Recovering Casing and Punctuation using Conditional Random Fields

Marco Lui, Li Wang
NICTA VRL
Department of Computing and Information Systems
University of Melbourne
mhlui@unimelb.edu.au, li@liwang.info

Abstract

This paper describes the winning entry to the ALTA Shared Task 2013. The theme of the shared task was recovery of casing and punctuation information from degraded English text. We tackle the task as a sequential labeling problem, jointly learning the casing and punctuation labels. We implement our sequential classifier using conditional random fields, trained using linguistic features extracted with off-the-shelf tools with simple adaptations to the specific task. We show the improvement due to adding each feature we consider, as well as the improvment due to utilizing additional training data beyond that supplied by the shared task organizers.

1 Introduction

The ALTA Shared Task 2013[1] required participants to recover casing and punctuation information from degraded English text. The task focused specifically on English text where casing and punctuation were entirely absent (i.e. all characters had been reduced to lowercase, and all non-alphanumeric symbols had been omitted). Participants were required to submit word-level predictions, identifying on a word-by-word basis whether (1) the word in its original form has any characters in uppercase, and (2) whether the word is followed by one of a closed set of punctuation marks.

Our approach to the task treats the task as a sequential labeling problem, a common technique in modern natural language processing (NLP). We implement a word-level sequential classifier using conditional random fields (CRFs), a sequential labeling technique that has been successfully applied to a variety of NLP problems. We make use of a number of linguistic features extracted using off-the-shelf NLP tools, with some simple adaptations to the specific problem at hand. We also make use of additional training data beyond that supplied by the shared task organizers, and show that this has a large impact on the final result. Overall, our approach was the most effective in the shared task by a reasonable margin, outperforming 50 other participants (22 of which outperformed the organizer-supplied baseline).

[1]http://www.alta.asn.au/events/sharedtask2013

2 Task Description

Participants were required to implement an automated system that could accept as input a stream of words without any casing or punctuation. On the basis of this stream, participants were asked to infer which words in the stream should (1) have some form of casing, and (2) be followed by punctuation. The setting was meant to simulate scenarios whereby such information is missing, such as from audio transcriptions, or from user-generated content in social media.

For purposes of the shared task, the text supplied to participants was an automatically-converted version of original English documents with "standard" casing and punctuation. These documents constitute the "original" goldstandard, and were not supplied to participants. Participants only received the transformed (i.e. lowercased and punctuation removed) version. Additionally, for training documents, participants were provided with a "simplified" goldstandard consisting of word-level annotation of which words in the original text (1) contained any uppercase characters and/or (2) were followed by a punctuation mark. Participants were only required to provide predictions corresponding to the "simplified" goldstandard, not to restore the text to the "original" goldstandard.

The shared task was hosted on Kaggle[2], a web platform for crowdsourced competitive data analytics. Kaggle provides the infrastructure for running a shared task, including user management, results tabulation and discussion forums. For evaluation, the macro-averaged F-score across the two types of word-level labels (casing and punctuation) was used. Participants were able to submit two attempts daily and received immediate feedback on the score obtained, which was also posted on a publicly-visible ranking known as the leaderboard. The initial data released to participants consisted of a "basic" training set of ≈66k words and a test set of ≈64k words, with a further ≈300k words from English Wikipedia. The source of the "basic" training data and the test data was not officially revealed, but manual inspection showed that it was newswire data.

3 Methodology

Our main focus was to treat the task as a sequential labeling problem, which in recent NLP research has frequently been tackled using conditional random fields (Lafferty et al., 2001), a class of probabilistic graphical model that integrates information from multiple features, and has enjoyed success in tasks such as shallow parsing (Sha and Pereira, 2003). We apply CRFs to learn a sequential labeler for the shared task on the basis of 4 automatically-extracted features: the surface form of the word (WORD), the part-of-speech of the word in context (POS), IOB tags for verb and noun phrases (CHUNK) and named entity recognition with NE type such as person (NER).

