SEMISUPERVISED NAMED ENTITY RECOGNITION

TAYLOR P. TURPEN* AND DANIEL M. DUNLAVY†

1. Introduction. With the ever increasing number of documents readily available to the casual reader, careful observer or scientific analyst it is becoming more important to quickly recognize names of people places, locations, etc., within those documents. Although it may be a relatively easy task to extract *named entities* through manual inspection, parsing the million of blogs created daily[1] is untractable with such an approach.

In this paper we focus on statistical methods for solving the Named Entity Recognition (NER) problem [4], i.e., the identification of words and phrases that are associated with a given set of entity categories. Specifically, we will focus on sequence-based machine learning methods, with an emphasis on semisupervised learning [1, 8], an iterative approach to predictive modeling that uses both labeled and unlabeled data in generating models.

The definition of the NER problem used in the paper is as follows. Given a sequence of words, $\tilde{\mathbf{x}} = \{\tilde{x}_1, \ldots, \tilde{x}_n\}$, and associated named entity labels $\tilde{\mathbf{y}} = \{\tilde{y}_1, \ldots, \tilde{y}_n\}$, the goal is to generate a model for predicting the labels of new word sequences. Spefically, we seek to find a model, f, such that

$$\hat{y} = f(\tilde{x}) \,, \tag{1.1}$$

"best agrees" with $\tilde{\mathbf{y}}$. Throughout the remainder of this paper, we will refer to $\tilde{\mathbf{x}}$ as the *labeled data*, $\tilde{\mathbf{y}}$ as the *true labels*, and $\hat{\mathbf{y}}$ as the *predicted labels*.

A set of features associated with a word is used in determining the most probable label for that word. Examples of features explored in this paper include the position of the sequence (e.g., sentence beginning, end, or other) and capitalization; the statistics on the use of such features as they relate to the labeled data and true labels are used to set parameters in the NER model, f.

Within the class of sequence-based NER are also rule-based methods, i.e., sets of prioritized logical conjunctions that are tested for each sequence being labeled. One benefit of statistical methods, over rule-based approaches, is that all that is required is a sequence of labeled data and the associated true labels; there is no need to know grammatical constructs specific to the particular subject domain or genre in which the text is written in order to generate models. However, there is evidence that incorporation of such information into statistical models may be useful and we plan to explore which method may work best for a particular data set or type of data in future work.

A major challenge in solving the NER problem is handling word sense ambiguity, the fact that words often have more than one meaning and thus entity type. For example, the word "Washington" may be of type *person* (Geroge Washington), *place* (Washington, D.C. or the state of Washington), or *organization* (Washington University). One benefit in using statistical-based methods is that most models compute likelihoods or probabilities of all entity types specified for each word. For example, consider the following sentence.

`John attends George Washington University with George Washington.`

Typical computation performed by a statistical model may be as follows for this example (assuming only four possible entity types).

*San Diego University, tturpen-10@sandiego.edu
†Sandia National Laboratories, dmdunla@sandia.gov
[1] http://technorati.com/blogging/state-of-the-blogosphere/

TABLE 1.1
Different ways of combining these probabilities lead to different algorithms.

	Probability			
Word	*Person*	*Place*	*Organization*	*Other*
John	0.95	0.01	0.01	0.03
attend	0.01	0.01	0.01	0.97
George	0.07	0.04	0.85	0.04
Washington	0.07	0.04	0.85	0.04
University	0.01	0.10	0.75	0.05
with	0.01	0.01	0.01	0.97
George	0.75	0.21	0.01	0.03
Washington	0.79	0.19	0.01	0.01

Machine learning methods build predictive models using statistical inferences derived from training data (e.g., the labeled data described above). There are three main machine learning approaches: unsupervised, supervised and semisupervised. Unsupervised techniques are given no structural or implicit information about the data they are processing. For the NER problem, unsupervised methods (see, e.g., [3]) do not use any information about the entity labels associated with the training data. Supervised learning uses both the labeled data and the true labels to build predictive models.

A sequence-based semisupervised NER approach trains a model on an initial set of labeled data and true labels, makes predictions on a separate set of unlabeled data, and then iteratively attempts to create improved models using predictions of previously generated models (plus the original labeled data and true labels). The main advantage of semisupervised approaches is that it is possible to create more accurate NER models from less training data. Since annotating training data is a task which requires time and money resources, and the data may be related to a niche technical area, proprietary, and/or classified, requiring subject matter experts or limited personnel, this benefit of semisupervised learning can become crucial.

