Exploiting Unannotated Corpora for Tagging and Chunking

Rie Kubota Ando
IBM T.J. Watson Research Center
19 Skyline Dr., Hawthorne, NY 10532
rie1@us.ibm.com

Abstract

We present a method that exploits unannotated corpora for compensating the paucity of annotated training data on the chunking and tagging tasks. It collects and compresses feature frequencies from a large unannotated corpus for use by linear classifiers. Experiments on two tasks show that it consistently produces significant performance improvements.

1 Introduction

This paper presents a method for exploiting large unannotated corpora for the tagging and chunking tasks. We report experiments on *entity mention detection*[1] and *part-of-speech (POS) tagging*. To apply classification techniques to chunking tasks, a common approach is to cast the task to that of token tagging, where token tags encode chunk information, e.g., 'B-PERSON' (beginning of person chunk), 'I-PERSON' (inside of person chunk), and 'O' (outside of any entity chunk). The challenge for a classifier is to learn unknown relationships between token tags and *features* (such as token strings and context information) from tagged examples. To achieve reasonable performance, a *sufficiently large* number of *representative* examples are required. Our goal is to compensate for the paucity of tagged examples or their differences from test data, by using untagged examples.

One type of approaches to this problem involves iterative and automatic tagging of the untagged data such as *bootstrapping* or *co-training*. *Expectation Maximization (EM)* also uses untagged data for iteratively improving model parameters. Another type uses untagged corpora for improving *feature representation*, e.g. (Schüetze, 1992). We take the latter approach.

To see how unannotated corpora may help tagging, consider the following examples:

$\cdots$ the president/B-PERSON and $\cdots$
$\cdots$ our chairman/B-PERSON is $\cdots$

Suppose that "president" appeared in the training data, but "chairman" didn't, and that in a large corpus, both words ("chairman" and "president") often appear as the subject of "said", "visited", etc., and that both are often modified by "vice", "powerful", etc. It is intuitive that such corpus statistics would help a classifier to tag "chairman" correctly even if "chairman" did not appear in the training data.

Given some set of features designed for the task (see Figure 1 for example), we count feature occurrences in all the word instances in the unannotated corpus to generate feature-by-word co-occurrence frequency matrices. When we encounter a training or test instance of word w, we generate two kinds of features. One is the features observed in that instance (as usual). The other is the features derived from the columns (corresponding to w) of the feature-by-word co-occurrence matrices – collections of w's context in the untagged corpus – which we call *corpus-context features*.

Our experiments show that the corpus-context features consistently improve performance on the two tasks. There are two important elements for achieving such effectiveness in this simple framework. One is a high-performance linear classifier, *Robust Risk Minimization (RRM)* (Zhang et al., 2002), which has an ability to ignore irrelevant features while coping with mutually-dependent features. (RRM learns feature weights by minimizing classification errors with regularization on the tagged training data.) Therefore, we take a 'feature-rich' strategy to use a variety of types of cor-

[1] The task objective of entity mention detection is to detect and classify text spans that *mention* (or refer to) certain types of entities in the real world such as persons and organizations. We experiment with the data from the ACE (Automatic Content Extraction) program (http://www.nist.gov/speech/index.htm).

pus context information. To enable classifier training with many types of corpus statistics, such vast amounts of information from a large corpus must be compressed. Hence, the second key element is a dimension reduction technique. We adapt a variation of LSI, specifically designed for feature occurrence frequencies (Ando, 2004). As such, the objective of this paper is to show that a right combination of techniques produces a useful tool for coping with the paucity of tagged training data.

2 Method

2.1 Collecting corpus statistics

From a given set of features designed for the task (see Figure 1 and Figure 6 for example), we use context features only (i.e., excluding features that strongly depend on words[2]) to generate feature-by-word co-occurrence matrices. We generate one matrix for each type, e.g., a 'left adjacent word'-by-word matrix, a 'right adjacent word'-by-word matrix, and so forth.

2.2 Vector compression

To compress feature-by-word matrices, we adapt a procedure proposed for semantic lexicon construction (Ando, 2004). That is to apply *singular value decomposition (SVD) only* to a smaller matrix consisting of *several selected* columns of the co-occurrence matrix and to 'fold in' the rest of the columns to the reduced dimensions. The choice of columns is important. The columns corresponding to the *most frequent words* should be selected. The intuition behind its theoretical justification (Ando, 2004) is that more reliable statistics from high-frequency words should produce a better representation space, which should result in improving statistically 'poor' vectors for low-frequency words. Thus, we choose k most frequent words and reduce the dimensions to h. The dimensionality h should be no smaller than the number of target classes[3].

We compress each of feature-by-word co-occurrence matrix *independently* of one another. This is important, as it gives more freedom to sophisticated classifiers to weight relevant types of features more heavily than irrelevant ones. If all are compressed together, the classifiers can not tear them apart. For efficient training, though optionally, we further reduce non-zero entries by zeroing out all but n entries that have the largest absolute values in each compressed vector. We call the entries of the resultant vectors *corpus-context features*. For a training or test instance of word w, we have two kinds of features: features derived from the instance (as usual), and the corpus-context features generated from w's context in the corpus.

