tagging. Because our statistical approach does not use domain specific features or significant amounts of labeled data, we are able to easily adapt the same learning across domains, and in this paper, demonstrate this by adapting the same system to recognize food dishes in restaurant reviews. In Section 3, we present this knowledge-intensive generative model called N-gram Language Markov Model (NLMM).

3. N-GRAM LANGUAGE MARKOV MODEL

In this section, we propose a generative model for recognizing named entities that combines ideas from HMM and n-gram language models into a novel N-gram Language Markov Model (NLMM). For convenience and clarity, we use the task of recognizing three types of named entities: PERSON, LOCATION, and ORGANIZATION to explain our framework.

At test time, we use NLMM to find and segment entity mention boundaries. During this stage, we do not separate PERSON, LOCATION and ORGANIZATION. All entity types are tagged as *foreground*, and all other words are tagged as *background*. After finding the entity mentions, we then use three models, one for each type, to classify the entity mentions into their categories.

In training time, however, we do not use any hand-labeled data. One main contribution of our framework is that we replace the large amounts of hand-labeled training data for POS tagging, noun phrase chunking, segmentation and type classification with large-scale weak supervision signals to estimate the parameters in NLMM. While not all parameters of NLMM can be estimated precisely given that we do not use labeled data, the scale of the weak supervision signals still makes our model very competitive as shown in Section 6.

[5] http://officialblog.yelp.com/2011/07/four-score-and-20-million-reviews-ago.html

3.1 Motivation

Our motivation is to create a generalized NER system that 1) is expressive enough to model and predict entity mentions accurately, and 2) allows use of weak supervision to estimate the model parameters so that the model can easily learn new entity classes and adapt to different domains. Before presenting our model, we first analyze several machine learning models for sequential tagging including HMM (Figure 1a, left), MEMM (Figure 1a, right) and CRF (Figure 1b, left).

We first consider discriminative models such as MEMM and CRF. Typically discriminative models need a large amount of labeled data to estimate a model. While there is prior art for using weak supervision signals to reduce the amount of hand-labeled training data, Li et al. show that performance may degrade significantly [35]. Even the smallest 500 label set used by Li et al. will not scale as we increase the number of domains and entity types our generalized NER system needs to handle. Therefore, discriminative models such as MEMM and CRF are not a good fit for building a generalized NER system.

Finally, we consider the HMM. Given a word sequence $W_{1\ldots n} = w_1\ w_2\ \ldots\ w_n$, the goal is to find the stochastic optimal tag sequence $T_{1\ldots n} = t_1\ t_2\ \ldots\ t_n$ with the standard BIO tagging schema (Beginning, Inside and Outside of an entity). By applying Bayes' rule we have the following generative model for HMM:

$$argmax_T P(T_{1\ldots n}|W_{1\ldots n}) = argmax_T \frac{P(T_{1\ldots n}, W_{1\ldots n})}{P(W_{1\ldots n})}$$
$$= argmax_T P(T_{1\ldots n}, W_{1\ldots n})$$

HMM then models the above joint probability as follows:

$$P(T_{1\ldots n}, W_{1\ldots n}) = P(T_{1\ldots n})P(W_{1\ldots n}|T_{1\ldots n})$$

However, we argue that HMM does not have enough expressivity for named entity recognition. Consider the generative process for a first-order HMM:

$$t_i \sim t_{i-1} \quad w_i \sim t_i$$

Where the tag t_i only depends on the previous tag $t_{i-1}$and the word w_i is generated from t_i. This model does not capture enough context information. Consider the following phrase:

listening to am

In HMM, $w_i \sim t_i$ will cause *am* to almost never be labeled as an entity where in fact, *am*[6] is a name of a musician. However, if we observe the count of the bigram *to am*, which is rare, we can then learn that *am* is likely an entity.

3.2 Generative Story

In NLMM, we bring in the idea of using a language model here – the generation of the current word does not only depend on the current tag, but also on the n-gram history *h*. The definition of *h* is context dependent; we will give a precise definition later.

Intuitively, if a word belongs to the *background*, or the O-tag, this word should be generated using the background language model. On the other hand, if a word belongs to the foreground, B-tag or I-tag, this word should be generated using the foreground language model we build from named entities. Moreover, we also want to explicitly model the boundaries of entity mentions. For example, to switch from an *I-tag* to an *O-tag,* we want to consider how likely the previous word appears at the end of an entity name. Hence, the generation of the current tag also depends on the previous word as shown in Figure 1. This is one of the key differences between NLMM and HMM.

