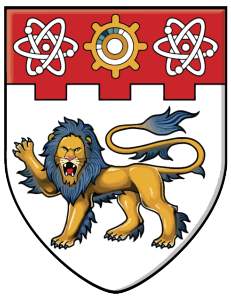


NANYANG
TECHNOLOGICAL
UNIVERSITY

Product Name Recognition and Normalization in Internet Forums

YAO YANGJIE

SCHOOL OF COMPUTER ENGINEERING

2014

Product Name Recognition and Normalization in Internet Forums

Yao Yangjie

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Master of Engineering

2014

Acknowledgments

I would like to express my sincere thanks to my supervisor Prof. Aixin Sun from School of Computer Engineering, who gives me invaluable assistance and continuously encouragement. Under his supervision, I am fortunate to learn state-of-the-art techniques on information retrieval.

I also want to thank my colleagues, Surendra Sedhai, Longke Hu, Zongyang Ma, Chenliang Li and Newsha Ghoreishi. Discussions with them were really helpful and pleasant.

My immense gratitude goes to my family for their great love and continuous encouragement in both my studies and life.

Contents

List of Figures

List of Tables

Abstract

Collecting user feedback of products is a common practice for the product providers to better understand consumers' concerns or requirements and to further improve their products or marketing strategies. Even though dedicated review sites (*e.g.,* Epinions, Amazon, CNET reviews) supply the relatively straightforward approach as user feedback about one specific product is usually well organized in a list, collecting user feedback from Internet forums is challenging. One reason is that user feedback about a product often spreads in different discussion threads in forums. More importantly, users often mention product names with a large number of name variations. On the other hand, Internet forums cover feedback from many more users. Thus, user feedback in more comprehensive aspects can be obtained.

We propose a method named GREN to recognize and normalize mobile phone names from Internet forums. Instead of directly recognizing phone names from sentences as in most named entity recognition tasks, we propose an approach to generating candidate names as the first step. The candidate names capture short forms, spelling variations, and nicknames of products, but are not noise free. To predict whether a candidate name mention in a sentence indeed refers to a specific phone model, a CRF-based name recognizer is developed. The CRF (Conditional Random Field) model is trained by using a large set of sentences obtained in a semi-automatic manner with minimal manual labeling effort. Lastly, a rule-based name normalization component maps a recognized name to its formal form.

For evaluation, we randomly select 20 threads related to 20 mobile phones from an Internet forum. Each thread contains about 100 post messages. We manually labeled the mobile phone name mentions in these posts and mapped the true mentions to their formal names. In total, about 4000 sentences have been manually labeled which contain about 1000 phone name mentions. Evaluated on labeled data, GREN

outperforms all baseline methods. Specifically, it achieves precision and recall of 0.918 and 0.875 respectively, with the best feature setting. Comparing to Stanford NER which is considered as a strong baseline, 134% improvement on recall is observed. We also provide detailed analysis of the intermediate results obtained by each of the three components in GREN and observe that features from Blown clustering are the most effective features. Removing them results in the largest degradation in F_1 from 0.896 to 0.804.

Two implications for NER tasks are further made based on our observation. First, if candidate named entities are able to be pre-generated, a large number of training examples may be generated at very low cost for manual annotation. Second, if we can segment the sentences and pre-generate the text chunks, we are able to rewrite the sentences. The rewriting enables us to take surrounding words of a candidate named entity to be its context in a more natural manner.

Chapter 1

Introduction

1.1 Background and Motivation

Reading relevant discussions and reviews has become a common practice for many users before they purchase consumer products like mobile phones and digital cameras. Many users also seek for recommendations by posting questions on Internet forums and social networking sites (*e.g.,* Facebook, Google+, and Twitter). Such questions, review comments and discussions are important resources for product providers to better understand consumers' concerns or requirements and to further improve their products or marketing strategies. Because social networking sites often limit data crawling for privacy or other reasons, our discussion in this thesis will be focusing on publicly accessible Internet forums.

Crawling product reviews from dedicated review sites (*e.g.,* Epinions, Amazon, CNET reviews) is relatively straightforward because review reports about one specific product are usually well organized. Collecting user feedback (*e.g.,* questions , comments, comparison with other products) about *a specific product* from Internet forums is much more challenging. One reason is that user feedback about a product often spreads in different discussion threads in forums. More importantly, users often mention product names with a large number of name variations containing informal abbreviations, misspellings and nicknames. As an example, Table 1.1 lists 25 name variations of *"Samsung Galaxy SIII"* (both LTE and non-LTE models), each used by at least 10 users in an Internet forum used in our experiments. We also observe more than 6 forms of misspells of "Ericsson" when users mention Sony Ericsson phones. On the other hand, Internet forums cover feedback from more users. Thus,

Table 1.1: Name variations of *"Samsung Galaxy SIII"*

Name variation	#users	Name variation	#users
1. galaxy s3	553	14. lte s3	46
2. s3 lte	343	15. galaxy s3 lte	45
3. samsung galaxy s3	284	16. s3 non lte	32
4. s iii	242	17. samsung galaxy siii	32
5. galaxy s iii	225	18. sgs 3	27
6. samsung s3	219	19. samsung galaxy s3 lte	22
7. sgs3	187	20. sg3	21
8. siii	149	21. gsiii	16
9. samsung galaxy s iii	145	22. samsung galaxy s3 i9300	15
10. i9300	120	23. samsung i9300 galaxy s iii	13
11. gs3	82	24. s3 4g	11
12. galaxy siii	61	25. 3g s3	11
13. i9305	52	–	

user feedback in more comprehensive aspects can be obtained. Furthermore, a user may choose not to buy a product (hence will not write a review report) after reading recommendations and discussions from other forum users. Understanding the reason behind helps a product manufacturer in many aspects ranging from product design to marketing strategy.

In order to obtain user feedback of a product from Internet forums, the intuitive approach is to collect all the post messages containing the name mentions of the product. The approach can be divided into two steps. Name mentions of products are identified at first. As a name mention may refer to a product or a noise based on the context, most of existing systems for Named Entity Recognition (NER) adopt sequence classification methods such as Hidden Markov Model (HMM) [1] and Conditional Random Field (CRF) [2] which utilize contextual information. Given a product, the post messages containing the name mentions of the product are further collected. However, based on our observation, both of the steps are beset by several problems which lead to a poor performance of user feedback collection.

- In general, a post message is short and sparse which cannot supply enough contextual information. In our dataset, the average length of post messages is only 11.6 words. Furthermore, the qualities of post messages are not guaranteed. Some post messages may be written by professionals which are high-quality

while most of post messages contain a large number of ill-formed words. Even though the state-of-the-art NER systems show satisfied results on formal text, when they work on user-generated short text like post messages, the performance drops dramatically.

- Given a product, as a number of name variations may exist, expensive and time-consuming human work needs to be conducted to obtain these name variations in order to collect user feedback of the product in a high coverage.

The solution for these problems is to develop a NER system which performs well on user-generated short text. Then the Named Entity Normalization (NEN) process is designed to map a detected name mention with the formal name it refers to. By associating name mentions in post messages with their corresponding formal names, it is convenient for a product manufacture to collect all the user feedback of a specific product.

1.2 Project Objective

We aim to recognize product name mentions in Internet forums and map each name mention to its formal name. However, different products may follow very different naming conventions depending on the characteristics of the products (*e.g.,* camera lenses vs mobile phones). In this research, we focus on mobile phones and aim to *recognize and normalize phone name mentions in Internet forums.* Even though some other social media platforms such as Facebook and Twitter are more popular, our discussion in this thesis will be focusing on publicly accessible Internet forums for two reasons. First, social networking sites often limit data crawling for privacy or other reasons making data collection a challenge. Second, social media sites like Facebook and Twitter cover many diverse topics while Internet forums are domain-specific. In our study, we extract mobile phone names from domain specific forums and the extracted names could be used to extract names from other social media platforms such as Twitter.

Table 1.2 lists 4 example sentences, taken from a forum. Our task is to recognize the phone name mentions (highlighted in boldface) and map the recognized names

Table 1.2: Mobile phone name mentions (highlighted in boldface), and their formal names [in brackets] in 4 example sentences

1. True, **Desire** [HTC Desire] might be better if compared to **X10** [Sony Ericsson Xperia X10] but since I am using **HD2** [HTC HD2], it will be a little boring to use back HTC ...
2. I just wanna know what problems do users face on the **OneX** [HTC One X]... of course I know that knowing the problems on **one x** [HTC One X] doesn't mean knowing the problems on **s3** [Samsung Galaxy SIII]
3. Still prefer **ip 5** [Apple iPhone 5] then **note 2** [Samsung Galaxy Note II]...
4. oh, the mono rich recording at **920** [Nokia Lumia 920] no better than stereo rich recording at **808** [Nokia 808 PureView].

to their corresponding formal names (shown in brackets). We choose mobile phone as the product of interest for two reasons. First, mobile phones, particularly smart phones, have very large user base, leading to massive relevant discussions/comments in forums. Second, because of the strong competition in the market, mobile phones are released and updated at a very fast pace. Efficient and effective mining of user feedback about mobile phones is a major challenge for all phone makers.

1.3 Approach and Methodology

In our problem setting, we assume two sets of inputs: (i) a collection of formal/official mobile phone names, and (ii) a collection of discussion threads from an Internet forum that are relevant to mobile phones. Note that, discussions in an Internet forum are usually organized into a few *discussion boards*; each discussion board consists of a number of threads and each thread has one or more post messages. Here, we assume the threads are collected from an Internet forum that is dedicated to mobile phone discussions like xda-developers forum[1] or from a specific discussion board that is about mobile phones, *e.g.,* the board titled "Mobile Communication Technology" in HardwareZone forum.[2]

[1] `http://forum.xda-developers.com/`
[2] `http://forums.hardwarezone.com.sg/`

Our proposed solution for mobile phone name recognition and normalization, named Gren, consists of three main components, namely, *candidate name Generator*, *CRF-based name REcognizer*, and *rule-based name Normalizer*.

Candidate Name Generator. As the name suggests, this component generates candidate names that might be used to mention a mobile phone. Examples are the names listed in Table 1.1. The generation consists of two phases. The first phase is to find name variations of phone brands including their common misspells and abbreviations (*e.g.,* bberry for BlackBerry). The next phase is to generate phone name variations by assuming that (i) a phone name is noun phrase, and (ii) a phone name shall either contain a brand variation or appear after a brand variation at least once. As the result of the second phase, we obtain a relatively large collection of candidate names. However, not all candidate names obtained are truly mobile phone names. For example, the word "battery" appears more than once after a brand (*e.g.,* Sony battery); then "battery" will be considered as a candidate name.

CRF-based Name Recognizer. The name recognizer is a classifier based on the linear chain CRF model. Given a sentence which contains at least one candidate name mention, the classifier predicts whether the mention indeed refers to a mobile phone. The prediction is based on the local context derived from the sentence and global context associated with the mentioned candidate name. To learn the classifier, we use three types of features including lexical features, grammatical features, and features derived from candidate names. We highlight that a large number of training instances are semi-automatically labeled in our implementation with minimal human annotation effort.

Rule-based Name Normalizer. The last component of Gren maps a recognized name variation to its formal name (*e.g.,* "sgs3" is mapped to "Samsung Galaxy SIII"). The normalization is mainly based on lexical rules. The confidence of a mapping is measured by the co-occurrence of the name variation and the formal name in selected forum threads.

Although Gren is evaluated in the setting of mobile phone name recognition and normalization in forums, the proposed approach is generic. Candidate name generator is probably the only component among the three that needs to be modified to reflect the naming convention of products in a different domain.

1.4 Contributions

We highlight the contributions of our research as below.

- We propose a novel solution named Gren for mobile phone mention recognition and normalization in Internet forums. Instead of running named entity recognizer on sentences directly, we generate candidate names and then predict whether a candidate name mention indeed refers to a mobile phone. The recognized names are then normalized to formal names.

- We propose a semi-automatic approach to generating training examples for CRF-classifier learning. A large number of training examples are obtained with little manual effort. More specifically, $33,072$ sentences are obtained as training examples by manually labeling only 500 candidate names. A candidate name is a word or a phrase, which is easy to label.

- We conducted extensive experiments to evaluate the proposed Gren method and compared it against baseline methods. Our results show that Gren significantly outperforms all baseline methods. For mobile phone name recognition and normalization, with the best feature setting, Gren achieves precision and recall of 0.918 and 0.875 respectively.

1.5 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 summaries related works. The overview of the Gren method and the detailed description about mobile phone mention recognition including Candidate Name Generator and CRF-based Name Recognizer are presented in Chapter 3. Rule-based Name Normalizer which is used to map a recognized name variation to its formal name is presented in Chapter 4. We evaluate our proposed method and study the results obtained by the different components in the Gren method in Chapter 5. Finally, Chapter 6 concludes the work that has been done and presents our future work.

Chapter 2

Related Work

Our work is related to both Named Entity Recognition (NER) and Named Entity Normalization (NEN). In general, NER detects name mentions in sentences, then NEN links a detected name mention to its formal name or the entity it refers to.

2.1 Named Entity Recognition

The task of NER is to extract and classify information units like person, organization, location and other entity types [3–8]. The task can be divided into two aspects. The first aspect is to detect named entities on formal text like news articles while another aspect is to detect named entities on user-generated short text like tweets.