For our experiments, we set $(k, h, n) = (1000, 50, 6)$ using held-out data (the development set described below). Performance is relatively insensitive to the changes of these parameters[4]. We use the same parameter setting for both entity mention detection and part-of-speech tagging experiments.

- token, capitalization, POS in 3-token window
- bi-grams of adjacent words in 5-token window
- words in the same syntactic chunk.
- head words in 3-chunk window
- word uni- and bi-grams based on subject-verb-object and preposition-noun constructions.
- syntactic chunk types
- tags in 2-token window to the left
- tri-grams of POS, capitalization, and word ending
- tri-grams of POS, capitalization, and left tag

Figure 1: Features for entity detection

3 Entity mention detection experiments

3.1 Experimental framework

Entity classes and evaluation metric We experiment with 10 classes from the ACE entity classes – obtained by combining five entity types (Person, Organization, Facility, GPE, Location) and two mention types (Name, Nominal), which make 21-way classification when chunk boundary information is encoded into token tags. Proposed mention chunks are counted as correct *only if* both mention boundaries and classes are correct. We combine precision and recall into F-measure with equal weight.

Features Figure 1 describes features used for entity mention detection experiments. We generate corpus-context features from the features

[2]For instance, it is useless to count 'co-occurrences' of words and their endings. Moreover, features that are nearly conditionally independent of words given classes are more useful for the purpose, since ultimately we want to capture correlations of words to classes (through their co-occurrences with features) rather than their correlations to specific features.

[3]Intuitively, there need at least h dimensions to express correlations to h classes.

[4]On the held-out data, $k \in [1000,5000]$ produced essentially similar performance, and so did $h \in [30,60]$ and $n \in [6,10]$.

	annotated			unannotated
type	training	dev	test	–
ACE	185K (21K)	21K (2.4K)	52K (5.5K)	WSJ: 40M
CNS	–	–	32K (4.5K)	CNS: 3M

Figure 2: Data statistics. # of tokens and # of mentions in parentheses.

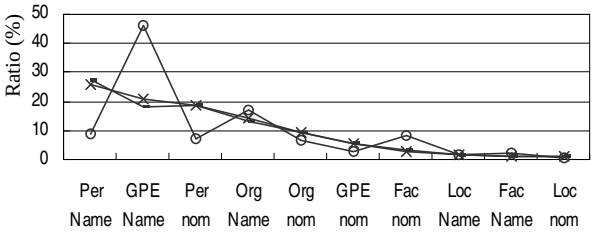

Figure 3: Target class proportions. The line with circles is on the CNS corpus. The other two lines (almost overlapping) are on the ACE training and test sets.

in the first five lines of the figure *excluding* those overlapping with the current word. The rest are not suitable either because they have strong dependency on the word, or because they require token tags, which are not available in the untagged corpus.

Data We use two annotated corpora. Data statistics is summarized in Figure 2. The first annotated corpus is the ACE phase-2 data available from Linguistic Data Consortium. The sources of the documents are newswire, newspapers and broadcasts. The data consists of the training set (January–June 1998), and the test set (52K words; October–December 1998). We held out one tenth of the training set for the development purpose, and used the remaining nine tenth as tagged training data. The second annotated corpus is part of the *CNS corpus*, a collection of reports on the development of nuclear technologies from Center for Nonproliferation Studies. We manually annotated 32K words of this corpus for generating test data, and use the rest (3 million words) as an unannotated corpus. Figure 3 plots the proportions of target entity classes annotated in the ACE corpus and in the above 32K words of the CNS corpus. The ACE training and test sets have almost identical distributions while the CNS test set exhibits clearly different class distributions. Our interest is in whether use of the unannotated portion of the CNS corpus can compensate for such differences between the ACE training set and the CNS test set. As for an ACE-type unannotated corpus, we use Wall Street Journal (WSJ) articles (January 1991 through March 1992; 40 million words).

	RRM+corpus-ctx		RRM	Baseline
	WSJ	CNS	–	–
7K	**63.8** (+15.4)	59.6	48.4	31.9
11K	**69.1** (+14.2)	62.5	54.9	30.0
22K	**71.3** (+10.4)	67.2	60.9	46.4
40K	**73.1** (+7.9)	69.7	65.2	50.4
76K	**74.3** (+6.0)	72.6	68.3	54.4
185K	**77.0** (+2.6)	75.5	74.4	60.5

Figure 4: Entity detection F-measure results on the *ACE test set*. Numbers in parentheses are differences from RRM without corpus-context features.

