Modeling OOV Words With Letter N-Grams in Statistical Taggers: Preliminary Work in Biomedical Entity Recognition

Teemu Ruokolainen[1] *Miikka Silfverberg*[2]

(1) Aalto University, Helsinki, Finland
(2) University of Helsinki, Helsinki, Finland

teemu.ruokolainen@aalto.fi, miikka.silfverberg@helsinki.fi

We discuss sequential tagging problems in natural language processing using statistical methodology. We propose an automatic and domain-independent approach to modeling out-of-vocabulary (OOV) words, that is words that do not occur in training data. Our method is based on using probabilistic letter n-gram models to model orthography of different tags. We show how to combine the approach with two widely used statistical models Hidden Markov Models and Conditional Random Fields. Instead of taking the common approach of directly using sub-strings as features resulting in an explosion in the number of model parameters, we compress orthographic information into a small number of parameters. Experiments in biomedical entity recognition on the Genia corpus show that the approach can alleviate the OOV problem resulting in improvement in overall model performance.

KEYWORDS: Biomedical Entity Recognition, CRF, HMM, Letter N-Grams, OOV, Tagging.

1 Introduction

We discuss sequential tagging problems in natural language processing (NLP) including tasks such as part-of-speech (POS) tagging (Brants, 2000), shallow parsing (Tjong Kim Sang and Buchholz, 2000), and named entity recognition (NER) (Tjong Kim Sang and De Meulder, 2003). Modern NLP approaches rely on data-driven statistical models trained using manually tagged training corpora. Commonly applied models include Hidden Markov Models (HMMs) (Brants, 2000), Maximum Entropy Markov Models (MEMMs) (McCallum et al., 2000), and Conditional Random Fields (CRFs) (Lafferty et al., 2001). In this work, we present preliminary work on an automatic, domain-independent approach for extracting orthographic features useful for modeling rare words and out-of-vocabulary words, that is words which were not observed in the training data.

All existing supervised statistical models utilize some form of feature functions, which extract relevant information from training sentences. In sequential word tagging problems, the word forms themselves and the words surrounding them are by default the most useful features. Consider for example POS tagging given a relatively large training corpus. For words that are frequently observed during training, it is reasonable to assume that all of the possible tags of the words are present in the training data. For these words, tagging is thus reduced to *disambiguation* between a limited set of known tags based on neighboring words and their tags.

Unfortunately, the word form itself is not a useful feature for out-of-vocabulary (OOV) words. For OOV words essentially all tags are possible, which makes disambiguation based on word and tag context more difficult. The OOV problem is typically most severe when only a small amount of data is available for model training, and when dealing with languages with rich morphology. The OOV problem is important, because manually tagging large amounts of training data is resource intensive. In order to alleviate the OOV problem, it is beneficial to enrich the model with *sub-word features* which model *word orthography and morphology*.

In some modeling problems, designing the sub-word features requires little knowledge about the domain of the problem. For example in POS tagging, simple features describing capitalized letters, presence of digits and suffixes of varying lengths, are sufficient for many languages (Brants, 2000). However, for morphologically complex languages and tasks such as NER this approach is inadequate. Instead, domain-specific knowledge is required for manually writing a set of regular expressions, which are then used in feature extraction. An alternative domain independent and automatic approach is to utilize sub-strings as features. In this work, we discuss a modification of this approach. In contrast to using a large amount of sub-string features, out approach is based on a small number of features utilizing *letter n-gram models*, which are easy to employ using existing efficient toolkits. We describe how to incorporate the features with HMM and CRF models.

In order to evaluate our method, we present experiments in biomedical entity recognition, a special case of NER, on the Genia corpus (Kim et al., 2004). Our results show, that the letter n-gram features can improve the model performance with small training sets. The improvements are more notable with CRFs, which as a discriminative log-linear model can naturally incorporate complex, non-independent features. Although the present work focuses on biomedical entity extraction, we believe that the approach could be beneficial in other domains as well.