2.1.1 NER on Formal Text

The task of NER on formal text is a well-studied problem [4–6, 8]. As an example, F_1 of 0.8686 on CoNLL03 data is reported by Stanford NER [4]. Stanford NER is one of best publicly available systems for NER tasks. Most of currently used techniques in NER systems are based on statistical hidden state sequence models such as Hidden Markov Model (HMM) [1] and Conditional Random Field (CRF) [2]. These models are developed based on a Markov property which the decision about the state of current position in a sequence is only depended on a small local window around it. The reason behind it is that this property makes the inference process tractable computation. However, the situation in real world is much complicated. The task sometimes benefits by involving non-local structure. In order to model

long-distance dependency, the authors replace Viterbi algorithm by Gibbs sampling in the inference process. As the state of the word in current position takes the state of words in the rest positions into account, Gibbs sampling captures much more information which gives better results.

LBJ-NER is another state-of-the-art NER system which gives satisfied results on formal text [5]. LBJ (Learning Based Java) [9] is a modeling language for the discriminative model. The details of feature extraction, learning, model evaluation, and inference are all abstracted away from the programmer which makes him more focus on his application. The programmer supplies target-specific local decision methods and their constraint specifications. Then, LBJ uses Integer Linear Programming to automatically learn the decision function by combining the outputs of local decision methods into a coherent, exact, global inference.

Ratinov *et al.* [5] use regularized averaged perceptron model [10] and a number of features including gazetteers extracted from Wikipedia, expressive non-local features and word class features derived from unlabeled text. The model and features are specified in LBJ which obtains F_1 of 0.908 on CoNLL03 data. The result is even better than the result obtained by Stanford NER. They further analyse some of fundamental challenges in order to design an efficient and robust NER system. In the task of NER, the representation of text segments [11] is important. Different representation schemes cause different performance on NER systems. Specifically, they evaluate the performance of their system with two most popular schemes: BIO (Begin-In-Out) and BILOU (Begin-In-Last-Out-Unit) respectively. The results show that BILOU scheme outperforms BIO scheme significantly as BILOU scheme is able to learn a more expressive information by increasing a small number of parameters. They further compare the performance of three inference (decoding) algorithms named greedy left-to-right decoding, Viterbi and Beamsearch. Beanmsearch is a heuristic search algorithm which uses breadth-first search to build a search tree. At each level of the tree, it generates all partial solutions (states), sorts them in order of increasing heuristic values and stores only top k status where k is predetermined. Greedy left-to-right decoding can be considered as Beanmsearch with k equal to one. Viterbi finds the most likely assignment to a second-order model in sentences. Compared to Beanmsearch and Viterbi, greedy left-to-right decoding captures less

local information. However, surprisingly, it performs well in which the performance drops only a little while the speed is much faster.

2.1.2 NER on User-Generated Short Text

Even though relatively good recognition accuracy has been achieved for NER on formal text, NER on user-generated short text such as Twitter messages and post messages in Forums, remains challenging. The reason behind it is that short text is unable to supply enough contextual information for NER. What is worse, the user-generated text is not of high quality. Some text may be written by professionals which has high quality while most text contains a large number of ill-formed words. The ill-formed words include typos, ad-hoc abbreviations, phonetic substitutions and emoticons, which causes grief for text processing tools [12]. For instance, we trained Stanford NER to detect mobile phone mentions using our labeled data. Given a sentence "In that aspect, the Sony Xperia Z scores a brownie point.", "Sony Xperia Z" is extracted by Stanford NER as all the words in the sentence are formal words. However, given a sentence "I wanna know what problems users face on hox... wanna buy it 2morrow", although "hox" refers to "HTC ONE X", Stanford NER misses it as "hox" is an informal abbreviation. Ratinov *et al.* [13] propose to restore ill-formed words to their canonical lexical forms in standard English. For each ill-formed word, they build a candidate list storing its possible canonical lexical forms based on morphological and phonetic variation. The canonical lexical forms are further ranked based on the similarities between their context and the context of the ill-formed word. However, this work does not address the problem of NER.

2.1.2.1 Supervised Learning

Most of current works related to NER adopt supervised learning methods [14–17]. However, obtaining a considerable amount of labeled training data is time-consuming and expensive. Possible solutions of insufficient training data include transfer learning and semi-supervised learning. Transfer learning aims to train the model using formal text and then apply it to user-generated short text. However, as the nature of two types of text is quite different, transfer learning is especially difficult [14]. Semi-supervised learning makes use of unlabeled data for training, typically a small

number of labeled data with a large number of unlabeled data. It proves useful when labeled data is scarce and hard to construct while unlabeled data is abundant and easy to access [15].

Liu *et al.* [15] propose a semi-supervised learning framework to handle the insufficient training data problem. The framework consists of K-Nearest Neighbors (KNN) classification and CRF model. For each word in a tweet, KNN utilizes its text window with the word in the middle and forms bag-of-word features. The label of current word is assigned based on the global coarse evidence (the information encoded in all tweets). Then, CRF takes the outputs of KNN into account and utilizes orthographic features, lexical features and gazetteers related features. CRF refines the results generated by KNN using local information (the information encoded in a single tweet). A confident score is further calculated to measure the confidence of assigning labels to the words in a tweet. If the confident score is larger than presettled threshold, the tweet and the labels of words inside are stored into training dataset.

As the nature of user-generated short text is huge different with the nature of formal text, Ritter *et al.* [17] re-build the different components of Nature Language Process (NLP) including Part of Speech Tagging (T-POS), shallow parsing (T-CHUNK), capitalization classification (T-CAP). Each component captures the specific nature of twitter data. A tweet-based NER system called T-NER is designed which utilizes the output of each component. Specifically, T-POS adopts both Penn TreeBank tag set and Twitter specific tag set including retweets, @usernames, #hashtags and urls. CRF is trained using POS dictionaries, Brown clusters, spelling and contextual features. Similarly, T-CHUNK is trained by CRF. In addition to general features used in [18], they take the output of T-POS into account. The purpose of T-CAP is to determine whether a capitalization in a tweet is informative or not. T-NER is separated into two tasks: named entity segmentation (T-SEG) and named entity classification (T-CLASS). CRF is utilized in T-SEG with a number of features including orthographic, contextual and lexical features, and the outputs of T-POS , T-CHUNK, and T-CAP. T-CLASS is implemented based on Labled-LDA [19] which utilizes the external knowledge base Freebase[1] and the redundancy inherent in tweets.

[1] `http://www.freebase.com/`

2.1.2.2 Unsupervised Learning

Compared to supervised learning methods, unsupervised learning methods do not require training data. However, one of the major drawbacks is that unsupervised learning methods are not able to classify detected named entities to the corresponding classes such as PERSON, LOCATION, ORGANIZATION and PRODUCT. Thus, they actually solve a sub-problem in NER which detects named entities in all types. There is only few works utilizing the unsupervised learning methods.

Li *et al.* [20] present an unsupervised NER system for tweets, called TwiNER. The system is divided into two steps. In the first step, each tweet is divided into a sequence of consecutive segments which contains one or more words. Each segment is calculated a stickiness score. If the score is high, it is not suitable to further split current segment as it may break a named entity into different parts. The stickiness score is measured by Symmetric Conditional Probability (SCP) [21] where the conditional probabilities used are supplied by Microsoft Web N-Gram Services[2]. Wikipedia[3] is further used to refine the score as a segment has a high chance to be a valid named entity if it appears in some specified positions in Wikipedia pages. For each tweet, they use dynamic programming algorithm to divide it into segments by maximising the overall stickiness score. A graph containing all the segments in tweet set is constructed. Random walk model then applied on graph which captures the relationship between segments based on the current tweet set. Finally, the segments are ranked based on the confident scores generated by random walk model and top segments are retained as named entities.

In [22], Li *et al.* further refine the performance of segmentation. At first, a number of state-of-the-art NER systems are used to detect named entities (segments) in tweets. Each segment is assigned with a probability to indicate in which degree the segmentation is correct. In the iteration process, the stickiness score of each segment is calculated based on global knowledge and local contextual knowledge. For global knowledge, they again adopt Microsoft Web N-Gram Services and refine the scores using Wikipedia. For local contextual knowledge, they utilize the confident probabilities of previously detected segments. In each iteration, the segments with high stickiness score are extracted and used to guide the next iteration process.

[2] `http://web-ngram.research.microsoft.com/info/`

[3] `http://dumps.wikimedia.org/enwiki/`

2.1.3 NER in Other Domains

As training data of user-generated short text is hard to obtain, Finin *et al.* [16] investigate the use of Amazon Mechnaical Turk and CrowFlower in order to label a large amount of data at a relatively low price. They further design approaches to judge the quality of the human work. Benson *et al.* [23] develop a graphical model to extract artists and venues related to musical performances from tweets and cluster tweets according to their corresponding events. Singh *et al.* [24] extract named entities in text advertisements. They propose a semi-Markov conditional random field model. The model is built based on a set of unlabeled advertisements with domain specific constraints.

NER tasks on queries have also attracted researcher interests. As reported in [25], about 71% of search queries contain named entities and less than 1% of the queries contain two or more named entities. Thus, extracting named entities in queries helps search engineers to better understand their semantics. However, search query is too short to generate useful information. Thus, poor results are obtained if traditional NER systems work on search queries directly. Given a query, Alasiry *et al.* [26] retrieve a number of documents using the search engineer. The words surrounding the query in documents provide enough contextual information. Grammar annotation and query segmentation are further conducted based on the documents to accurately identify the boundaries of named entities.

Biomedical NER represents another line of active research. Yoshida *et al.* [27] utilize lexical features, orthographic features, semantic features and syntactic features, such as part-of-speech and shallow parsing. These features are used by a reranker method which is based on a log-linear classifier to detect proteins, genes, cell lines, DNAs, cell types, and RNAs. Wang [28] introduces NER on clinical notes. A data set is manually annotated which contains eleven concept categories related to clinical concepts. Orthographic features, lexical features and semantic features are used in CRF.

2.1.4 Sequence Classification Methods

Even though a number of techniques have been proposed, statistical hidden state sequence models, such as HMM and CRF are still most widely used for NER tasks

nowadays. For any sentence $x_1, \ldots, x_n$ where $x_i \in \mathcal{V}$ (the set of all words) for $i = 1 \ldots n$ and any tag sequence $y_1 \ldots y_n$ where $y_i \in \mathcal{L}$ (the set of all tags) for $i = 1 \ldots n$, HMM models the joint probability of the sentence and tag sequence.

$$p(x_1 \ldots x_n, y_1 \ldots y_n) = \prod_{i=1}^{n+1} q(y_i|y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i|y_i) \tag{2.1}$$

The model is a Trigram HMM. $q(y_i|y_{i-2}, y_{i-1})$ is the probability of current word's label being y_i given the labels of previous two words in the sentence. $e(x_i|y_i)$ is the probability of tag y_i generating the word x_i. $q(y_i|y_{i-2}, y_{i-1})$ and $e(x_i|y_i)$ are obtained from training dataset. As for assigning the tag to each word in the sentence, the tag sequence which maximizes the joint probability is selected. The intuitive method is to conduct brute force search which calculates the joint probabilities with all the tag sequences and selects the tag sequence with maximal value. However, in this way, the time complexity is $O(|L|^n)$ which is the number of tags with the power n as the number of words in a sentence. Thus, it becomes hopeless inefficient to do brute force search. The approach to solve this problem is Viterbi Algorithm which is a dynamic programming algorithm. At each position i in the sentence, Viterbi Algorithm maintains a dynamic programming table $\pi(i, u, v)$ which records the maximal probability of a tag sequence ending in tags u, v at position i. Thus, when calculates the probabilities of tag sequences including the next position, Viterbi Algorithm only uses the values stored in the dynamic programming table in the previous step rather than calculate the probabilities of whole tag sequences. With this method, the time complexity drops to $O(n|L|^3)$.

The biggest drawback of HMM is that the model is fixed. It is impossible to plug additional information into the model, which causes the limitation to use the model in various applications. The reason behind it is that HMM is a generative model which calculates the joint probability $p(\mathbf{x}, \mathbf{y})$. The joint probability includes the model of $p(\mathbf{x})$. It is difficult to calculate when many highly dependent features are involved. For example, in order to model the joint probability tractably, HMM assumes that each observation variable x_i depends only on the current state y_i. However, this assumption may ignore other useful information.

Compared to HMM, CRF is a discriminative model based on a model of the conditional probability.

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y},\mathbf{x})}{\sum_{\mathbf{y}} p(\mathbf{y},\mathbf{x})} = \frac{\exp\{\sum_{i=1}^{n}\sum_{k=1}^{K}\lambda_k f_k(y_i, y_{i-1}, x_i)\}}{\sum_{\mathbf{y}}\exp\{\sum_{i=1}^{n}\sum_{k=1}^{K}\lambda_k f_k(y_i, y_{i-1}, x_i)\}} \tag{2.2}$$

The main difference is that conditional probability does not include a model of $p(x)$, which is not needed for classification anyway. Thus, a number of feature functions $f_k(y_i, y_{i-1}, x_i)$ can be assigned to CRF without considering the inner relationship among themselves. The feature functions are assigned by users which capture the information of a specific task more completely and accurately. In order to learn feature weights λ_k in the formula, the conditional probability is maximized given training data. All the sentences in training data are used and log likelihood is taken.

