ner and pos when nothing is capitalized

Stephen Mayhew, Tatiana Tsygankova, Dan Roth
University of Pennsylvania
Philadelphia, PA, 19104
{mayhew, ttasya, danroth}@seas.upenn.edu

Abstract
For those languages which use it, capitalization is an important signal for the fundamental NLP tasks of Named Entity Recognition (NER) and Part of Speech (POS) tagging. In fact, it is such a strong signal that model performance on these tasks drops sharply in common lowercased scenarios, such as noisy web text or machine translation outputs. In this work, we perform a systematic analysis of solutions to this problem in English, modifying only the casing of the train or test data using lowercasing and truecasing methods. While prior work and first impressions might suggest training a caseless model, or using a truecaser at test time, we show that the most effective strategy is a concatenation of cased and lowercased training data, producing a single model with high performance on both cased and uncased text. As shown in our experiments, this result holds across tasks and input representations. Finally, we show that our proposed solution gives an 8% F1 improvement in mention detection on noisy out-of-domain Twitter data.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Task</th>
<th>Cased</th>
<th>Uncased</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiLSTM-CRF w/ ELMo</td>
<td>NER</td>
<td>92.45</td>
<td>34.46</td>
</tr>
<tr>
<td>BiLSTM-CRF w/ ELMo</td>
<td>POS</td>
<td>97.85</td>
<td>88.66</td>
</tr>
</tbody>
</table>

Table 1: Modern tools trained on cased data perform well on cased test data, but poorly on uncased (lowercased) test data. For NER, we evaluate on the test set of CoNLL 2003, and the scores are reported as F1. For POS, we evaluate on PTB sections 22-24, and the scores represent accuracy. ELMo refers to contextual representations from Peters et al. (2018).

Table 1 demonstrates how popular modern systems trained on cased data perform well on cased data, but suffer dramatic performance drops when evaluated on lowercased text.

Prior solutions have included models trained on lowercase text, or models that automatically recover capitalization from lowercase text, known as truecasing. There has been a substantial body of literature on the effect of truecasing applied after speech recognition (Gravano et al., 2009), machine translation (Wang et al., 2006), or social media (Nebhi et al., 2015). A few works that evaluate on downstream tasks (including NER and POS) show that truecasing improves performance, but they do not demonstrate that truecasing is the best way to improve performance.

In this paper, we evaluate two foundational NLP tasks, NER and POS, on cased text and lowercased text, with the goal of maximizing the average score regardless of casing. To achieve this goal, we explore a number of simple options that consist of modifying the casing of the train or test data. Ultimately we propose a simple preprocessing method for training data that results in a single model with high performance on both cased and uncased datasets.

1 For POS tagging, this happens in tagsets that explicitly mark proper nouns, such as the Penn Treebank tagset.
2 Related Work

This problem of robustness in casing has been studied in the context of NER and truecasing.

Robustness in NER. A practical, common solution to this problem is summarized by the Stanford CoreNLP system (Manning et al., 2014): train on uncased text, or use a truecaser on test data. We include these suggested solutions in our analysis below.

In one of the few works that address this problem directly, Chieu and Ng (2002) describe a method similar to co-training for training an upper case NER, in which the predictions of a cased system are used to adjudicate and improve those of an uncased system. One difference from ours is that we are interested in having a single model that works on upper or lowercased text. When tagging text in the wild, one cannot know a priori if it is consistently cased or not.

Truecasing. Truecasing presents a natural solution for situations with noisy or uncertain text capitalization. It has been studied in the context of many fields, including speech recognition (Brown and Coden, 2001; Gravano et al., 2009), and machine translation (Wang et al., 2006), as the outputs of these tasks are traditionally lowercased.

Lita et al. (2003) proposed a statistical, word-level, language-modeling based method for truecasing, and experimented on several downstream tasks, including NER. Nebhi et al. (2015) examine truecasing in tweets using a language model method and evaluate on both NER and POS.

More recently, a neural model for truecasing has been proposed by Susanto et al. (2016), in which each character is associated with a label U or L, for upper and lower case respectively. This neural character-based method outperforms word-level language model-based prior work.

3 Truecasing Experiments

We use our own implementation of the neural method described in Susanto et al. (2016) as the truecaser used in our experiments. Briefly, each sentence is split into characters (including spaces) and modeled with a 2-layer bidirectional LSTM, with a linear binary classification layer on top.

We train the truecaser on a dataset from Wikipedia, originally created for text simplification (Coster and Kauchak, 2011), but commonly used for evaluation in truecasing papers (Susanto et al., 2016). This task has the convenient property that if the data is well-formed, then supervision is free. We evaluate this truecaser on several data sets, measuring F1 on the word level (see Table 2). At test time, all text is lowercased, and case labels are predicted.