The rest of the paper is as follows. In Section 2, we briefly discuss previous, related research. In Section 3, we describe the letter n-gram models and how to use them to extract orthographic

features for the HMM and CRF models. Experiments conducted in biomedical entity recognition are described In Section 4 along with a discussion. We provide conclusions in Section 5.

2 Related Work

In the experimental section of this work, we provide results on the Genia corpus, the data set used in the shared task of biomedical entity recognition in the joint workshop of BioNLP/NLPBA 2004 (Kim et al., 2004). Among the participants of this task, the most commonly adopted approach to modeling OOV words was to use manually constructed regular expressions and affixes combined with gazetteers and POS features. This was the approach taken by the best performing system described by (Zhou and Su, 2004). We compare our n-gram-based feature extraction approach with the regular expression set used in their system, originally described by (Shen et al., 2003), and show that our method performs better[1].

The idea of constructing a domain-independent set of orthographic features utilizing letter n-grams is not new. Probabilistic letter n-gram models have been used in term recognition. For example (Vasserman, 2004) employed them as classifiers in chemical name recognition in a similar manner they have been utilized in language identification (Vatanen et al., 2010). Our use of the n-gram models to capture word orthography closely resembles theirs. However, we investigate incorporating letter n-gram models as *feature extractors* in statistical taggers. Also, plain letter n-grams (raw sub-strings of words) have been used extensively as features in taggers, see for example (Rössler, 2004). However, this approach results in an explosion in the number of model parameters, which could be avoided by using the feature extraction approach presented in this work.

Automatic and domain-independent orthographic feature extraction was recently discussed by (Fujii and Sakurai, 2012). The central idea in their work was to employ combined tag level and character level language models based on a hierarchical Bayesian approach. They also described experiments on the Genia corpus. However, our CRF model appears to outperforms their model. Additionally, although orthographic features are used to combat the OOV problem and have little effect on recognition of frequent words, the discussion provided by (Fujii and Sakurai, 2012) lacks this aspect completely.

To our knowledge, the highest performance on the Genia corpus has been reported by (Vishwanathan et al., 2006). Our work is similar to theirs in that we use a CRF model. The key differences are that their results were obtained using an extensive regular expression set and manually prepared features. Unfortunately, the description of the features in the work was severely underspecified and we could not reproduce their results. This made it impossible to compare the system to our system. Additionally, their evaluation is non-standard in that they fitted model parameters directly to the test set instead of using a development set. This results in overly optimistic performance. The reason for the non-standard evaluation in their work was that they focused on comparing time complexities of model optimization algorithms instead of model performances.

3 Methods

In this section, we describe the methods. We first describe the letter n-gram models. Then, we go on to describe how to utilize the n-grams in the lexical model of an HMM and for feature extraction in CRFs.

[1]Note, however, that their system incorporates a wide variety of other contextual features thus achieving higher overall performance.

3.1 Obtaining Tag-Specific Word Likelihoods Using Letter N-Grams

Letter-based language models are used to assign a probability for a word represented as a sequence of letters $(w_1, w_2, \ldots, w_{m-1}, w_m)$. Given the sequential nature, a suitable approach to obtaining the probability is to use a factorization

$$p(word) = \prod_{i=1}^{m} p(w_i \mid w_1, \ldots, w_{i-1}), \tag{1}$$

where $p(w_i \mid w_1, \ldots, w_{i-1})$ denotes the probability of letter w_i given the context $(w_1, \ldots, w_{i-1})$. Due to sparsity, the context is usually approximated utilizing only the $n-1$ previous letters (Markov assumption) yielding the approximation

$$p_{ng}(word) \approx \prod_{i=1}^{m} p_{ng}(w_i \mid w_{i-n+1}, \ldots, w_{i-1}), \tag{2}$$

where the distribution $p_{ng}(w_i \mid w_{i-n+1}, \ldots, w_{i-1})$ is referred to as the *letter n-gram model*. The individual probabilities $p_{ng}(w_i \mid w_1, \ldots, w_{i-1})$ can be obtained using smoothed maximum likelihood estimates (Chen and Goodman, 1999) or the maximum entropy model (Chen and Rosenfeld, 2000).