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{x}) &= \sum_{m=1}^{M}\sum_{i=1}^{n}\sum_{k=1}^{K}\lambda_k f_k(y_i^{(m)}, y_{i-1}^{(m)}, x_i^{(m)}) \\ &- \sum_{m=1}^{M} log\sum_{\mathbf{y}}\exp\{\sum_{i=1}^{n}\sum_{k=1}^{K}\lambda_k f_k(y_i, y_{i-1}, x_i)\})\end{aligned} \tag{2.3}$$

As a large number of parameters are in the formula, penalty function needs to be added in order to prevent overfitting.

$$\begin{aligned}l(\theta) &= \sum_{m=1}^{M}\sum_{i=1}^{n}\sum_{k=1}^{K}\lambda_k f_k(y_i^{(m)}, y_{i-1}^{(m)}, x_i^{(m)}) \\ &- \sum_{m=1}^{M} log\sum_{\mathbf{y}}\exp\{\sum_{i=1}^{n}\sum_{k=1}^{K}\lambda_k f_k(y_i, y_{i-1}, x_i)\}) - \sum_{k=1}^{K}\frac{\lambda_k^2}{2\sigma^2}\end{aligned} \tag{2.4}$$

Where θ represents the feature weights which should be learned from training data. As the function $l(\theta)$ is concave, Stochastic Gradient Descent (SGD) is used as the optimization method for maximizing $l(\theta)$. At each iteration, $l(\theta)$ is steepest ascent along the gradient. When the function reaches to the global maximum, the values of feature weights θ at the point are stored. If a new sentence comes in, a dynamic programming algorithm which is similar to Viterbi algorithm for HMM is conducted to find the optimal label sequence.

2.2 Named Entity Normalization

The task of Named Entity Normalization (NEN) which is also known as Named Entity Disambiguation is to map a detected name mention to the real world entity it refers to. A knowledge base like Wikipedia is commonly used to represent a real world entity. In Wikipedia, each article (or page) represents a real world entity and consists of hyperlinks to other articles. A name mention is used to represent the surface form of the hyperlink. However, a name mention may refer to different real word entities based on the context. Thus, disambiguation pages are specifically created for ambiguous mentions, and consist of links to articles defining the different real world entities. On the other side, a real world entity may be referred by different name mentions. Thus, redirect pages are created. A redirect page stores all the name mentions that may be used to refer to a real world entity.

A number of applications could benefit from this task such as automatic construction of a semantic web [29] and information retrieval systems [30]. Most of studies have been done for NEN on Web documents while a number of studies focus on NEN on user-generated short text like tweets.

2.2.1 NEN on Web Documents

The existing works can roughly be divided into two categories [31]. The first category mainly includes methods that map a name mention to a real world entity at each time. The methods in second category map a set of related name mentions to corresponding real world entities simultaneously.

The methods in first category mainly consider the similarity between a mention-entity pair. In [29], Mihalcea *et al.* build a system called Wikify!. Two approaches are introduced to map name mentions to their corresponding Wikipedia pages. The first method measures the mention-entity similarity by calculating contextual overlap between current paragraph with the name mention inside and candidate Wikipedia pages. The candidate Wikipedia pages are obtained through the links in the disambiguation page of the name mention. The second method extracts features from the name mention and its surrounding words, and compares it to the training examples

obtained from the entire Wikipedia. Naive Bayes classifier is used to predict which Wikipedia page the name mention should be mapped to.

Milne *et al.* [32] also train a classifier based on the surrounding contextual information. As a name mention may refer to different Wikipedia pages, they measure the commonness of a name mention and a Wikipedia page at first. The commonness between a name mention and a Wikipedia page is defined by the number of times the name mention is used as an anchor that links to the Wikipedia page. They further assign the weights to contextual terms. They assume that the term that is always used as a link when it appears in Wikipedia pages is important. Furthermore, if the Wikipedia page that a contextual term maps to has similar semantic meaning with the Wikipedia page that the name mention maps to, the contextual term earns more weight. A classifier that utilizes these features is designed.

Compared to the methods in the first category, the methods in second category leverage the global coherence between entities which can improve the NEN accuracy. Shen *et al.* [33] propose a NEN framework named LINDEN which processes one document at a time. For each name mention in a document, a candidate entity set is generated at first. Contextual concepts in the document are further detected by Wikipedia-Miner[4]. As both candidate entities and contextual concepts are Wikipedia pages, they are connected based on the hyperlinks and taxonomic relations supported by YAGO. Thus, a semantic network of the document is constructed. Semantic associativity based on Wikipedia Link-based Measure [34], semantic similarity based on the structure of YAGO ontology [35], and global coherence are calculated for a candidate entity by averaging the values between it and connected contextual concepts. Lastly, Support Vector Machine takes the values of a candidate entity as features to predict its score. The candidate entities in each candidate entity set are ranked based on the scores.

Han *et al.* [36] introduce a graph-based representation, called Referent Graph. The graph consists of all name mentions in a document and candidate entities associated with them. Two types of relationship between nodes in Graph are calculated. The first type models the likelihood of a name mention referring to a entity. The second type measures the semantic relation between entities based on Wikipedia

[4] `http://wikipedia-miner.cms.waikato.ac.nz/`

Link-based Measure. A prior knowledge is further calculated to measure the importance of a name mention to a document based on TF-IDF. Random Walk is utilized to infer the referent entities of all name mentions in a document.

2.2.2 NEN on User-Generated Short Text

Similar to NER, the performance of NEN is beset by the characteristics of user-generated short text.

Meij *et al.* [37] generate a list of candidate entities for each n-gram in a tweet at first. The candidate entities in a list are further ranked. They compare three families of approaches including lexical matching [38, 39], language modeling [40], and other state-of-the-art methods [32, 41]. Then, they apply supervised techniques to determine which of the candidate concepts should keep in a list. By removing a number of noises in a list, the precision is enhanced. They present a huge number of features that can be used. These features are belonged to four categories. *N-Gram Features* are based on the information from an n-gram. *Concept Features* adopt the knowledge from Wikipedia pages. *N-Gram + Concept Features* utilize the relation between an n-gram and an entity. *Tweet Features* consider the information related to the tweet. Different supervised learning methods with different combination of features are compared.

Liu *et al.* [31] propose a collective inference method which maps a set of name mentions from multiple tweets with entities simultaneously. Three types of similarities are measured including mention-entity similarity, entity-entity similarity and mention-mention similarity. Specifically, mention-entity similarity measures the similarity between a mention and an entity. Entity-entity similarity is calculated between two entities based on their associated Wikipedia pages. The most important and novel similarity is mention-mention similarity which capture the relationship between two mentions. A collective inference method is designed to utilize three similarities. Based on the mention-mention similarity, OOV (Out of Vocabulary) mentions are able to associate with the corresponding entities in Wikipedia.

2.3 Consumer PRODucts Contest

In 2012, International Conference on Data Mining (ICDM)[5] organized a Consumer PRODucts Contest[6]. Given a hundreds of thousands textual content found in web pages and web forum posting pages, a product catelog with over fifteen million formal names of products, and hundreds of manually labeled product mentions in textual content which are mapped to the formal names of products they refer to, the goal of the contest is to determine the state-of-the-art methods to automatically recognize product mentions in the textual content and map them to the correct formal names of products in the product catelog they refer to. The task of Consumer PRODucts Contest is similar to the task we aim to address in this thesis. Being a similar problem, it also shows that our research problem is recognized as an important one.

Most of the solutions are rule-based [42, 43]. However, the rules are usually heavily custom tailored to the given data set which cannot be extended. Wu *et al.* [44] achieve the best performance in the contest. In their solution, name mentions are extracted by three models. The first model simply stores the manually labeled product names in the training data and conducts string match on the test data. The second model is the rule-based model which is designed by the authors' observations and assumptions. The third model adopts CRF. The results of three models are further combined. For name normalization, all product names in the product catelog which contain a detected name mention are first retrieved. Words in the retrieved product names form a word set. The name mention is extended to a product name using words in this set and then mapped to the product catalog.

While the contest considers many different kinds of products, we focus on a specific product (*e.g.,* mobile phone). Different from [44] and other submissions to the contest, instead of recognizing product names directly from forum messages, we generate candidate phone names and then use CRF-based classifier to predict whether a candidate name mention is a mobile phone name. For name normalization, we use lexical rules to directly map a candidate name to its formal name.

[5] http://icdm2012.ua.ac.be/

[6] http://www.kaggle.com/c/cprod1

2.4 Summary

For NER and NEN tasks, even though a number of approaches have been proposed, they are beset by some fundamental problems.

For NER tasks using supervised learning methods, large amount of training data is essential. However, obtaining training data is time-consuming and costly. In our work, we propose a semi-automatic approach to generating training examples for constructing CRF-classifiers. A large number of training examples are obtained with little manual effort. Furthermore, most of named entity recognition methods run on sentences directly with a number of features. The named entity recognizer conducts two sub-tasks. The first task is to extract text chunks in sentences. The second task is to identify which categories the extracted text chunks belong to. Both tasks may involve mistakes and adversely affect the overall accuracy. In our work, we generate candidate names. Each candidate name is considered as one token in the sentence. This enables us to take surrounding words of a candidate name to be its context in a more natural manner, which gives us better result in return.

For NEN tasks, a knowledge base like Wikipedia is often necessary. However, a knowledge base may contain only a small number of real world entities in some fields. For example, in our study, we found Wikipedia only holds a very small number of mobile phone formal names. The performance of existing NEN methods drops dramatically if the knowledge base is incomplete. In our approach, we use rule-based approach to map the name variations.

Chapter 3

Mobile Phone Name Recognition

In this chapter, we first define our research problem and then give an overview of the proposed GREN method. Candidate Name Generator and CRF-based Name Recognizer used to conduct mobile phone mention recognition are further presented in detail.

3.1 Problem Definition

Let $\mathcal{T}$ be a thread in a discussion board relevant to mobile phones in an Internet forum. Let $S \in \mathcal{T}$ be a sentence from a post message in $\mathcal{T}$. Our task is to recognize from S a span of words $s = \langle w_1 w_2 \dots w_n \rangle$ $(n \geq 1)$ that refers to a mobile phone and map s to its formal name f. The collection of formal (or official) names of mobile phones is provided as an input to our task. The notations used in this thesis are listed in Table 3.1.

Taking the first sentence in Table 1.2 as an example, our task is to recognize mobile phone name mentions *Desire*, *X10*, and *HD2*, from the sentence and map these mentions to their formal names *HTC Desire*, *Sony Ericsson Xperia X10*, and *HTC HD2* respectively.

3.2 Overview of Gren Method

Figure 3.1 gives an overview of the proposed GREN method. Shown in the figure, candidate name generator takes two inputs: a collection of sentences in an Internet forum and a set of formal names of mobile phones. The output of the candidate name

Table 3.1: Notations and semantics

$\mathcal{T}$	a thread in a discussion board
f	a formal name of a mobile phone
$\mathcal{L}^f$	list of name variations for a formal name f
c	a candidate mobile phone name
$\mathcal{C}$	collection of candidate names, $c \in \mathcal{C}$
$\mathcal{W}_b$	word cluster containing a phone brand word
$\mathcal{W}_m$	word cluster containing a word used to name a phone model

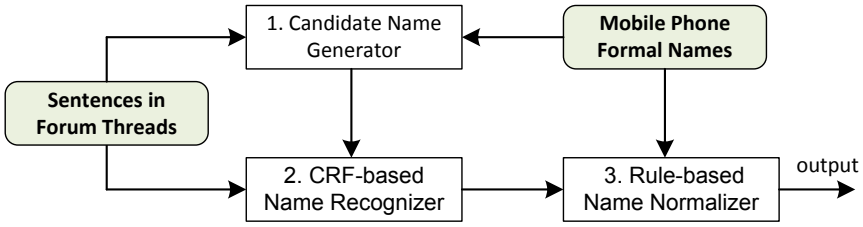


Figure 3.1: Overview of Gren

generator includes mobile phone name variations like the examples listed in Table 1.1, and phrases that are wrongly taken as candidate phone names. Examples of the latter include *software*, *samsung service center*, and *iphone price*. We denote the set of candidate names $\mathcal{C}$. We will detail the candidate name generator in Section 3.3 and explain why some phrases are wrongly included as candidate names.

Given a sentence, if a span of words in the sentence matches any candidate name $c \in \mathcal{C}$ (*i.e.,* a candidate name mention), then the name recognizer is utilized to predict whether the mention is truly a mobile phone name or not based on the current context. The prediction is made by a CRF-based classifier with an array of features detailed in Section 3.4.

If a mention is predicted to be a true mobile phone name, then the last component in Gren, name normalizer, maps the phone name to its formal name. For example, *X10* is normalized to *Sony Ericsson Xperia X10*. With name normalization, one can easily retrieve all sentences mentioning a specific phone, regardless of the mention is through which name variation of this phone. The name normalization is presented in Chapter 4.

Table 3.2: Example mobile phone formal names

Brand	Model Name	Model Number
Samsung	Galaxy SIII	I9300, I9305
Nokia	Lumia 920	–
Nokia	8210	–
Sony	Xperia S	LT26

3.3 Candidate Name Generation

A mobile phone formal name usually contains two components: *brand* and *model name*. Some mobile phones (about 22% in our dataset) have *model numbers* as the examples shown in Table 3.2. Note that a mobile phone may have more than one model number indicating small specification differences (*e.g.,* I9300 and I9305 for "Samsung Galaxy SIII"). A model number is often in the form of alphanumeric starting with a letter then followed by numerical digits, and is *independent* from a mobile phone name. That is, a mobile phone can be identified either by its *brand and model name* or by its *brand and model number*. For example, a model like "Samsung Galaxy SIII" can be identified without using its model number. Numbers like 920, 8210 shown in Table 3.2 are considered parts of model names but not model numbers. In the following discussion, we mainly focus on model names unless stated otherwise.

Figure 3.2 gives an overview of the candidate name generation process. We take an specific case of generating candidate names about mobile phones of "Samsung" as an example. Brown clustering is first adopted to group words with similar meaning and syntactical function together. The intuition behind it is that a word used to form a mobile phone name variation has similar context with a word used to form the formal name of this mobile phone. The result of Brown clustering is a number of clusters where each word is belonged to only one cluster. Based on the result of Brown clustering, the cluster containing word "Samsung" and candidate words used to form brand variations of "Samsung" is extracted. A number of rules are designed to obtain the brand variations of "Samsung" from the cluster. Similarly, the clusters containing words used to form model names of mobile phones of "Samsung" and candidate words used to form the variations of these model names are extracted.

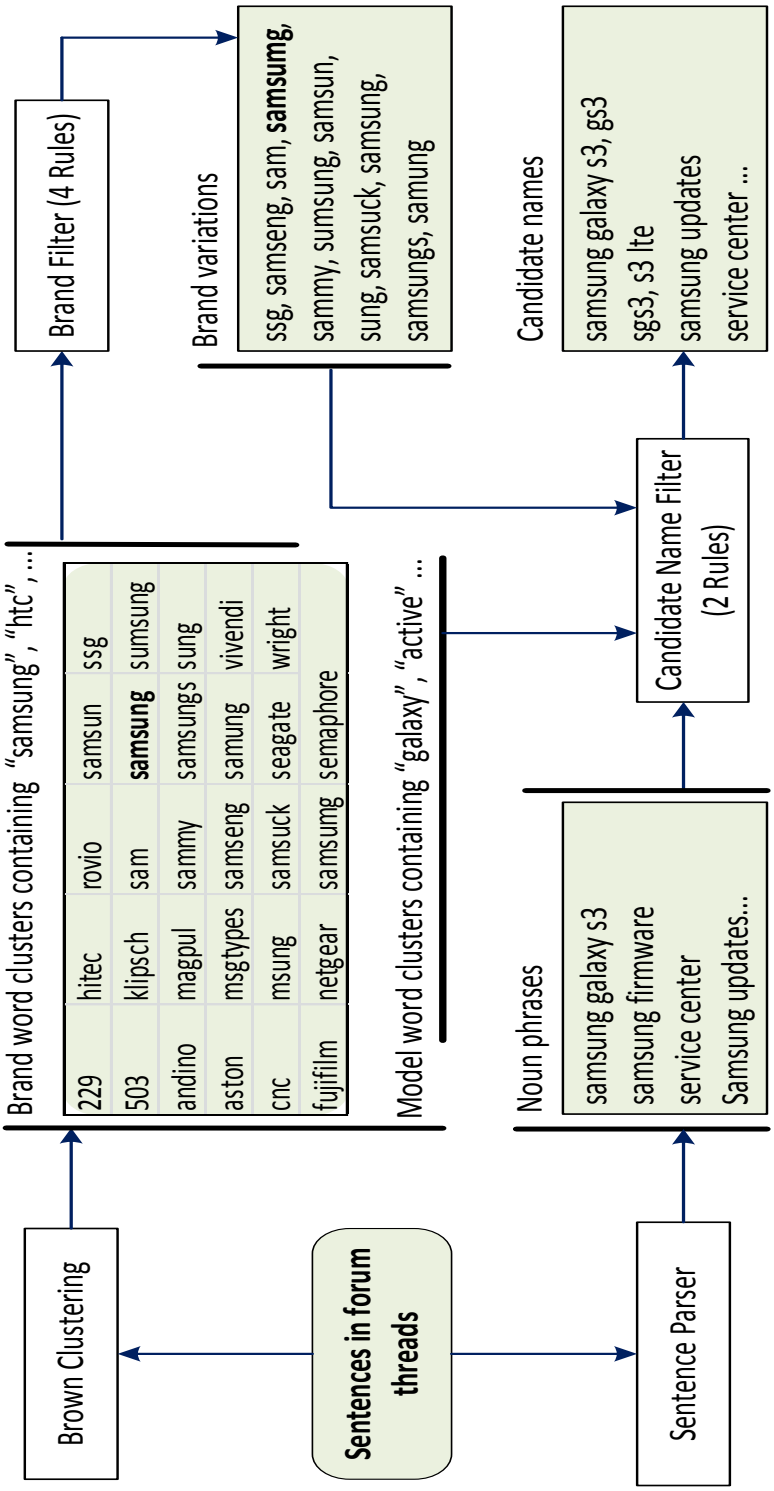


Figure 3.2: Overview of candidate name generation using brand "Samsung" as an example

Based on the extracted clusters, the noun phrases which may refer to mobile phones of "Samsung" are detected from sentences in the forum. However, as detected noun phrases consist of a large number of noises, we construct a candidate name filter based on the obtained brand variations of "Samsung" in order to generate candidate names of mobile phones of "Samsung" with a reasonable number of noises.

3.3.1 Brown Clustering

Brown Clustering is a hierarchical clustering algorithm which groups the words with similar meaning and syntactic function together [45]. The clustering is conducted by maximizing the mutual information of the bi-gram language model [46]. Assume $\mathcal{V}$ is the set of all words seen in the corpus $w_1, w_2 \ldots, w_t$, brown clustering algorithm measures the likelihood when two clusters are merged by calculating $p(w_1, w_2 \ldots, w_t)$ which is showed in Equation 3.1.

$$p(w_1, w_2 \ldots, w_t) = \prod_{i=1}^{n} e(x_i|G(x_i))q(G(x_i)|G(x_{i-1})) \tag{3.1}$$

where x_i is a word in a sentence, $G(x_i)$ is the cluster it belongs to and $G(x_{i-1})$ is the cluster of the previous word x_i belonging to. $e((x_i)|G(x_i))$ represents the probability of the generation of word x_i given the cluster $G(x_i)$ while $q(G(x_i)|G(x_{i-1})$ represents the probability of the appearance of current cluster given the cluster the previous word belongs. When calculating the probability, the log likelihood is usually used.

$$\begin{aligned} \log p(w_1, w_2 \ldots, w_t) &= \sum_{i=1}^{n} \log e(x_i|G(x_i))q(G(x_i)|G(x_{i-1})) \\ &\approx \sum_{g=1}^{k}\sum_{g'=1}^{k} p(g, g') \log \frac{p(g, g')}{p(g)p(g')} \end{aligned} \tag{3.2}$$

Equation 3.2 is used to measure the likelihood when two clusters are merged where g and g' represent the clusters. The number of clusters is assigned by k. $p(g, g')$ equals to $\frac{n(g,g')}{\sum_{g,g'} n(g,g')}$ while $p(g)$ equals to $\frac{n(g)}{\sum_g n(g)}$. $n(g)$ is the number of times cluster g occurs in the corpus and $n(g, g')$ is the number of times g' following g.

The intuitive idea is to first assign the number of clusters to the number of unique words in the corpus. At each time, two clusters are merged hypothetically and the value of equation 3.2 is calculated. The highest value is selected and the

corresponding two clusters are merged. However, this method is a greedy bottom up approach and the time complexity is $O(|\mathcal{V}^5|)$ where $\mathcal{V}$ is the number of unique words in the corpus. In order to boost the speed, top m most frequent words are selected and each word belongs to its own cluster. After two clusters being merged, a new word is picked and assigned to a new cluster. Two clusters are further chosen to be merged among the rest $m-2$ clusters, the cluster from emergence of original two clusters and the new cluster. In this approach, the time complexity is dropped to $O(|\mathcal{V}^2|m^2+n)$ where n is corpus length.

Given all sentences from relevant threads, by applying the Brown clustering, we obtain a collection of clusters. Each word belongs to exactly one word cluster. To avoid misspellings used by very few users, we only consider the words that appear at least 10 times in the dataset when conducting Brown clustering. As a word used to form a mobile phone name variation has similar context with a word used to form the formal name of this mobile phone, they have a high chance to be grouped together. On the other hand, most of clusters contain the words that are not related to mobile phones at all. These clusters can be considered as noises in the result. Thus, we need to extract useful clusters which contain the words that have high change been used to form the mobile phone name variations.

Given formal names F of all mobile phones, we store the words in brands into set F_b and words in model names into set F_m. If a cluster contains at least one word in F_b, we call it a *brand word cluster* while if a cluster contains at least one word in F_m, we call it a *model word cluster*. A specific situation is that a cluster may contain the words from both set F_b and F_m. In this case, the cluster is considered as both brand word cluster and model word cluster. The words in brand word clusters have high chance to be used to form brand variations while the words in model word clusters have high chance to be used to form model variations.

3.3.2 Brand Variations

It is obvious that noisy words exist in both brand word clusters and model word clusters. The noisy words are not elements used to form a mobile phone name variation. What is worse, they may hurt the performance when we further generate

candidate names of mobile phones. Compared to phone models, phone brands have three features which make it possible to remove the noises from brand word clusters.

(1) Compared to the number of different phone models, the number of brands is much smaller.

(2) The words used to form brands are usually unique and not even formal English words.

(3) Due to clear context in word usage, almost all variations of the same brand are grouped into the same brand word cluster through Brown clustering. An example brand word cluster containing brand "Samsung" is shown in Figure 3.2. The cluster has 29 words and many of them are spelling variations of "Samsung" and its nicknames (*e.g., sam* and *sammy*).

Because of the small number of phone brands with unusual words inside, complicated and target-specific rules can be designed. Furthermore, because of the relatively good quality of word grouping for mobile phone brands, the application range of rules is limited which leads to better results.

We apply 4 rules as *brand filter* to identify brand variations from a brand word cluster. Let $\mathcal{W}_b$ be the word cluster containing brand b. We consider a word $w \in \mathcal{W}_b$ is a brand variation of b if w satisfies one of the following 4 rules:

(i) The phonemic edit distance between w and b is 0. That is, the two words have the same pronunciation. The phonemic edit distance is measured by using Metaphone algorithm [47].

(ii) The first and the last characters in w and b are the same. The character comparison is case-insensitive.

(iii) The first three characters in w and b are the same. The character comparison is case-insensitive.

(iv) Brand b contains more than one upper-case character (*e.g.,* "BlackBerry") and the prefix of w matches all upper-case characters in b in sequence. The match is case-insensitive. For example "bb", "bbry", and "bberry" are variations of "BlackBerry" matched by this rule.

We assume that a brand formal name and its variations share common lexical characteristics. These common lexical characteristics are therefore utilized in designing our rules. As an example of the results generated by 4 rules, Figure 3.2 lists the 12 brand variations for "Samsung". Note that, if a brand contains more than one word such as "Sony Erricson", brand word clusters are extracted based on each word inside and brand variation are obtained by combining the matches from all the clusters.

3.3.3 Candidate Mobile Phone Names

Recall that a mobile phone formal name contains brand and model name. A model word cluster has high chance containing words used to form model variations. However, unlike phone brands, there are a large number of phone models. Besides, many words that are used to form model names are valid English words (*e.g.,* galaxy, active, one, desire) and some of them are widely used in very different contexts. A rule based approach to recognizing model variations from word clusters becomes less practical.

Based on the assumption that mobile phone name variations are noun phrases, we filter noun phrases in sentences to obtain candidate phone names. In order to focus on detecting noun phrases of mobile phones, we extract the noun phrases that satisfy one of following conditions.

- All the words in the phrase appear in model word clusters.
- The phrase is begun with the words in brand word clusters and is followed by at least one word in model word clusters. The words that are not in brand word clusters and model word clusters are not included in the phrase.

The reason we make above two conditions is that we believe all the words used to form the mobile phone name variations should be in the similar context as their formal names. Therefore, we rely on Brown clustering to group these words together.

We further make the assumption that a phrase is a candidate phone name only if the phrase contains a brand variation, or the phrase appears after a brand variation

at least once in the whole dataset. It is noticed that because of this assumption, only the sentences that contain at least one brand variation need to be parsed.

Shown in Figure 3.2, we capture most mobile phone name variations such as "sumsang galaxy s3" and the other names listed in Table 1.1. However, due to the noisy word usage contexts of model names like "one" and "active", the collection of candidate names contains a number of noises such as service center", "samsung updates".

