

Support Vector Machines to Weight Voters in a Voting System of Entity Extractors

Deborah Duong, Ben Goertzel, Jim Venuto, Ryan Richardson, Shawn Bohner, Edward Fox,
Virginia Tech

Abstract—Support Vector Machines are used to combine the outputs of multiple entity extractors, thus creating a composite entity extraction system. The composite system has a significantly higher f-measure than any of the component systems. Compared to a standard voting technique for combining the results of multiple entity extractors, the SVM approach produces comparable precision and recall statistics but tends to utilize fewer of the component entity extractors, thus providing superior computational efficiency, which is critical in practical applications. In this paper, we present our experimental results of comparing a standard voting technique with SVM that each aggregate four entity extractors. We also describe our future plans of integrating agent-based technology into our experimental testbed where we examine the evolution of composite techniques as part of the analysis stream. Given that much of the improvement comes from tuning the algorithms to the data stream with a human-in-the-loop, we are considering the merits of employing cognitive agents that are strategically embedded in the workflow for processing data. As we tune the algorithms for better performance on the data streams, we envision agents learning the patterns of data streams and apply the appropriate tuning to ensure optimality.

I. INTRODUCTION

Multiple software systems carrying out named entity recognition exist, each with their own strengths and weaknesses. It is a natural idea to create a composite entity extractor by combining the outputs of multiple entity extractors to get a single combined signal. However, the optimal way to perform this combination is not clear. Previous authors have used a simple voting approach which, for example, accepts the assignment of a word to an entity category if two out of three entity extractors agree that it is an entity in that category [1]. Here we explore a more sophisticated approach to the output combination problem, utilizing Support Vector Machines (SVM) [2]. SVM offers a trainable linear classification approach that through our research here is showing promising results.

While voting is a powerful approach for combining different estimates, it falls short in cases where the number of estimates being combined is very small. It may perform

suboptimally due to its failure to account for the dependencies among the component estimates. This phenomenon may be relevant to entity extraction because entity extraction systems are often black boxes that use many techniques, some of them alike and some different. Moreover, each product may perform several functions, and may be more accurate in some tasks than in others. These functions may be performed on many types of data, and some products might be better on some types, while other products might be better on other types.

Support Vector Machines are a supervised learning technique used to apply linear classification to non-linear classification problems. It has been used for text classification for several years [3]. SVM works by mapping the positive and negative data points into a high-dimensional space and attempts to find a hyperplane which separates them with the maximum possible margin. This hyperplane is called the maximum-margin hyperplane, and the vectors closest to the hyperplane are called the support vectors. Boser, Guyon and Vapnik developed the technique of using a (possibly non-linear) kernel to transform the feature space into a higher-dimensional one. Thus the maximum-margin hyperplane may be non-linear in the original input space, while being linear in the high-dimensional transformed space. In our research, we have used a linear kernel since they can be trained more quickly.

The practical motivation underlying this work is a desire to improve the quality of named entity recognition available to the intelligence analysis community. In this application domain, the primary goals are to achieve increased precision without paying too large a price in terms of recall, and to achieve this without too much increase in computational complexity. Precision is often more important to the intelligence community than recall because there is far more data available than individual intelligence analysts are able to consume, and it does no good to recall information that can not be processed anyway. Computational efficiency is important for similar reasons: applying entity extraction to huge document stores requires considerable processing resources.

