Automated Annotation of Text Using the Classification-based Annotation Workbench (CLAW)

R. George, H. Nair, K. A. Shujaee
Department of Computer and Information Science
Clark Atlanta University
Atlanta, GA, USA
Email: rgeorge@cau.edu

D. A. Krooks, C. M. Armstrong
Construction Engineering Research Laboratory
Champaign, IL, USA

***Abstract*—Text annotation is used to mark up text using highlights, comments, footnotes, tags, and links. Manual annotation is a human intensive process and is not feasible for a large corpus of text. Classification is a technique that may be used to automate the annotation process. This paper develops a Classification-based Text Annotation Workbench (CLAW), an annotation assistance tool that incorporates automated classification to reduce the difficulty of manual annotation. There are several technical challenges posed by the practical nature of the text corpus and the annotation methodology. The text corpus, is large and consists of numerous reports, lessons learnt and best practices. Complexity is introduced due to the size of the documents, the variety of formats and the range of subject matter. The annotation taxonomy is extensive and unstructured and may be applied to the text body without constraints. Consequently, the search space for the label(s) become prohibitively large and it becomes necessary to adopt strategies that reduce the complexity of the classification process. We introduce a simplification technique to reduce the large classification search space. We improve precision by supplementing these predictive algorithms with similarity based measures and evaluate CLAW for performance using both prediction-based metrics and ranking-based metrics. It is shown that CLAW performs better than a competing algorithm on all evaluation metrics.**

Keywords-Text Annotation, Multi-label Classification, Bayes Theorem, Annotation Workbench.

I. INTRODUCTION

Text annotation is the practice of marking up text using highlights, comments, footnotes, tags, and links. This may include notes written for a reader's private purposes, as well as shared annotations written for the purposes of collaborative writing, editing, commentary, social sharing, and learning. Annotation of these documents is the first step in the automation of the processing of such documents with applications such as identification of socio-cultural constructs, and improved methods of query and retrieval.

Manual annotation is a human intensive process and is not feasible for a large corpus of text. Classification is a technique, well-researched in data mining and machine learning that may be used to automate the annotation process. Classification separates data into distinct classes characterized by some distinguishing features and rules relate class labels to these features. Automated classification has been used in a variety of domains including textual data such as e-mails, web pages, news articles; audio; images and video; medical data; or even annotated genes (Read, 2010). Each example is associated with an attribute vector, which represents data from its domain. Labels represent concepts from the problem domain such as subject categories, descriptive tags, genres, gene functions, and other forms of annotation. The training set is readily available in practical scenarios, usually in the form of human-annotation by a domain expert. A supervised classifier trains its model on these examples and continues the labeling task thereafter automatically. Single-label classification is the task of associating each example with a single class label. Classes may also overlap, in which case, the same data may belong to all of the many classes that overlap. In such instances, it becomes necessary to collect the details or features of all the classes that the data belongs to in order to perform a complete classification that is also accurate. When each example may be associated with multiple labels simultaneously, this is known as multi-label classification. In this paper, the terms class, label and tag are used interchangeably.

This paper presents the Classification-Based Text Annotation Workbench (CLAW), an annotation assistance tool that reduces the difficulty of the annotation process. The area of application for the tool is a large corpus (~1G) supplied by the U.S. Army Corps of Engineers with the objective of annotating the text using a classification taxonomy provided. The purpose of the annotation is to introduce common terminology in order to facilitate an understanding of the dominant themes within the corpus. The corpus consists of numerous reports, lessons learned, and best practices drawn from peace keeping and nation building operations. There are several technical challenges posed by this application domain. The document set is complex with respect to size, variety of formats and range of subject matter. The subject matter in these documents is extensive and includes reports on social and cultural institutions, physical infrastructure, education, agriculture, etc. The annotation taxonomy is large and unstructured with the flexibility of labels being applied orthogonally. Furthermore the annotation is required at the phrase level. Consequently, the search space for the label(s) is large and it becomes necessary

 ISBN: 978-1-61208-269-1

to adopt strategies that reduce the complexity of the classification process. The work reported in this paper is the development of a practical solution to this problem. We investigate the applicability of the Naïve Bayes classification technique to the corpus and compare to a more complex text classification technique, the MLkNN. It is found that the Naïve Bayes provides sufficient accuracy for automating the classification process for the target taxonomy, and text corpus. This evaluation is used to justify the incorporation of the algorithm into the automation workbench, CLAW.

The paper is organized as follows. Section II reviews published literature on text processing and multi-label classification in text. Section III describes the approach to the problem and the architecture of CLAW. Section IV describes the results and compares them with a benchmark algorithm. Section V concludes this paper and points to directions of future research.