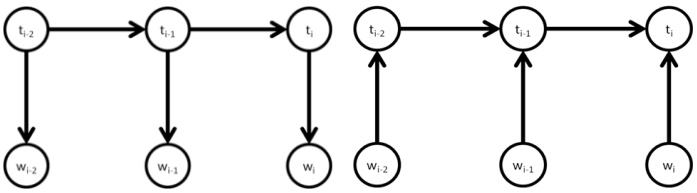

Figure 1a. Hidden Markov Model (HMM) [left] and Maximum-Entropy Markov Model (MEMM) [right]

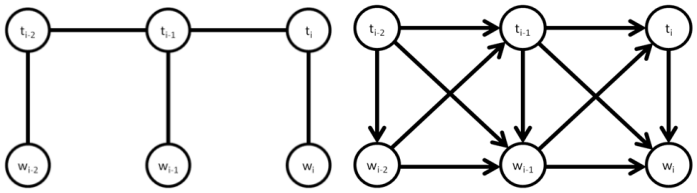

Figure 1b. Conditional Random Fields (CRF) [left] and First-order N-gram Language Markov Model (NLMM) [right]

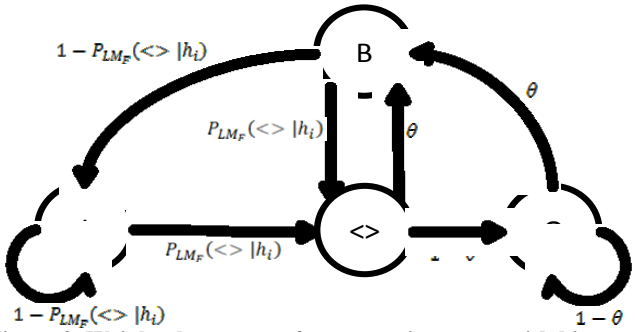

Figure 2. Weighted automata for generating tag t_i with history h_i

In NLMM, the probability of generating the next tag depends on both previous tags and words. More formally, for tag t_i and a k-th order NLMM model, we have the following:

$$t_i \sim T_{(i-k)\ldots(i-1)}, W_{(i-k)\ldots(i-1)}$$

The parameter specification for generating tag t_i is shown in Figure 2. In addition to the BIO tags, we have a symbol <> we call the *boundary token*. We explicitly model the boundary of an entity mention with the boundary token. A prior θ is included every time we enter the foreground language model generating a beginning of an entity mention (B-tag). Once we are in the foreground (B or I) we use the words appearing in the entity to decide how likely the next word should also be part of this entity (I-tag). Therefore, $P_{LM_F}(<> |h_i)$ is the probability we exit the current entity mention and hence we call it the exit probability. Note that the <> state does not generate any word and will continue to generate the next entity mention (B-tag) or a background tag (O-tag). The history *h* here is the last *k* words in the current entity name and is generated from $T_{(i-k)\ldots(i-1)}, W_{(i-k)\ldots(i-1)}$.

By explicitly modeling entity name boundaries with exit probabilities, we can correctly segment informal text such as the following tweet:

[6] http://en.wikipedia.org/wiki/AM_(musician)

if they got Ice Cube Nas Immortal and Jay Z on the same track.

Because of the missing commas between the first three names, NER systems that do not explicitly model entity name boundaries and rely on, for example, POS-tagging or capitalization will likely treat the first three musician names as a single entity name on the incorrect assumption that grammar is regular in informal text.

To generate the word w_i, we have the following:

$$w_i \sim t_{(i-k)\dots i}, w_{(i-k)\dots(i-1)}$$

With t_i already generated, we select the context and language model to generate w_i. If t_i = O, we generate w_i using the background language model with last k words as history h. If t_i = B, we generate w_i using the foreground language model without any history. If t_i = I, we generate w_i using the foreground language model with the last k words in the current entity name.

The probability of a word sequence given a tag sequence with p entities each with length v_j, q O-tags and n-gram history h_u for token w_u is hence:

$$P(W_{1\dots n}|T_{1\dots n}) = \left[\prod_{j=1}^{p}\prod_{u=1}^{v_j} P_{LM_F}(w_u|h_u)\right]\left[\prod_{j=1}^{q} P_{LM_B}(w_j|h_j)\right]$$

Given a word sequence at test time, we can then use dynamic programming to find the best tag sequence.

Once we have our entity segmentations resulting from consecutive BI-tags, for entity classification, we directly compare the log-likelihood of each entity E with length v_j in each entity class language model as follows:

$$E_{class}$$
$$= argmax_{class}\, P_{LM_{class}}\left(E_{v_j}\right)$$
$$= argmax_{class}\left[\prod_{u=1}^{v_j} P_{LM_{class}}(w_u|h_u)\right] P_{LM_{class}}\left(<> \,|h_{v_j+1}\right)$$

As mentioned earlier, in addition to the combined foreground language model used for entity segmentation, we also have one language model for each entity class we use for entity classification. The supervision signals we use to train these language models are identical. In the next two sections, we formally define the background and foreground language model and how to use large-scale weak supervision signals to set the parameters of NLMM and build these language models.

4. BUILDING THE BACKGROUND LANGUAGE MODEL

The background language model is a statistical representation of phrases in the corpus of the domain we want NLMM to tag as non-entity, i.e. O in the BIO tagging schema. To minimize incorrect tagging, this background language model should not contain entity names. We have the following:

$$LM_B \propto Corpus - Entities$$

Removing entity names from the corpus can generally be done in two ways. The first is to hand-label in-domain documents as training data. This is the approach being taken by other NER systems that we want to avoid. Another approach is to iteratively remove named entities that the algorithm is more confident about via methods such as class-n-gram language models. However, because of the low precision of current state-of-the-art NER systems on tweets, as shown in Section 6, it is likely that a naïve implementation of such an approach will not suffice and therefore, we leave this open to future work.

