# Automatic Labeling for Entity Extraction in Cyber Security

Robert A. Bridges*, Corinne L. Jones§, Michael D. Iannacone*, and John R. Goodall*

*Computational Sciences and Engineering Division
Oak Ridge National Laboratory, Oak Ridge, TN 37830
{bridgesra, iannaconemd, jgoodall}@ornl.gov

§Department of Mathematics
The Pennsylvania State University, University Park, PA 16802
clj180@psu.edu

***Abstract*—Timely analysis of information in cyber-security necessitates automated information extraction from unstructured text. Unfortunately, state-of-the-art extraction methods require training data, which is unavailable in the cyber-security domain. To avoid the arduous task of hand-labeling data, we develop a very precise method to automatically label text from several data sources by leveraging article-specific structured data and provide public access to corpus annotated with cyber-security entities. We then prototype a maximum entropy model that processes this corpus of auto-labeled text to label new sentences and present results showing the Collins Perceptron outperforms the MLE with L-BFGS and OWL-QN optimization for parameter fitting. The main contribution of this paper is an automated technique for creating a training corpus from text related to a database. As a multitude of domains can benefit from automated extraction of domain-specific concepts for which no labeled data is available, we hope our solution is widely applicable.**

## I. Introduction

Online security databases, such as the National Vulnerability Database (NVD), the Open Source Vulnerability Database (OSVBD), and Exploit DB, are important sources of security information, in large part because their well defined structure facilitates quick acquisition of information and allows integration with various automated systems.[1] On the other hand, newly discovered information often appears first in unstructured text sources such as blogs, mailing lists, and news sites. Hence, in many cases there is a time delay, sometimes months, between public disclosure of information and appropriate classification into structured sources (as noted in [18]). Additionally, many of the structured sources include a text description that provides important details (e.g., Exploit DB). Timely use of this information, both by security tools and by the analysts themselves, necessitates automated information extraction from these unstructured text sources.

For identifying more general entity types, many "off-the-shelf" software packages give impressive results using proven supervised methods trained on enormous corpora of labeled text. Because the training data is only annotated with names, geo-political entities, dates, etc., these general entity recognition tools are rendered useless when expected to extract the relatively foreign entities that occur in domain-specific documents. As exemplified by our need for entity extraction in the cyber-security domain, there are many domain-specific applications for which entity extraction will be very beneficial, although usually no labeled training data is available; moreover, manual annotation of a sufficiently large amount of domain text is extremely costly.

This paper describes an automated process for creating an annotated corpus from text associated with structured data that can produce large quantities of labeled text very quickly. More specifically, the wealth of structured data available in the cyber-security domain is leveraged to automatically label associated text descriptions, and made publicly available online[2]. While labeling these descriptions may be useful in itself, the intended purpose of this corpus is to serve as training data for a supervised learning algorithm that accurately labels other text documents in this domain, such as blogs, news articles, and tweets.

To ensure the usefulness of the automatically annotated corpus, we evaluate precision and recall, obtaining very good results. Additionally, en route to a trained classifier for identifying cyber-security concepts, we construct a maximum entropy model and evaluate three techniques for training the model, showing the Collins

The Department of Homeland Security sponsored the production of this material under DOE Contract Number DE-AC05-00OR22725 for the management and operation of Oak Ridge National Laboratory.



[1] http://nvd.nist.gov/, http://www.osvdb.org/, http://www.exploit-db.com/

[2] https://github.com/stucco/auto-labeled-corpus

Perceptron decisively gives the best results.

## II. Background

*1) Entity Extraction in Cyber-Security:* Our overall goal of automatically labeling cyber-security entities is similar to a few previous efforts. In order to instantiate a security ontology, More et al. [19] attempt to annotate concepts in the Common Vulnerability Enumeration (CVE)[3] descriptions and blogs with OpenCalais, an "out-of-the-box" entity extractor [22]. Mulwad et al. [20] expand this idea by first crawling the web and training a decision classifier to identify security relevant text. Then using OpenCalais along with the Wikipedia taxonomy, they identify and classify vulnerability entities.