II. RELATED WORK

In document classification a large number of attributes are used to characterize the document. The attributes of the examples to be classified are the words in the text phrases, and the number of different words can be quite large. McCallum and Nigam [8] clarify the two different first order probabilistic generative models that are used for text classification, both of which make the Naïve Bayes assumption. The first model is a multi-variate Bernoulli model, which is a Bayesian network with no dependencies between words and binary features. The second model is the multinomial model, which specifies that a document is represented by the set of word occurrences in the document. The probability of a document is a product of the probability of each of the words that occur.

Lauser et al. [6] propose an approach to automatically subject index full-text documents with multiple labels based on binary Support Vector Machines (SVMs). The authors incorporate multilingual background knowledge in the form of thesauri and ontologies in their text document representation. Godbole et al [2] present methods for enhancing and adapting discriminative classifiers for multi-labeled predictions. Their approach exploits the relationship between classes, by combining text features and the features indicating relationship between classes. They also propose enhancements to the margin of SVMs for building better models in the event of overlapping classes. In [3] the authors evaluate the preprocessing combination of feature reduction, feature subset selection, and term weighting is best suited to yield a document representation that optimizes the SVM classification of particular datasets. Ikonomakis et al. [4] describe the text classification process using the vector representation of documents, feature selection, and provide definitions of evaluation metrics.

Tsoumakas and Katakis [13] give a good introduction to multi-label classification using methods such as algorithm adaptation and problem transformation. The different techniques are compared and evaluated using metrics, after they are applied to classify some benchmarked data sets. Zhang et al. [17] present a multi-label lazy learning approach named MLkNN, which is derived from the traditional k-Nearest Neighbor (kNN) algorithm. Using experiments on three different multi-label learning problems, i.e., yeast gene functional analysis, natural scene classification and automatic web page categorization, the authors show that MLkNN achieves better performance when compared to some well-established multi-label learning algorithms.

Younes et al. [15] describe an adaptation of MLkNN that takes into account dependencies between labels (DMLkNN). The authors use a Bayesian version of kNN. Experiments on simulated and benchmarked datasets show the efficiency of this approach compared to other existing approaches. Tsoumakas et al. [11] describe a new enhancement on the multi-label algorithm called label powerset (LP) that considers each distinct combination of labels that exist in the training set as a different class value in a single-label classification task.

Previous work in work benches for text annotation includes the Koivunen [5] system. This work describes Annotea, a semantic web-based project. Metadata is generated in the form of objects such as web annotations, reply threads, bookmarks, topics etc. Users can easily create RDF metadata that may be queried, merged and mixed with other metadata. In Zeni et al. [16], a software tool (Biblio) is described for automatically generating a list of references and an annotated bibliography, given a collection of published research articles. Finlayson [1] describes the Story Workbench, a software tool that facilitates semantic annotation of text documents. The tool uses Natural Language Processing tools to make a best guess as to the annotation, presenting that guess to the human annotator for approval, correction, or elaboration. This is a semi-automatic process. Annotation is generalized into a "tagging" procedure with parts-of-speech tags as well as general tags for "tooltips" or "infotips" in a GUI.

The problem that we address in this research is unique to the domain in several respects- the need to annotate at an atomic level, i.e., the noun phrase and verb phrase level; the unstructured labeling taxonomy supplied to annotate text; and finally, the need to find a practical solution to automating a supplied corpus and taxonomy. The taxonomy gives rise to a very large labeling search space, which makes accurate classification of text difficult. The algorithms discussed previously are developed for a much smaller set of classifications, and are applicable at a grosser level than that required here (paragraph or document). The software tools in general with the exception of Annotea are not geared towards the annotation process.

III. APPROACH

The first phase of the process is to input previously classified text phrases to the Stanford Parts Of Speech (POS) Tagger [9]. The Text Preprocessor component performs pre-processing on POS Tagged data. Pre-processing includes steps such as the filtering of records that do not contain either noun phrases or verb phrases, and retaining only those features (words) that have appropriate parts-of-speech tags