The key idea of our approach for is to learn a separate letter n-gram model for each tag in the tag set. This can be achieved by dividing the words observed in the training corpus into L sets, where L is the number of tags, so that each set contains words that have been tagged at least once with the corresponding tag. Then, a tag-specific n-gram model

$p_{ng}(w_i \mid w_1, \ldots, w_{i-1}, tag)$

can be trained on each of the words sets. Hence, the likelihood of a word being generated by a n-gram model corresponding to a particular tag can then be written as

$$p_{ng}(word \mid tag) = \prod_{i=1}^{m} p_{ng}(w_i \mid w_{i-n+1}, \ldots, w_{i-1}, tag). \tag{3}$$

3.2 Hidden Markov Models

Model. Hidden Markov Models (HMMs) are a well known generative statistical model suitable for tagging and segmentation tasks (Brants, 2000). In this work, we employ a second degree HMM. In a second degree HMM for sentence tagging, the estimate of the probability of a sequence of tags $\boldsymbol{y} = (y_1, \ldots, y_t, \ldots, y_T)$ given an input sentence $\boldsymbol{x} = (x_1, \ldots, x_t, \ldots, x_T)$ is

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{T} p(x_t \mid y_t) p(y_t \mid y_{t-1}, y_{t-2}). \tag{4}$$

The estimates $p(y_t \mid y_{t-1}, y_{t-2})$ are called *transition probabilities* and can be readily computed from a training corpus. To deal with data sparseness, some form of smoothing is usually necessary. We use deleted interpolation following (Brants, 2000).

Two extra tags y_0 and y_{-1} are needed in formula (4) for the trigram estimates $p(y_2 \mid y_1, y_0)$ and $p(y_1 \mid y_0, y_{-1})$. These are so called buffer tags #, which are appended to tag sequence boundaries.

Lexical modeling utilizing letter n-grams. The conditional probabilities $p(x_t \mid y_t)$ are called *lexical emission probabilities*. They are determined by a *lexical model*. Our HMM implementation uses a lexical model, which estimates $p(x_t \mid y_t)$ for known words based on smoothed ML estimates computed from training data. It estimates the conditional probability $p(x_t \mid y_t)$ for an OOV word x_t and a tag y_t as the probability $p_{ng}(x_t \mid y_t)$ given by the letter n-gram model for tag y_t. The lexical emission probability is thus

$$p(x_t \mid y_t) = \begin{cases} (\mathrm{C}(x_t, y_t) + \alpha)/(\mathrm{C}(x_t) + N\alpha) & \text{if } x_t \text{ is known.} \\ p_{ng}(x_t \mid y_t) & \text{if } x_t \text{ is OOV.} \end{cases} \quad (5)$$

The term N in formula 5 denotes the number of unique word forms in the training data and its smoothing coefficient α is determined by minimizing the tagging error on a development data set.

3.3 Linear-chain Conditional Random Fields

Model. Linear-chain Conditional Random Fields (CRFs) are a discriminative, log-linear model for tagging and segmentation presented originally by Lafferty et al. (2001). The central idea of the linear-chain CRF is to exploit the dependencies between the output variables using a chain structured undirected graph, also referred to as a Markov random field.

Formally, the model for input $\boldsymbol{x}$ (words in sentence) and output $\boldsymbol{y}$ (tags) is written as

$$p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{w}) \propto \prod_{t=1}^{T} \exp\left(\boldsymbol{w}^\top \boldsymbol{f}(y_t, \boldsymbol{x}, t)\right) \prod_{t=2}^{T} \exp\left(\boldsymbol{w}^\top \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)\right), \quad (6)$$

where t indexes the characters, T denotes sentence length, $\boldsymbol{w}$ the model parameter vector, $\boldsymbol{f}$ the vector-valued feature extraction function, and $\boldsymbol{w}^\top \boldsymbol{f}$ dot product between $\boldsymbol{w}$ and $\boldsymbol{f}$. The purpose of $\boldsymbol{f}$ is to capture the co-occurrence behavior of the output configurations (y_t) and (y_{t-1}, y_t) and a set of features describing the input $\boldsymbol{x}$ at word position t.