While the two sources above rely on standard entity recognition software, such tools are not trained to identify domain specific concepts, and they unsurprisingly give poor results when applied to more technical documents (as shown in Figure 2). This is due to the general nature of their training corpus; for example, the Stanford Named Entity Recognizer is trained on the CoNLL-2003 data, consisting of news documents from the Reuters Corpus with approximately 300,000 labeled tokens – mainly the names of people, places, and organizations [13], [26]. Similar findings are noted in Joshi et al.'s recent work [17], where OpenCalais, The Stanford Named Entity Recognizer, and the NERD framework [23] were all generally unable to identify cyber-security domain entities. Because these tools do not use any domain specific training data, specific domain entities are either unrecognized or are labeled with descriptions that are too general to be of use (e.g., "Industry Term"). The Joshi et al. paper later supplies the Stanford Named Entity Recognizer framework with appropriate hand-labeled training data from this domain, and it is then able to produce good results for most of their domain specific entity types.

More specifically, the Joshi et al. [17] work also addresses the problem of entity extraction for cyber-security with a similar solution, namely, by training a supervised learning algorithm to identify desired entities and then populating an RDF linked data representation. Unlike our approach, which introduces an automated way to generate an arbitrarily large corpus and writes an entity extraction algorithm completely from scratch, their approach, conversely, involves painstakingly hand-annotating a small corpus that is then fed into Stanford's "off-the-shelf" template for training a conditional random field entity extractor [13]. In all, they label a training corpus of 350 short text descriptions, mostly from CVE records, with categories surprisingly similarly to ours. We note that their publication was made available only days before this paper was submitted, and the two research efforts occurred independently and in parallel. While their work has identified the same cyber-security problem, they do not furnish a data set labeled for this domain, nor do they address the more general problem of how to automate the labeling process when no training data exists.

Given this general lack of domain specific training data, there has been some work considering semi-supervised methods instead of supervised methods because they generally need much less training data. Although a thorough discussion of semi-supervised methods for entity extraction is outside the scope of the current manuscript, such techniques have yielded worthwhile results; for example see [6], [16], [4], [8], [9], and [15]. To our knowledge only one such effort focuses on cyber-security; current work by McNeil et al. [18] performed in parallel to ours develops a novel bootstrapping algorithm and describes a prototypical implementation to extract information about exploits and vulnerabilities. Using known databases for seeding bootstrapping methods is also not uncommon; for example, see [14].

*2) Auto-Labeling:* Previous work has incorporated variations of auto-labeling in several different contexts where NLP is needed and no training data exists. "Distant labeling" generally refers to the process of producing a gazetteer for each database field and performing a dictionary look-up to label text that is not directly associated with a given database record. While gazetteers give poor results in an unconstrained setting [5], accurate results can be achieved when the text has little variation. An example is Seymore et al. [24] who use a database of BibTeX entries and some regular expressions to produce training data for a hidden Markov model by labeling headers of academic papers.

In general, more accurate labels are possible if there is a direct relationship between a given database record and the text entry to be labeled, such as if a text description occurs as a field of a database, or a separate text document is referenced for each record, as is the case for our setting. For example Craven and Kumlien [11] train a naive Bayes classifier to identify sentences containing a desired entity via "weak labeling". Specifically, given a database record that includes an entity name along with a reference to an academic publication, positive sentences occurring in the article's abstract are automatically labeled. This is shown to yield better precision and recall scores than using a smaller hand-annotated training corpus and obviates the tedious manual labor.

More recently, Bellare and McCallum [2] also use a BibTeX database to label corresponding citations and then train a classifier to segment a given citation into authors, title, date, etc. Because their goal is to create a text segmentation tool, they rely on the implicit assumption that every token will receive a label from the given database field names. As our goal is to identify and classify specific entities in text, no such assumption can be leveraged.