Without hand-labeled training data, we instead model the background language model as the raw unlabeled corpus which includes the entity names we want to identify. In Section 5, we present the foreground language model and its more complex properties.

5. BUILDING THE FOREGROUND LANGUAGE MODEL

Instead of using hand-labeled training data, we use large-scale weak supervision signals to recognize named entities. This representation is called the foreground language model LM_F. We model this foreground language model as a mixture model interpolated across the different data sources. More formally, with D_i data sources, we define the n-gram foreground language model as the following:

$$LM_F = \sum_{d=1}^{D_i} \alpha_d LM_d$$

Tuning α_d is discussed in Section 6.4. As mentioned in Section 3, NLMM uses one foreground language model that is the combination of all entity classes for entity segmentation. During entity type classification, we have one language model for each entity type, LM_{class}. All of the language models in NLMM are mixture models interpolated across their respective data sources as described in the equation above.

In building LM_d, we need two pieces of information as weak supervision signals – entity mentions and entity popularity counts. In the next two Sections 5.1 and 5.2, we talk about these two weak supervision signals. Finally, a common question regarding language models is their ability to recognize out-of-vocabulary (OOV) words. We discuss this in Section 5.3.

5.1 Entity Mentions

The first step is to create a gazetteer, or dictionary of entity mentions (i.e. surface forms), with which to train our foreground language model. Prior art in this area describe how to best utilize Wikipedia categories to build gazetteers [30, 48] quite well, therefore, in this section we focus on presenting the abstract formulation of this process.

Traditional gazetteers only have canonical entity names without alternate entity mentions that refer to the same entity. While this may be sufficient in the news domain, informal content contains more lexical variants we need to capture by including all entity mentions. A typical example is *LOTR*[7] for *The Lord of the Rings*. Therefore, given a seed list of canonical entity names or entity mentions, we need to collect all entity mentions for a particular entity class. Although using Wikipedia allows us to collect alternate entity references more easily, we can still build our list of entity mentions from other sources such as search queries or from pre-curated lists.

Consider an undirected graph with n vertices, $G_n = v_1, v_2 \dots v_i \dots v_j \dots v_n$, where vertices represent entity mentions and edge weights e_{ij} represent the confidence or

[7] http://en.wikipedia.org/wiki/LOTR

probability two vertices v_i, v_j belong to the same entity class. Entity mentions with k different meanings are represented as k independent vertices and share a non-zero edge weight if and only if they have a possibility of sharing the same entity class. The goal here is to iteratively propagate the edge weights to reach as many entity mentions for a given entity class.

In Section 6.4, we outline how we initialized the graph vertices and edges using Wikipedia as the data source. In the next section, we describe how to use entity popularity to bias entity mentions.

5.2 Entity Popularity

The ideal source of popularity data is the counts of entities mentioned in the domain *corpus itself*. Unfortunately, this data does not exist so we estimate it with popularity data from other sources, i.e. Wikipedia page view counts, in the same way we estimate entity mentions with Wikipedia article titles and redirects as weak supervision signals for entity mentions on Twitter.

To improve the precision of our model, we normalize the popularity data by discounting entity mentions which have more alternate meanings. We call this the ambiguity factor. Because language model probabilities sum to one, the ambiguity factor allows us to more aggressively identify entity names that we are more confident by discounting ambiguous entity mentions. More formally, given an entity with popularity count (C_j) and n meanings of the same phrase we normalize C_j with the following:

$$C_{j\,normalized} = C_j \times \frac{C_j}{\sum_1^n C_k}$$

For unambiguous terms such as *Microsoft*, n = 1 leaving popularity count unchanged. As shown in Section 6, empirically we find that this normalization factor increases precision by 18% for ORGANIZATION and 12% overall.

5.3 Out-Of-Vocabulary

Out-of-vocabulary (OOV) refers to a class of problems where entity mentions will never be identified, no matter how strong the context, because they do not exist in the foreground language model vocabulary. Assuming we have already used all data sources at our disposal, NLMM alleviates OOV by training both the foreground and background language model with the same joint vocabulary. We also assign a very small non-zero probability to tokens that are in neither the foreground or background vocabulary; this allows NLMM to look at the surrounding context. In Section 7, we identify a class-n-gram background language model as future work to further address this issue.

6. EVALUATION

The motivation of our paper is to create a generalized NER system that can relatively easily learn new entity classes and adapt to recognize entities in different domains. Finding another NER system that does not require hand-labeled training data is challenging because recent work in NER has focused on using discriminative models to outperform baselines in formal text.

In this section, we compare our NLMM-based NER system with two state-of-the-art CRF-based systems. First, we evaluate the Stanford NER[8] on Twitter using the models that came with the package without hand-labeling new Twitter messages to retrain the model. This is a fair baseline because no new hand-labeled training data is being used for NLMM and we therefore evaluate the domain adaptability of the state-of-the-art NER systems. We expect our NLMM-based NER system to outperform Stanford NER due to the poor domain adaptability of discriminative models. We also compare our system with a CRF-based NER that is trained on Twitter messages, Ritter et al.'s NER[9]. This is not an entirely fair comparison because of the difference in amount of hand-labeled training data used so we expect Ritter et al.'s system to outperform our system.

