Abstract

We present a named entity recognition (NER) system for tagging fiction: LitNER. Relative to more traditional approaches, LitNER has two important properties: (1) it makes no use of hand-tagged data or gazetteers, instead it bootstraps a model from term clusters; and (2) it leverages multiple instances of the same name in a text. Our experiments show it to substantially outperform off-the-shelf supervised NER systems.

1 Introduction

Much of the work on applying NLP to the analysis of literature has focused on literary figures/characters in the text, e.g. in the context of social network analysis (Elson et al., 2010; Agarwal et al., 2013; Ardanuy and Sporleder, 2015) or analysis of characterization (Bamman et al., 2014). Named entity recognition (NER) of person names is generally the first step in identifying characters; locations are also a prevalent NE type, and can be useful when tracking different plot threads (Wallace, 2012), or trends in the settings of fiction.

There are not, to our knowledge, any NER systems that are specifically targeted at literature, and most related work has used Stanford CoreNLP as an off-the-shelf solution (Bamman et al., 2014; Vala et al., 2015). In this paper, we show that it is possible to take advantage of the properties of fiction texts, in particular the repetition of names, to build a high-performing 3-class NER system which distinguishes people and locations from other capitalized words and phrases. Notably, we do this without any hand-labelled data whatsoever, bootstrapping a text-level context classifier from a low-dimensional Brown clustering of the Project Gutenberg corpus.

2 Related work

The standard approach to NER is to treat it as a supervised sequential classification problem, typically using conditional random fields or similar models, based on local context features as well as properties of the token itself. Relevant to the present work is the fact that, despite there being some work on enforcing tag consistency across multiple instances of the same token (Finkel et al., 2005) and the use of non-local features (Ratinov and Roth, 2009) to improve supervised sequential models, the consensus seems to be that this non-local information has a relatively modest effect on performance in standard datasets, and as a result off-the-shelf NER systems in practice treat each sentence as a separate document, with multiple instances of the same token in different sentences viewed as entirely independent classification problems. We also note that although supervised NER is the norm, there is a smaller body of work in semi-supervised and unsupervised approaches to NER and semantic lexicon induction, for instance pattern bootstrapping (Nadeau et al., 2006; Telen and Riloff, 2002; McIntosh et al., 2011) and generative approaches (Elsner et al., 2009).

In the context of literature, the mostly closely related task is character identification (Vala et al., 2015), which is itself an intermediate task for character speech identification (He et al., 2013), analysis of characterization (Bamman et al., 2014), and analysis of social networks (Elson et al., 2010; Agarwal et al., 2013; Ardanuy and Sporleder, 2015). In addition to NER, character identifica-
tion also involves clustering multiple aliases of the same character, and discarding person names that don’t correspond to characters. Vala et al. (2015) identify some of the failures of off-the-shelf NER with regards to character identification, and attempt to fix them; their efforts are focused, however, on characters that are referred to by description, which is orthogonal to our proposed approach.

3 Method

3.1 Corpus preparation and segmentation

The corpus we use for building and testing our NER system is the 2010 image of the (US) Project Gutenberg corpus,¹ a reasonably comprehensive collection of out-of-copyright English literary texts, to our knowledge the largest that is publicly available in a machine-readable, full-text format. We access the texts via the GutenTag tool (Brooke et al., 2015), which allows both filtering of texts by genre as well as within-text filtering to remove Project Gutenberg copyright information, front and back matter (e.g. table of contents), and headers. We focus here only on fiction texts (i.e. novels and short stories); other kinds of literature (e.g. plays) are rare in the corpus and have very different properties in terms of the distribution of names. The final corpus size is 10844 texts.

GutenTag also provides an initial segmentation of tokens into potential names, using a simple rule-based system which segments contiguous capitalized words, potentially with common intervening function words like of as well as leading the (e.g. the King of Westeros). It largely (but not entirely) overcomes the problem of sentence-initial capitalization in English by generalizing over an entire text; as long as a capitalized word or phrase appears in a non-sentence initial position at least once in a text, it will be tagged in the sentence-initial position as well. To improve precision, the name tagger in the version of GutenTag used for this paper (0.1.3) has lower bounds on token count (at least 10) and an upper bound on the length of names (no longer than 3 words). For this work, however, we remove those restrictions to maximize recall. Though not our primary concern, we return to evaluate the quality of the initial segmentation in Section 5.

3.2 Brown clustering

The next step is to induce Brown clusters (Brown et al., 1992) over the pre-segmented corpus (including potential names), using the tool of Liang (2005). Briefly, Brown clusters are formed using an agglomerative hierarchical cluster of terms based on their immediate context, placing terms into categories to maximize the probability of consecutive terms over the entire corpus. Note that using information from Brown clusters is a well established technique in NER, but more typically as features within a supervised framework (Miller et al., 2004; Liang, 2005; Ritter et al., 2011); we are unaware of any work using them directly as a source of bootstrapped training examples. We used default settings except for the number of clusters (c): 50. The rationale for such a small cluster size—the default is 1000, and NER systems which use Brown clusters as features do better with even more (Derczynski et al., 2015)—is that we want to have clusters that correspond to major noun categories (e.g. PERSON and LOCATION), which we consider the next most fundamental division beyond part-of-speech; 50 was selected because it is roughly comparable to the size of the Penn Treebank tagset (Marcus et al., 1993). We did not tune this number, except to observe that larger numbers (e.g. 100 or 200) resulted in increasingly fragmented clusters for our entities of interest.

To automatically extract a seed list of people and locations, we ranked the clusters by the total (token) count of names (as identified by GutenTag), and took the first cluster to be PERSON, and the second to be LOCATION; all other clusters are considered OTHER, our third, catch-all category. Alternatively, we could have set c higher and manually grouped the clusters based on the common words in the clusters, adding a thin layer of supervision to the process; with a low c, however, this was unnecessary since the composition and ranking of the clusters conformed exactly to our expectations. The top-5 clusters by token count of names are given in Table 1.²

Note the presence of the multiword name New York in the second cluster, as a result of the segmentation.

The most common words in the first two clusters correspond well with expectations, though there is a bit of noise, e.g. Him included as a place. The other clusters are messier, but still in-

¹http://www.gutenberg.org

²Note that each cluster generally includes large numbers of non-names, which we ignore.
names that appear only in one text; by limiting our context for common words to a single text, we simulate the task for rarer words: Mary is a common name, and may be a major character in one text, but a minor one in another; hence, we build a classifier that deals with both context-rich and context-poor situations. The noisy training set thus constructed has about 1 million examples.

Our feature set consists of filtered word features in a 2-word window \((w_{-2}, w_{-1}, w_0, w_1, w_{+1}, w_{+2})\) around the token occurrences \(w_0\) of a target type in a given text, made up of position-indexed unigrams \((w_{-2}, w_{-1}, w_1, w_{+1})\) and bigrams \((w_{-2}w_{-1}, w_{+1}w_{+2})\) and bigrams \((w_{-2}w_{-1}, w_{+1}w_{+2})\) and bigrams \((w_{-2}w_{-1}, w_{+1}w_{+2})\), excluding unigrams when a subsuming bigram feature matched (e.g. if we match trust in, we do not add trust and in). For this we used the name-segmented corpus, and when one of the words in the context was also a name, we attempted to improve the generalization of the model by taking the category from the Brown clustering as the word (so \(w_0^\text{London}\) for London to \(w_0^\text{New}\) York is LOCATION, not New). Across multiple tokens of the same type, we count the same context only once, creating a binary feature vector which was normalized to the unit hypersphere once all contexts were collected.

To be included as features, the \(n\)-grams had to occur with \(\geq 10\) different \(w_0\) target word types. Note that given our bootstrapping setup, the word type itself cannot be used directly as a feature.

For classification, we use logistic regression from scikit-learn (Pedregosa et al., 2011) trained with SGD using L2 regularization \((C = 1)\).\(^3\) The only non-standard setting that we used was the “balanced” option, which weights classes by the inverse of their count in the training set, countering the preference for the majority class; we did this in part because our bootstrapped distribution is an unreliable reflection of the true distribution, and also because it makes it a fairer comparison to off-the-shelf models with no access to this distribution.

### 3.4 Improved phrase classification

Relative to (true) supervised models, our bootstrapped model suffers from being able to use only context, and not the identity of the name itself. In the case of names which are phrases, this is troubling because there are many generalizations

---

\(^3\)Using cross-validation over the training data, we tested other solvers, L1 regularization, and settings of the \(C\) parameter, but saw no appreciable improvement in performance.
to be made, for instance names ending with City are locations. Our final model addresses this failing somewhat by using more information from our Brown clustering: from each of the initial and final words across all names, we extract a set of words \( W_s \) that appear at least ten times in position \( s \in S, S = \{ \text{initial}, \text{final} \} \) across all phrases. Let \( c(w, t, s) \) be the number of times a word \( w \in W_s \) appears in the corpus at position \( s \) in phrases which were Brown clustered into the entity type \( t \in T \), and \( p(t|r) \) be the original probability of phrase \( r \) being type \( t \) as determined by the logistic regression classifier. For our two homogenous entity types (PERSON and LOCATION), we calculate a new score \( p' \):

\[
p'(t|r) = p(t|r) + \sum_{s \in S} \left( \frac{c(r_s, t, s)}{\sum_{t' \in T} c(r_{t'}, t', s)} - \frac{\sum_{w' \in W_s} c(w_{t'}, t, s)}{|W_s|} \right)
\]

The first term in the outermost summation in Equation 1 is the proportion of occurrences of the given expression in position \( t \) which correspond to type \( t \). To avoid applying too much weight to the homogeneous classes, the second term in the summation subtracts the average number of occurrences in the given position for all words in \( W_s \). As such, the total effect on the score can be negative. Note that if \( p_s \notin W_s \), no modification is made, and for the other type \( p'(t|r) = p(t|r) \).

Once we have calculated \( p'(r, t) \) for each class, we take the \( t \) with the highest \( p'(r, t) \) as the classification.

### 4 Evaluation

Our interest is in a general NER system for literature. Though there are a few novels which have been tagged for characters (Vala et al., 2015), we wanted to test our system relative to a much wider range of fiction. To this end, we randomly sampled texts, sentences, and then names within those sentences from our name-segmented Project Gutenberg corpus to produce a set of 1000 examples. These were tagged by a single annotator, an English native speaker with a PhD in English Literature. The annotator was presented with the sentence and the pre-segmented name of interest, and asked (via written instructions) to categorize the indicated name into PERSON, LOCATION, OTHER, UNCERTAIN due to ambiguity, or segmentation error. We ran a separate two-annotator agreement study over 200 examples which yielded a Cohen’s Kappa of 0.84, suggesting high enough reliability that a single annotator was sufficient. The class distribution for the main annotation was 66.9% PERSON, 10.2% LOCATION, 19.0% OTHER, 2.4% UNCERTAIN, and 1.5% segmentation error. For the main evaluation, we excluded both UNCERTAIN examples and segmentation errors, but had our annotator provide correct segmentation for the 15 segmentation errors and carried out a separate comparison on these.

We compare our system to a selection of publicly available, off-the-shelf NER systems: OpenNLP,\(^4\) LingPipe,\(^5\) and Stanford CoreNLP (Finkel et al., 2005), as well as the initial Brown clustering. OpenNLP allowed us to classify only PERSON and LOCATION, but for Stanford CoreNLP and LingPipe we used the existing 3-entity systems, with the ORGANIZATION tag collapsed into OTHER (as it was in our guidelines; instances of ORGANIZATION are rare in literature). Since the exact segmentation guidelines likely varied across these systems—in particular, we found that Stanford CoreNLP often left off the title in names such as Mr. Smith—and we didn’t want to focus on these issues, we did not require exact matches of our name segmentation: instead, we consider the entire name as PERSON or LOCATION if any one of the tokens were tagged as such (names with both tags or no tags were considered OTHER). For our system (LitNER), we test a version where we use only the immediate sentence context to make the classification (“sentence”), and versions based on text context (“text”) with or without our phrase improvement (“±-phrase”).

We evaluate using two standard metrics: accuracy (“Acc”), and macroaveraged F-score (“\( F_M \)”).

