Corpus Based Classification of Text in Australian Contracts

Michael Curtotti
Australian National University
Canberra, Australia
michael.curtotti@anu.edu.au

Eric Mccreath
Australian National University
Canberra, Australia
eric.mccreath@anu.edu.au

Abstract

Written contracts are a fundamental framework for commercial and cooperative transactions and relationships. Limited research has been published on the application of machine learning and natural language processing (NLP) to contracts. In this paper we report the classification of components of contract texts using machine learning and hand-coded methods. Authors studying a range of domains have found that combining machine learning and rule based approaches increases accuracy of machine learning. We find similar results which suggest the utility of considering leveraging hand coded classification rules for machine learning. We attained an average accuracy of 83.48% on a multi-class labelling task on 20 contracts combining machine learning and rule based approaches, increasing performance over machine learning alone.

1 Introduction

Contracts govern economic transactions and commercial or organisational relationships from trivial purchases to major national infrastructure projects. Any large organisation (whether private or public) must unavoidably invest significant resources in developing and concluding contracts, as the contracts it enters define its legal relations with organisations and individuals with which it interacts. As one author has noted, contracts are an integral part of any business enterprise and it is "difficult to overstate their importance to the business world" (Khoury and Yamouni, 2007, p16). In addition, drafting contracts represents a major economic activity for the legal industry.

Ambiguity is an unresolved problem in natural language processing. It is also an important problem affecting the drafting and operation of contracts. The costs of ambiguity in a contract can be enormous. The presence of ambiguity increases the risk of litigation, project failure and loss of commercial relationships. Instances of litigated ambiguity are used in drafting manuals as instructional tales for the unwary drafter (Aitken and Butt, 2004, p37).

Our future research goal is to investigate approaches that will help identify such ambiguity, however, a precursor for addressing the problem of ambiguity is accurate labelling or classification of the constituent parts of contract documents.[1] In this paper we report work on classifying 'lines' in contracts into 32 classes. We compare three different classification approaches in order to determine which approach may provide the best performance with respect to this task: supervised machine learning; a hand-crafted rule based tagger; or combining hand-crafted rules based tagging with machine learning.

2 Background

Natural language processing involves applying a pipeline of transformations to text. A typical first step is to segment the text into its sentences, after which further processing is applied (Jurafsky and Martin, 2009, p 69). This works well in usual English prose, but is not necessarily well suited to contracts because of the heavy use of layout to embody structure and semantics. In a contract, a sentence may occur within a single paragraph of text, or may be spread over several line breaks due to the use of sub-paragraphing as an aid to comprehension. Such sub-paragraphs may embody disjunctive or conjunctive conditions associated with

[1] This task can also be thought of as a document segmentation task, although we here consider it as a classification exercise.

a rule. A 'clause'[2] is the typical structural unit which embodies a legal rule or set of rules. Sometimes clauses are organised into sub-clauses and clauses and sub-clauses may be numbered in hierarchical fashion (e.g. 1, 1.1, (a), (i), (A))(Aitken and Butt, 2004, pp 23, 27). Line breaks often occur immediately before data such as headings, clauses, sub-clauses, party names, dates or execution blocks. Being able to label a line as to its character, key content or function thus enriches available information for later NLP. The line is thus a valuable starting point for applying NLP.

3 Related Work

3.1 Classification and Segmentation of Contracts and Legislative Documents

There would appear to be little published work on the application of natural language processing (NLP) or machine learning specifically for the classification of text in legal contracts. An exception to this is (Indukuri and Krishna, 2010) who apply a support vector machine over feature vectors extracted from contract clauses. These feature vectors are comprised of n-grams up to size n = 4. They classify contract sentences as clauses or non-clauses and then sub-classify clauses as concerned with payment terms or not. Indukuri and Krishna note the lack of published work on classification of sentences in contract documents for workflow and monitoring purposes.