6.1 Data Utilization

In labeling our evaluation set, we follow the CoNLL 2003 annotation guidelines for PERSON, LOCATION and ORGANIZATION to hand-label 1300 randomly sampled English tweets across a few weeks with two-fold verification. Given the discrepancies we have seen in reported F_1 scores, we contacted Ritter et al., and found that the sampling method makes a noticeable difference in NER performance.

We normalize all tweets to remove retweets (RT), @usernames and #hashtags as these are tokens that may easily confuse any of the NER systems but can also be easily recognized by simple regular expressions.

To train the foreground language model we use publicly available Wikipedia monthly dumps[10] (34GBs) to get page titles (11M), redirects (5M) and category information (661K unique categories). Aside from the freshness and accessibility of Wikipedia dumps, we use category metadata to filter entity classes and redirects to alleviate lexical variation and alternate entity mentions. We also have access to 300GBs of Wikipedia page view counts[11], representing one month's worth of page views information we use as popularity data. To train the background language model, we use 2 million tweets randomly sampled from one day[12]. We normalize language models by removing all capitalization and punctuation then build a trigram language model with Good-Turing smoothing with a shared vocabulary. Both foreground and background language models should be built and normalized the same way to maintain consistency.

6.2 Comparison to Stanford NER

We compare our NER system with two current state-of-the-art CRF-based systems trained on large amounts of hand-labeled data, Stanford NER and Ritter et al. NER, both of which are publicly available. Both systems return entity types in addition to the above three annotated categories so we discard these labels without hurting precision or recall. We measure precision, recall and F_1 score, which are widely used in evaluating NER systems.

Tables 1-4 show results of our NLMM based NER in comparison with Stanford NER trained on both CoNLL and MUC. We evaluate NLMM with only entity names as weak supervision signals (NLMM Title), with entity names and page view popularity (NLMM Title + PV) as well as with normalizing with the ambiguity factor (NLMM Title + PV + Norm).

We expected both CoNLL and MUC to have high precision and low recall because these corpora are news articles with formal capitalization and grammar causing the POS-tagger and noun-phrase chunking features to trigger when tokens are capitalized.

[8] http://nlp.stanford.edu/software/CRF-NER.shtml

[9] https://github.com/aritter/twitter_nlp

[10] http://dumps.wikimedia.org/enwiki/

[11] http://dammit.lt/wikistats

[12] On Oct. 17, 2011, Twitter reported 250 million tweets per day

This explains why precision for MUC is a lot higher than recall but we are unsure why the same is not true for CoNLL. In both Stanford NER models, ORGANIZATION has extremely low precision and recall. The reason this happens is because in news articles, when a capitalized noun phrase is detected but not found in the gazetteer used to train the model, this often means that the noun phrase is a new ORGANIZATION since new PERSON and LOCATION names are much less common. Hence, all the harder cases for which the discriminative model is not confident about are defaulting to the ORGANIZATION entity class making the other two entity classes score higher than they would otherwise.

As shown in the same tables, our NLMM based NER system outperforms both of Stanford NER's F_1 scores by 23% and 12% for CoNLL and MUC respectively. In the ORGANIZATION entity class, we outperform F_1 scores by a wide margin of 86% and 70% for CoNLL and MUC respectively. As we expected, without newly hand-labeled training data, existing NER systems using discriminative models adapt poorly to other domains.

Table 1. NER on tweets results for PERSON

	Precision	Recall	F_1 score
Stanford NER (CoNLL)	0.58	0.54	0.56
Stanford NER (MUC)	0.78	0.42	0.54
Ritter et al. NER	0.48	0.52	0.50
NLMM Title	0.40	0.34	0.37
NLMM Title + PV	0.55	0.56	0.55
NLMM Title + PV + Norm	**0.59**	**0.57**	**0.58**

Table 2. NER on tweets results for LOCATION

	Precision	Recall	F_1 score
Stanford NER (CoNLL)	0.45	0.47	0.46
Stanford NER (MUC)	0.68	0.45	0.55
Ritter et al. NER	0.63	0.40	0.49
NLMM Title	0.29	0.17	0.21
NLMM Title + PV	0.53	0.43	0.47
NLMM Title + PV + Norm	**0.59**	**0.39**	**0.47**

Table 3. NER on tweets results for ORGANIZATION

	Precision	Recall	F_1 score
Stanford NER (CoNLL)	0.17	0.28	0.21
Stanford NER (MUC)	0.27	0.20	0.23
Ritter et al. NER	0.60	0.34	0.43
NLMM Title	0.21	0.22	0.22
NLMM Title + PV	0.33	0.39	0.36
NLMM Title + PV + Norm	**0.39**	**0.39**	**0.39**

Table 4. Overall NER on tweets results

	Precision	Recall	F_1 score
Stanford NER (CoNLL)	0.35	0.43	0.39
Stanford NER (MUC)	0.54	0.35	0.43
Ritter et al. NER	0.62	0.42	0.50
NLMM Title	0.30	0.25	0.27
NLMM Title + PV	0.46	0.46	0.46
NLMM Title + PV + Norm	**0.51**	**0.45**	**0.48**

6.3 Comparison to Ritter et al. NER

Ritter et al. [47] use over 350k tokens of labeled news and Twitter data representing thousands of hand-labeled tweets to rebuild all the components of a pipeline NER system including a part-of-speech (POS) tagger, noun-phrase chunking, capitalization classifier and named entity segmentation. Tables 1-4 also show results of Ritter et al.'s NER system in comparison to ours. Similar to Stanford NER, we expect to see a higher precision than recall and in fact, this is what we observed.

Ritter et al.'s NER performs worse than Stanford NER for both PERSON and LOCATION entity classes, however, does very well in ORGANIZATION. This is likely because the hand-labeled Twitter data is helping the discriminative model identify what organizations entities are as opposed to default assigning the ORGANIZATION class for unknown noun phrases based on capitalization and POS-tagging. This therefore, means that some of the more difficult terms in tweets are also assigned to PERSON and LOCATION, hence we see a performance degrade for both these categories compared to the Stanford NER.

Even with all the additional hand-labeled tweets, Ritter et al.'s NER system outperforms our system only by a slight margin. Another surprise is that Ritter et al.'s system took significantly longer to analyze 1300 tweets from the test set. Our NLMM-based NER system takes less than two minutes to analyze the same messages. The F_1 score we measure, 0.50, is lower than Ritter et al.'s reported score of 0.59. We contacted Ritter and found that they sampled the Twitter API with temporal keywords[13] and used this corpus for their training and test set. We suspect that these correctly spelled temporal keywords may have biased the data towards more formal and regular text.

In comparison, our model is able to tag these messages correctly where CRF-based hand-labeled NER systems such as both the Stanford NER and Ritter et al.'s NER have difficulty:

you youngins' just can't define it with gucci mane and souja boy!

Via Waiting in line for Sharon jones, broken bells and spoon.

if they got Ice Cube Nas Immortal and Jay Z on the same track.

Even with a significant effort of hand-labeled training data, it is unclear that the current state-of-the-art CRF-based NER systems can consistently outperform our NLMM-based NER system. More important is the difficulty in hand-labeling new training data for each new domain and entity class therefore, making it difficult to adapt. On the other hand, Twitter is only one of the many domains

[13] Such as *today*, *yesterday*, *January*, *Sunday*…

of informal content our NLMM-based NER system can be easily trained for. It is also easy for NLMM to recognize entities at a more granular level such as *politicians, artists, musicians* all of which would require hand-labeled training data in CRF-based NER systems to be re-labeled. In Section 6.6, we demonstrate the flexibility of our NLMM-based NER system by adapting our NER system to recognize food dishes from restaurant reviews.

6.4 Details of the Foreground Language Model

In Section 3, we present the first-order NLMM which uses bigram language models to simplify the formulation. Experimentally, we use and evaluate the second-order NLMM with trigram foreground and background language models. In general, higher order NLMMs will result in sparsity and will likely only benefit systems with larger training corpus.

For weak supervision signals, we obtain entity mentions from Wikipedia by starting off with a bag of words for each entity type[14] generated from the CONLL annotation guidelines. We then find all categories in Wikipedia where the category name contains one or more of these words. Once we have this seed list of categories, we initialize the undirected graph, as outlined in Section 5.1, with Wikipedia page titles as vertices and set the edge weight between vertices to 1 if the two vertices share one or more seed categories and 0 otherwise. We also set edge weights between vertices for Wikipedia page redirects if the destination page contains at least one seed category. To improve accuracy and coverage, we can use advanced graph propagation and regularization techniques such as ones used to analyze search query-click behaviors [25,28,36]. However, not all data sources have this structure that allows for an iterative propagation and therefore, we chose to evaluate our model at iteration zero to serve as a baseline. Finally, we apply Wikipedia page view counts as weak supervision signal for popularity as described in Section 5.2.

As presented in Section 5, the foreground language model is a linear interpolation of all data sources with interpolation parameters α_d. We recommend tuning these parameters with a small set of hand-labeled data, 10 tweets for example, via standard algorithms such as gradient descent especially when using multiple data sources. There are also more advanced techniques of representing this mixture of language models [26, 31]. However, because we primarily only use Wikipedia as the data source and wanted to simplify our presentation, we tried a few values {1, 10, 100, 1000} for α_d and eyeballed the results on 10 tweets.

The most important parameter and one that we do tune is the transition probability θ as described in Section 3. We use 10 hand-labeled tweets to run NLMM with $\theta = \{0.1, 0.2, \ldots 0.9\}$ and chose the best performing parameter. In Section 7, we talk about using Expectation Maximization (EM) to tune transition probabilities conditioned on the n-gram history context via class-n-gram language models.

6.5 Varying Background Language Model

The most important aspect in training the background language model is the corpus size. This corpus needs to be large enough to be statistically representative of messages in the domain we want to analyze but beyond that yields diminishing returns. Exact corpus size should vary with domain but in Figure 3 we show the impact on F_1 score for corpus size (# tweets) of 1k, 3k, 10k, 30k, 100k, 300k, 1M and 2M. Notice that we start seeing diminished returns after 100k tweets and after 1M the gain is no longer statistically significant. It would be interesting to measure whether perplexity of the background language model is indicative of the marginal gain in increasing the corpus size. However, training language models efficiently is a well-understood problem [7,19,27,44] and a corpus of 2M tweets (24M tokens) and 2M restaurant reviews (28M tokens) take only 15 minutes on a single machine with little optimization so we leave investigations to future work.