In practice, voting systems are rarely used for entity extraction in the intelligence analysis community because they are computationally inefficient. A voting system that uses four voters takes roughly four times as long as a single entity extraction system. Our results, reported here, suggest that using SVMs rather than simple voting to combine the

D. Duong, J. Venuto, and B. Goertzel are with the Virginia Tech Applied Research Laboratory for National and Homeland Security, 2000 N. 15th Street, Suite 503, Arlington, VA 22201. (phone: 703-875-0158; fax: 703-875-0161; e-mail: dduong@vt.edu). R. Richardson and E. Fox are in the Digital Libraries Research Lab at Virginia Tech in Blacksburg, Virginia. S. Bohner is in the Virginia Tech Systems and Software Engineering Lab at Virginia Tech in Blacksburg, Virginia.

results of multiple entity extractors may provide a better solution for some significant cases. We show that using SVMs to combine the results of four open-source entity extractors on the standard CONLL-2002 test corpus [4] results in precision and recall statistics similar to that of simple voting, but with significant increases in computational efficiency. Employing learning combination rules that, for each category, completely ignore some of the component entity extractors (because they contribute nothing for classifying entities in that category beyond what is already contributed by the SVM combination of the other entity extractors) we found that the computational requirements were substantially reduced to produce comparable precision and recall. For this reason, we believe that using SVMs to combine the results of multiple entity extractors may be the best current approach to entity extraction for intelligence analysis applications, and perhaps for other applications as well.

II. PROCEDURE

We have implemented an algorithmic approach that provides high-precision annotation of entities (persons, locations, and organizations) in text, by running several different entity extractors and combining their results using a machine learning algorithm, Support Vector Machines [5].

We trained three different SVMs using the open source package SVM-Light [6], one for each function of named entity extraction: person, location, and organization extraction. We took four major open source entity extractors, the University of Sheffield's GATE [7], Aliasi's LingPipe [8], MITRE's Alembic [9], and the OpenNLP project on source forge [10], and ran them on the 494 training documents from the Reuters corpus [11] manually marked for persons, places, and organizations for the CONLL-2002 [4] named entity recognition task. We then train SVM-Light with the four entity extractors' decisions on the annotation of an entity. For example, if GATE assigns a word as a "person" but LingPipe, Alembic, and OpenNLP do not, and the manually marked documents assign the word as a person, then the SVM trained on persons gets trained, for this particular instance, on a positive example consisting of a 1 for GATE, a 0 for LingPipe, a 0 for Alembic, and a 0 for Open NLP. There are 6058 persons, 3945 organizations, and 5863 locations in these 492 CONLL training documents, constituting the training data for the three SVMs

After the three SVMs are trained, each SVM is then tested on the CONLL test set. This test set is manually marked with 2282 persons, 2421 organizations, and 1995 locations, enough for statistically significant results at the .99 level of confidence.

We judged the performance of the system via a weighted f-measure in which precision is worth five times recall, a measure chosen because of the paramount importance of precision in the intelligence analysis application domain. Of course for other applications, this f-measure may be reduced

showing preference towards recall.

III. RESULTS

Our results suggest that the SVM is equally as effective as the standard voting technique in raising the precision of annotations while keeping recall at a reasonable level. However, the SVM approach provides an advantage in terms of computational efficiency, because it turns out that on two of the three entity categories, the SVM does not use all of the component entity extractors. So, in addition to improving precision without sacrificing recall unacceptably, the SVM approach also serves to tell which extractors do not even have to be run on the data because they do not improve the results. This is critical in practical applications due to the opportunity it provides for saving computation time.

The F measure was calculated according to the formula, $(a+1)pr/r+(ap)$, where a is the weight of recall, $0 - \infty$, when precision is weighted 1. So, when precision is weighted at 5 times recall, " a " is set to .2, and when they are equally weighted, precision is set to 1.

When annotating persons, the SVM learned the rule that the vote of any two of the four entity extractors should be equally weighted and considered useful for determining if words in text are a person's name. Since here it simply learned to vote, the SVM had the same precision as voting, 0.91, resulting in a weighted f measure of 0.83. This is 6 points higher than the highest scorer of the individual extractors, LingPipe.

When annotating locations, the SVM learned the rule that a vote of any of the three extractors besides OpenNLP counts towards the result. With this rule, the SVM exceeds by one point the performance of the voting system, which requires all four extractors; and exceeds the performance of the next individual entity extractor, GATE, by seven points. Its weighted f measure is 0.78.