The Stanford Natural Language Processing Group has created software for solving the NER problem [2], known as the Stanford Named Entity Recognizer (SNER). SNER uses a supervised learning approach called conditional random fields [3, 6], where a model maximizing the conditional probabilities of the sequence of true labels, $\tilde{\mathbf{y}}$, given the labeled data, $\tilde{\mathbf{x}}$, is generated. Our work has focused on creating a semisupervised approach by wrapping an outer loop of iteration around SNER, where at each iteration the training data used by SNER changes. Our wrapper implementation is called the Semisupervised Utility for Named Entity Recognition (SUNER) and is discussed in more detail in Section 3. SUNER accomplishes two tasks: 1) performance of NER modeling for different types highly depends on the various features used in creating the NER models and 2) performance using semisupervised learning can be comparable to that of supervised learning using only a fraction of the size of training data used by supervised learning.

2. Named Entity Recognition. Named Entity Recognition is a difficult task. Polysemy and synonomy don't make it any easier. Polysemy is defined as "the coexistence of many possible meanings for a word or phrase."[5]. Whereas, synonomy is "the state of being synonomous. . . having the character of a synonym. . . one of two or more words or expressions of the same language that have the same or nearly the same meaning in some or all senses."[7] And that's just the beginning. How should NER deal with slang terms? Do words used in professional applications have the same meaning as those used in casual emails or blogs?

NER can also be used to identify proteins, acids, gene sequences etc.

2.1. Statistical Inference. As we discussed earlier, there are two main approaches to sequence based methods: Rules and Statistics. We also mentioned something known as a machine learning feature set, which describes words in numerical or boolean values. A NER semisupervised algorithm trains itself by building a statistical model based on what it knows to be significant. This is broken into two stages: initial training and iterative training. Initial and iterative training constitue the entirety of semisupervised learning to be discussed in section 3. Initial training, which we will touch on briefly here is how the program remembers which words correspond to which names by recording which features were true for those words. In essence it builds a percentage confidence associated with different features being true. For example the sentence:

```
I saw Washington<person> walking down Central<street>
with Boogie<monster>.
```

Evaluate the word "Washington". After reading the sentence, it can be assumed that a word that does not begin the sentence nor end the sentence and has a capital letter is a person. The same goes for street. However, as far as statistical modeling is concerned, any word that has a capital letter and ends the sentence is a monster. When this concept is extrapolated to thousands of documents it becomes more easy to generalize each feature set and the values associated with each entity label.

2.2. Applications. While we have tested SUNER on the University of Pennsylvania Biomedical Database and Newswire Documents there are many other applications and ways to think about named entities such as:

- Variable cross-language labels can be useful because their evaluation is consistant regardless of data subject matter.
  - PERS, ORG, LOC, etc.
- Static part of speech tagging is useful for many translation applications as well as being used in conjunction with the general cross-language labels.
  - Noun, Verb, Predicate, Adjective, Adverb etc.
- Variable improvisational tags are flexible and imaginative where their use comes from the importance of their application.
  - Terrorist, first baseman, gene-protein, car salesman, radioactive isotope etc.

3. Semisupervised Learning. Sequence based semisupervised learning is a machine learning technique that reads through an already annotated training document word by word. It builds the statistical inferences from that training data, initial training, and then uses them to make predictions on an unannotated document. The highest of those predictions are then assumed to be part of the annotated corpus and are used to train a new statistical model, also known as iterative training. Usually semisupervised learning will continue to iteratively train statistical models until some stopping criteria is met. This is known as self-training because in a way the model is training itself without absolute certainty as to which labels are true, only an educated guess as to which labels are statistically likely.

3.1. Initial training. Initial training can be viewed as describe in equation 1.1. Where $\hat{y}$ is the sequence of predicted labels created after the training process f on the training data $\tilde{x}$. Take this one step further and include what is known as a weighting scheme. Because it may be more important that a word includes a capital letter than it is that the word is at the beginning of the sentence, the function becomes:

$$\hat{y} = f(\tilde{x}, w_d) \,, \tag{3.1}$$

Where w_d is the default weighting scheme. SNER uses a Quasi-Newtonian Minimizing scheme in order to achieve the most ideal weight for each feature depending on $\tilde{x}$.