3.4 CRF-based Name Recognition

We now have a set of candidate names that might be used to refer to mobile phones in a forum. Given a sentence mentioning at least one candidate name, our task is to predict whether this mention indeed refers to a specific mobile phone, *i.e.,* a binary classification task. Many classifiers have been developed in past decades and been used in text domain, *e.g.,* Naïve Bayes, K Nearest Neighbor, Support Vector Machines etc. [48]. However, most of these classifiers cannot utilize the local contextual information in a sentence that is useful for the prediction in our task. In this study, we adopt linear-chain CRF model.

3.4.1 Training Examples

CRF is a supervised classification method which needs to learn the weight of each feature function through training examples. However, obtaining a reasonable number of high quality training examples is always a major challenge. In most classification tasks, training examples are manually annotated, which is costly and time consuming. Thus, before introducing the features used in our task, we first propose a semi-automatic manner with minimal manual labeling effort to generate training examples for our CRF model.

Before we create labeled sentences for CRF training, we create two sets of mobile phone names: a set of positive names ($\mathcal{P}$) and a set of negative names ($\mathcal{N}$). In our problem definition (see Section 3.1), a set of formal names is given as one input. Each formal name (brand and model name, without model number) is inserted into $\mathcal{P}$. If a formal name contains Roman number such as "ii" and "iii", a separate copy

of this formal name is inserted into $\mathcal{P}$ by mapping the Roman number into Arabic number. Similarly, if a formal name contains number in English like "one", the separate copy which English number is transferred into Arabic number is inserted into $\mathcal{P}$. To further enlarge the list of positive names, if a model name has more than one word (*e.g.,* "galaxy note"), then the model name is inserted into $\mathcal{P}$. For example, given a formal name "Samsung Galaxy SIII", the following four names are considered positive names and are inserted into $\mathcal{P}$: "Samsung Galaxy SIII", "Samsung Galaxy S3", "Galaxy SIII", and "Galaxy S3".

Populating the set of negative names $\mathcal{N}$, however, is through manual annotation. In Section 3.3, we get the set of candidate names $\mathcal{C}$ and we note that many of its entries are not true mobile phone names. A number of entries from $\mathcal{C}$ that can be easily identified not phone names are manually selected to form $\mathcal{N}$. Example entries in $\mathcal{N}$ include "debug", "service center", "retail prices", "toolbar", "firmware", "update". We argue that annotating words/phrases that are not phone names is much easier and less time-consuming compared to annotating sentences directly.

With sets $\mathcal{P}$ and $\mathcal{N}$, we select sentences satisfying the following two conditions to be training examples.

(i) The sentence contains at least one entity in either set $\mathcal{P}$ or set $\mathcal{N}$; and

(ii) The sentence does not contain any entry appears in $\mathcal{C}$ but not in $\mathcal{N}$ or $\mathcal{P}$ (*i.e.,* any entry in $\mathcal{C} \setminus (\mathcal{P} \cup \mathcal{N})$).

The first condition is set because when we generate training data, only the entities in $\mathcal{P}$ are considered as positive entities while only the entities in $\mathcal{N}$ are considered as negative entities.

The second condition is set because we assume that all the words that are not in the candidate set $\mathcal{C}$ are not mobile phone names. However, the entities in candidate set have high chances to be mobile phone names. As we do not label the entities that are not in set $\mathcal{P}$ and set $\mathcal{N}$, we would not expect they appear when we generate training data.

Table 3.3: Original and rewritten sentence with labels for "ip_5"

Original sentence:	Still	prefer	ip 5	then	note 2
Rewritten sentence:	Still	prefer	ip_5	then	note_2
"ip_5" (w_i):	w_{i-2}	w_{i-1}	w_i	w_{i+1}	w_{i+2}

Table 3.4: Feature description for lexical features (L_1, L_2), grammatical features (G_1, G_2), and name features (N_1, N_2)

L_1	The current word and its surrounding two words $w_{i-2}w_{i-1}w_iw_{i+1}w_{i+2}$, and their lower-cased forms.
L_2	Word surface feature of the current word: Initial capitalization, all capitalization, containing capitalized letters, all digits, containing digits and letters.
G_1	POS tagger of the current word and its surrounding two words.
G_2	Path prefixes of length 4, 6, 10, 20 (*i.e.,* maximum length) of the current word by Brown clustering.
N_1	Flags to indicate whether the current word and its surrounding two words are candidate phone names
N_2	The brand entropy of the current word and its surrounding two words.

3.4.2 Features for CRF Model

Because our task is to predict whether a candidate name mention truly refers to a mobile phone, we consider each candidate name "*a single token*" in the sentence. More specifically, we rewrite the sentence by adding underscores to the contained words in a candidate name, as shown in Table 3.3. The rewriting enables us to take surrounding words of a candidate name to be its context in a more natural manner.

In the example shown in Table 3.3, after rewriting, candidate names *ip_5* and *note_2* each becomes one word. We propose to use 6 features in our CRF model (listed in Table 3.4), mostly considering the current word w_i, and its surrounding two words on both sides: $w_{i-2}w_{i-1}$ to its left-hand side and $w_{i+1}w_{i+2}$ to its right-hand side.

The two lexical features include the word and the surrounding words from w_{i-2} to w_{i+2} (L_1) and the surface forms of the word (L_2). The two grammatical features include the POS taggers of the 5 words from w_{i-2} to w_{i+2} (G_1) and the path prefixes obtained from Brown clustering (G_2). Note that, the path prefixes of the words are obtained by running Brown clustering on the rewritten sentences. The resultant word clusters therefore are different from the word clusters obtained for candidate

name generation in Section 3.3. We argue that by rewriting each candidate name mention to a single word, word usage context patterns, particularly the usage patterns surrounding candidate names, can be better captured by Brown clustering algorithm on the rewritten sentences than the original sentences.

For the candidate name features, the first feature (N_1) is to flag the candidate names. The corresponding feature is assigned a value of 1 if the word is rewritten from a candidate name (*e.g., ip_5*) and assigned 0 otherwise. The last feature (N_2) is the *brand entropy* of a candidate name. Recall that in candidate name generation (Section 3.3.3), a candidate name either contains a brand variation or appears after a brand variation at least once. By intuition, a mobile phone name should appear after only one brand (or its variations) but not appear after many different brands. If a candidate name appears after many different brands, then it is unlikely a true phone name. The brand entropy feature is computed as following:

- If a word in a sentence is not a candidate name, then feature N_2 is set to 0.
- If a word is a candidate name (*i.e.,* the rewritten form) that contains a brand variation, then feature N_2 is set to 1.
- If a word is a candidate name that does not contain a brand variation, then we compute the probability of this candidate name appears after a brand (or its variation). Let m be the number of times a candidate name c appears after any brand variation, and m_b be the number of times c appears after a brand variation of brand b. Then the probability of c following brand b is $P(c, b) = \frac{m_b}{m}$. Let B be the set of brands that c appears after at least once, then the brand entropy of c is $H(c) = -\sum_{b \in B} P(c, b) \log P(c, b)$. If a candidate name, for example "galaxy siii", appears only after one brand "Samsung", then $H(c) = 0$. If a candidate name (*e.g.,* "software", "phone", or "battery") appears after many different brands, then $H(c)$ is a much larger value. The value of feature N_2 is set to 1 if $0 \leq H(c) \leq 1$ and set to 2 if $H(c) > 1$.

3.4.3 Sentence Labeling

CRF model is flexible, allowing different kinds of labeling schemes. In our proposed solution, we use "YNO" (Yes-No-Out) scheme. Given a rewritten sentence as a

training example, if a word is rewritten from a positive name in $\mathcal{P}$, then the word is labeled "Y"; if a word is rewritten from a negative name in $\mathcal{N}$, the word is labeled "N". The remaining words in the sentence are labeled "O".

Chapter 4

Mobile Phone Name Normalization

After we obtain the CRF classifier, if a name mention is recognized as a phone name variation, our next task is to map this name variation to its formal name. The last component of GREN (Rule-based Name Normalizer) is used for this purpose. In this chapter, a detailed description about rule-based name normalizer is presented.

4.1 Overview of Rule-based Name Normalizer

Because of the way we train our CRF classifier, most phone name variations detected are originated from the candidate name set $\mathcal{C}$. That means we can pre-normalize candidate names in $\mathcal{C}$ to their corresponding formal names with some degree of noises because not all candidate names are true mobile phone names.

The overview of rule-based name normalizer is showed in Figure 4.1. Specifically, it contains two components named matchable candidate name generator and disambiguator. In terms of matchable candidate name generator, it takes two inputs: candidate name set $\mathcal{C}$ and formal names of mobile phones with their model numbers if exist. A number of rules are designed to extract matchable candidate names from the candidate name set $\mathcal{C}$ and link each matchable candidate name to the formal names of mobile phones it may refer to. A matchable candidate name is the candidate name that may refer to a mobile phone. On the other hand, if a candidate name is not selected as a matchable candidate name, it will not be mapped to the mobile phone formal name even though it is recognized by the CRF classifier. As

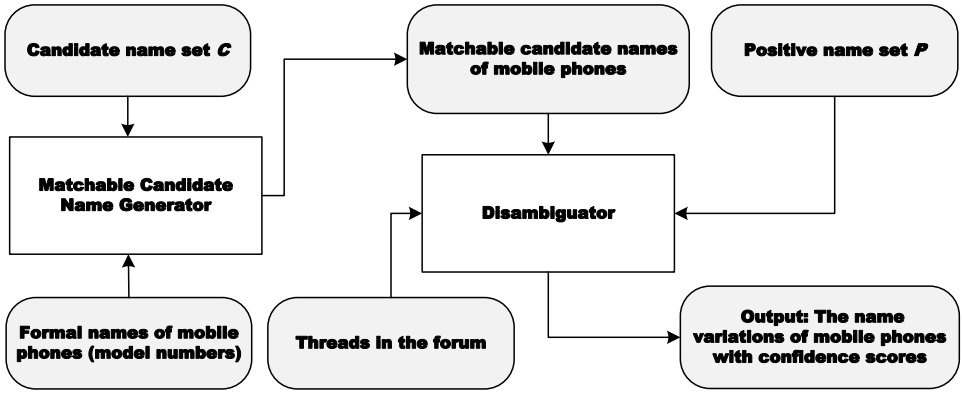


Figure 4.1: Overview of Rule-based Name Normalizer

matchable candidate name generator may link a matchable candidate name to multiple formal names, disambiguation needs to be conducted. Disambiguator takes the results obtained from matchable candidate name generator, the threads in the forum and positive name set $\mathcal{P}$ generated in Section 3.4, and calculates the confidence score of each matchable candidate name to each formal name it is linked to. The matchable candidate name is mapped to the formal name which holds the highest confidence score. As most processes are conducted by rules, we name our approach rule-based name generator. The output of rule-based name generator is the lists of mobile phones where each list contains the mobile phone possible variations from $\mathcal{C}$ ranked by their confidence scores.

Algorithm 1 outlines our algorithm for rule-based name normalizer. For each formal name f containing brand f_b, model name f_m, and model number f_r, we build a list of its possible variations from $\mathcal{C}$ with confidence scores, denoted by $\mathcal{L}^f$. A matchable candidate name in list $\mathcal{L}^f$, if detected to be true by the CRF classifier, is normalized to formal name f. The algorithm can be divided into three steps. Specifically, step 1 (lines 2-6) and step 2 (lines 7-10) form matchable candidate name generator while disambiguation is conducted in step 3 (lines 11-14). Figure 4.2 gives an demonstration on how the rule-based name normalizer works using "Samsung Galaxy SIII" as an example. We give a detailed description of each step in the algorithm and use the demonstration to give a straightforward illustration.

Input: f, a formal name consisting of brand f_b, model name f_m, and model number f_r;
$\mathcal{C}$, set of candidate names
Output: $\mathcal{L}^f$, list of name variations of f

```
1  L^f ← Φ;
2  forall the c ∈ C do
3      if sequenceContainment(f_b, f_m, c) then
4          L^f ← L^f ∪ {c};
5      if containsModelNumber(c, f_r) then
6          L^f ← L^f ∪ {c};
7  forall the f' ∈ L^f do
8      forall the c ∈ C \ L^f do
9          if wordMatch(c, f') then
10             L^f ← L^f ∪ {c};
11 forall the f' ∈ L^f do
12     f'.conf ←confidenceScore(f', f);
13     if f'.conf == 0 then
14         remove f' from L^f;
15 return L^f;
```

Algorithm 1: Name variation generation for formal name f

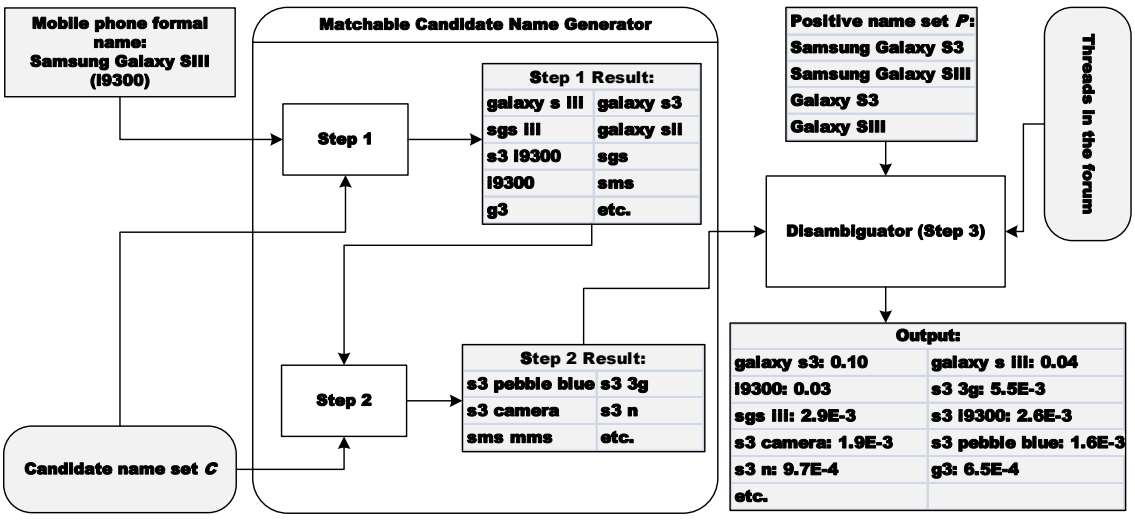


Figure 4.2: Rule-based Name Normalizer using "Samsung Galaxy SIII" as an example

4.2 Matchable Candidate Name Generator

Matchable candidate names are generated and linked to the formal names of mobile phones they may refer to by step 1 and step 2. As only a matchable candidate name can be mapped to a mobile phone formal name, matchable candidate names that cover a large number of name variations with some degree of noises from the candidate name set $\mathcal{C}$ are expected.

