Lattice: Data Adaptation for Named Entity Recognition on Tweets with Features-Rich CRF

Tian TIAN
Lattice / 1 Maurice Arnoux
92120 MONTROUGE
tian.tian@live.cn

Marco Dinarelli
Lattice / 1 Maurice Arnoux
92120 MONTROUGE
marco.dinarelli@ens.fr

Isabelle TELLIER
Lattice / 1 Maurice Arnoux
92120 MONTROUGE
isabelle.tellier@univ-paris3.fr

Abstract

This article describes our CRF named entity extractor for Twitter data. We first discuss some specificities of the task, with an example found in the training data. Then we present how we built our CRF model, especially the way features were defined. The results of these first experiments are given. We also tested our model with dev_2015 data and we describe the procedure we have used to adapt older Twitter data to the data available for this 2015 shared task. Our final results for the task are discussed.

1 Introduction

In this shared task, we have to extract 10 types of (or not typed) named entities in Twitter data. We have at our disposal two labelled corpora: train and dev. The first section shows some specificities of the data, from an example it contains. We then construct a CRF model for the task, using the software Wapiti. Our features for this CRF are chosen according to the state-of-the-art, they are described in the second section. The third section focuses on some experiments with train and dev and gives the obtained results. The fourth section is about the procedure we have used to build our final model, by applying a domain adaptation strategy. In the last section, we discuss some future work for this shared task.

2 Data Analysis

Although named entity recognition is a traditional task of natural language processing (NLP) which has given rise to a large body of works for written English (Finkel et al., 2005) or news wires in French (Stern and Sagot, 2010), the same task with Twitter data remains difficult (Ritter et al., 2011). This is not only because of the task itself, but also because of the way tweets are written.

Today wasz Fun cusz anna Came juss for me <3: hahaha

Figure 1: An example of tweet

Figure 1 shows an example of tweet. The correct sentence should be: Today was fun because Anna came just for me <3: hahaha. We can note the following phenomena:

- spelling mistakes: wasz (was), cusz (because), juss (just)
- confusion of upper/lower cases: Fun (fun), anna (Anna), Came (came)
- emoticon: <3
- interjection: hahaha

We remark here that the only name has no upper case letters whereas other words have upper cases (like "Fun", "Came"). So, it would be difficult for a named entity extractor to correctly detect this person name.

3 CRF Implementation and Features

3.1 CRF Features

We used the CRF implementation *Wapiti 1.5.0* [1] to create our CRF model. The optimization algorithm we chose was rprop+. The features for the tokens are all in unigrams and within a window of size 3 (previous token, current token and next token). The bigrams are only made of labels, characterizing label transitions. Table 1 shows the features we implemented. These templates have been chosen following (Suzuki and Isozaki, 2008), (Lavergne et al., 2010), (Nooralahzadeh et al., 2014) and (Constant et al., 2011)

[1]https://wapiti.limsi.fr

token value
fstUpper shortCap longCap mixCap hasUpper allUpper capType: combination of 6 binary values allLetter singleLetter tokenType: punctuation, 9, x or X
hasNumber allNumber isDecimal onePunct allPunct hasPunct longPunct hasQuotation hasAtLeast2periodes finishedByPeriode hasDash
lower returnUnicodeVector
isEmal isURL isRT isUSR isHashTag isDate isTime isAbbrev
prefixe_n, suffixe_n (n = 1..5)
postag in PTB: with binary values
category in Brown cluster: in binary tree

Table 1: CRF features

The capType features regroup 6 binary features: allUpper, shortCap, longCap, allLower, fstUpper, mixCap. The tokenType feature transforms a token into a ”skeleton”: in this skeleton, all numbers are replaced by 9, all letters in lower case by x, all letters in upper case by X and the punctuations remain unchanged. The part-of-speech tags (postags) of the Penn Tree Bank (PTB) (Marcus et al., 1993) generate 45 distinct features. Each tag in the PTB becomes a feature with a binary value. The ”category in Brown cluster” uses the result of Brown clustering (Brown et al., 1992) executed with 56,345,753 tweets available at `http://www.ark.cs.cmu.edu/TweetNLP/`. The class of each token is represented with 13 binary values. These values represent therefore a binary tree. Each value means one level in the binary tree. So we took the first value for each token, i.e. its category with only one level (two possible values).We then took the first two values of each token, resulting in the clustering of twitter tokens into four classes, etc. We took until all 13 values, to get the classes of the token at every level of the binary tree.

	precision	recall	FB1
dev	69.01%	33.15%	44.78%
dev_2015	43.26%	22.43%	29.54%

Table 2: Experiment results with model trained on train file

3.2 Use of Lexical Resources

As they were attached with the available baseline, we processed a set of entity dictionaries. We tried to associate these dictionaries with the 10 types of entities defined for the shared task. We deleted duplicated data (as we kept only cap.1000 but not cap.10 nor others, etc). Then we read every item of the lists. As some items (entities) contain more than one token, we extracted the first tokens (or the only token for one-token-entities) and the remaining ones before storing them into different lists. So, for every dictionary we had, we created 2 lists: a ”B-dictionary” and a ”I-dictionary”, preparing the BIO labelings. Finally, we integrated these dictionaries into the model by binary values. For each token, if it is present in a dictionary (B-dictionary or I-dictionary), its value for the corresponding feature is set to 1, and 0 otherwise. And we could always try with other ressources like FreeBase `https://www.freebase.com/` and dbpedia `http://dbpedia.org/`.