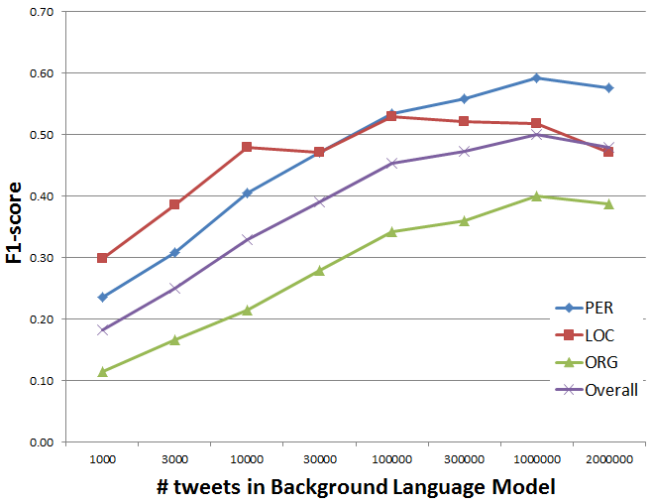

Figure 3. Effect of varying corpus size of background LM on F_1 score

6.6 Learning a new Entity Class and Domain

Because our approach has not used any domain specific features, a natural question is to ask whether we can apply the same method to other domains. The ability to easily adapt to another domain and learn a new entity class is the main advantage of our NLMM-based NER system. In this section, we train different foreground and background language models while using the same NLMM as we did with NER on twitter to not only a completely different domain, restaurant reviews, (*domain adaptation* [17]) but also an entirely different entity class recognition task, FOOD, (*multi-task learning* [9]).

For recognizing food dishes in restaurant reviews we define the FOOD entity class broadly as including food, drinks and other edible things. To evaluate our food dishes NER we hand-label 200 reviews from the Yelp and Citysearch review corpus. We use 2M unlabeled restaurant reviews to build our background language mode. For the foreground language model, we provide insight into how different data sources perform for this recognition task.

Table 5 shows precision, recall, and F_1 score for NLMM when trained to recognize food dishes in restaurant reviews. The goal here is to show that domain adaptation and learning new entity classes is effortless compared to CRF-based or other methods that require a large amount of hand-labeled data. To provide insight into how different data sources perform for a given recognition task, we try three data sources: menu items crawled from

[14] PERSON = {alumni, people, births, artists},

LOCATION = {countries, landmarks, capitals, states, cities},

ORGANIZATION = {companies, organizations, universities, bands, groups, magazines, institutions, teams, newspapers}

restaurant web sites (MI) with numbers designating cut-off frequency, query-URL-click logs from Bing (Query), and Wikipedia article titles from the same way we extract article titles for NER on tweets except with food categories (Wiki). For popularity data we use Wikipedia page view counts, query click counts, and menu item counts each with $\alpha_d = 1$. Tables 1-4 already evaluate the effects of entity popularity (page view counts, PV) and normalizing ambiguity factor so in Table 5 we focus on presenting effects of different data sources. All NLMM performances in Table 5 use entity name, entity popularity and normalization (Title + PV + Norm).

Table 5. Food dishes recognition on Yelp & Citysearch reviews

	Precision	Recall	F_1 score
Lookup (MI-10)[15]	0.00	0.02	0.01
Lookup (MI-100)	0.27	0.27	0.27
Lookup (MI-1000)	0.25	0.07	0.11
Lookup (Wiki + MI-100)	0.10	0.37	0.16
NLMM (Query + MI-100)	0.41	0.26	0.32
NLMM (Wiki)	0.58	0.48	0.53
NLMM (Wiki + MI-100)	**0.58**	**0.64**	**0.61**

As baselines, we present four dictionary lookups each with a different list of food dishes. The entire menu items dataset has 980k unique menu items. Lookup (MI-10) represents items in this menu item corpus that have at least 10 occurrences (39k items) while Lookup (MI-1000) represents items in this menu item corpus that have at least 1000 occurrences (295 items). As shown from the Lookup baselines, the menu item dataset is extremely low quality.

We also try generating food dish names from queries issued to Bing resulting in a 19% improvement in F_1 score over the baseline. Our best performing model combines Wikipedia data with menu items improving F_1 score by 126% over the best dictionary lookup baseline. To verify that we did not just create a high quality gazetteer, we take the same corpus used to train our best performing foreground language model and use that in a dictionary lookup, Lookup (Wiki + MI-100). We find that this gazetteer is actually very dirty with a precision as low as 0.10 further reinforcing that our NLMM is working as intended.

In this section, we show that NLMM can easily adapt to learn a new entity class, FOOD, and adapt to another domain, restaurant reviews, with no hand-labeled training data. This level of adaptability in an NER system is a novel contribution that even state-of-the-art systems such as Stanford NER and Ritter et al.'s NER system cannot match.