[3] http://cve.mitre.org/

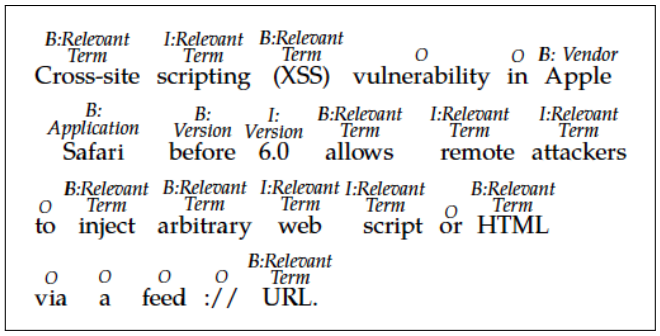


Fig. 1. NVD text description of CVE-2012-0678 with automatically generated labels.

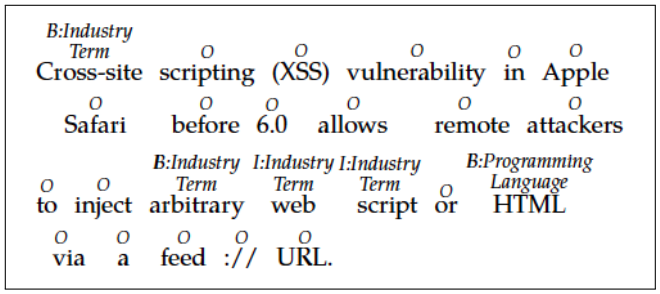


Fig. 2. NVD text description of CVE-2012-0678 with labels from OpenCalais.

While a few instances of automated labeling have occurred in the literature, to our knowledge no previous work has addressed the accuracy of the automatically prescribed labels. Rather, an increase in accuracy of the supervised algorithm is usually attributed to the improved training data, which is facilitated by the automated process. We note that the precision and recall of an algorithm's output is determined by comparison against the training data, which may or may not have correct labels. In order to address the quality of our auto-labeling, we have randomly sampled sentences for manual inspection (see Results Subsection III-5).

## III. Automatic Labeling

*3) Data Sources:* To build a corpus with security-relevant labels, we seek text that has a close tie to a database record and use its field names for the record to label matching entries in the text. When a vulnerability is initially discovered, the Common Vulnerability Enumeration (CVE) is usually the first structured source to ingest the new information and it provides, most importantly, a unique identification number (CVE-ID), as well as a few sentence overview. Shortly afterward, the National Vulnerability Database (NVD) incorporates the CVE record and adds additional information such a classification of the vulnerability using a subset of the Common Weakness Enumeration (CWE) taxonomy[4], a collection of links to external references, and other fields. Hence, the NVD provides both informative database records and many structured fields to facilitate auto-labeling. All NVD descriptions from January 2010 through March 2013 have been auto-labeled and comprise the lion's share of our corpus.

While our main source for creating an auto-labeled corpus is the NVD text description fields, the universal acceptance of the CVE-ID allows text from other sources to be unambiguously linked to a specific vulnerability record in the database. The Microsoft Security Bulletin provides patch and mitigation information and gives a wealth of pertinent text related to a specific vulnerability identified by the CVE-ID.[5] Specific text fields include an "executive summary" as well as "revision", "general", "impact", "target set", "mitigation", "work around", and "vendor fix" descriptions; moreover, while not all text fields are populated for a given record, many times a single text field will have multiple descriptions by different authors. Every description for the previous year's MS-Bulletin entries were added to our corpus.

Lastly, the Metasploit Framework contains a database of available exploits, which includes a text description, several categorizations and properties, and a reference to the associated vulnerability, usually the CVE-ID.[6] By linking these text sources to the NVD via CVE-IDs we are able to leverage the structured data for very precise labeling of the unstructured data.

Overall, a corpus of over 850,000 tokens with automatic annotations are available online at https://github.com/stucco/auto-labeled-corpus.

*4) Auto-Labeling Details:* Given a database record and a block of text, our algorithm assigns labels to the entities in the text as follows:

- **Database Matching.** Any string in the text that exactly matches an entry of the record is labeled with the name of the database field.
- **Heuristic Rules.** A variety of heuristic rules are used for identifying entities in text that are not direct matches of database fields. For example, the database lists every version number affected by a vulnerability, but such a list is almost never written in text; rather, short phrases such as "before 2.5", "1.1.4 through 2.3.0", and "2.2.x" usually appear after a software application name; consequently, a few regular expressions combined with rules identifying both labels and features of previous words give precise identification of version entities. Similarly, source code file names, functions, parameters, and methods, although not in the database, are often referenced in text. As file names end in a file extension (e.g., ".dll") and the standards of camel- and snake-case (e.g., camelCaseExample, snake_case_example) are universal, such entities are easily distinguishable

[4]http://cwe.mitre.org/

[5]http://technet.microsoft.com/en-us/security/bulletin

[6]http://www.metasploit.com/

by their features.