Organizations are difficult for all of the extractors, and SVM learned the rule to only consider GATE's vote, making its performance equal GATE's, and exceed the voting entity extractors by two points.

The use of SVM also gives us results that cannot be inferred by isolated testing. For instance, in the location task, OpenNLP looks much better than Alembic based on precision and f-measure, but OpenNLP actually added no value when used in combination. This could be because Alembic finds particular location names that the other tools, including OpenNLP, do not. Also, for the Organizations task, GATE and LingPipe appear comparable based on recall and unbiased f-measure, but when combined, LingPipe does not contribute anything more.

In total, the weighted f measure for the voting and SVM systems is comparable, at 0.72, and 7% higher than the overall best individual entity extractor, GATE. However, compared to voting, the SVM saves us three runs of extractors for organizations and one run of an extractor for locations, and thus cuts the overall computational time significantly, assuming that the tools allow us to switch off

one extractor (like the organization extractor) while still running the others. Often a given application will only require one particular type of entity and not all three, and in these cases we can see larger savings. For instance, in the Organization name-finding task, SVM tells us that we only need to run GATE, saving us the computation time of the three other extractors. For sake of comparison, when the precision is set equal to recall, the unweighted f measure of the SVM and voting techniques are the same, at 0.6, and 5% higher than GATE. Tables 1 -4 summarize these results.

PERSONS	Precision	Recall	F (Precision = Recall)	F (Precision = 5 X Recall)
SVM	0.91	0.57	0.70	0.83
Voting 2 / 4	0.91	0.57	0.70	0.83
GATE	0.84	0.53	0.65	0.76
LingPipe	0.87	0.50	0.64	0.77
OpenNLP	0.86	0.30	0.44	0.66
Alembic	0.72	0.42	0.60	0.64

Table 1. SVM, Voting System and Individual Entity Extractor Results for Persons.

As you can see in Table 1, we compare the voting system, the individual entity extractors, and the SVM results for identifying individual persons. Note that the precision and recall numbers are identical here for Voting and SVM (and they exceed the performance of the individual entity extractors by a significant margin). Further, when the f-measure is calculated with the precision = 5X recall, the performance with respect to what we believe is the case for precision being more important than recall for the intelligence community, it is increased 15.66%.

LOCATIONS	Precision	Recall	F (Precision = Recall)	F (Precision = 5 X Recall)
SVM	0.78	0.78	0.78	0.78
Voting 2 / 4	0.76	0.78	0.78	0.77
GATE	0.69	0.81	0.74	0.71
LingPipe	0.66	0.70	0.68	0.67
OpenNLP	0.83	0.40	0.54	0.71
Alembic	0.45	0.49	0.47	0.46

Table 2. SVM, Voting System and Individual Entity Extractor Results for Locations.

When we localized our results to location entities extracted, the results flattened out a bit in Table 2. While SVM's precision was slightly better at 0.78 vs. 0.76, the recall was not effected and the f-measure was only 0.01 higher for the SVM. It is worth noting that OpenNLP had slightly higher precision than that of SVM and Voting. However, for the recall and ultimately for the f-measures, it

was not competitive.

ORGANIZATIONS	Precision	Recall	F (Precision = Recall)	F (Precision = 5 X Recall)
SVM	0.59	0.15	0.24	0.40
Voting 2 / 4	0.59	0.13	0.23	0.38
GATE	0.59	0.15	0.24	0.40
LingPipe	0.43	0.16	0.23	0.33
OpenNLP	0.49	0.07	0.13	0.25
Alembic	0.43	0.12	0.19	0.31

Table 3. SVM, Voting System and Individual Entity Extractor Results for Organizations.

Table 3 depicts our results for organization entities, the most difficult area for the extractions for all of the approaches, but the one that illustrates a strength of using SVM. Note that the precision numbers were identical for Voting and SVM (and even GATE), and the recall numbers for GATE (the best performer of the individual approaches) and SVM were identical – making SVM, in terms of the intelligence community, as competent as the best performer. However it also takes only the computational time of the best performer, as opposed to the greater computational time of the voting system.