While also limited, there has been some work on segmentation or classification of legislative corpora using machine learning. (Mencía, 2009) reports 100% precision and recall in machine learning on classification of legislation into articles, sections and parts (although to a task for which he is able to craft an equally accurate regular expression based segmenter). (Bacci et al., 2009) automatically classify plain text legislative documents for the purpose of XML mark up. (Francesconi, 2006), on whose work Bacci et al. build, describes four separate modules for dealing with plain text legacy content which they are seeking to convert to XML tagged text. (Hasan et al., 2008) carry out segmentation of Spanish legislative bulletins using the table of contents as an aid to segmentation. This work, in the legislative field, provides a parallel application domain for the work undertaken here in respect of contracts, although it is generally true that legislation will tend to conform to far tighter stylistic rules, because of the central control of legislative drafting.

3.2 Combining Machine Learning and Hand Coded Approaches

A number of researchers have combined hand-coded approaches and machine learning to produce better overall results.

(Kipp, 2006) augments handcrafted rules with machine learnt rules for gesture generation.

(McCreath and Kay, 2003) used such a combined approach to improve performance on categorizing emails.

(Park and Zhang, 2003) address the problem of text chunking for the Korean language, combining hand-crafted rules with a memory based machine learning method. A rule based method is used to determine whether an instance may constitute an exceptional case, with machine learning used to correct classifications assigned by the rule based method. They report up to a 2.83% improvement in F-scores over memory-based learning alone.

(Rochery et al., 2002) report the use of hand-crafted rules to enhance the accuracy of machine learning utilising boosting. They apply the method to a multi-labelling classification problem where they are seeking to classify spoken utterances. They attribute increased accuracy to the fact that the rules they develop are not represented in their data at all or do not appear sufficiently to have statistical impact during the training process. They note that the combination has the greatest effect when there are less then 100 examples, with the benefit of the combination decreasing as the number of training examples increases. The method therefore has direct applicability to reducing the data requirement for machine learning (a costly aspect of supervised machine learning).

(Takahashi et al., 2005) utilise a combination of SVM based learning and hand-crafted rules for the task of classifying occupations from social surveys. They effect a combination by using rule derived labels as features for the machine learner. Again the work occurs in a multi-class labelling context (with around 200 occupation codes). They find that SVM approaches are superior to rule-based approaches alone but that combining rule-based and SVM learning produces the best results. They also examine the effect of the number of training examples, noting that the differential ben-

[2]The term 'clause' here refers to a legal rule or rules appearing in a contract, rather than the linguistic entity.

efits of combining methods reduces with increasing data.

In our study we use class labels assigned by a hand-coded tagger as additional features for machine learning to obtain an increase in accuracy in the domain of text classification.

The work reported above, and our own paper, come from diverse domains and illustrate the application of combined rule/ML approaches using a variety of machine learning algorithms.

This suggests the wide utility of using hand-coded rules to improve accuracy of machine learning. Symetrically we may say that machine learning may potentially be used to improve the accuracy of hand-coded rules. Here we essentially follow a hybrid model of development that examines each method separately as well as in combination.

4 Data

4.1 Australian Contract Corpus

The data which forms the basis of this study is a corpus of 256 contracts ('the Australian Contract Corpus') which has been compiled from the web using the Google Australia web search: `http://www.google.com.au`.[3] The corpus is constituted of 1043364 tokens (words and punctuation), 42910 sentences, and a vocabulary of 14217. The lexical diversity of this corpus is 73.39.[4] As far as we are aware there is no published work based on a corpus of Australian contract language, however space does not permit a fuller discussion of the design and analysis of the corpus.

4.2 Data Representation

For the purpose of this study we randomly selected 30 contracts from the Australian Contract Corpus to produce data sets for application of machine learning. The 30 contracts were divided into three sets: a training set (Set A)[5], a second set (Set B)[6] (primarily used as a test set, although sometimes combined with Set A to form a larger training set) and an additional test set (Set C)[7], each of 10 contracts. