Overview of the 2013 ALTA Shared Task

Diego Molla
Department of Computing
Macquarie University
Sydney, NSW 2109
diego.molla-aliod@mq.edu.au

Abstract

The 2013 ALTA shared task was the fourth in the ALTA series of shared tasks, where all participants attempt to solve the same problem using the same data. This year's shared task was based on the problem of restoring casing and punctuation. As with last year, we used Kaggle in Class as the framework to submit the results and maintaining a leaderboard. There was a strong participation this year, with 50 teams participating, of which 21 teams submitted results that improved on the published baseline. In this overview we describe the task, the process of building the training set, and the evaluation criteria, and present the results of the submitted systems. We also comment on our experience with using Kaggle in Class.

1 Introduction

There are many situations when a piece of English text does not have information about capitalisation or punctuation. These situations may arise, for example, when the text is the result of automatic transcription from speech, or when the text has been typed in a hurry, such as when taking quick notes or in quick responses in Web forums, or when using media such as text messages. If such a text is to be processed automatically, one may want to restore the missing capitalisation and punctuation, so that the text can be processed using conventional text processing tools and resources. It has been shown that introducing a preliminary step that automatically restores case information improves the results of machine translation (Lita et al., 2003) and information extraction from speech transcripts (Niu et al., 2004).

The task of case and punctuation restoration takes text such as the following as input:

> ... stored at the ucla television archives the archived episodes were telecast march 8 16 and 24 1971 april 1 and ...

The expected output is:

> ... stored at the UCLA Television Archives. The archived episodes were telecast: March 8, 16, and 24, 1971, April 1 and ...

An interesting feature of case and punctuation restoration is that training data can be obtained cheaply. One only needs to take a piece of text and remove case and punctuation. By doing this we can obtain the input data (the text with the case and punctuation information removed) and the target data (the original text). It has been observed that using this approach to generate training data suffices to obtain reasonable results (Niu et al., 2004), and this observation agrees with the results obtained in this shared task, as we will show in this paper.

2 The 2013 ALTA Shared Task

Case and punctuation restoration can be formulated as a text classification task. Baldwin and Joseph (2009) used a multi-label classification approach where a word can have multiple labels, each label indicating the information to be restored in the word. For example, the set of labels `CAP1+FULLSTOP+COMMA` indicates that the word has the first character as uppercase and is followed by a full stop and a comma. Thus, if the word was *corp*, the labels indicate that the word should be restored to *Corp.,*. There is a label `ALLCAPS` to indicate that all letters in the word need to be uppercased, and the specific label `NOCHANGE` indicates that the word does not need any special restoration.

Diego Molla. 2013. Overview of the 2013 ALTA Shared Task. In *Proceedings of Australasian Language Technology Association Workshop*, pages 132−136.

```
ID WORD
255 stored
256 at
257 the
258 ucla
259 television
260 archives
261 the
262 archived
263 episodes
264 were
265 telecast
266 march
267 8
268 16
269 and
270 24
271 1971
271 april
273 1
274 and
```

Figure 1: Example of input text

The ALTA shared tasks primarily target university students with programming experience, but without necessarily much background on text processing techniques. For this reason, the 2013 ALTA shared task is a simplification of the more general task of case and punctuation restoration. The participants are asked to build automatic systems that predict where a word should have any of its characters in uppercase, and whether the word is followed by any punctuation mark. They are not required to predict which specific characters are in uppercase, or which specific punctuation marks are attached to the word. Furthermore, the only punctuation characters to consider for the task are:

,.;:?!

The shared task was presented as a task of multi-label classification with two possible labels: `Case` and `Punct`. A word could be labelled with any of the labels, both, or none. The participants were given text that had been tokenised, all case removed, and all punctuation (,.;:?!) removed. Figures 1 and 2 show an example of input text and the target, using the specific format required for the task. According to the example in the figures, word with ID 258 (*ucla*) has at least one character in uppercase, and word with ID 260 (*archives*) has uppercase characters and punctuation marks.

```
Id,documents
Case,258 259 260 261 266 272
Punct,260 265 267 268 270 271
```

Figure 2: Example of target output

3 The Training and Test Sets

We used the data by Baldwin and Joseph (2009) to produce a training set and two test sets, plus text from Wikipedia to produce additional training data.