	RRM+corpus-ctx		RRM	Baseline
	WSJ	CNS	–	–
7K	52.5	**58.6** (+19.2)	39.4	27.9
11K	60.9	**65.0** (+19.6)	45.4	34.5
22K	63.9	**69.4** (+19.3)	50.1	39.6
40K	66.7	**71.2** (+15.4)	55.8	48.4
76K	69.2	**74.2** (+11.2)	63.0	56.0
185K	70.2	**76.2** (+10.9)	65.3	60.8

Figure 5: Entity detection F-measure results on the *CNS test set*. Numbers in parentheses are differences from RRM without corpus-context features.

3.2 Results

In all the experiments in this section, annotated training examples are randomly drawn from the ACE training set.

Results on the similar test data Figure 4 shows F-measure results on the ACE test set in relation to the number of tagged training examples (the left-most column). The right-most column shows the baseline performance, obtained by choosing the tag most frequently assigned to the word if that word appears in the training data, and chooses 'O' (outside of entity chunks) otherwise. The best performance is achieved when corpus-context features are generated from the untagged corpus similar to test data (WSJ). It outperforms the RRM classifier that does not use corpus-context features by 2.6% to 15.4%.

Results on the dissimilar test data The results on the CNS test set (Figure 5) indicate that an untagged domain corpus (the CNS cor-

· token, capitalization in 5-token windows
· ending (length 1 to 4)
· uni- and bi-grams of tags at the left
· tag-word bi-grams in 3-token windows
· bi-grams of adjacent words in 5-token windows

Figure 6: Features for POS tagging

	RRM		HMM	
	Corpus-ctx	–	with BW	w/o
5K	**90.2** (+7.4)	82.8	82.1 (+5.0)	77.1
9K	**92.7** (+5.0)	87.7	84.9 (+2.7)	82.2
19K	**93.7** (+2.8)	90.9	87.1 (+0.3)	86.8
38K	**94.7** (+1.8)	92.9	89.8 (-0.2)	89.6
75K	**95.2** (+1.6)	93.6	91.2 (-0.6)	91.8
149K	**95.6** (+0.9)	94.7	92.3 (-1.0)	93.3

Figure 7: POS tagging accuracy results. Numbers in parentheses are differences from their counterparts that do not use the untagged corpus.

pus), indeed, compensates for the differences between tagged training data (ACE) and test data (CNS). The other classifiers are apparently suffering from the dissimilarity.

4 POS Tagging Experiments

Features Figure 6 shows the features we use for POS tagging. Among them, we use word uni- and bi-grams that do not overlap with the current word, to generate corpus-context features.

Baseline As our baseline, we implement an HMM tagger with and without *Baum-Welch reestimation* (EM for HMM). We smooth transition probabilities by deleted interpolation. For unseen and low-frequency words, word emission probabilities are estimated as Weischedel et al. (1993) do while interpolating emission probabilities of words and endings (length 1 to 4). We estimate these probabilities by relative frequencies in tagged training corpora, and perform 10 EM iterations using unannotated data. To avoid underestimating the baseline, we report its best performance among the iterations.

POS tagging results We report results on the standard Brown corpus. The test data was fixed to arbitrarily-drawn one fifth of the corpus (230K words). We use the rest (930K words) as tagged and untagged training data: all 930K words as untagged data for collecting corpus context and for the BW reestimation; and arbitrarily-drawn various portions as tagged training data. Figure 7 shows accuracy (# of correctly tagged words divided by # of words) in relation to the number of tagged training examples. The performance differences between HMM and RRM mainly derive from the differences in the 'richness' of information they make use of. The additional features[5] used by RRM are apparently effective for compensating for the paucity of the tagged data. Corpus-context features further improve the performance up to 7.4%. This is in contrast to the Baum-Welch reestimation, which sometimes rather degrades performance.

5 Conclusion

The method we present is intended for the chunking/tagging tasks in which words serve as strongly effective features. Performance improvements obtained by corpus-context features are especially large when tagged training is small or different from test data, which is useful for expediting the adaptation of the system to new domains.

Acknowledgements

This work was supported by the Advanced Research and Development Activity under the Novel Intelligence and Massive Data (NIMD) program PNWD-SW-6059.

References

Rie Kubota Ando. 2004. Semantic lexicon construction: Learning from unlabeled data via spectral analysis. In *Proceedings of CoNLL-2004*.

Hinrich Schüetze. 1992. Dimensions of meaning. In *Proceedings of Supercomputing'92*, pages 787–796.

Ralph Weischedel, Marie Meteer, Richard Schwartz, Lance Ramshaw, and Jeff Palmucci. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19(2):359–382.

Tong Zhang, Fred Damerau, and David Johnson. 2002. Text chunking based on a generalization of Winnow. *Journal of Machine Learning Research*, 2:615–637.

[5] As many of the features used with RRM are mutually dependent, there is no easy way to exploit them with HMM. However, we note that when trained with over one million tagged examples, RRM (with and without corpus context) and HMM taggers produce essentially similar high accuracy. That is, the mutually-dependent features become redundant once sufficiently large tagged data becomes available.