ALL ENTITIES	Precision	Recall	F (Precision = Recall)	F (Precision = 5 X Recall)
SVM	0.8	0.48	0.6	0.72
Voting 2 / 4	0.79	0.48	0.6	0.72
GATE	0.72	0.48	0.57	0.67
LingPipe	0.68	0.48	0.53	0.62
OpenNLP	0.79	0.48	0.38	0.58
Alembic	0.53	0.24	0.41	0.49

Table 4. SVM, Voting System and Individual Entity Extractor Results for All Entities.

Combining all of the analyses, Table 4 summarizes our results for all the examined entities. SVM has comparable results to the voting systems and both significantly outperform the individual entity extraction techniques. Again, the payoff here is in the computational savings gained by the SVM techniques which are less compute intensive. In the next section, we examine some of our preliminary work in examining cognitive agents for this purpose.

IV. META-ANALYSIS

So, what does this all mean for our research? Much of our work at the Virginia Tech Applied Research Laboratory for National and Homeland Security is focused on exploring

combinations of technologies to improve the performance of intelligence analysis for our sponsors.

SVM finds a good subset of voters which perform as well as a voting system, thus saving computational time. However, it does not find the set which optimizes results on the basis of criteria for computational cost, monetary cost, precision and recall. We intend to combine the SVM with a coevolving agent based system that will be useful for both integrating and evaluating tools that can be used in combination.

For the future research here, we examined the three top distributed agent architectures in an in-depth comparative analysis, Grasshopper [12], the Java Agent Development Environment (JADE) [13] and the Cognitive Agent Architecture (COUGAAR) [14]. All of these were supported by Java-based platforms, although Cougaar significantly outperformed the others in our benchmarks [15,16].

Cougaar is an open source, distributed agent architecture – a product of over eight years of research and development, and over a \$150 million investment by the Defense Advanced Research Projects Agency (DARPA) under the Advanced Logistics Program (ALP) and the Ultra*Log program [17,18]. It is a large-scale, component-based, distributed agent architecture where agents communicate with one another by a built-in asynchronous message-passing protocol. These agents cooperate with one another to solve problems, storing the intermediate solutions in distributed blackboards across the agents. Agents are composed of related tasking modules that dynamically rework the solution as the problem parameters, constraints, and/or execution environment changes [16]. Agents consist primarily of a blackboard and a set of plug-ins. The blackboard is essentially a container of objects that adhere to publish/subscribe semantics. A Cougaar plug-in implements a piece of the core business logic associated with a given agent and is referentially uncoupled (e.g., a task or set of tasks implementing a given behavior). Plug-ins can publish objects, remove objects or publish changes to existing objects via the Blackboard. They can create subscriptions to be notified when objects are added, removed or changed in the Blackboard.

Cougaar agents can be arranged into a society that collectively solves a problem or class of problems. This architectural capability serves our research testbed. A society can comprise one or more communities of agents that share the same functional purpose or organizational commonality. Using Cougaar as an underpinning for our architecture provides a way to integrate the various components of our entity extraction and document understanding research into a scalable and measurable framework that can readily support our research advancements.

Cougaar incorporates a model of human cognition that enables agents to reason, plan, monitor, execute, and dynamically replan. Agents are autonomous, goal-oriented, taskable, and capable of learning. These aspects lend

themselves well to the tasks of integrating coevolutionary technology with SVM in analyzing data streams for the intelligence community. While these ideas are still being formulated, we see promise with the results that we obtained from the comparison of Voting, SVM, and individual entity extraction techniques.

V. CONCLUSION

We have run computational experiments comparing SVMs with a standard voting technique for combining the outputs of four open-source entity extractors on the CONLL test corpus. The SVM found simple rules with weighted f-measure comparable to that of the voting system, but saved computational cost compared to the latter. This savings is a critical benefit in the intelligence analysis domain and any other area where the number of documents being processed is very large.