Step 1 (Lines 2 - 6): Given a candidate name c, if all its characters are contained in the brand and model name of the formal name and are arranged in the same sequence, then we consider c is a matchable candidate name of f (Lines 3 and 4). The match is case-insensitive, and Roman and English numbers match Arabic numbers. For example, all characters in candidate name "sgs iii" are contained in "Samsung Galaxy SIII" and are in the same sequence. As the result, *sequenceContainment* function returns true for "sgs iii", "galaxy s3", and "galaxy s iii" etc. If a candidate name c contains the model number of f, then the candidate name is also added to $\mathcal{L}^f$, leading to the addition of "i9300" and "s3 i9300" as examples. If a phone formal name does not contain a model number, then the *containsModelNumber* function (Lines 5 and 6) is skipped. Figure 4.2 shows the part of results generated by step 1. It is noticed that this step may generate some noises such as "g3", "sms", and "sgs". On the other hand, due to the strict rules, a number of name variations may be ignored. For example, if a name variation contains the letters that are not appeared in a formal name, it will not be considered as a matchable candidate name of the mobile phone.

Step 2 (Lines 7 - 10): We get the initial set of matchable candidate names in $\mathcal{L}^f$ through step 1. However, as a number of name variations may be ignored, we enlarge the number of matchable candidate names by learning from the current set of matchable candidate names on how users name this phone. For this purpose, we tokenize all names currently in $\mathcal{L}^f$ to get all words people use to refer to this phone. The brand variations and single-character words are ignored. Example words obtained include "s3", "galaxy", and "sgs". Then, if any word in a candidate name that is currently not in $\mathcal{L}^f$ matches one of these words, the candidate name is added

to $\mathcal{L}^f$ (Lines 7 - 10). With this step, we get matchable candidate names such as "s3 phone", "s3 3g" and "s3 pebble blue" listed in figure 4.2.

Through the above two steps, each formal name f is now associated with a list of matchable candidate names stored in $\mathcal{L}^f$. However, one matchable candidate name may appear in multiple $\mathcal{L}^f$'s. For example "sgs ii" appears in the list of "Samsung Galaxy SII" and the list of "Samsung Galaxy SIII" because function *sequenceContainment* returns true for both formal names. Then which is the correct normalization? Disambiguator is used to address this issue.

4.3 Disambiguator

An Internet forum contains a large number of threads. Each thread has a title and a description (*i.e.,* the first post message) and the replying posts. Recall that in Section 3.4, to train the CRF classifier, for each formal name, we generate a few positive names (*e.g.,* "Samsung Galaxy SIII" leads to the generation of "Samsung Galaxy SIII", "Samsung Galaxy S3", "Galaxy SIII", "Galaxy S3"). To compute the confidence score, for each formal name, we obtain all the threads such that the title and description of the thread contain only the formal name or any positive names derived from it but not any other matchable candidate names. We assume such thread has clear topical focus on the phone with the given formal name. Accordingly, when users post messages in this thread, name variations of this phone have higher chance to surface. We count the frequency of matchable candidate names that appear in all these threads obtained for a formal name. The confidence score of a matchable candidate name is the relative frequency with respect to the most frequent matchable candidate names in these threads (Lines 11 and 12). If a matchable candidate name has 0 confidence score, meaning that it never appears in any discussion threads specific to the phone model in the whole forum, then the matchable candidate name is removed from $\mathcal{L}^f$ (Lines 13 - 14).

After computing the confidence score, if a matchable candidate name appears in multiple $\mathcal{L}^f$'s, only the instance with the highest confidence score is kept in the corresponding $\mathcal{L}^f$. Figure 4.2 gives the part of outputs of "Samsung Galaxy SIII". It shows all the matchable candidate names listed in the results of step 1 and step 2

with their confidence scores. If a matchable candidate name is in the result of step 1 or step 2, but does not appear in the output, it means it has been removed based on the confidence score. For example, in the figure, we can see that "sgs ii" which is generated by step 1 has been removed in the output as it has the highest confidence score with "Samsung Galaxy SII". The noises which are not the name variations of "Samsung Galaxy SIII" hold relatively low confidence scores.

4.4 Remark

The matchable candidate name that is assigned to a formal name might not be a true mobile phone name variation because of the noises in candidate name generation. Only the names that are predicted true by the CRF classifier will trigger the normalization rule. On the other hand, even though a candidate name is recognized by the CRF classifier, it will not be considered as a name variation if it is not selected by the matchable candidate name generator. The reason is that the CRF classifier may predict wrongly and the detected candidate name is not a name variation of any mobile phones.

Chapter 5

Experimental Evaluation

In this chapter, we evaluate the proposed GREN method and compare it with baseline methods. We further study the results obtained by the different components in the GREN method.

5.1 Dataset

Forum Data. We collected forum data from the discussion board titled "Mobile Communication Technology" in HardwareZone forum, probably the most popular forum in Singapore.[1] Most discussions in this board are about mobile phones and a few user interest groups are formed for a few brands: BlackBerry, Nokia, Samsung, Motorola, Sony, LG, HTC, and Apple's iPhone. In total, 25,251 discussion threads were collected, containing 1,026,190 post messages. The time duration of the discussion is from March 15, 2002 to May 2, 2013 in our data collection.

Mobile Phone Formal Names. The formal names of mobile phones were crawled from GSMArena.com.[2] GSMArena lists all phone models released by all phone makers in the global market. However, not all phone markers released their products in Singapore. Hence, we only consider the mobile phone models from the 8 brands where user interest groups are found in HarewareZone: BlackBerry, Nokia, Samsung, Motorola, Sony (Sony Ericsson), LG, HTC, and Apple's iPhone. For these 8 brands, 2,623 formal mobile phone names were obtained from GSMArena. Again, only a

[1] http://forums.hardwarezone.com.sg/

[2] http://www.gsmarena.com/

Table 5.1: The 20 mobile phones, number of distinct name variations in threads ($\mathcal{T}$), and number of distinct name variations covered in normalization list ($\mathcal{L}^f$); Columns titled "Re ($\mathcal{L}^f$)" and "AP ($\mathcal{L}^f$)" report the recall and average precision of $\mathcal{L}^f$ respectively; The last column "AP($\mathcal{L}^f_r$)" reports the average precision of $\mathcal{L}^f_r$ after removing negative entries from $\mathcal{L}^f$ by the name recognizer.

Mobile Phone	$\mathcal{T}$	$\mathcal{L}^f$	Re($\mathcal{L}^f$)	AP($\mathcal{L}^f$)	AP($\mathcal{L}^f_r$)
apple iphone 4	3	3	1.000	0.643	1.000
apple iphone 4s	3	2	0.667	0.284	0.500
apple iphone 5	7	3	0.429	0.742	0.600
blackberry bold 9650	4	2	0.500	0.600	0.500
blackberry bold 9700	7	5	0.714	0.692	0.800
blackberry z10	4	3	0.750	0.800	1.000
htc desire hd	2	2	1.000	0.833	1.000
htc hd2	2	2	1.000	0.750	1.000
htc one x	4	3	0.750	0.549	1.000
lg nexus 4	3	3	1.000	0.700	1.000
motorola milestone 2	7	3	0.429	1.000	1.000
nokia e72	1	1	1.000	1.000	1.000
nokia lumia 900	5	4	0.800	0.714	1.000
nokia lumia 920	6	5	0.833	0.778	1.000
samsung galaxy note ii	8	7	0.875	0.666	1.000
samsung galaxy s iii	9	8	0.889	0.840	0.800
samsung omnia ii	4	3	0.750	0.875	1.000
sony ericsson xperia arc	2	2	1.000	0.361	1.000
sony xperia s	2	2	1.000	0.214	0.667
sony xperia z	3	3	1.000	0.507	1.000
Average	4.3	3.3	0.819	0.677	0.893

subset of these 2,623 phone models were released in Singapore and those released before March 2002 are not covered by our dataset.

Ground Truth Labeling. To evaluate the accuracy of name recognition and normalization, we select 20 mobile phones (the first column in Table 5.1) which are relatively popular in the forum. For each phone, we randomly selected a thread with about 100 post messages, then manually labeled the mobile phone name mentions in these posts and mapped to their formal names. In our manual annotation process, we observed that some of the names are used to refer to a series of phones. For example, in this sentence "I still prefer iphone than any android phones", "iphone" here does not refer to any specific model (*e.g.,* iPhone 4s or iPhone 5). That means we cannot even manually normalize this mention to any formal name and we ignore

Table 5.2: Brand variations for 8 brands

Brand	**Brand variation**
Apple, HTC, LG	–No brand variations–
Nokia	nokia, nokie, nk
BlackBerry	blackberry, bbry, blackbery, bb, bberry
Motorola	motorola, moto, motorolla, mot
Samsung	ssg, samseng, sam, samsumg, sammy, sumsung, samsun, sung, samsuck, samsung, samsungs, samung
Sony Erricson	sony erricson, sony ericsson, sony ericson, sony ericcson, sony-ericsson, sony ericssion, sn, sony, sonyeric

such cases in our evaluation. Names like "s3 lte" and "3g s3" were labeled positive because each name indeed refers to a specific phone model with emphasis on specification variants.

In total, we labeled 4,121 sentences within which there were 946 phone name mentions. The 946 name mentions include name variations for the 20 selected mobile phones as well as name variations of other phone models, because users often compare phones with many different models in their discussion. The number of distinct phone name variations for each of the 20 phones is listed in the second column in Table 5.1.

5.2 Implementation and Intermediate Results

Before we jump to the results for name recognition and normalization, we report the software packages used in Gren implementation, and the intermediate results obtained, *e.g.,* the number of candidate name obtained, the number of sentences used for training the CRF classifier etc..

Candidate Name Generation. For preprocessing, we adopted Stanford Tokenizer[3] for the sentence splitting task. Brown Clustering was conducted using Liang's code.[4] The number of clusters was set to 1000, ignoring the words that appeared fewer than 10 times in the dataset. On average, each resultant word cluster contains 32 words.

In terms of brand variation extraction, although simple and even seem to be ad-hoc, the four rules give surprisingly good results. For brands with easy spelling

[3] `http://nlp.stanford.edu/software/tokenizer.shtml`

[4] `https://github.com/percyliang/brown-cluster`

Table 5.3: Number of distinct name variations in the 20 manually labeled threads $\mathcal{T}$'s and in candidate name set $\mathcal{C}$, with user frequency of 10 and above

Mobile Phone	No. in $\mathcal{T}$'s	No. in $\mathcal{C}$	Recall
The 20 selected phones	74	67	0.905
Other phones mentioned	93	85	0.914
All phones in the 20 threads	167	152	0.910

(*e.g.,* Apple, HTC, and LG), no variations are found. More variations are obtained for brands with more than 5 characters (*e.g.,* BlackBerry, Motorola, Samsung, and Sony Erricson). Table 5.2 lists the brand variations for the 8 brands.

After obtaining the brand variations for the 8 brands, we generated the candidate phone names. The sentences were parsed by using Stanford Parser with careless English model.[5] In total, $4,258$ candidate names were obtained. Each of the $4,258$ candidate names has been used by at least 10 users. We consider a candidate name to be less interesting if it is used by very few users only. Many such candidate names are results of typos. The next question is: *does the set of candidate names cover all phone name variations?*

It is infeasible to directly evaluate the recall of $\mathcal{C}$ unless all the $1,026,190$ post messages are fully manually labeled. We therefore partially evaluate the recall of $\mathcal{C}$ using the phone variations obtained from the 20 manually labeled phones. In total, we have 86 distinct name variations for the 20 selected phones, listed in the second column in Table 5.1. Among them, 74 name variations each has been used by at least 10 users in the whole dataset. Because $\mathcal{C}$ only includes candidate names with user frequency equal or greater than 10, we compute the recall to be the ratio of these 74 name variations that are also included in $\mathcal{C}$, which is 0.905. Because users often compare different phones, we also get another 93 phone name variations (not for the 20 selected phones), each has user frequency not lower than 10. The recall for these 93 phone name variations is 0.914, computed in a similar manner. Table 5.3 summarizes the results. Overall, the recall of $\mathcal{C}$, evaluated using the 167 distinct phone name variations is 0.910, which is a very high value.