POS, CHUNK and NER were extracted using `SENNA` v3.0 (Collobert et al., 2011), an off-the-shelf shallow parsing system based on a neural network architecture. We chose `SENNA` for this task due to its near state-of-the-art accuracy on tagging tasks and relatively fast runtime. One challenge in using `SENNA` is that it expects input to be segmented at the sentence level. However, this information is obviously missing from the stream-of-words provided for the shared task. Restoring sentence boundaries is a non-trivial task, and automatic methods (e.g. Kiss and Strunk, 2006) typically make use of casing and punctuation information (e.g. a period followed by a capitalized word is highly indicative of a sentence boundary). In order to obtain POS, CHUNK and NER tags from `SENNA`, we segmented the text using a fixed-size sliding window approach. From the original stream of words, we extracted pseudo-sentences consisting of sequences of 18 consecutive words. The start of each sequence was offset from the previous sequence by 6 words, resulting in each word in the original appearing in three pseudo-sentences (except the words right at the start and end of the stream). `SENNA` was used to tag each pseudo-sentence, and the final tag assigned to each word was the majority label amongst the three. The rationale behind this overlapping window approach was to allow each word to appear near the beginning, middle, and end of a pseudo-sentence, in case sentence position had an effect on `SENNA`'s predictions. In practice, for over 92% of words all predictions were the same. We did not carry out an evaluation of the accuracy of `SENNA`'s predictions due to a lack of gold-standard data, but anecdotally we observed that the POS and CHUNK output generally seemed sensible. We also observed that for NER, the output appeared to achieve high precision but rather low recall; this is likely due to `SENNA` normally utilizing casing and punctuation in carrying out NER.

[2]`http://www.kaggle.com`

3.1 Sequence Labeler Implementation

To implement our sequence labeler, we made use of `CRFSUITE` version 0.12 (Okazaki, 2007). `CRFSUITE` provides a set of fast command-line tools with a simple data format for training and tagging. For our training, we used L2-regularized stochastic gradient descent, which we found to converge faster than the default limited-memory BFGS while attaining comparable extrema. We also made use of the supplied tools to facilitate sequential attribute generation. We based our feature template on the example template included with `CRFSUITE` for a chunking task. For WORD, single words are considered for a (-2,2) context (i.e. two words before and two words after, as well as word bigrams including the current word. For POS, CHUNK and NER, we used a (-2,2) context for single tags, bigrams and trigrams. This means that for word bigrams, we also utilized features that captured (1) two words before, and (2) two words after, in both cases excluding the target word itself.

We treated the task as a joint learning problem over the casing and punctuation labels, reasoning that the two tasks are highly mutually informative,

Feature	Case	Punc	Avg
WORD	0.469	0.453	0.461
+POS	0.607	0.577	0.592
+CHUNK	0.627	0.592	0.610
+NER	0.636	0.597	0.617

Table 1: F-score attained by adding each feature incrementally, using only the organizer-supplied `train` data, broken down over the two component tasks. The average of the two components is the metric by which the shared task was judged.

as certain punctuation strongly influences the casing in the immediate context, e.g. a period often ends a sentence and thus a word followed by a period is expected to be followed by a capitalized word. We trained the labeler to output four distinct labels: (FF) the word should not be capitalized and should not be followed by punctuation, (FT) the word should not be capitalized and should be followed by punctuation, (TF) the word should be capitalized and should not be followed by punctuation, and (TT) the word should be capitalized and should be followed by punctuation.

We applied the same pseudo-sentence segmentation to the text that was carried out to pre-process the word stream for use with SENNA, and again the majority label amongst the three predictions was used as the final output.

Table 1 summarizes the effect of adding each feature to the system, using only the "basic" training data. The result attained using only word features is marginally better than the organizer-supplied hidden Markov model baseline (0.461 vs 0.449). The biggest gain is seen by adding POS, and further improvements are achieved by using CHUNK and NER.

4 Additional Data Used

The feature set and labeler setup we outline above, combined with the "basic" training data provided by the shared task organizers resulted in a system that attained F-score of 0.617, comfortably exceeding the organizer-posted baseline of 0.449 utilizing a hidden Markov model on the same training data. Adding the organizer-provided Wikipedia data further improved this result to 0.664.

Banko and Brill (2001) observed that for confusion set disambiguation, the performance of learners can benefit significantly from increasing the training set size substantially. Confusion set disambiguation is representative of many NLP tasks, in that it relies on statistical models of word sequences. Due to the large vocabulary, only a very small proportion of valid sequences is observed in any given dataset, and hence adding more data allows us to better estimate the parameters of the model. Our task suffers the same problem, and we thus expect that increasing the quantity of training data will increase performance, a hypothesis is supported by the results obtained by adding the Wikipedia data to the "basic" training data.

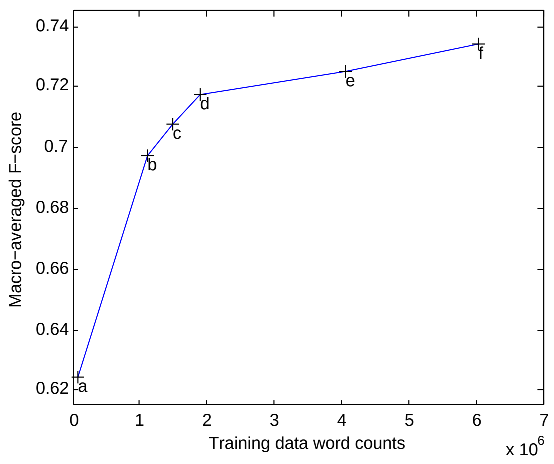

Figure 1: Effect of adding training data. Left to right, each point represents the cumulative addition of training data. (i.e. `a` uses only `train`, and `f` uses all the data in Table 2.) Datasets are added in the order listed in Table 2.