4 Some Experiments and Results

With the templates defined in the previous section, we used rprop+ as optimization algorithm in Wapiti and we did some experiments (only with the 10 distinct types of entities) with models trained with ”train” and tested on ”dev”, and later tested on ”dev_2015”. Table 2 shows some of these results.

5 CRF Model Training with Domain Data Adaptation

As we can see in the previous section, our first model performs poorly on dev_2105 data compared to dev. This suggests that the data in dev_2015 are very different from the data in dev and train. This intuition has indeed been confirmed by a quick data analysis.

As a consequence, we had the idea to perform a kind of domain data adaptation, inspired by the work of (Raymond and Fayolle, 2010). In this context, the data we want to adapt is called *source domain*. In our case, train and dev data play the role of this source domain. The role of *target domain* is played by the new version of tweet data provided for the shared task, that is dev_2105 data. The approach described in (Raymond and Fayolle, 2010) mixes together data from the source domain and from the target domain in order to train a CRF model. The originality of this approach consists in using more CRF features for the part of the data constituting the target domain than features for the data constituting the source domain. The consequence of this choice is that the CRF models learn word-label dependencies from both domains, but put much stronger importance (feature scores) on features in the target domain, since they are described by more information (features).

We annotated afterwards the training data, which we have already seen during the training phase, with such a model. If the model can apply stronger dependencies learned from the target-domain part of the training data, it will apply such dependencies performing thus the desired adaptation. Otherwise it will apply the dependencies learned from the source-domain part of the training data, thus keeping the old annotation.

We only applied an approximation of this domain adaptation procedure of (Raymond and Fayolle, 2010), because of a serious lack of time. In order to create our final model, we trained our first CRF model (with the templates mentioned in the previous section) with dev_2015. We then applied this first CRF model to train and dev to obtain train_crf and dev_crf. So, these data are labelled with our first CRF model. We got rid of the original labels for train and dev. And, in the end, we trained our final model (always with the same templates) with dev_2015, train_crf and dev_crf all together. We did the same procedure for the 10 types entities and for no typed data. Our results are described in Table 3.

	precision	recall	FB1
10 types	55.17%	9.68%	16.47%
no type	58.42%	25.72%	35.71%

Table 3: Results with model trained on dev_2015 then applied to train and dev files

Compared to results with dev_2015, we had a better precision, which confirms that the adaptation was worth doing. However we also had a much worse recall, which could be someway predicted since the dev_2015 data is much smaller than the training data. It thus creates a serious low covering problem. Such problem can be overcome by applying the exact adaptation procedure described in (Raymond and Fayolle, 2010), together with the use of more external resources (such as name lists).

6 Future work

In the future, we could do some proper experiments in cross validation with the training data, in order to find better templates, and find the best L1 and L2 regularization parameters of the CRF. We believe that correctly performing the adaptation procedure of (Raymond and Fayolle, 2010) and thus obtaining a better CRF model for our named entity extractor would lead to much better results.

References

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, December.

Matthieu Constant, Isabelle Tellier, Denys Duchier, Yoann Dupont, Anthony Sigogne, and Sylvie Billot. 2011. Intégrer des connaissances linguistiques dans un CRF : application à l'apprentissage d'un segmenteur-étiqueteur du français. In *TALN*, volume 1, page 321, Montpellier, France, June.

Jenny R. Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale crfs. In *Proceed-*

ings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pages 504–513, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *COMPUTATIONAL LINGUISTICS*, 19(2):313–330.

Farhad Nooralahzadeh, Caroline Brun, and Claude Roux. 2014. Part of speech tagging for french social media data. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 1764–1772.

Christian Raymond and Julien Fayolle. 2010. Reconnaissance robuste d'entités nommées sur de la parole transcrite automatiquement. In *Conférence Traitement automatique des langues naturelles, TALN'10*, Montréal, Québec, Canada, July. ATALA.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1524–1534, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rosa Stern and Benoît Sagot. 2010. Resources for named entity recognition and resolution in news wires. In *Entity 2010 Workshop at LREC 2010*.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using gigaword scale unlabeled data. In *In ACL*.