 ISBN: 978-1-61208-269-1

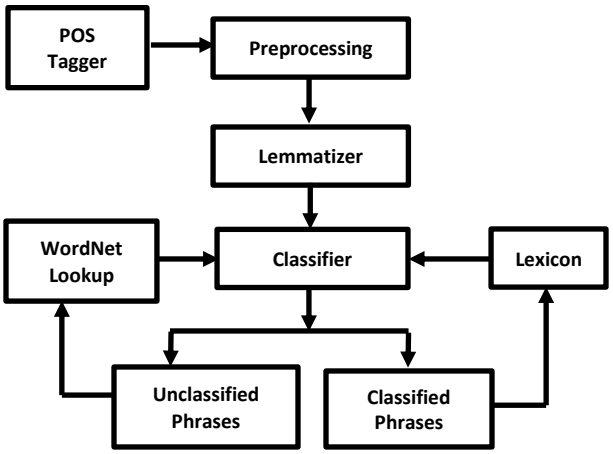

Figure 1. Process Flow Diagram for CLAW

for noun phrases and verb phrases. This component produces as output a delimited ASCII text file for next phase of lemmatizing. The lemmatizer uses WordNet® [14] database to extract synonyms or lemmas of input phrases to build an expanded input set for the next stage of classification. The Java API for WordNet Searching (JAWS, 2012) interface to WordNet is utilized in the lemmatizer. The lemmatized phrases are input into the Naïve Bayes classifier. Unclassified phrases are passed into a WordNet Lookup component to extract synonyms and returned to the Naïve Bayes Classifier for another attempt at classification. The lexicon is stored in a SQLite database [10] that stores the data. The process flow diagram for *CLAW* is shown in Figure 1.

Naïve Bayes is a standard algorithm for learning to classify text. Naïve Bayes classifiers are faster than other algorithms discussed in literature such as SVMs, since they learn a probabilistic generative model in just one pass of the training data even though they may sacrifice some classification accuracy. Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. If *B* represents the dependent event and *A* represents the prior event, Bayes' theorem is stated as follows.

Prob (B given A) = Prob (A and B)/Prob (A) (1)

i.e., the probability of classification, *B*, given a phrase *A*, is the probability of *A* classified as *B* in the training data, divided by the probability of *A* occurring in the training data. The Training data consists of text phrases classified /annotated by human annotator, which is input into the *CLAW* system as discussed previously.

The labels or tags for this domain are verb phrase tags (*task*, *state*, *role*, and *other*) and noun phrase tags. Noun phrase tags are further subdivided into Level 1 (*L1*), and Level 2 (*L2*) tags shown in TABLE 1 and TABLE2 respectively. $L2^{(i)}$ denotes the i^{th} most frequent *L2* tag.

As previously noted, we have a very large search space of labels, which could be used to tag noun phrases, since the *L1* and *L2* tags may be orthogonally applied multiple number of times (depending on the context that we are annotating), with the only condition that every NP has at least one L1 tag. In the case of the verb phrases the process is simpler since there are only four possible tags to be applied. A simplification strategy is used to reduce the search space in the case of noun phrases. We calculate the frequency count of the labels to isolate and replace L1 labels with the most frequent ten classes and L2 labels in a particular instance with $L2^{(i)}$ where *i=1..4*, in the data set (if present in that string) to create particular models to be learnt by the Bayesian classifier. This selection strategy transforms the multi-label learning problem to a single label learning problem. Thus, based on frequency counts of labels, different models are learnt by the Naïve Bayes classifier. The instances that correspond to the top ten Level 1 Tags alone are also tested separately, and a model corresponding to L1 tags is learnt.

TABLE 1. LEVEL 1 TAGS

L1 Tags
entity/agents
entity/events
entity/info
entity/institutions
entity/materials
entity/organizations
entity/physical_behaviors
entity/physical_infrastructures
entity/places
entity/services
entity/social_behaviors
entity/social_infrastructures
entity/technical_capabilities
entity/time

There is only one learning model for Verb Phrases (VP) test data since there are only four possible VP tags. Rare classes with percentage frequency less than 1% in the data set are matched using simple string matching. In the final step the individually predicted labels are composed together. This approach avoids the need to learn a combinatorial number of classes directly by simplifying the problem, however restoring the complexity of the labeling by composition in the final step.

IV. RESULTS AND COMPARISONS

Prediction-based metrics and Ranking-based metrics are standard measures used to evaluate performance of text classification. Ranking based metrics (TABLE 3) evaluate the label ranking quality depending on the ranking or scoring function. Hamming Loss is used as the basis for Ranking Function in our classification. Lower Hamming Loss implies

TABLE 2. LEVEL 2 TAGS

L2 Tags		
administrative	global	private
agreement	governance	psychological
agriculture	health	public
authority	Illicit	public opinion
civilian	indigenous	purpose
communication	labor	relationships
conditions	language	relief
conflicting	liaison	religion
contractor	licit	requirements
criminal	local_governance	return
definition	military	routine
dislocated	negotiation	security
economy	oversight	social
education	perspective	transition
environment	pets	transportation
extremism	political	utilities
food		

TABLE 3. DEFINITIONS OF RANKING-BASED METRICS

Hamming Loss = # of misclassified records in Test Data/(# of records of Test Data* Size of Label Set)

Subset Accuracy = (No. of Exact Matched records with True Predicted Classes)/(No. of Test Records)

Average Precision: Average fraction of labels ranked above a particular label (Best value is 1)

Coverage: Average # of steps needed to move down the ranked label list in order to cover all the labels assigned to a test instance. Smaller value of this metric is desirable.