The data by Baldwin and Joseph (2009) are from the AP Newswire (APW) and New York Times (NYT) sections of the English Gigaword Corpus. Of the two test sets, one was used as a "public" test set that participants could use to check their progress in the development of their systems. The participants did not have access to the target output but they could submit the output of their systems and they would receive instant feedback with the results and how they compare against other participants in the leaderboard. The second test set was a "private" test set that was used to determine the final scores. By having separate "public" and "private" test sets we aimed to reduce the risk of some systems overfitting to the actual test set, since each participant could submit up to two runs every day. As training data we used the third partition from Baldwin and Joseph (2009) plus an extract from Wikipedia.

To download the Wikipedia text, shuffle the paragraphs, and split the contents into smaller files we used a method and scripts based on a blog post[1]. We then used the Python NLTK toolkit (Bird et al., 2009) to tokenise the words. We lowercased the tokens and removed those that matched our list of punctuation marks.

The Wikipedia training data consisted of 18 files with a total of 306,445 words. The data from Baldwin and Joseph (2009) consisted of a "train" file with 66,371 words, the "public" test file with 64,072 words, and the "private" test file with 65,903 words.

[1] http://blog.afterthedeadline.com/2009/12/04/generating-a-plain-text-corpus-from-wikipedia/

4 Kaggle in Class

Kaggle[2] is a Web-based framework for the creation of data-driven competitions. Kaggle provides the means for competition designers to upload the training data to distribute among participants, and private test data that is used when participants submit a run. The winner is automatically determined as the team who produces the highest score on the private data.

Kaggle targets data analytics companies who can use this framework to hire data modelling specialists. At a cost, Kaggle offers their outreach solutions to find participants to the competitions. Kaggle in Class[3] is a free variant that allows class instructors and organisations hold shared tasks without incurring in management fees. The main differences between Kaggle and Kaggle in class are that Kaggle in Class has limited support services, it has less flexibility in its setup, and does not use Kaggle's outreach services.

A very attractive feature of Kaggle in Class is its ability to maintain a leaderboard that allows participants to keep track of how they stand against other participants. Participants can submit results up to two times a day using the test set. The test set has a "public" partition that is used to display the participants' results in a public leaderboard, and a "private" partition that is used for the final rating. Kaggle in Class also includes a competition-specific Web forum for communication among participants and between organisers and participants.

Kaggle in Class offers an array of evaluation metrics. For this shared task we used the macro-averaged F1 score, which allowed the evaluation of the output of multi-label classification tasks by averaging the class-specific F1 score. For example, if the target output is as shown in Table 2, and a system returns the following output:

```
Id,documents
Case,258 259 260 262 270
Punct,259 260 265 270
```

Then the computed F-scores are:

Case: P = 3/5; R = 3/6; F1 = 0.54

Punct: P = 3/4; R = 3/6; F1 = 0.6

Final score: (0.54+0.6)/2 = **0.57**

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

[3]http://inclass.kaggle.com

Training data	F1 (private)	F1 (public)
Train data	0.2895	0.4355
Wikipedia 0-5	0.2761	0.4077
Wikipedia 0-10	0.2791	0.4173
Wikipedia 0-17	0.2789	0.4226
Train + Wikipedia	0.2876	0.4493

Table 1: Impact of training data on the baseline

5 The Baseline

We built a simple baseline and made the code available to the participants. The baseline was written in Python and it used NLTK's Hidden Markov Model (HMM) trained on a single-labelling variant of the task. The single-labelling variant had 4 classes, one for each combination of the `Case` and `Punct` labels. Table 1 shows the result of the system when trained on the "train" data, and when trained on increasing portions of the Wikipedia train data. We observed that the "train" data was better than the Wikipedia train data, but adding more Wikipedia data might have improved the results. These findings are in line with the findings of the winner of the shared task (Lui and Wang, 2013, in these proceedings), who observed better results as they added more training data. We also observed a considerable difference between the results of the "public" and the "private" test. This may indicate that these two partitions do not represent each other, although as we will observe in Section 6, the results of the top participants are consistent across the two test partitions.

6 Results

The specific format required for submitting the results to Kaggle in Class using the macro-averaged F1 score did not allow to specify "public" and "private" partitions on the test file. For this reason we created two Kaggle in Class competitions: a "public" competition where participants could submit and observe the results in the leaderboard, and a "private" competition for the final results. However, it turned out that many participants who submitted to the public competition did not submit to the private competition. Table 2 shows the results of all teams in the public competition, including the baseline (in **boldface**), and a test system that used the same training data as the baseline plus the private set. Table 3 shows the results of the private submissions. All team names

have been anonymised, and we have kept the same names in both tables.