- **Relevant Terms Gazetteer.** In order to extract short phrases that give pertinent information about a vulnerability, a gazetteer of relevant terms is created, and phrases in the text matching the gazetteer are labeled "relevant term". As mentioned above, each record in the NVD includes one (of twenty) CWE classifications, which gives the vulnerability type (e.g., SQL injection, cross-site-scripting, buffer errors). As the goal of CWE is to provide a common language for discussing vulnerabilities, many phrases indicative of the vulnerability's characteristics occur regularly. To construct the gazetteer of relevant terms, the NVD is sorted by CWE type, and statistical analysis of the text descriptions for a given CWE classification is used to find the most prevalent unigrams, bigrams, and trigrams. Commonly occurring but uninformative phrases (e.g., "in the", "is of the") were discarded manually. We note that Python's Natural Language Toolkit facilitated tokenization and frequency distributions of $n$-grams [5]. Examples of relevant terms include "remote attackers", "buffer overflow", "execute arbitrary code", "XSS", and "authentication issues".

All together, the following is the comprehensive list of labels used: vendor, application name, hardware, operating system, version, update, edition, CVE-ID, file, function or method, parameter, and relevant term.

Because many multi-word names are commonplace, standard IOB-tagging is used; specifically, the first word of an identified entity name is labeled with a "B" (for "beginning") followed by the entity type, and any word in an entity name besides the first is tagged with an "I" (for "inside") followed by the entity name. Unidentified words are labeled as "O". An example of an automatically labeled NVD description is given in Figure 1.

*5) Auto-Labeling Results:* As the overall goal is to produce a machine learning algorithm that will identify entities in a much broader class of documents, thereby aiding security analysts, the accuracy of the algorithm, and therefore the training data, is very important. While both high precision and recall are ideal, precision is more important for our purposes as reliable information is mandatory. More specifically, in a high recall but low precision setting, nearly all desired entities would be returned along with many incorrectly labeled ones; hence, the quality of the data returned to the user would suffer. On the other hand, if all information extracted from text sources is correct, anything returned is an immediate value-add. In general, this is guaranteed by high precision in the auto-labeling process, which we ensure by constructing precise heuristics and using a specific database record to label closely related text.

In order to test the accuracy of the auto-labeling, about 30 randomly sampled text descriptions from each source were manually labeled. Because the labeled relevant terms are exactly those phrases appearing in the gazetteer and deciding the validity of relevance of an arbitrary phrase is highly subjective, our scoring ignored the relevant term labels.

- **NVD** Precision: $100\%$, Recall: $77.8\%$, F=.875
- **MS-Bulletin** Precision: $99.4\%$, Recall: $75.3\%$, F=.778
- **Metasploit** Precision: $95.3\%$, Recall: $54.3\%$, F=.691

To our knowledge, similar work has assumed correct automatically generated labels and ignored investigating the accuracy of the labels. In total over 850,000 tokens (almost three times the size of the training corpus used to train the Stanford NER) have been labeled very quickly and with high accuracy, and increasing the corpus size as necessary is both expedient and easy. We hope the proposed method can facilitate labeling data in many other domains.

## IV. Maximum Entropy Model

While the auto-labeled corpus may be useful in its own right, the overall goal is to train a classifier that can apply domain-appropriate labels to a wider class of documents including news articles, security blogs, and tweets. As maximum entropy models (MEMs) are state-of-the-art for sequential tagging problems, we are currently developing a customized version. Performance, both in terms of accuracy and computational/temporal expense, is highly dependent on implementation details, most notably feature selection and parameter estimation techniques. This section gives a proof-of-concept implementation of a MEM with only binary tags (1 for an entity name, 0 else) in order to make implementation decisions in a computationally efficient setting before scaling-up to the full set of labels. Our results illustrate the vast performance difference between three methods for learning the parameter vector from the training data; specifically, the Collins Perceptron is shown to vastly outperform models fitted via maximum likelihood estimation with the L-BFGS or OWL-QN optimization algorithms. A brief mathematical overview of a history-based MEM is followed by the implementation details used in our experiment and results.