As a discriminative log-linear model, the CRFs can naturally incorporate arbitrary, overlapping features which might be useful for the prediction. The downside of this freedom in feature design is the larger computational cost of model training compared to the HMM models.

Enriching Orthographic Feature Extraction With Letter N-Grams. The performance of the CRF model depends heavily on the quality of the features included in the feature vector. As our default feature set, we use binary-valued features describing the input $\boldsymbol{x}$ at position t with *word identities* in position $t-2, \ldots, t+2$. This feature set performs well if there exists ample training data, or generally, if the number of OOV words is small.

In order to alleviate the OOV problem, we propose modeling word orthography utilizing tag-specific letter n-gram models, $p_{ng}(word|tag)$, presented in section 3.1. They key idea is to to represent any arbitrary word using L features, where L is the number of tags in tag set. The value of each of these features is the *posterior probability* of the corresponding tag given the word, $p_{ng}(tag|word)$. Therefore, instead of using the *likelihoods*, the CRFs exploit the posterior probabilities which directly hold the relevant information when choosing best tags for words. The features are combined with different tag configurations in a standard manner. The aim of

these features is simply to associate a high tag posterior value to a high probability of assigning the word with that particular tag.

Formally, the posterior probability of a word given a tag is given by the standard Bayes' rule as

$$p_{ng}(tag|word) = \frac{p_{ng}(word|tag)p(tag)}{\sum_{tag'} p_{ng}(word|tag')p(tag')} \quad (7)$$

where the likelihoods $p_{ng}(word|tag)$ are obtained as presented in Section 3.1. The tag priors $p(tag)$ are simply the counts of tags in the training data normalized to sum up to one.

Parameter estimation. In general, the CRF model parameters $\boldsymbol{w}$ are estimated using a training set of annotated text using, for example, the maximum likelihood criterion as in (Lafferty et al., 2001). In this work, we estimate the parameters using the averaged structured perceptron algorithm presented by Collins (2002). The perceptron algorithm proceeds one training instance at a time by performing maximum a posteriori (MAP) inference, that is prediction of most likely output configuration over the graph, using the standard Viterbi search, and by updating the model parameters using a simple additive rule if an erroneous prediction occurs. Subsequent to training, test instances are tagged again using the Viterbi search. The structured perceptron has a single hyper-parameter, namely the number of passes over training data, which is optimized using a separate development set.

4 Experiments

We present experiments conducted in biomedical entity recognition on the extended GENIA (version 3) corpus (Kim et al., 2004). We compare three feature sets, namely word identities for baseline, word identities combined with a set of regular expressions adapted from the ones used by (Shen et al., 2003), and word identities combined with letter n-gram based features using HMM and CRF models as described in Section 3. Appendix A contains the complete set of regular expressions that we used.

4.1 Setup

Task and data. Bio-entity recognition is the task of finding and classifying biological entities such as the protein *CD28 surface receptor* in the sentence fragment "Activation of the CD28 surface receptor provides...". The task is viewed as tagging the words in a sentence using three basic tag types: *beginning of an entity (B)* , *inside an entity (I)* and *outside an entity (O)* as illustrated in Figure 1.

```
MTIIa   mRNA    level   increased   significantly   .
B-RNA   I-RNA   I-RNA   O           O               O
```

Figure 1: Biological entities `X` are chains `B-X I-X ...`

The GENIA corpus (Kim et al., 2004) consists of biomedical abstracts with a manually prepared named entity annotation. Each word token in the corpus is annotated with a bio-entity tag or an *O* tag if the word does not belong to a bio-entity. The tag set consists of 11 entity tags: the *O* tag and {*B, I*} tags for five entity classes corresponding to DNA, RNA, protein, cell line and cell type respectively. Additionally, we assume sentence start and end tags and markers.

Data Set	Sentences	Words
Train.	114	2798
Train.	229	5865
Train.	459	11667
Train.	918	23675
Train.	1836	48291
Train.	3672	97290
Train.	7345	194637
Train.	14690	390058
Devel.	3856	102493
Test	3856	101039

Table 1: Number of sentences and word tokens in training, development and test data sets.