We can observe a number of participants with the same score as the baseline. Since the code of the baseline was made available, it is likely that these participants simply ran the baseline. The two top participants in the public competition submitted to the private competition and obtained similar results. We could not locate one of the two remaining participants of the private competition in the public competition, but we observed a very different score for one participant ("Team A") across the two competitions. Unfortunately too few participants submitted to the private competition to confirm whether the "private" test data tends to lower the scores of poor submissions. Given that our baseline also had a reduced score with the private test data, it appears that this is the case.

7 Conclusions

This year's shared task had a much larger participation than in past tasks. The main reason for this was the use of the task as part of an assignment of a Masters unit at University of Melbourne.

A large percentage of participants outdid our baseline task, and the top participants did much better than our baseline. The best results outperformed the results reported by Baldwin and Joseph (2009), who achieved an F-score of 0.619. Even though our shared task was a simplification, it shows the good skills of the top participants, who were PhD and Masters students. The top team used Conditional Random Fields and is described elsewhere in these proceedings (Lui and Wang, 2013).

We observed that a key component to improve the results was the use of additional training data. Since training data is easy to obtain for this task, the only issue would be the increasing computational costs involved in adding additional data.

The use of Kaggle in Class was very convenient due to its easy interface for the creation of the task, its ability to maintain a leaderboard, and its automatic partition into public and private test data. Unfortunately, the actual evaluation score that we used, macro-averaged F1, did not allow the automatic partition into public and private test sets. Our solution was to create an additional "private" competition, but very few participants submitted to the new competition, possibly because they could observe that they were not at the top of

Rank	Team	Score
1	Winner	0.73763
2	Second	0.68360
3	(anonymous)	0.63232
4	(anonymous)	0.63109
5	(anonymous)	0.60251
6	(anonymous)	0.60147
7	(anonymous)	0.59517
8	(anonymous)	0.58332
9	(anonymous)	0.56832
10	(anonymous)	0.56747
11	(anonymous)	0.55793
12	(anonymous)	0.55606
13	(anonymous)	0.55087
14	(anonymous)	0.52261
15	(anonymous)	0.51954
16	(anonymous)	0.51167
17	(anonymous)	0.49311
18	(anonymous)	0.47622
19	(*test system*)	0.46667
20	(anonymous)	0.46490
21	(anonymous)	0.45986
22	(anonymous)	0.45291
	Baseline	**0.44930**
23	(8 systems)	0.44930
32	(anonymous)	0.44914
33	(anonymous)	0.42710
34	(anonymous)	0.42257
35	(anonymous)	0.41692
36	(anonymous)	0.40239
37	(anonymous)	0.38812
38	(anonymous)	0.38113
39	(anonymous)	0.32594
40	(anonymous)	0.32320
41	(anonymous)	0.30988
42	(anonymous)	0.29891
43	(anonymous)	0.29304
44	(anonymous)	0.27642
45	(anonymous)	0.23504
46	Team A	0.23108
47	(anonymous)	0.21930
48	(anonymous)	0.21771
49	(anonymous)	0.21291
50	(anonymous)	0.20226
51	(anonymous)	0.13397
52	(anonymous)	0.00000

Table 2: Results of the public submissions

Rank	Team	Score
1	**Winner**	**0.73660**
2	Second	0.64934
3	(anonymous)	0.30037
4	Team A	0.07656

Table 3: Final results of the private submissions

the leaderboard.

References

Timothy Baldwin and Manuel Paul Anil Kumar Joseph. 2009. Restoring Punctuation and Casing in English Text. In *AI '09 Proceedings of the 22nd Australasian Joint Conference on Advances in Artificial Intelligence*, pages 547–556.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. tRuEcasIng. In *Proc. ACL 2003*, pages 152–159.

Marco Lui and Li Wang. 2013. Recovering casing and punctuation using conditional random fields. In *Proceedings of the 2013 Australasian Language Technology Workshop*, Brisbane, Australia.

Cheng Niu, Wei Li, Jihong Ding, and Rohiri K. Srihari. 2004. Orthographic Case Restoration Using Supervised Learning without Manual Annotation. *International Journal on Artificial Intelligence Tools*, 13(1):141–156, March.