*6) Mathematical Overview:* Following Berger et al. [3] and Ratnaparkhi [21], we implement a MEM, a discriminative model shown to excel in sequential labeling. Derived by maximizing Shannon's Entropy in the presence of constraint equations, MEMs provide a principled mathematical foundation that ensures only the observed features design the probability model.

For a given sentence $w = (w_1, \ldots, w_n)$ and corresponding tag sequence $t = (t_1, \ldots, t_n)$, the conditional probability of $t$ given $w$ is estimated as

$$p(t|w) \equiv \prod_{i=1}^{n} p(t_i|t_{i-2}, t_{i-1}, w_{i-2}, w_{i-1}, w_i)$$

with $t_0, t_{-1}, w_0, w_{-1}$ defined to be distinguished start symbols. Hence the probability of tag $t_j$ being assigned

to word $w_j$ is conditioned on the previous two tags, as well as the current word and previous two words. For notational ease we let $\bar{t}_i = (t_{i-2}, t_{i-i}, t_i)$, and similarly for $\bar{w}$. As prescribed by the MEM,

$$p(t_i|t_{i-2}, t_{i-1}, w_{i-2}, w_{i-1}, w_i) \equiv \frac{e^{f(\bar{t}_i, \bar{w}_i)\cdot v}}{z(\bar{t}_i, \bar{w}_i)} \quad (1)$$

where $f = (f_1, \ldots, f_m)$ denotes a feature vector, $v = (v_1, \ldots, v_m)$ the parameter vector to be learned from the training data, and $z(\bar{t}_i, \bar{w}_i) \equiv \sum_{\hat{t}} \exp[f(t_{i-2}, t_{i-1}, \hat{t}, \bar{w}_i) \cdot v]$, i.e., $z$ is the appropriate constant to ensure the sum of Equation 1 over the sample space is one. An example of a feature (i.e., a component of the feature vector) is

$$f_1(\bar{t}_i, \bar{w}_i) = \begin{cases} 1 & \text{if } t_i = \text{ B: Vendor}, w_{i-1} = \text{ "the"} \\ 0 & \text{else.} \end{cases}$$

After fixing a set of features, one must decide on the "best" parameter vector $v$ to use and many techniques for fitting the model to the training data (i.e., learning $v$) exist [12]. Perhaps most common is maximum likelihood estimation (MLE), which assumes each (sentence, tag)-pair is independent and chooses the $v$ that maximizes the probability of the training data. To prevent overfitting, we employ $l^1$ regularization, or equivalently, use a Laplacian prior on $v$. Moreover, as is common for MEM, the log-likelihood is maximized; specifically, we seek the argument maximum of

$$L(v) = \sum_{(w,t)} \sum_{i=1}^{n} [f(\bar{t}_i, \bar{w}_i) \cdot v - \log z(\bar{t}_i, \bar{w}_i)] - \frac{\lambda}{2} \sum_{j=1}^{m} |v_j|,$$

where $\lambda$ is the regularization parameter and the left most sum is over the training set of (sentence,tag)-pairs. Note that the regularized log-likelihood, being strictly concave, has a unique maximum and so we can use gradient-based methods to find the optimal $v$. We experiment with two optimization algorithms, namely L-BFGS [7], a well-known limited-memory quasi-Newton method, and OWL-QN [1], an algorithm designed specifically for $l^1$ regularization. Building on the quasi-Newton approach of L-BFGS, OWL-QN avoids the non-differentiability issues that arise when an element of $v$ is zero by approximating a quadratic function on an orthant containing the current point at each step.

In addition, we implement the Collins Perceptron [10], a heuristic method that loops over the training corpus several times, updating $v$ at each step $n$ by setting $v_n = v_{n-1} + f(w, t) - f(w, t')$, where $t$ is the "gold standard" tag sequence for $w$ given by the training data and $t'$ the most probable tag sequence for the sentence given by $v_{n-1}$.