Table 2 summarizes the training data we used in our best-scoring system. As previously discussed, manual inspection of the training data suggested that it was derived from a newswire source. We thus sought additional training data from newswire sources. We made use of all the treebanked Wall Street Journal (WSJ) data from the Penn Treebank (Marcus et al., 1993), as well as a sample of data from RCV1 (Lewis et al., 2004). We opted not to use the organizer-supplied sample of English Wikipedia in our further experiments, instead utilizing our own sample which gave us access to a larger number of documents. For purposes of discussion, we divide the Wikipedia and Reuters data into two partitions each.

Table 2 also shows the score attained using each dataset individually as training data. Here we see the effect of affinity between datasets; the organizer-supplied `train` is assumed to be the most similar to the test data, and hence attains a

Label	Source	Size (words)	F-score
train	Competition Organizer	66371	0.617
wsj	Wall Street Journal	1082959	0.617
enwiki1	English Wikipedia	372547	0.553
enwiki2		383368	0.564
reut1	Reuters RCV1	2244959	0.604
reut2		2052119	0.606

Table 2: Datasets used in our best-scoring system. F-score is the macro-averaged F-score for the task attained by using each dataset individually as training data, using the full feature set.

high score despite having a relatively low word count. An interesting comparison can be made with the wsj result, where much more data is required to attain the same score. Figure 1 illustrates the effect of training models on successively greater amounts of training data. The order in which data was added for Figure 1 was largely arbitrary, though several trends can be observed: adding each successive dataset reduces the improvement obtained per word added, and for each of Wikipedia and RCV1, adding the same amount of data results in a near-linear increase in performance.

5 Negative Results

CRFSUITE is not able to train a model incrementally, and so adding additional training data required retraining the entire model from scratch. We attempted to circumvent this problem by implementing a voting system over models from different partitions of the training data, but found that the performance was not competitive with the monolithic model.

We also tested crfsgd,[3] but found that CRFSUITE was faster and attained slightly better F-score for comparable setups. We believe that the default parametrization of the CRF model differs slightly between tools, but did not investigate.

Much of the additional training data we used was pre-tokenized in Penn Treebank format, which utilizes special sequences to represent different types of brackets. As the organizer-supplied data did not follow this convention, we tested "de-escaping" the Penn Treebank brackets (e.g. converting -LCB- back to {). Surprisingly, we found that this actually reduced our F-score, though the reasons for this remain unclear.

We introduced gazetteer features on the basis of observing that despite low recall, the NER features produced by SENNA had an overall positive impact on our task score. Our gazetteer features are based on word lists of common named entities. For place names, we used data from GeoNames,[4] and for first names we used a wordlist included with Apple's OS X. We used each wordlist to produce a single Boolean feature indicating whether the word was present in the given wordlist. This approach showed some promise in internal cross-validation, but was abandoned as we found that it did not improve our score on the leaderboard.

6 Further Work

The CRF feature template (i.e. the set of rules for generating sequential features) that we used was derived with minimal modification from a template for a chunking task. Tuning the template to this task may yield further improvements. Furthermore, CRFs are able to provide a probability distribution over labels, and this information may be useful in weighting a voting approach to model combination. Finally, the amount of data we used was primarily limited by computation resources available, so it is likely that increasing data quantity will further improve performance.

7 Conclusion

In this paper we detailed the winning entry to the ALTA 2013 Shared Task. We treat the task as a sequence labeling problem, jointly learning the casing and punctuation labels. We implemented a classifier using conditional random fields, trained using linguistic features extracted using off-the-shelf tools, using a simple windowing approach to generate pseudo-sentences. We find that the linguistic features out-perform simple word features, and that further improvements can be made by further adding training data. We briefly discussed

[3]http://leon.bottou.org/projects/sgd

[4]http://www.geonames.org

negative results in our development process, and outlined some avenues for further work.

Acknowledgments

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the ACL and 10th Conference of the EACL (ACL-EACL 2001)*, pages 26–33. Association for Computational Linguistics, Toulouse, France.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289. Williamstown, USA.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397.

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.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs). URL `http://www.chokkan.org/software/crfsuite/`.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 3rd International Conference on Human Language Technology Research and 4th Annual Meeting of the NAACL (HLT-NAACL 2003)*, pages 213–220. Edmonton, Canada.