One-Error: Calculates how many times the top-ranked label i.e., the label with highest ranking score, is not in the set of labels for the appropriate instance. Smaller value of this metric is desirable.

Ranking loss: Average fraction of label pairs that are reversely ordered i.e., number of times irrelevant labels are ranked higher than relevant labels for an instance. This does not happen in our case. Smaller value of this metric is desirable.

higher rank for a label. The most relevant label has highest rank of 1.

The MULAN [12] package has implemented the MLkNN (Multi-label lazy learning k-NN) and provides a good comparison metric for the approach taken in this paper. The software generates classification metrics automatically when supplied with train-test data.

TABLE 4. RANKING METRICS FOR NOUN AND VERB PHRASES.

Classification Algorithm	Ranking Metrics	L1 Tags only for Noun Phrases	Verb Phrase Tags only
Naïve Bayes	Hamming Loss	0.03	0.05
	Subset Accuracy	0.7	0.8
	Average Precision	0.78	0.5
	Coverage	2	1
	One Error	0.0023	0.005
	Ranking Loss	0	0
MULAN MLkNN	Hamming Loss	0.1	0.25
	Subset Accuracy	< 0.01	< 0.01
	Avg. Precision	0.53	0.5454
	Coverage	2.1336	1.255
	One Error	0.6889	0.76
	Ranking Loss	0.2371	0.4183

For the experiments we used a training set of approximately 1000 classified phrases, with 200 phrases as the test set. In the first part of the experiments, we consider L1 Noun Phrase (NP) tags alone and in a separate experiment Verb Phrases alone. The results of this experiment are shown in TABLE 4.

From TABLE 4 it may be seen that, while the MLkNN has marginally better Average Precision for Verb Phase test data, all other evaluation metrics are considerably better for the Naïve Bayes classifier used in this work. TABLE 5 establishes this for a series of $L1L2^{(i)}$ models tested. It should be noted that these conclusions are specific to the annotation taxonomy and the domain of application used here.

A. Integrating the Learning Component into CLAW

The learning component described in the previous section is integrated into the software tool, to develop the Classification-based Text Annotation Workbench (CLAW). CLAW uses the learning component, and the corpus database to provide hints for annotation to the user. SQLite is chosen as the database for the work bench because of its light weight footprint. The workbench is developed using the .NET framework, with the learning components constructed as Java services.

TABLE 5. RANKING METRICS FOR L1L2$^{(l)}$ MODELS

	Ranking Metrics	L1L2$^{(1)}$	L1L2$^{(2)}$	L1L2$^{(3)}$	L1L2$^{(4)}$
Naïve Bayes	Hamming Loss	0.026	0.027	0.0236	0.05
	Subset Accuracy	0.59	0.65	0.72	0.55
	Average Precision	0.91	0.92	0.75	0.89
	Coverage	4	3	3	2
	One Error	0.0043	0.0055	0.0024	0
	Ranking Loss	0	0	0	0
MULAN MLkNN	Hamming Loss	0.0625	0.0769	0.1	0.1111
	Subset Accuracy	< 0.01	<0.01	<0.01	<0.01
	Average Precision	0.4448	0.4863	0.4883	0.5208
	Coverage	4.2735	3.1209	2.4633	2.2038
	One Error	0.6795	0.7005	0.7267	0.6943
	Ranking Loss	0.2849	0.2601	0.2737	0.2755

V. CONCLUSIONS AND FUTURE WORK

This work approached the problem of text annotation in two phases. In the first phase we annotated the text manually based on a taxonomy provided, and in the second phase we used the annotations to develop algorithms to perform automated annotation. There were several challenges to developing the automated annotation component- the text corpus is wide ranging encompassing a broad range of topics; the taxonomy is unstructured and large; and finally the annotations may be applied combinatorially. We introduced a multi-modal approach that reduces the combinatorial nature of the problem, making the automation feasible. The resulting model was validated against a benchmark algorithm for text classification. Finally, we integrated the approach into a novel system, Classification-based Text Annotation Workbench (CLAW) that facilitates both manual annotation and incorporates supervised annotation (based on automated classification) to reduce the complexity of annotating text. CLAW provides a flexible environment with the ability to change the taxonomy depending on the domain of application. The CLAW tool also has the capability to ensure a consistent basis to annotation, since it generates annotations based on the deterministic learning component. Processing times within the tool are fairly reasonable, but this could change as the repository gets larger.