By default, the corpus is divided into training (18,546 sentences, 492,551 tokens) and test (3,856 sentences, 101,039 tokens) sets. We form a separate development set by extracting the last 3,856 sentences (102,493 tokens) from the training set. In order to observe the effect of training data set size on model performances, we form eight training sets by taking first n sentences from the total 14,690 training instances for various values of n, see Table 1.

Model evaluation. The model performances are evaluated using the *f-score*. The f-score is the geometric mean of micro-averaged *precision* (the percentage of correctly assigned entities with respect to all assigned entities) and *recall* (the percentage of correctly assigned entities with respect to the gold standard entities).

Feature extraction. We compare three different feature sets. The first set is the WORD set, which includes only the current word to be tagged for HMMs and a five word window around the current word for CRFs as described in Sections 3.2 and 3.3, respectively. The second set is the REGEXP set, which additionally contains the regular expressions originally presented by (Shen et al., 2003) (see Appendix A). The third set is the N-GRAM set, which consists of the WORD features combined with letter n-gram features as described in Sections 3.2 and 3.3, respectively.

N-gram model, CRF and HMM training. We trained the tag-specific letter n-gram models on training sets of various sizes described in Table 1. We used the SRILM toolkit (Stolcke et al., 2011) to implement the n-gram models. More specifically, we used n-grams of length 9 with Witten-Bell smoothing, which appeared to work sufficiently well in preliminary experiments on the development data set. The HMM and CRF models were trained on the training data sets using three feature sets (WORD, REGEXP, N-GRAM) as described in Sections 3.2 and 3.3. As to CRF model training, the perceptron algorithm is terminated when development set performance has not improved during the last five passes over the training set.

Software. We used our own implementations of the CRF and HMM models. For training the letter n-gram models, we used SRILM as described above.

Train.	OOV	WORD			REGEXP			N-GRAM		
114	35.8	34.1	22.7	27.3	33.2	26.8	29.7	29.4	37.5	**33.0**
229	28.3	41.5	28.2	33.5	40.4	33.2	36.4	33.1	46.1	**38.5**
459	22.1	48.8	35.5	41.1	48.0	40.1	43.7	41.2	47.8	**44.2**
918	17.4	52.8	41.0	46.1	52.3	40.1	48.7	46.0	53.0	**49.3**
1836	13.5	56.7	44.3	49.8	56.8	49.2	52.7	52.4	56.8	**54.5**
3672	10.0	61.1	51.2	55.7	59.8	54.8	57.2	57.0	61.7	**59.3**
7345	7.2	64.2	58.7	61.3	63.6	61.0	62.2	61.9	65.4	**63.6**
14690	5.2	67.5	64.2	66.5	65.8	65.5	66.0	65.5	69.4	**67.4**

Table 2: Results for the CRF model using varying training data set sizes (in sentences) and feature sets. For each feature set, the figures are recall, precision and f-score.

Train.	OOV	WORD			REGEXP			N-GRAM		
114	35.8	22.0	35.6	**27.2**	20.5	34.5	25.7	16.5	7.7	10.5
229	28.3	27.2	37.0	**31.4**	26.3	35.7	30.3	27.1	15.1	19.4
459	22.1	33.4	42.3	**37.3**	32.5	40.8	36.2	34.6	32.4	33.5
918	17.4	37.0	44.8	40.6	39.1	44.7	**41.7**	41.5	40.3	40.0
1836	13.5	40.5	47.7	43.8	42.8	47.8	45.1	47.1	45.9	**46.5**
3672	10.0	46.7	51.0	48.7	49.0	50.5	49.7	53.4	50.9	**52.1**
7345	7.2	52.0	53.8	52.9	53.5	53.0	53.3	57.1	55.1	**56.1**
14690	5.2	56.5	56.6	56.6	57.8	56.3	57.0	60.4	57.4	**58.9**

Table 3: Results for HMMs using varying training data set sizes (in sentences) and feature sets. For each feature set, the figures are recall, precision and f-score.