7. FUTURE WORK

Echoed by many others working on NER and even more so for tweets [37,38,47], due to the wide range of entity types, entity mentions, and lexical variations, having a high-quality large dataset is extremely important as observed by Ratinov and Roth [46] and others [14,30,48]. Aspects of Wikipedia we have not yet taken full advantage of include disambiguation pages, link text as alternate surface forms [15], category hierarchies, lists and tables. In addition to Wikipedia, Freebase [47], Wordnet [39, 48], and other domain-specific data sources such as IMDB[16] for movies and CrunchBase[17] for startups are all interesting datasets to consider. Once there are more data sources, learning the interpolation parameters for the mixture model will also need to be considered.

Even without improving our gazetteers there are a lot of improvements we can explore in how we train our language models. In the current method, we use a single probability to model the transition likelihood from the background language model to an entity mention, or foreground language model. This is clearly an over-simplification because the transition likelihood should depend on the context. The likelihood of g*ame* referring to the rapper in *listening to [game]* should be much higher than *playing a [game]*. We can model this context dependent transition probability further by using a class-based n-gram model [8] for the background language model. Furthermore, in light of recent work to efficiently train complex language models such as maximum entropy language models (MELM) and neural network language models (NNLM) [41,51], an interesting area of future work would be to evaluate whether MELM, NNLM or combinations of such will yield a better performance [40] with our NLMM.

As we move away from formal text and consider informal content, normalization of lexical variations, whether misspelled or not, becomes increasingly important especially as users move towards using smartphones to author their content[18]. We can adapt language modeling techniques from well-studied domains such as query spelling correction [10,16,18,23,34], to help NLMM recognize entities such as *jstin beeberz* which are missing from current gazetteers.

Training our NER system to recognize more entity types such as products, movies and songs will not only increase the overall utility of this system, but also increase the quality of results in existing entity types. Although segmentation algorithms that consider punctuation and capitalization may help in the *I Love You Phillip Morris* example, a foreground language model for movies would also help the system understand that this is actually a movie and therefore, not label *Philip Morris* as a person.

Last but not the least, we would like to explore non-local dependencies not currently captured by NLMM. Cucerzan [15] uses Wikipedia links and disambiguation pages to disambiguate entities. In social media due to the brevity of messages, we would need to consider non-entity tokens as well to help in classification and disambiguation of identified entities.

8. CONCLUSION

Our motivation to build a generalized NER for any domain and entity class is showing promising results. In this paper, we propose a novel generative model that combines ideas from HMM and n-gram language models we call N-gram Language Markov Model (NLMM). NLMM allows us to explicitly model entity name boundaries and use weak supervision signals to replace

[15] Menu items crawled from the web with at least 10 occurrences

[16] http://www.imdb.com/interfaces

[17] http://www.crunchbase.com/help/api

[18] http://allthingsd.com/20110927/nearly-half-of-tweets-originate-from-mobile-says-twitter-engineering-head/

hand-labeled training data. We outperform domain adaptability of Stanford NER trained on news data and match Ritter et al.'s NER system trained on a large amount of hand-labeled tweets. We also demonstrate that our NLMM-based NER system can easily adapt from recognizing person, location, and organization in tweets to identifying food dishes in restaurant reviews.

We hope that this paper will inspire the NER research community to consider taking steps towards building a generalized NER without using hand-labeled training data. Demonstrating the use of existing non-domain-specific data sources and minimal amounts of hand-labeled data to build NER for arbitrary domains has opened the door to a new avenue of research into generalized NER that promises to have significant practical impact on information extraction and web search for informal content.

9. REFERENCES

[1] Harshavardhan Achrekar, Avinash Gandhe, Ross Lazarus, Ssu-Hsin Yu, and Benyuan Liu. 2011. Predicting flu trends using Twitter data. In CCW IEEE, pages 702-707.

[2] Andrew Arnold, Ramesh Nallapati, and William W. Cohen. 2008. Exploiting feature hierarchy for transfer learning in named entity recognition. In ACL, pages 245-253.

[3] Sitaram Asur and Bernardo A. Huberman. 2010.Predicting the future with social media. In WIIAT, pages 492-499.

[4] Daniel M. Bikel, Richard Schwartz and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. Machine Learning Journal Special Issue on Natural Language Learning.

[5] John Blitzer, Ryan Mcdonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In EMNLP, pages 120-128.

[6] Johan Bollen, Huina Mao, and Xiao-Jun Zeng. 2011. Twitter mood predicts the stock market. In Journal of Computational Science 2(1), pages 1-8.

[7] Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In EMNLP, pages 858-867.

[8] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. In ACL.

[9] Rich Caruana. 1997. Multitask learning. In Machine Learning Vol. 28 No. 1, pages 41–75.

[10] Qing Chen, Mu Li, and Ming Zho. 2007. Improving query spelling correction using web search results. In EMNLP, pages 181-189.

[11] Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Frederick Reiss, and Shivakumar Vaithyanathan. 2010. Domain adaptation of rule-based annotators for named-entity recognition tasks. In EMNLP, pages 1002–1012.

[12] Massimiliano Ciaramita and Yasemin Altun. 2005. Named-entity recognition in novel domains with external lexical knowledge. In Advances in Structured Learning for Text and Speech Processing Workshop.

[13] Massimiliano Ciaramit and Olivier Chapelle. 2010. Adaptive parameters for entity recognition with perceptron HMMs. In DANLP, pages 1-7.