First, we evaluate the truecaser on the same test set as Susanto et al. (2016) in order to show that our implementation is near to the original. Next, we measure truecasing performance on plain text extracted from the CoNLL 2003 English (Tjong Kim Sang and De Meulder, 2003) and Penn Treebank (Marcus et al., 1993) train and test sets. These results contain two types of errors: idiosyncratic casing in the gold data and failures of the truecaser. However, from the high scores in the Wikipedia experiment, we suppose that much of the score drop comes from idiosyncratic casing. This point is important: if a dataset contains idiosyncratic casing, then it is likely that NER or POS models have fit to that casing (especially with these two wildly popular datasets). As a result, truecasing, since it can’t recover these idiosyncrasies, is not likely to be the best plan.

Notably, the scores on CoNLL are especially low, likely because of elements such as titles, bylines, and documents that contain league standings and other sports results written in uppercase. The higher scores on Penn Treebank corpus suggest that the capitalization standards are more traditional. Many errors are where the truecaser fails to correctly capitalize such words as “Federal” or “Central”. In addition, there are many

<table>
<thead>
<tr>
<th>System</th>
<th>Test set</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Susanto et al., 2016)</td>
<td>Wikipedia</td>
<td>93.19</td>
</tr>
<tr>
<td>BiLSTM</td>
<td>Wikipedia</td>
<td>93.01</td>
</tr>
<tr>
<td></td>
<td>CoNLL Train</td>
<td>78.85</td>
</tr>
<tr>
<td></td>
<td>CoNLL Test</td>
<td>77.35</td>
</tr>
<tr>
<td></td>
<td>PTB 01-18</td>
<td>86.91</td>
</tr>
<tr>
<td></td>
<td>PTB 22-24</td>
<td>86.22</td>
</tr>
</tbody>
</table>

Table 2: Truecaser word-level performance on English data. This truecaser is trained on the Wikipedia corpus. Wikipedia refers to the test set from Coster and Kauchak (2011). CoNLL Test refers to testb. PTB is the Penn Treebank.
occasions where the truecaser fails to capitalize named entities, for example “Mr. susulu”.

4 Methods

In this section, we introduce our proposed solutions. In all experiments, we constrain ourselves to only change the casing of the training or testing data with no changes to the architectures of the models in question. This isolates the importance of dealing with casing, and makes our observations applicable to situations where modifying the model is not feasible, but retraining is possible.

Our experiments aim to answer the extremely common situation in which capitalization is noisy or inconsistent (as with inputs from the internet). In light of this goal, we evaluate each experiment on both cased and lowercased test data, reporting individual scores as well as the average. Our experiments on lowercase text can also give insight on best practices for when test data is known to be all lowercased (as with the outputs of some upstream system).

We experiment on five different data casing scenarios described below.

1. **Train on cased** Simply apply a model trained on cased data to unmodified test data, as in Table 1.

2. **Train on uncased** Lowercase the training data and retrain. At test time, we lowercase all test data. If we did not do this, then scores on the cased test set would suffer because of casing mismatch between train and test. Since lowercasing costs nothing, we can improve average scores this way. As such, cased and uncased test data will have the same score.

3. **Train on cased+uncased** Concatenate original cased and lowercased training data and retrain a model. Test data is unmodified. Since this concatenation results in twice the number of training examples than other methods, we also experimented with randomly lowercasing 50% of the sentences in the original training corpus. We refer to this experiment as 3.5 Half Mixed. We also tried ratios of 40% and 60%, but these were slightly worse than 50% in our evaluations.

4. **Train on cased, test on truecased** Do nothing to the train data, but truecase the test data. Since we lowercase text before truecasing it, the cased and uncased test data will have the same score.

5. **Truecase train and test** Truecase the train data and retrain. Truecase the test data also. As in experiment 4, cased and uncased test data will have the same score.

One way to look at these experiments is as dropout for capitalization, where a sentence is lowercased with respect to the original with probability $p \in [0, 1]$. In experiment 1, $p = 0$. In experiment 2, $p = 1$. In experiment 3, $p = 0.5$. Our implementation is somewhat different from standard dropout in that our method is a preprocessing step, not done randomly at each epoch.

5 Experiments

Before we show results, we will describe our experimental setup. We emphasize that our goal is to experiment with strong models in noisy settings, not to obtain state-of-the-art scores on any dataset.