For a fixed $v$ and (unlabeled) sentence $w$, we choose the most probable tag sequence $t$. Because our model is history-based, the Viterbi algorithm [27], a dynamic programming technique, is used to find the optimal tag sequence and increase computational performance.

*7) Parameter Estimation Results:* To test our methods we consider various sized subsets of the 15,192 auto-tagged NVD text descriptions, partition each into training and test sets, and compare precision, recall, and F-score for the three methods of training $v$. For computational efficiency we relabel the words as "1" for an an entity and "0" for non-entities. Additionally, to develop robust features, part-of-speech tags are applied to each word using NLTK, and regular expressions are used to identify words that begin with a digit, contain an interior digit, begin with a capital letter, are camel-case, are snake-case, or contain punctuation. We then generate binary features that relate the current tag to the previous two tags and the current or previous two words including their parts-of-speech and matched regular expressions. The results are reported in Table I. The number of iterations of the Collins Perceptron is 20 for all sets, save the largest, where 10 was used to save time.

Note that we calculate the precision, recall, and F-score with respect to the auto-labeled data, which biases the precision downward and recall upward. Upon examining the output, we find that when the auto-labeler labels a word "O" and the log-linear model labels it "1", the log-linear model is nearly always correct. Thus, the true precision lies much closer to 100%.

While all of the optimization algorithms yield high precision for the various training and test set sizes, their recall values differ drastically. OWL-QN appears to stop prematurely in some trials, failing to find the correct maximum of the regularized log-likelihood function. Therefore, its recall suffers, and varies between 3% and 70%. L-BFGS, on the other hand, comes much closer to finding the true maximum, but produces very low recall (21-41%). Perhaps surprisingly, Collins' Perceptron unwaveringly generates the overall best results. On a training set of size 15,192 it gives 98.6% precision and 66.3% recall, for an F-score of 0.79. Previous testing has shown much more competitive results among methods [25].

We expect the Collins Perceptron to similarly produce the best results when implementing the model with the full set of labels. Also included in the table is the value of the MLE at the various $v$-values found by each method. Note that the $v$ produced by the Collins Perceptron provides good classification results while being far from the optimal $v$; hence in these cases, the MLE does a poor job of helping with the end-goal of accurate labeling.

## V. Conclusion and Future Work

Our auto-labeling techniques give an expedient way annotate unstructured text using associated structured database fields and we provide a publicly available corpus for entity extraction in the cyber-security domain. As the structured and unstructured sources harnessed for auto-labeling provide regular updates, we envision

TABLE I
RESULTS FROM THE LOG-LINEAR MODEL

| Size | Algorithm | $L(v)$ | Precision | Recall | F-score |
|---|---|---|---|---|---|
| 100 | L-BFGS | 2.65E+52 | 1.0000 | 0.2085 | 0.3450 |
| 100 | OWL-QN | -3.80E+02 | 0.7660 | 0.6710 | 0.7160 |
| **100** | **Perceptron** | **-3.63E+03** | **0.9766** | **0.5901** | **0.7357** |
| 500 | L-BFGS | 2.27E+55 | 0.9195 | 0.3248 | 0.4800 |
| 500 | OWL-QN | -1.80E+03 | 1.0000 | 0.0277 | 0.0540 |
| **500** | **Perceptron** | **-9.62E+03** | **0.9769** | **0.6870** | **0.8065** |
| 2500 | L-BFGS | 5.22E+44 | 0.9836 | 0.2585 | 0.4094 |
| 2500 | OWL-QN | -3.66E+03 | 1.0000 | 0.0027 | 0.0053 |
| **2500** | **Perceptron** | **-3.43E+04** | **0.9724** | **0.6582** | **0.7850** |
| 10000 | L-BFGS | 6.57E+42 | 0.9758 | 0.2405 | 0.3859 |
| 10000 | OWL-QN | -2.87E+04 | 0.9377 | 0.4136 | 0.5740 |
| **10000** | **Perceptron** | **-1.09E+05** | **0.9852** | **0.6643** | **0.7935** |
| 15192 | L-BFGS | | | | |
| 15192 | OWL-QN | -8.58E+04 | 0.8385 | 0.7035 | 0.7651 |
| **15192** | **Perceptron** | -1.66E+05 | **0.9857** | **0.6628** | **0.7926** |