We will pursue further experiments applying machine learning techniques to combine the results of a larger number of entity extractors, including commercial entity extractors. Our suspicion is that when the number and quality of entity extractors is significantly expanded the results will be even more dramatic, not only improving computational efficiency but also achieving better weighted f-measures than voting. It may also be worthwhile to explore the application of this technique to entity extraction in other domains, for example biomedical research articles, an area in which there is a large number of complex entity extractors implementing fundamentally different algorithms [19].

Further, we examined cognitive agent-based technology to help with the combining and tuning of entity extraction techniques as well as the other workflow aspects of processing data streams for document understanding.

REFERENCES

- [1] Henderson, Eric Brill. *Exploiting Diversity in Natural Language Processing: Combining Parsers*. In: Proceedings of the Fourth Conference on Empirical Methods in Natural Language Processing (EMNLP-99), pp. 187–194. College Park, Maryland, 1999.
- [2] Boser, B.E, Guyon, I., and Vapnik, V., A Training Algorithm for Optimal Margin Classifiers. *In Proceedings of the 5th ACM Conference on Computational Learning Theory (COLT '92)*: 144-152, 1992.
- [3] Joachims, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of the 10th European Conference on Machine Learning*, Chemnitz, Germany, 137-142, 1998.
- [4] CoNLL-2002 Language Independent Named Entity Recognition Task, The Sixth Workshop on Computational Language Learning, Taipei, Taiwan, 2002.
- [5] Vapnik, Vladimir N. *The Nature of Statistical Learning Theory*. Springer, 1995.

- [6] Joachims, Thorsten . *Making Large-Scale SVM Learning Practical*. LS8-Report, 24, Universität Dortmund, LS VIII-Report, 1998.
- [7] Cunningham, H.; D. Maynard, K. Bontcheva, V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*. Philadelphia, July 2002.
- [8] Baldwin, Breck. *CogNIAC: A Discourse Processing Engine*. Ph.D. Dissertation. University of Pennsylvania Computer and Information Sciences Department, 1995.
- [9] Day, David; John Aberdeen, Lynette Hirschman, Robyn Kozierok, Patricia Robinson and Marc Vilain. Mixed-Initiative Development of Language Processing Systems. Fifth Conference on Applied Natural Language Processing, Association for Computational Linguistics, 31 March -- 3 April, Washington D.C., U. S. A, 1997.
- [10] Open NLP, <http://opennlp.org/>, Feb 2006.
- [11] Reuters Corpus, Volume 1, English language, 1996-08-20 to 1997-08-19 (Release date 2000-11-03, Format version 1, correction level 0).
- [12] Grasshopper Website (<http://www.grasshopper.de>). 2004.
- [13] Java Agent Development Environment (JADE – <http://jade.tilab.com>). 2006.
- [14] Cognitive Agent Architecture (COUGAAR – <http://www.cougaar.org>). 2006.
- [15] Hartman, B., An In-Depth, Comparative Analysis of Three Agent Architectures – Grass-hopper, JADE and COUGAAR, in Computer Science. 2004, Virginia Tech. p. 108.
- [16] Bohner, S., Hartman, B., Eltoweissy, M., and Gracanin, D., Agents in Service-Oriented Wireless Sensor Networks. *International Journal of Wireless and Mobile Computing*, 2005.
- [17] COUGAAR Architecture Document for Cougaar Version 11.4. 2004, BBN Technologies.
- [18] Helsinger, A., Lazarus, R., Wright, W., Zinky, J. Tools and Techniques for Performance Measurement of Large Distributed Multiagent Systems. in 2nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS). 2003. Melbourne, Australia.
- [19] Kim, Jim-Dong et al. Introduction to the Bio-NLP Entity Task at JNLPBA 2004. In *Proceedings of JNLPBA 2004*.