### 5 Results

The results in Table 2 show that our system easily bests the off-the-shelf systems when it is given the contextual information from the entire text; the difference is more stark for accuracy (+0.085 absolute), though consistent for \( F_M \) (+0.041 absolute). Stanford CoreNLP is the only competitive off-the-shelf system—the other two are far too conservative when encountering names they

\(^4\)http://opennlp.apache.org/
\(^5\)http://alias-i.com/lingpipe
Table 2: Performance of NER systems

<table>
<thead>
<tr>
<th>System</th>
<th>Acc</th>
<th>$\mathcal{F}_M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>All PERSON baseline</td>
<td>.696</td>
<td>—</td>
</tr>
<tr>
<td>OpenNLP</td>
<td>.435</td>
<td>.572</td>
</tr>
<tr>
<td>LingPipe</td>
<td>.528</td>
<td>.536</td>
</tr>
<tr>
<td>Stanford CoreNLP</td>
<td>.786</td>
<td>.751</td>
</tr>
<tr>
<td>Brown clusters</td>
<td>.803</td>
<td>.672</td>
</tr>
<tr>
<td>LitNER sentence + phrase</td>
<td>.757</td>
<td>.671</td>
</tr>
<tr>
<td>LitNER text – phrase</td>
<td>.855</td>
<td>.771</td>
</tr>
<tr>
<td>LitNER text + phrase</td>
<td>.871</td>
<td>.792</td>
</tr>
</tbody>
</table>

For the segmentation errors, we compared our corrected segmentations with the segmentation provided by the CRF-based Stanford CoreNLP system, our best competitor. Only 2 of the 15 were segmented correctly by Stanford CoreNLP. This potential 0.002 improvement is tiny compared to the 0.085 difference in accuracy between the two systems.

6 Discussion

Aspects of the method presented here could theoretically be applied to NER in other genres and other languages, but one important point we wish to make is that our approach is clearly taking advantage of specific properties of (English) literature. The initial rule-based segmentation, for instance, depends on reliable capitalization of names, which is often not present in social media, or in most non-European languages. We have found more subtle genre effects as well: for comparison, we applied the preliminary steps of our approach to another corpus of published texts which is of comparable (token) size to the Project Gutenberg corpus, namely the Gigaword newswire corpus (Graff and Cieri, 2003), and noted degraded performance for both segmentation and Brown clustering. With respect to the former, the obvious issue is considerably more complex proper nouns phrases such as governmental organizations and related titles; for the latter, there were several clusters in the top 10 (including the first one) which corresponded to LOCATION, while the first (fairly) clean PERSON cluster was the 15th largest; in general, individual people, organizations, and other groupings of people (e.g. by country of origin) were not well distinguished by Brown clustering in the Gigaword corpus, at least not with the same low number of clusters that worked well in the Project Gutenberg corpus.

Also less than promising is the potential for using text-level classification in other genres: whereas the average number of token occurrences of distinct name types within a single text in the Project Gutenberg corpus is 5.9, this number is just 1.6 for the much-shorter texts of the Gigaword corpus. Except in cases where it is possible to collapse texts into appropriately-sized groups where the use of a particular name is likely to be both common and consistent—an example might be a collection of texts written by a single author, which in social media such as Twitter seems to obey the classic one-sense-per-discourse rule (Gella et al., 2014)—it’s not clear that this approach can be applied successfully in cases where texts are relatively short, which is a far more common situation. We also note that relying primarily on contextual classification while eschewing resources such as gazetteers makes much less sense outside the context of fiction; we would expect relatively few fictitious entities in most genres.

LitNER tags names into only two main classes, PERSON and LOCATION, plus a catch-all OTHER. This coarse-grained tag set reflects not only the practical limitations of the method, but also where we think automatic methods have potential to provide useful information for literary analysis. The other clusters in Table 1 reflect word categories which are relatively closed-class and much less central to the fictional narratives as character and setting; though the fact they appear as clusters means we should be able to identify them automatically, we actually don’t see a compelling case for doing so. When these as well as non-entities are excluded from OTHER, what remains is eclectic, including names referring to small groups of people (e.g. families), animals, gods, ships, and titles of other works of literature. If we were to target any of these kinds of names, it would be primarily to prevent them from being included as PERSON or LOCATION. Looking at errors made by the system, we note that any named entity with some kind of agency is likely to be mistaken for...
PERSON, whereas passive containers of all kinds tend to be classified as a LOCATION.

7 Conclusion

In this paper, we have presented LitNER, an NER system targeted specifically at fiction. Our results show that a simple classifier, trained only with noisy examples derived in an unsupervised fashion, can easily beat a general-purpose supervised system, provided it has access to the full context of the text. Finally, we note that the NER tagging provided by LitNER has been integrated into the latest version of the GutenTag tool (0.1.4).6

References


6See http://www.projectgutentag.org