[14] William W. Cohen and Sunita Sarawagi. 2004. Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In KDD.

[15] Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In EMNLP, pages 708-716.

[16] Silviu Cucerzan and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In EMNLP, pages 293-300.

[17] Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. In Journal of Artificial Intelligence Research 26, pages 101-126.

[18] Huizhong Duan and Bo-June (Paul) Hsu. 2011. Online Spelling Correction for Query Completion. In WWW, pages 117-126.

[19] Marcello Federico and Mauro Cettolo. 2007. Efficient handling of n-gram language models for statistical machine translation. In Workshop on Statistical Machine Translation, pages 88-95.

[20] Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in Twitter data with crowdsourcing. In Proceedings of the NAACL Workshop on Creating Speech and Text Language Data With Amazon's Mechanical Turk. Association for Computational Linguistics, June.

[21] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In ACL, pages 363-370.

[22] Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In NAACL-HLT, pages 1-8.

[23] Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In COLING, pages 358-366.

[24] Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu and Zhong Su. 2009. Domain Adaptation with Latent Semantic Association for Named Entity Recognition. In NAACL-HLT, pages 281-289.

[25] Xiaofei He and Pradhuman Jhala. Regularized query classification using search click information. Pattern Recognition, 41(7):2283–2288, 2008.

[26] Bo-June (Paul) Hsu. 2007. Generalized linear interpolation of language models. In ASRU.

[27] Bo-June (Paul) Hsu and James Glass. 2008. Iterative language model estimation: Efficient data structure & algorithms. In Proc. Interspeech.

[28] Ming Ji, Jun Yan, Siyu Gu, Jiawei Han, Xiaofei He, Wei Vivian Zhang, and Zheng Chen. Learning search tasks in queries and web pages via graph regularization. In SIGIR, pages 55–64, 2011.

[29] Jing Jiang and ChengXiang Zhai. 2006. Exploiting domain structure for named entity recognition. In NAACL-HLT, pages 74-81.

[30] Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In EMNLP, pages 698-707.

[31] Dietrich Klakow. 1998. Log-linear interpolation of language models. In ICSLP.

[32] George R. Krupka and Kevin Hausman. 2001. IsoQuest, Inc.: Description of the NetOwlTM extractor system as used for MUC-7. In MUC-7.

[33] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In ICML, pages 282-289.

[34] Mu Li, Muhua Zhu, Yang Zhang, and Ming Zhou. 2006. Exploring distributional similarity based models for query spelling correction. In ACL, 1025-1032.

[35] Xiao Li, Ye-Yi Wang, and Alex Acero. 2009. Extracting Structured Information from User Queries with Semi-Supervised Conditional Random Fields. In SIGIR, pages 572–579.

[36] Xiao Li, Ye-Yi Wang, and Alex Acero. 2008. Learning query intent from regularized click graphs. In SIGIR, pages 339–346.

[37] Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. Recognizing named entities in tweets. In ACL, pages 359-367.

[38] Brian Locke and James Martin. 2009. Named entity recognition: Adapting to microblogging. In Senior Thesis, University of Colorado.

[39] Bernardo Magnini, Matteo Negri, Roberto Prevete, and Hristo Tanev. 2002. A wordnet-based approach to named entities recognition. In SemaNet, pages 38–44.

[40] Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan "Honza" Cernocky. 2011. Empirical evaluation and combination of advanced language modeling techniques. In Interspeech, pages 605-608.

[41] Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, Jan "Honza" Cernocky. 2011. Strategies for training large scale neural network language models. In ASRU.

[42] Scott Miller, Michael Crystal, Heidi Fox, Lance Ramshaw, Richard Schwartz, Rebecca Stone, Ralph Weischedel, and the Annotation Group. 1998. BBN: Description of the SIFT system as used for MUC-7. In MUC-7.

[43] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. Linguisticae Investigationes, 30:3-26.

[44] Patrick Nguyen, Jianfeng Gao, and Milind Mahajan. 2007. MSRLM: A scalable language modeling toolkit. In MSR-TR-2007-144.

[45] Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In ICWSM.

[46] Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In Proceedings CoNLL-2009, pages 147-155.

[47] Alan Ritter, Sam Clark, Mausam and Oren Etzioni. 2011. Named Entity Recognition in Tweets: An Experimental Study. In EMNLP, pages 1524-1534.

[48] Antonio Toral and Rafael Munoz. 2006. A proposal to automatically build and maintain gazetteers for named entity recognition by using Wikipedia. In EACL 2006.

[49] Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner, and Isabell M. Welpe. 2010. Predicting elections with Twitter: What 140 characters reveal about political sentiment. In ICWSM.

[50] Dan Wu, Wee Sun Lee, Nan Ye, and Hai Leong Chieu. 2009. Domain adaptive bootstrapping for named entity recognition. In EMNLP, pages 1523-1532.

[51] Puyang Xu, Asela Gunawardana, Sanjeev Khudanpur. 2011. Efficient subsampling for training complex language models. In EMNLP, pages 1128-1136.

[52] GuoDong Zhou and Jian Su. 2002. Named entity recognition using an HMM-based chunk tagger. In ACL, pages 473-480.