5.1 NER

We use the standard BiLSTM-CRF architecture for NER (Ma and Hovy, 2016), using an AllenNLP implementation (Gardner et al., 2018). We experiment with pre-trained contextual embeddings, ELMo (Peters et al., 2018), which are generated for each word in a sentence, and concatenated with GloVe word vectors (lowercased) (Pennington et al., 2014), and character embeddings. ELMo embeddings are trained with cased inputs, meaning that there will be some mismatch when generating embeddings for uncased text.

In all experiments, we train on English CoNLL 2003 Train data (Tjong Kim Sang and De Meulder, 2003) and evaluate on the CoNLL 2003 Test data (testb). We always evaluate on two different versions: the original version, and a version with all casing removed (e.g. everything lowercase).

5.2 POS Tagging

We use a neural POS tagging model built with a BiLSTM-CRF (Ma and Hovy, 2016), and GloVe embeddings (Pennington et al., 2014), character embeddings, and ELMo pre-trained contextual embeddings (Peters et al., 2018).

As our experimental data, we use the Penn Treebank (Marcus et al., 1993), and follow the training splits of (Ling et al., 2015), namely 01-18 for train,
6 Results

Results for NER are shown in Table 3, and results for POS are shown in Table 4. There are several interesting observations to be made.

Primarily, our experiments show that the approach with the most promising results was experiment 3: training on the concatenation of original and lowercased data. Lest one might think this is because of the double-size training corpus, results from experiment 3.5 are either in second place (for NER) or slightly ahead (for POS).

Conversely, we show that the folk-wisdom approach of truecasing the test data (experiment 4) does not perform well. The underwhelming performance can be explained by the mismatch in casing standards as seen in Section 3. However, experiment 5 shows that if the training data is also truecased, then the performance is good, especially in situations where the test data is known to contain no case information.

Training only on uncased data gives good performance in both NER and POS – in fact the highest performance on uncased text in POS – but never reaches the overall average scores from experiment 3 or 3.5.

We have repeated these experiments for NER in several different settings, including using only static embeddings, using a non-neural truecaser, and using BERT uncased embeddings (Devlin et al., 2019). While the relative performance of the experiments varied, the conclusion was the same: training on cased and uncased data produces the best results.

When using uncased BERT embeddings, we found that performance on the uncased test set (U) was typically higher than that of ElMo, while the maximum performance on the cased test set (C) was typically lower. This again exemplifies the challenge of using capitalization as a signal while being robust to its absence.

7 Application: Improving NER Performance on Twitter

To further test our results, we look at the Broad Twitter Corpus (Derczynski et al., 2016), a dataset comprised of tweets gathered from a broad variety of genres, and including many noisy and informal examples. Since we are testing the robustness of our approach, we use a model trained on CoNLL 2003 data. Naturally, in any cross-domain experiment, one will obtain higher scores by training on in-domain data. However, our goal is to show that our methods produce a more robust model on out-of-domain data, not to maximize performance on this test set. We use the recommended test split of section F, containing 3580 tweets of varying length and capitalization quality.

Since the train and test corpora are from different domains, we evaluate on the level of mention detection, in which all entity types are collapsed into one. The Broad Twitter Corpus has no annotations for MISC types, so before converting to a single generic type, we remove all MISC predictions from our model.

Results are shown in Table 5, and a familiar pattern emerges. Experiment 3 outperforms experiment 1 by 8 points F1, followed by experiment 3.5 and experiment 5, showing that our approach holds when evaluated on a real-world data set.

---

4https://github.com/GateNLP/broad_twitter_corpus
Table 5: Results on NER+ELMo on the Broad Twitter Corpus, set F, measured as mention detection F1.

<table>
<thead>
<tr>
<th>Exp.</th>
<th>Mention Detection F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Cased</td>
<td>58.63</td>
</tr>
<tr>
<td>2. Uncased</td>
<td>53.13</td>
</tr>
<tr>
<td>3. C+U</td>
<td><strong>66.14</strong></td>
</tr>
<tr>
<td>3.5. Half Mixed</td>
<td>64.69</td>
</tr>
<tr>
<td>4. Truecase Test</td>
<td>58.22</td>
</tr>
<tr>
<td>5. Truecase All</td>
<td>62.66</td>
</tr>
</tbody>
</table>

8 Conclusion

We have performed a systematic analysis of the problem of unknown casing in test data for NER and POS models. We show that commonly-held suggestions (namely, lowercase train and test data, or truecase test data) are rarely the best. Rather, the most effective strategy is a concatenation of cased and lowercased training data. We have demonstrated this with experiments in both NER and POS, and have further shown that the results play out in real-world noisy data.

9 Acknowledgments

For their valuable feedback and suggestions, we would like to thank Jordan Kodner, Shyam Upadhyay, and Nitish Gupta.

This work was supported by Contracts HR0011-15-C-0113 and HR0011-18-2-0052 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References