While the CLAW tool provides a good infrastructure for annotation of text, there are several possible enhancements to the tool that can improve the quality and repeatability of the annotation process.

- Automated generation of learning models. The quality of the automated annotations provided by CLAW depends on the learning model used (refer the CLAW user guide). While the learning model is generated manually, the corpus database could provide a mechanism for the automated creation of models. Multiple models may be created and used within CLAW. The use of multiple models would improve the recall of the classification process. A further possibility might be the application of active learning into this component.
- Qualitative evaluation of model outputs. In the current version of CLAW every model output is treated identically. A quantitative approach that evaluates each model independently, and ranks the outputs could provide the user with greater confidence in the results. This would also help the user select annotation suggestions based on quantitative measures.
- Traceability of CLAW suggestions. In the current version of CLAW, the annotation suggestions are provided to the user, however there is no mechanism that informs the user as to how the suggestion was being made. Providing traceability of the model outputs would improve confidence in the system.

ACKNOWLEDGMENT

This research is funded in part by the Construction Engineering Research Laboratory, Engineering Research and Development Center (ERDC-CERL) under Contract No: W913T-12-C-007, ARL under Grant No: W911NF-12-2-0067 and ARO under Grant Number W911NF-11-1-0168. Any opinions, findings, conclusions or recommendations expressed here are those of the author(s) and do not necessarily reflect the views of the sponsor.

REFERENCES

[1] M.A. Finlayson. "The Story Workbench: An Extensible Semi-Automatic Text Annotation Tool," *AAAI Technical Report* WS-11-18. Copyright 2011.

[2] S. Godbole and S.Sarawagi. "Discriminative Methods for Multi-labeled Classification," *PAKDD 2004*, LNAI 3056, pp. 22–30.

[3] T. Gonçalves and P. Quaresma. "Evaluating preprocessing techniques in a text classification problem," *XXV Congresso da Sociedade Brasileira de Computação*, July 2005, pp. 841-850

[4] M. Ikonomakis, S. Kotsiantis, and V. Tampakas. "Text Classification: A Recent Overview,". ICCOMP'05. *Proceedings of the 9th WSEAS International Conference on Computers.* Article no: 125. In ESWC 2005, UserSWeb workshop

[5] M.R. Koivunen. "Annotea and semantic web supported collaboration," Downloaded from http://ceur-ws.org/Vol-137/01_koivunen_final.pdf, November 2012.

[6] B., Lauser, A. Hotho, T. Koch, and I.T. Solvberg, (Eds.). "Automatic Multi-label Subject Indexing in a Multilingual Environment," *ECDL 2003*, LNCS 2769, pp. 140–151.

[7] Machine Learning Group at University of Waikato, 2012. http://www.cs.waikato.ac.nz/ml/weka/

[8] A. McCallum, and K. A. Nigam. "Comparison of Event Models for Naive Bayes Text Classification," In *AAAI-98 Workshop on 'Learning for Text Categorization'*, 1998, 41-48, WS-98-05

[9] POSTagger**,** http://nlp.stanford.edu/software/tagger.shtml

[10] SQLite. http://www.sqlite.org/

[11] G. Tsoumakas, I. Katakis, and I. Vlahavas. "Random k-Labelsets for Multilabel classification," *IEEE Transactions on Knowledge and Data Engineering*, July 2011, Vol 23, No. 7.

[12] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas. "MULAN: A Java library for Multi-Label Learning," *Journal of Machine Learning Research*, 12, 2411-2414.

[13] G. Tsoumakas and I. Katakis. "Multi-Label Classification: An Overview," *International Journal of Data Warehousing and Mining*, 3(3), July-Sept 2007, p.1-13.

[14] WordNet. http://wordnet.princeton.edu/

[15] Z. Younes, F. Abdallah, T. Denoeux, and H. Snoussi. "A Dependent Multilabel Classification Method Derived from the k-Nearest Neighbor Rule," In *Proceedings of EURASIP J. Adv. Sig. Proc*. 2011.

[16] N. Zeni, N. Kiyavitskaya, L. Mich, J. Mylopoulos, and J. R. Cordy. "A Lightweight Approach to Semantic Annotation of Research Papers," In *Proceedings of NLDB*. 2007, 61-72.

[17] M.L. Zhang and Z. H. Zhou. "Ml-knn: A lazy learning approach to multi-label learning," *Pattern Recognition*, 40, 2038-2048.

 ISBN: 978-1-61208-269-1