4.2 Results

The results for varying training data set sizes are presented in Tables 2 and 3 for the CRF and HMM models, respectively. The columns titled *OOV* correspond to the percentages of OOV word tokens in the test set. The performance values obtained using different feature sets are in order recall, precision, and f-score.

Using the CRF model, the regular expression feature set provided modest, yet consistent, improvement over the word identity context baseline. Utilizing the letter n-gram features yielded performance gains compared to both the word identity context baseline and the regular expression feature set. This improvement took place when small training sets were used.

The results for the HMM models using the different feature sets differ somewhat from the results of the CRF models. Both the REGEXP and N-GRAM features initially perform worse than the WORD features. Roughly the same performance for all features is achieved when there are 918 sentences of training data and in the next training set of 1836 sentences the N-GRAM features outperform the other feature sets. The result for the complete training set of 14690 sentences shows that the N-GRAM feature set is the best one achieving f-score 58.9 versus 57.0 for the REGEXP features and 56.6 for the WORD features. Overall, the HMMs consistently resulted in considerably worse performance compared to the CRF models for all training sets and features. The improvement given by the N-GRAM features is comparable to the improvements for the CRF model.

4.3 Discussion

For the complete data set, the letter n-gram features give the best results for both the HMM and the CRF. However, the CRF model clearly outperforms the HMM for all training data set sizes and all feature sets. The improvements given by the letter n-gram features are similar for the CRF and HMM models. However the CRF model with letter n-gram features consistently outperform the models with other feature sets for all training data set sizes. In contrast, the HMM with letter n-gram features performs worse than the models with word and regex features when the training set is smaller than 918 sentences.

Unfortunately, our baseline models are too weak to allow for a conclusive assessment of the impact of our approach. A model trained on suffixes and prefixes or all sub-strings of words should have been included in the evaluation. Such an approach could give higher performance, but would also result in a substantially higher number of features implying a longer training time and greater memory footprint.

The performance of the letter n-gram feature HMM clearly shows that the letter n-grams are not very reliable on small training sets. Nevertheless the letter n-gram model of the CRF model achieves improvements even for the smallest training set of 114 sentences. This is most likely a direct result of the discriminative training of CRF models, which can successfully integrate the uncertain information given by the letter n-grams.

The poor performance of the HMM compared to the CRF may partly be explained by the fact that there are two useful information sources which the HMM cannot use, but the CRF can utilize. Neighboring words cannot be used in the lexical model of the HMM. Our attempts to do this in the development phase were unsuccessful.

Finally, direct comparison of our models with the hierarchical Bayesian approach reported by (Fujii and Sakurai, 2012) is difficult because their model was trained using a *semi-supervised* approach. In other words, when using small data set sizes, their model exploited the remainder of the training corpus without the tagging information. Effectively, this means that compared to our approaches, their model utilized more data in training. However, a fair comparison can be made when all the models used the complete training data set of 18,546 sentences for training and model development. Using this full data, our CRF model outperformed their model regardless of the feature set. Additionally, the CRF model utilizing the letter n-gram features appeared to yield at least comparable if not improved performance compared to their model on smaller data sets.

5 Conclusions

We discussed biomedical entity recognition using statistical methodology. We proposed an automatic and domain-independent approach to modeling OOV words utilizing probabilistic letter n-gram models. We described how to combine the approach with Hidden Markov Models and Conditional Random Fields. The motivation behind our approach was to compress orthographic information captured by sub-strings in a small set of features. Experiments in biomedical entity recognition showed that the approach did alleviate the OOV problem resulting in improved overall model performance. Unfortunately, the true impact of the discussed approach is left unclear due to the lack of strong baselines in this preliminary work.

In future work, we aim to extend the experimental results provided in this paper with additional data sets, improved baseline models, and closer error analysis on OOV words.

Acknowledgments

This work was financially supported by the Academy of Finland under the grant no 251170 (Finnish Centre of Excellence Program (2012-2017)) and Langnet (Finnish doctoral programme in language studies).