Note: Size refers to the number of text descriptions, which contain about 50 words on average. Each data set is divided 80/20 into training and test sets. For each size, we ran all three algorithms on the same randomly chosen set to find $v$, so the difference in results is solely due to the algorithms. (L-BFGS run on the 15,192 sized set did not yet converge at the time of submission).

building an ever-growing auto-labeled corpus by labeling data as it is released, and making it publicly available. Additionally, preliminary testing using our corpus exhibits results showing Collins' Pereptron outperforms MLE with two optimization techniques for parameter fitting a log-linear model. In the future, we intend to train a classifier that incorporates the entire set of labels mentioned in Subsection III-4, trained with the Collins' Perceptron.

## REFERENCES


[1] G. Andrew and J. Gao. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 33–40, New York, NY, USA, 2007. ACM.

[2] K. Bellare and A. McCallum. Learning extractors from unlabeled text using relevant databases. In *Sixth International Workshop on Information Integration on the Web*, 2007.

[3] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[4] J. Betteridge, A. Carlson, S. A. Hong, E. R. Hruschka Jr, E. L. Law, T. M. Mitchell, and S. H. Wang. Toward never ending language learning. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*, pages 1–2, 2009.

[5] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. O'Reilly, 2009.

[6] S. Brin. Extracting patterns and relations from the world wide web. In *The World Wide Web and Databases*, pages 172–183. Springer, 1999.

[7] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, Sept. 1995.

[8] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr, and T. M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 101–110. ACM, 2010.

[9] A. Carlson, S. A. Hong, K. Killourhy, and S. Wang. Active learning for information extraction via bootstrapping, 2010.

[10] M. Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[11] M. Craven and J. Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86. AAAI Press, 1999.

[12] C. Elkan. Log-linear models and conditional random fields. *Tutorial notes at CIKM*, 8, 2008.

[13] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[14] J. Geng and J. Yang. Autobib: Automatic extraction of bibliographic information on the web. In *Proceedings of the International Database Engineering and Applications Symposium*, IDEAS '04, pages 193–204, Washington, DC, USA, 2004. IEEE Computer Society.

[15] R. Huang and E. Riloff. Bootstrapped training of event extraction classifiers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 286–295. Association for Computational Linguistics, 2012.

[16] R. Jones. *Learning to extract entities from labeled and unlabeled text*. PhD thesis, University of Utah, 2005.

[17] A. Joshi, R. Lal, T. Finin, and A. Joshi. Extracting cybersecurity related linked data from text. In *Proceedings of the 7th IEEE International Conference on Semantic Computing*. IEEE Computer Society Press, 2013.

[18] N. McNeil, R. A. Bridges, M. D. Iannacone, B. Czejdo, N. Perez, and J. R. Goodall. PACE: Pattern accurate computationally efficient bootstrapping for timely discovery of cyber-security concepts, 2013.

[19] S. More, M. Matthews, A. Joshi, and T. Finin. A knowledge-based approach to intrusion detection modeling. In *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on Semantic Computing and Security*, pages 75–81. IEEE, 2012.

[20] V. Mulwad, W. Li, A. Joshi, T. Finin, and K. Viswanathan. Extracting information about security vulnerabilities from web text. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 03*, WI-IAT '11, pages 257–260, Washington, DC, USA, 2011. IEEE Computer Society.

[21] A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, April 16 1996.

[22] T. Reuters. OpenCalais, 2009.

[23] G. Rizzo and R. Troncy. NERD: A framework for unifying named entity recognition and disambiguation extraction tools. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 73–76. Association for Computational Linguistics, 2012.

[24] K. Seymore, A. Mccallum, and R. Rosenfeld. Learning hidden markov model structure for information extraction. In *In AAAI 99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.

[25] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 134–141, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[26] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

[27] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theor.*, 13(2):260–269, Sept. 2006.