CRF-based Name Recognition. The CRF classifier for name recognition was implemented using the CRF++ toolkit.[6]

[5] `http://nlp.stanford.edu/downloads/lex-parser.shtml`

[6] `https://code.google.com/p/crfpp/`

We select the positive phone names and negative candidate names to generate training sentences. Using the 2,623 formal mobile phone names from GSMArena, we identified 412 positive names from the candidate name set $\mathcal{C}$. We then randomly selected 500 negative names from the remaining 3,846 names in $\mathcal{C}$, which were clearly not phone names. Following the rules stated in Section 3.4, we first selected 21,246 sentences each of which contained at least one positive name mention. In total, there were 22,884 positive phone name mentions. There were 250,644 sentences satisfying the condition with negative name mentions. However, using all these sentences leads to significant imbalance between positive and negative examples. We therefore randomly selected a subset of 11,826 sentences such that the total number of negative name mentions were the same as positive name mentions. In total, 33,072 sentences were used to construct the CRF-based name recognizer. Note that, in the process of generating the 33,072 training sentences, only the 500 negative names were selected manually from $\mathcal{C}$ with very low cost of manual annotation.

Regarding the CRF features (see Table 3.4), the POS tagger information was by Stanford POS Tagger with careless English model[7] and the Brown clustering was again by using Liang's code on the rewritten sentences. The performance of the CRF-based name recognizer will be reported in Section 5.3 in comparison with other baseline methods.

Rule-based Name Normalization. For each formal name f, we have a list of matchable candidate names $\mathcal{L}^f$. The matchable candidate names in $\mathcal{L}^f$ are originated from $\mathcal{C}$ that will be normalized to f if they are recognized to be true names by the CRF classifier. Now, we evaluate the precision and recall of $\mathcal{L}^f$ using the labeled data for the 20 selected phones.

In Table 5.1, the second column lists the number of distinct name variations for each of the 20 phones in the labeled data. Column 3 lists the number of distinct name variations that are included in the list $\mathcal{L}^f$ for each phone. The recall is computed by taking the ratio of the two numbers for each phone, reported in column 4. Observe that many phones get 100% recall and the overall recall average is 0.819. Note that this value is lower than the recall of $\mathcal{C}$ for the 20 phones (*i.e.,* 0.905) reported in

[7] `http://nlp.stanford.edu/downloads/tagger.shtml`

Table 5.3. One reason is that in computing the recall of $\mathcal{C}$ only name variations with user frequency not smaller than 10 are used. In Table 5.1, among the 86 distinct name variations for the 20 phones, 74 have user frequency equal or greater than 10. Because all matchable candidate names from $\mathcal{L}^f$ are originated from $\mathcal{C}$, then the 12 phone name variations with user frequency below 10 degrades the recall value of $\mathcal{L}^f$ on average.

Next, we evaluate the average precision of list $\mathcal{L}^f$. Recall that the matchable candidate names are ranked by confidence scores in $\mathcal{L}^f$. Using the ground truth for the 20 phones, we compute the average precision of each list and report the value in Table 5.1, column titled "AP($\mathcal{L}^f$)". Average precision of a ranked list of items is computed by averaging all precision values obtained at each rank position of the list from the top-ranked item to the bottom-ranked item [49]. The mean average precision is 0.677 for the 20 phones. We note that, the list $\mathcal{L}^f$ is generated from $\mathcal{C}$ and some entries in $\mathcal{L}^f$ may not be recognized as true phone name variations by the CRF classifier. If we remove the entries that are always predicted negative by the CRF classifier, then the mean average precision becomes 0.893, reported in the last column titled "AP($\mathcal{L}^f_r$)" in Table 5.1. Particularly, 14 mobile phones out of the 20 have perfect average precision of 1.0 for $\mathcal{L}^f_r$.

5.3 Name Recognition and Normalization

We report the accuracy of name recognition and normalization of Gren and compare it with baseline methods in this section.

5.3.1 Methods

We compare the proposed Gren method with other 3 baselines: Gren-nc, StanfordNER, and LexicalLookup. In the following, we detail the four methods.

- **Gren**: This is our proposed method with candidate name generation, CRF-based name recognition, and rule-based normalization. The training sentences were obtained by using 412 positive phone names and 500 negative phone names selected from the set of candidate names $\mathcal{C}$ reported in Section 5.2. The

features used to train the CRF model are listed in Table 3.4. Each candidate name mention was rewritten to a single word in both training and test sentences. We used YNO (Yes-No-Out) labeling scheme in CRF training: "Y" for a word rewritten from a positive name mention, "N" for a word rewritten from a negative name mention, and "O" for other words in the sentence (see Section 3.4).

- **Gren-nc**. In Gren, every candidate name mention is identified and rewritten to a single word. To evaluate the effectiveness of the pre-identification of candidate names, we propose Gren-nc method. In this method, no candidate name mention is identified and the sentences are kept in their original form. In training CRF, we used BILO (Begin-In-Last-Out) scheme. It is reported that BILOU (Begin-In-Last-Out-Unit) outperforms BIO (Begin-In-Out) scheme in named entity recognition [5, 50]. However, we cannot adopt the BILOU scheme because of the automatically generated training sentences. Recall in Section 3.4, the positive phone names are selected such that the model name (without brand) must contain at least two words. For this reason, no words in any sentences will be assigned label U (Unit). Therefore we use BILO (Begin-In-Last-Out) scheme without the U label. The same set of features in Table 3.4 were used except the last two features N_1 and N_2 because without sentence rewriting, a word by itself cannot represent a candidate name. In short, Gren-nc was trained using the same set of sentences as Gren with similar set of features except that Gren-nc did not take candidate names as an input.

- **StanfordNER**. This is of one of the most widely used package for named entity recognition. We adopted the default features provided by the package[8] and trained the CRF classifier using the same set of labeled sentences as in Gren and Gren-nc. Same as Gren-nc, the BILO labeling scheme was used. Note that, StanfordNER allows to use a gazetteer as external knowledge. However, the candidate name set $\mathcal{C}$ cannot be used as a gazetteer because many of its entries are not truly mobile phone names. The only difference between StanfordNER and Gren-nc is the features used for model training.

[8] http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/ie/crf/CRFClassifier.html

- **LexicalLookup**. We stored the 412 formal names that were used as positive names for sentence labeling in a dictionary. Phone name mentions were recognized by lexical lookup in this dictionary. Normalization is not necessary here because the names in the dictionary are formal names.

5.3.2 Evaluation Metric

We use the widely adopted *Precision* (Pr), *Recall* (Re) and F_1 to evaluate the performance of name recognition and the performance of name normalization respectively. The three measures for the name recognition subtask is defined as follows:

- Precision is the ratio of true phone name mentions among all mentions that are predicted positively.
- Recall is the ratio of correctly recognized name mentions among all phone name mentions annotated in the ground truth data.
- F_1 is the harmonic mean of precision and recall.

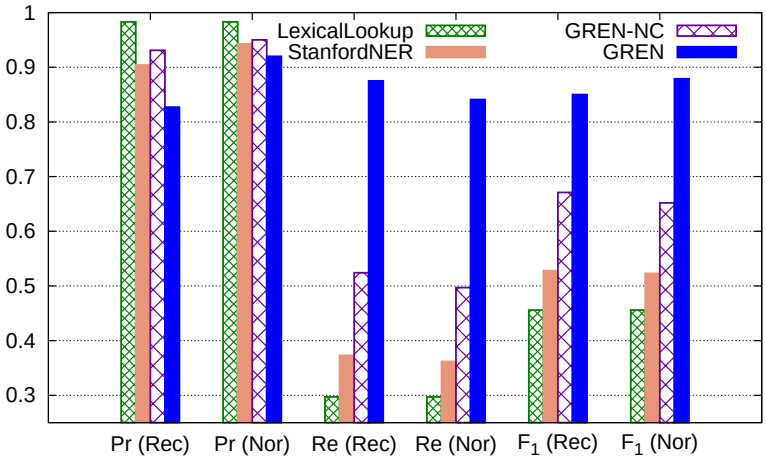
For the name normalization task, a phone name mention is correctly normalized only if it is first correctly detected by the name recognizer and then correctly mapped to its formal name by the name normalizer.

5.3.3 Experimental Results

The precision, recall and F_1 of the four methods for name recognition and name normalization are plotted in Figure 5.1.a and listed in 5.1.b for easy comparison.

We discuss the results on name recognition first. Observe from Figure 5.1, all the 4 methods achieve very high precision in name recognition. LexicalLookup gets nearly perfect precision as expected as all matched name mentions are formal names.[9] Both Gren-nc and StanfordNER achieve precision value higher than 0.9. Although the proposed Gren method has the lowest precision value of 0.827 among all methods, Gren significantly outperforms the other methods in recall. More specifically,

[9]The precision is not 1.0 because of a special case involving a space character. LexicalLookup recognizes "HTC Touch Diamond" from sentences containing word span "HTC Touch Diamond 2" and the correct recognition of the latter should be "HTC Touch Diamond2" (the formal name). "HTC Touch Diamond" is a different product.



5.1.a: Pr(Rec) and Pr(Nor) denote precision for name recognition and name normalization, respectively; the same applies to Re and F_1

Method	Name recognition			Name normalization		
	Pr	Re	F_1	Pr	Re	F_1
LexicalLookup	**0.983**	0.297	0.456	**0.983**	0.297	0.456
StanfordNER	0.904	0.373	0.528	0.943	0.362	0.523
Gren-nc	0.931	0.524	0.671	0.950	0.497	0.652
Gren	0.827	**0.875**	**0.850**	0.920	**0.841**	**0.879**

5.1.b: Pr, Re, and F_1 of the 4 methods with best results in boldface

Figure 5.1: Pr, Re, and F_1 of name recognition and name normalization for the 4 methods

Gren achieves recall of 0.875, which is 67% of improvement over the second best recall by Gren-nc, 134% improvement over StanfordNER, and nearly 200% improvement over LexicalLookup. Because of the much higher recall, Gren achieves F_1 of 0.850, followed by Gren-nc of 0.671 and StanfordNER of 0.528.

From this set of results, we argue that both candidate name generation and sentence rewriting are main factors contributing to the significant improvement of F_1 for Gren. Recall that the positive names used to generate training sentences are either formal phone names or model names (without brand) that contain at least two words. As the result, the name mentions seen by StanfordNER and Gren-nc in the training data do not contain any informal abbreviations or misspells. It becomes reasonable that StanfordNER and Gren-nc achieves better precision than Gren

but much poorer recall. The better recall by Gren-nc over StanfordNER attributes to the Brown clustering feature (G_2 in Table 3.4) which has been reported effective in NER from user-generated content like tweets [5, 17]. With candidate name generation and sentence rewriting, Gren is able to "focus" more on the surrounding context of a candidate name mention and at the same time to ignore detecting the boundaries of a candidate name because of the YNO labeling scheme. Nevertheless, not all candidate names are true phone name variations. The noises in candidate names result in slightly poorer precision for Gren, compared with StanfordNER and Gren-nc.

We now discuss the results on name normalization. There is no change in results of LexicalLookup because the names recognized are formal names by the method definition. For the rest three methods (*i.e.,* StanfordNER, Gren-nc, and Gren), improvement in precision and degradation in recall are observed compared to their corresponding results on name recognition. In our evaluation for name normalization, a positive instance refers to a name mention that is correctly recognized and correctly normalized to its formal name. Many of the mentions that are wrongly recognized as phone names by the name recognizer cannot be normalized to any formal names using the normalization rules are now considered negative. On the one hand, the normalization process removes errors from the results of name recognition, leading to increase in precision. On the other hand, due to the incompleteness of the rules, some of the correctly recognized names cannot be normalized to their corresponding formal names, *e.g.,* "Nozomi" is not normalized to "Sony Xperia S", leading to degradation in recall. Considering both precision and recall, Gren increases its F_1 from 0.850 to 0.879 after name normalization. Slightly worse F_1's are observed for both Gren-nc and StanfordNER.

5.3.4 Feature Analysis

Finally, we study the impact of the features listed in Table 3.4. Table 5.4 reports the precision, recall and F_1 of name normalization using all features in Gren and after removing one or two features (*e.g.,*-L_1, -N_1, N_2). Based on the results, we draw several conclusions along with their implications.