3.2. Iterative Training. Iterative training takes place on unannotated data. Basically, semisupervision takes the $\hat{y}$ achieved after initial training. If the prediction values, or percentage confidences of $\hat{y}$, are above a certain threshold, say 95%, then the program will train a new model considering those names to be true names. That gives us:

$$\hat{y}_0 = f(\tilde{x}, w_0, \hat{x}, t_d) \,, \tag{3.2}$$

Where $\hat{y}_0$ is the sequence of new predictions from $\tilde{x}$ original training data, w_0 weighting scheme as minimized for that data, and $\hat{x}$ the set of words and names from $\hat{y}_0$ with predictions higher than that of, t_d the prediction threshold. Note that once $\hat{x}$ is constructed via statistical predictions it does not become part of $\tilde{x}$. A prediction that falls above the acceptable threshold is only considered true for that iteration of iterative training. Predictions that persist through training iterations are classified under indelibility and will not be discussed in this paper.

3.3. Parameters. In order to control the iterative training portion of semisupervised learning there are several parameters that need to be evaluated.

3.3.1. Stopping Criteria.

- Performance threshold: Once the statistical model begins naming words with names that are incorrect, the model will stop iterative training.
- Numeric threshold: The model will stop iterative training once it has repeated a certain number of times. The performance metrics created as a result of numeric threshold implementation can then be used to maximize or minimize the number of iterations.
- Improvement threshold: Once the predicted names of words are the same predictions as the last iteration, because of a lack of idelibility, and no new predictions are being made, stop iterative training. This highlites a model's inability to improve its predictive power despite the lowering of t_d across iterations.
- Flexible Improvement threshold: This threshold is the same as the Improvement Threshold except that no new predictions are being made within an acceptable margin
- Floor threshold: Once the t_d drops below a certain percentage when using stepping(sec 3.3.2), the model will stop iterative training.
- t_d: This is not actually a stopping criteria but instead the threshold below which no predictions are accepted for potential word names

Often several of these stopping criteria will be used in conjunction with one another. When one of the parameters is exceeded the iterative training will stop regardless of the status of the others. Of course, some sort of marginal slush-factor can be used if the other parameters are nowhere close to being exceeded.

3.3.2. Stepping. Often to improve the quantity of accepted predictions, with little sacrifice to their quality, a stepping threshold is used. It is structured so that each iteration of iterative training results in a lower t_d. For example if the step size is 1% the first t_d may be 99%, then after the first iteration it will be 98% then 97% and so on.

Depending on the strength of the model being used the step size should be adjusted in order to reflect the model's ability to improve predictions. i.e. A very "strong" model would reflect poorly upon a large stepping size (greater than 1%). This is because a strong model will have little statistical variance (change in prediction values across $\hat{x}$) in comparable predictions. However, a weak model might perform well with one because the statistical

variance in predictions is usually larger with weaker models. The variability itself is part of the nature of f, and the size of $\tilde{x}$ when compared to $\hat{x}$. Stepping is also best used in conjunction with the improvement threshold. In that way, t_d will not be able to drop below acceptable levels if no literal predictive improvement is being made.

3.3.3. Named Entity Balancing. The goal of balancing is to moderate the preference of the statistical model. If the training data contains an unequal distribution of named entities (annotated words) then balancing is any sort of numeric "leash" put on the maximum number of accepted predicted named entities for any one set of names.

This helps to avoid what is known as "oversampling" or "undersampling" and prevents model preference toward a certain name that may not be a true representation of the global type of data being processed. Imagine wanting to create a model for scientific documents. If the model was trained on 500 newswire documents in the field of bioengineering, that model would more easily name words like "acid" or "rna" than it would "torque" or "pounds", even though the latter might be mentioned in the training set, they are not prevalent enough to make a statistical impression.

In SUNER we implement a simple linear balancing algorithm. The results for which we are still working on and will be added shortly.

4. Numerical Experiments. First, it will be shown that with all else held constant, semisupervised named entity recognition depends largely upon which features are being used. By providing evidence for the previous point it will also be shown that some features result in higher performance results than others. In order to demonstrate this a simple SUNER statistical model

using no balancing was implemented using k-fold cross validation. This data analysis technique involves dividing up a data set into k sections. Each section will be swapped out between being considered annotated(part of $\tilde{x}$), unannotated or "left-out" by the semisupervised program. When a document partition is considered unannotated, the program trains itself on the $k-2$ remaining documents considered to be annotated. It is important to train f on different annotated data partitions and then iteratively train it on other data partitions in order to avoid any sort of bias present within the data itself. Each partition, in turn, will also be left out of both training and iterative training in order to remove its bias on the model entirely.

The self-training also includes a number of "key features" which alternate on and off while all other features stay on. When a feature is "on" it will be read as true by the program if it applies to the word in question. If the performance evaluation metrics vary when the features are alternated, then the assumption that features affect the performance of semisupervised named entity recognition is true.

Second, it will be shown that with all else held constant, semisupervised named entity recognition can produce the same or better results than supervised named entity recognition with only a fraction of the size of training data. This will be done using leave-one-out k-fold cross validation and a balancing implemented self-training algorithm.

In order to minimize under or over sampling, a somewhat normal but randomized distribution algorithm was used to parse the data sets. Each trial includes the corresponding name/fold distribution table.

4.1. Results From 10-fold Cross Validation Self-Training With Key Features: 0,5,9. The first test consisted of a ten-fold cross validation on 261 Medline documents. When parsed into the Stanford Named Entity Recognition format, the document folds had the name distribution indicated by 4.1. The names, indixed by $L_0 ... L_5$ are as follows: Null-label(no name, or "other"), gene-protein, malignancy-histology, gene-rna, malignancy-sites,

malignancy-clinical-stages.

TABLE 4.1
10-fold Cross Validation Name Distribution

Fold	L_0	L_1	L_2	L_3	L_4	L_5
0	566	147	128	101	38	16
1	140	51	52	41	17	7
2	181	68	65	43	21	14
3	238	89	95	63	26	14
4	296	112	111	79	38	18
5	341	133	114	84	42	23
6	382	162	137	105	42	19
7	437	183	153	117	62	23
8	472	206	183	130	68	27
9	516	235	195	151	71	37

TABLE 4.2
10-fold Cross Validation, Fold 8, With Key Features: 0,5,9. Testing(and iterative training) File Length(lines): 4584 Training File Length:27794(lines), Number of Labels(6)

Fold	F-Measure	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
8	0.60545	0	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1
8	0.60401	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1
8	0.59907	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
8	0.59981	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
8	0.60545	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
8	0.60401	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
8	0.59907	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	0.59981	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The F-Measures for one of the folds is in table (4.1,4.2). A 0 indicates that that particular feature was not used for that iteration. A 1 indicates the opposite. The F-measure is the weighted harmonic mean of both precision and recall values. Where TP(true positives) the presence of a named entity was correctly predicted, TN(true negatives) the absence of a named entity was correctly predictued, FP(false positives) the presence of a named entity was incorrectly predicted and FN(false negatives) the presence of a named entity was not predicted when it should have been, are elements of performance evaluation in the following equations:

$$\text{precision} = \frac{TP}{TP + FN}\,, \tag{4.1}$$

$$\text{recall} = \frac{TP}{TP + FP}\,, \tag{4.2}$$

$$\text{F-Measure} = \frac{\text{Precision} \cdot \text{Recall}}{(1-\alpha)P + \alpha\text{Recall}}\,, \tag{4.3}$$

Where α is the weighting measure which usually equals 0.5 weighting *recall* twice as much as *precision*.

REFERENCES

[1] O. Chapelle, B. Sch"olkopf, and A. Zien, eds., *Semi-Supervised Learning*, MIT Press, Cambridge, MA, 2006.

[2] T. G. Jenny Rose Finkel and C. Manning, *Incorporating non-local information into information extraction systems by gibbs sampling*, in Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics, 2005, pp. 363–370.

[3] J. Lafferty, A. McCallum, and F. Pereira, *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*, in MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-, 2001, pp. 282–289.

[4] D. Nadeau and S. Sekine, *A survey of named entity recognition and classification*. 2006.

[5] *Oxford English Dictionary*, Oxford University Press, 2009.

[6] H. Wallach, *Conditional random fields: An introduction*, Rapport technique MS-CIS-04-21, Department of Computer and Information Science, University of Pennsylvania, 50 (2004).

[7] *Webster's Ninth New Collegiate Dictionary*, Merriam-Webster, Spingfield, MA 1984.

[8] X. Zhu, *Semi-supervised learning literature survey*, Tech. Rep. 1530, Computer Sciences, University of Wisconsin-Madison, 2005.