References

Brants, T. (2000). Tnt - a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing (ANLP-2000)*, Seattle, WA.

Chen, S. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Chen, S. and Rosenfeld, R. (2000). A survey of smoothing techniques for me models. *Speech and Audio Processing, IEEE Transactions on*, 8(1):37–50.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, volume 10, pages 1–8. Association for Computational Linguistics.

Fujii, R. and Sakurai, A. (2012). Technical term recognition with semi-supervised learning using hierarchical bayesian language models. *Natural Language Processing and Information Systems*, pages 327–332.

Kim, J.-D., Ohta, T., Tsuruoka, Y., Tateisi, Y., and Collier, N. (2004). Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, JNLPBA '04, pages 70–75, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

McCallum, A., Freitag, D., and Pereira, F. (2000). Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, volume 951, pages 591–598.

Rössler, M. (2004). Adapting an ner-system for german to the biomedical domain. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, JNLPBA '04, pages 92–95, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shen, D., Zhang, J., Zhou, G., Su, J., and Tan, C. (2003). Effective adaptation of a hidden markov model-based named entity recognizer for biomedical domain. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*, pages 49–56. Association for Computational Linguistics.

Stolcke, A., Zheng, J., Wang, W., and Abrash, V. (2011). Srilm at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*.

Tjong Kim Sang, E. F. and Buchholz, S. (2000). Introduction to the conll-2000 shared task: chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning - Volume 7*, ConLL '00, pages 127–132, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vasserman, A. (2004). Identifying chemical names in biomedical text: an investigation of the substring co-occurrence based approaches. In *Proceedings of the Student Research Workshop at HLT-NAACL 2004*, HLT-SRWS '04, pages 7–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vatanen, T., Väyrynen, J. J., and Virpioja, S. (2010). Language identification of short text segments with n-gram models. In Chair), N. C. C., Choukri, K., Maegaard, B., Mariani, J., Odjik, J., Piperidis, S., Rosner, M., and Tapias, D., editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., and Murphy, K. P. (2006). Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 969–976, New York, NY, USA. ACM.

Zhou, G. and Su, J. (2004). Exploring deep knowledge resources in biomedical name recognition. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, JNLPBA '04, pages 96–99, Stroudsburg, PA, USA. Association for Computational Linguistics.

A Regular expression features

The regular expression features in Python Re-syntax adapted from (Shen et al., 2003).

Feature name	Regular expression
Comma	`^[,]$`
Dot	`^[.]$`
Left Paren.	`^[(]$`
Right Paren.	`^[)]$`
Left Square Bracket	`^[\[]$`
Right Square Bracket	`^[\]]$`
Roman Numeral	`^[IVXCM]+$`
Greek Letter	`^(Alpha| ... |Omega|alpha| ... |omega)$`
Stop Word	`^(a|about|...)$`
ATCG Sequence	`^[ATCG]+$`
Digit	`^[0-9]$`
All Digits	`^[0-9]+$`
Digit Comma Digit	`^[0-9]+[,][0-9]+$`
Digit Dot Digit	`^[0-9]+[.][0-9]+$`
Capital	`^[A-Z]$`
All Capitals	`^[A-Z]+$`
Capital Low Alpha	`^[A-Z][a-z]+$`
Capital Mix Alpha	`^[A-Z]([A-Za-z]*[a-z][A-Za-z]*[A-Z][A-Za-z]*|$` `[A-Za-z]*[A-Z][A-Za-z]*[a-z][A-Za-z]*)$`
Low Mix Alpha	`^[a-z]([A-Za-z]*[a-z][A-Za-z]*[A-Z][A-Za-z]*|$` `[A-Za-z]*[A-Z][A-Za-z]*[a-z][A-Za-z]*)$`
Alpha Digit Alpha	`^[A-Za-z][0-9]+[A-Za-z]$`
Alpha Digit	`^[A-Za-z][0-9]+$`
Digit Alpha Digit	`^[0-9]+[A-Za-z][0-9]+$`
Digit Alpha	`^[0-9]+[A-Za-z]$`