Table 5.4: Name normalization accuracy after removing one or two features. Best results are highlighted in boldface

Method/Feature	Pr	Re	F_1
Gren	0.920	0.841	0.879
Gren-L_1	0.916	0.836	0.874
Gren-L_2	0.918	**0.875**	**0.896**
Gren-G_1	0.920	0.839	0.877
Gren-G_2	0.899	0.728	0.804
Gren-N_1, N_2	**0.961**	0.791	0.868
Gren-nc	0.950	0.497	0.652
Gren-nc-L_2	0.953	0.447	0.609

- We observe that much better F_1 is achieved by removing the word surface feature L_2 from Gren (*i.e.,* Gren-L_2). Removing L_2 feature leads to improvement of F_1 from 0.879 to 0.896, about 2%. For comparison, we include results of Gren-nc and Gren-nc-L_2 in the last two rows in the table. Observe that removing L_2 from Gren-nc adversely affects F_1 with a big drop in recall. While word surface features are effective in most NER tasks, our results suggest that such features derived from the artificially created words (*i.e.,* the single words rewritten from candidate names in Gren) confuse the CRF model.

- Among the other features, features from Brown clustering are the most effective features; removing G_2 results in the largest degradation in F_1 comparing Gren-G_2 against Gren.

- The two features derived from the rewritten candidate names (N_1 and N_2) are the second most effective features. Removing L_1 or G_2 each leads to marginal degradation in F_1 measure. Relatively, they are less effective compared to features from Brown clustering and the features from the rewritten candidate names.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In the research, we solve the problem of product name recognition and normalization in Internet forums. More specifically, we focus on mobile phones and aim to recognize and normalize phone name mentions in Internet forums. Traditional techniques may fail due to two reasons. Firstly, the post messages in Internet forums are short and contain a huge number of ill-formed words. Secondly, users often mention a mobile phone with its informal abbreviations or nicknames.

Different from most NER approaches where named entities are recognized from sentences directly, we generate possible name variations based on word usage patterns and mobile phone naming conventions at first. A collection of sentences from an Internet forum and a set of formal names of mobile phones are considered as inputs. Brown clustering is first adopted to group words with similar meaning and syntactical function together. Four rules are designed to extract brand variations from clusters. Although simple and even to be ad-hoc, the four rules give surprisingly good results in terms of brand variation extraction. Candidate names of mobile phones are further generated. A filter is built to remove a large number of noises based on the extracted brand variations. As the result, candidate names cover a large proportion of mobile phone name variations.

However, candidate names contain a number of noises which cannot be excluded. On the other hand, some candidate names may refer to mobile phones and noises based on different context in sentences. Thus, CRF-based name recognizer is built.

A sentence is rewritten based on the candidate names which is able to take surrounding words of a candidate name to be its context in a more natural manner. Lexical features, grammatical features and our target-specific features named candidate name features are used in CRF. Furthermore, the pre-generation of candidate names also enables us to obtain a large number of training examples for CRF at very low cost for manual annotation.

Finally, a rule-based name normalizer is designed to map a detected name variation to its formal name. As most phone name variations detected are originated from the candidate names, the candidate names can be pre-normalized to their corresponding formal names. The rule-based name normalizer contains two components named matchable candidate name generator and disambiguator. Matchable candidate name generator extracts matchable candidate names which may refer to mobile phones from candidate names. Each matchable candidate name is linked to the formal names of mobile phones it may refer to. Then, disambiguator calculates the confidence score of each matchable candidate name to each formal name it is linked to. The matchable candidate name is mapped to the formal name which holds the highest confidence score.

Through extensive experiments, we show that our proposed Gren method significantly outperforms baseline methods, particularly in recall measure for name recognition and normalization. Two implications are observed from the NER task. First, if candidate named entities are able to be pre-generated, a large number of training examples may be generated at very low cost for manual annotation. Second, if we can segment the sentences and pre-generate the text chunks, we are able to rewrite the sentences. The rewriting enables us to take surrounding words of a named entity to be its context in a more natural manner. Even though the problem is for mobile phones, we believe the Gren method is generic and flexible in the sense that both the candidate names and the name normalization rules can be easily modified to accommodate special cases in practical applications.

6.2 Future Work

In our approach, Brown clustering is considered as the first step. Even though satisfied performance is obtained, Brown clustering only works on existing dataset. Due

to this reason, our approach is limited to the existing mobile phone names while variations of newly released phones are not able to be recognized. Thus, a re-generation process of candidate names is necessary. As a series of mobile phones follows similar naming conventions, some rules may be designed to capture the patterns which partially solve the problem.

As the normalization component of Gren is mainly based on lexical rules, it may not cover all name variations of a mobile phone due to the flawed rules. For example, some users use codename "Nozomi" to refer the phone "Sony Xperia S" but the codename "Nozomi" is not included in $\mathcal{L}^f$ for this phone. On the other side, rules handcrafted based on observations and assumptions related to one application are not guaranteed to work well on other applications. Thus, we plan to improve the normalization component which does not only use lexical rules, but also incorporates additional information. A knowledge base which stores the profile of each mobile phone can be utilized. The profiles of mobile phones are obtained from Wikipedia and websites providing specific information of the mobile phones. Multiple methods can be used to map a name variation to the corresponding mobile phone profile. For example, the similarity can be measured by calculating contextual overlap between the sentence containing a name variation and mobile phone profiles. The name variation is mapped to the mobile phone profile which holds the highest similarity. The contextual information can also be used as features to generate a classifier which assigns a name variation to the corresponding mobile phone profile.

Shown in the experimental results of NER and NEN of mobile phones (see Section 5.3.3), NEN improves the results generated by NER in terms of the precision. However, as NER and NEN are conducted separately, there is no feedback from NEN to NER even though NEN recognizes that a detected name mention is not a mobile phone name variation. Thus, we suggest to build a system which conducts NER and NEN jointly. The improving performance of NER boosts the performance of NEN. On the other hand, NEN guides NER in terms of the recognition. The system conducts two tasks simultaneously by enabling them to interact with each other.

Appendix A

Author's Publications

- **Yangjie Yao**, Aixin Sun. "Are Most-viewed News Articles Most-shared?", *In Proc. of the 9th Asian Information Retrieval Societies Conference (AIRS)*, Singapore, 2013.
- Aixin Sun, **Yangjie Yao**. "Product Name Recognition and Normalization in Internet Forums", Technical Disclosure with NIEO, NTU. Reference No: TD/029/14.
- **Yangjie Yao**, Aixin Sun. "Product Name Recognition and Normalization in Internet Forums", *SIGIR Symposium on IR in Practice (SIRIP)* [SIGIR'14 Industry Track], Gold Coast, Australia, 2014.

References

[1] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[2] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 282–289, Morgan Kaufmann Publishers Inc., 2001.

[3] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, pp. 3–26, 2007.

[4] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 363–370, Association for Computational Linguistics, 2005.

[5] L. Ratinov and D. Roth, "Design challenges and misconceptions in named entity recognition," in *CoNLL*, pp. 147–155, ACL, 2009.

[6] D. M. Bikel, R. Schwartz, and R. M. Weischedel, "An algorithm that learns what‘s in a name," *Mach. Learn.*, vol. 34, no. 1-3, pp. 211–231, 1999.

[7] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, pp. 188–191, Association for Computational Linguistics, 2003.

[8] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Unsupervised named-entity extraction from the web: An experimental study," *Artif. Intell.*, vol. 165, no. 1, pp. 91–134, 2005.

[9] N. Rizzolo and D. Roth, "Modeling discriminative global inference," in *Proceedings of the International Conference on Semantic Computing*, pp. 597–604, IEEE Computer Society, 2007.

[10] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine learning*, vol. 37, no. 3, pp. 277–296, 1999.

[11] E. F. T. K. Sang and J. Veenstra, "Representing text chunks," in *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, pp. 173–179, 1999.

[12] A. Ritter, C. Cherry, and B. Dolan, "Unsupervised modeling of twitter conversations," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 172–180, Association for Computational Linguistics, 2010.

[13] B. Han and T. Baldwin, "Lexical normalisation of short text messages: Makn sens a #twitter," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pp. 368–378, Association for Computational Linguistics, 2011.

[14] B. Locke and J. Martin, "Named entity recognition: Adapting to microblogging," *Senior Thesis, University of Colorado*, 2009.

[15] X. Liu, S. Zhang, F. Wei, and M. Zhou, "Recognizing named entities in tweets," in *ACL*, pp. 359–367, ACL, 2011.

[16] T. Finin, W. Murnane, A. Karandikar, N. Keller, J. Martineau, and M. Dredze, "Annotating named entities in twitter data with crowdsourcing," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pp. 80–88, Association for Computational Linguistics, 2010.

[17] A. Ritter, S. Clark, Mausam, and O. Etzioni, "Named entity recognition in tweets: An experimental study," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1524–1534, Association for Computational Linguistics, 2011.

[18] F. Sha and F. Pereira, "Shallow parsing with conditional random fields," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pp. 134–141, Association for Computational Linguistics, 2003.

[19] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, pp. 248–256, Association for Computational Linguistics, 2009.

[20] C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B.-S. Lee, "Twiner: Named entity recognition in targeted twitter stream," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 721–730, ACM, 2012.

[21] J. F. da Silva and G. P. Lopes, "A local maxima method and a fair dispersion normalization for extracting multi-word units from corpora," in *Sixth Meeting on Mathematics of Language*, 1999.

[22] C. Li, A. Sun, J. Weng, and Q. He, "Exploiting hybrid contexts for tweet segmentation," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 523–532, ACM, 2013.

[23] E. Benson, A. Haghighi, and R. Barzilay, "Event discovery in social media feeds," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pp. 389–398, Association for Computational Linguistics, 2011.

[24] S. Singh, D. Hillard, and C. Leggetter, "Minimally-supervised extraction of entities from text advertisements," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 73–81, Association for Computational Linguistics, 2010.

[25] J. Guo, G. Xu, X. Cheng, and H. Li, "Named entity recognition in query," in *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 267–274, ACM, 2009.

[26] A. Alasiry, M. Levene, and A. Poulovassilis, "Detecting candidate named entities in search queries," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1049–1050, ACM, 2012.

[27] K. Yoshida and J. Tsujii, "Reranking for biomedical named-entity recognition," in *Biological, translational, and clinical language processing*, pp. 209–216, Association for Computational Linguistics, 2007.

[28] Y. Wang, "Annotating and recognising named entities in clinical notes," in *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*, pp. 18–26, Association for Computational Linguistics, 2009.

[29] R. Mihalcea and A. Csomai, "Wikify!: Linking documents to encyclopedic knowledge," in *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, pp. 233–242, ACM, 2007.

[30] M. A. Khalid, V. Jijkoun, and M. De Rijke, "The impact of named entity normalization on information retrieval for question answering," in *Proceedings of the IR Research, 30th European Conference on Advances in Information Retrieval*, pp. 705–710, Springer-Verlag, 2008.

[31] X. Liu, Y. Li, H. Wu, M. Zhou, F. Wei, and Y. Lu, "Entity linking for tweets," in *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics*, 2013.

[32] D. Milne and I. H. Witten, "Learning to link with wikipedia," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pp. 509–518, ACM, 2008.

[33] W. Shen, J. Wang, P. Luo, and M. Wang, "Linden: Linking named entities with knowledge base via semantic knowledge," in *Proceedings of the 21st International Conference on World Wide Web*, pp. 449–458, ACM, 2012.

[34] I. Witten and D. Milne, "An effective, low-cost measure of semantic relatedness obtained from wikipedia links," in *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pp. 25–30, 2008.

[35] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*, pp. 697–706, ACM, 2007.

[36] X. Han, L. Sun, and J. Zhao, "Collective entity linking in web text: A graph-based method," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 765–774, ACM, 2011.

[37] E. Meij, W. Weerkamp, and M. de Rijke, "Adding semantics to microblog posts," in *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, pp. 563–572, ACM, 2012.

[38] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien, "Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation," in *Proceedings of the 12th International Conference on World Wide Web*, pp. 178–186, ACM, 2003.

[39] P. N. Mendes, A. Passant, P. Kapanipathi, and A. P. Sheth, "Linked open social signals," in *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, pp. 224–231, IEEE Computer Society, 2010.

[40] E. A. Fox and J. A. Shaw, "Combination of multiple searches," *NIST SPECIAL PUBLICATION SP*, pp. 243–243, 1994.

[41] P. Ferragina and U. Scaiella, "Tagme: On-the-fly annotation of short text fragments (by wikipedia entities)," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pp. 1625–1628, ACM, 2010.

[42] O. Topchylo, "Identification and disambiguation of product mentions with information retrieval and problem specific methods," in *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, pp. 872–873, IEEE, 2012.

[43] B. Gődény, "Rule based product name recognition and disambiguation," in *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, pp. 858–860, IEEE, 2012.

[44] S. Wu, Z. Fang, and J. Tang, "Accurate product name recognition from user generated content," in *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, pp. 874–877, IEEE, 2012.

[45] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, pp. 467–479, 1992.

[46] P. Liang, "Semi-supervised learning for natural language," Master's thesis, Massachusetts Institute of Technology, 2005.

[47] L. Philips, "Hanging on the metaphone," *Computer Language Magazine*, vol. 7, no. 12, pp. 39–44, 1990.

[48] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, pp. 1–47, Mar. 2002.

[49] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[50] L. Ramshaw and M. Marcus, "Text Chunking Using Transformation-Based Learning," in *Proceedings of the Third Workshop on Very Large Corpora* (D. Yarovsky and K. Church, eds.), pp. 82–94, Somerset, New Jersey: Association for Computational Linguistics, 1995.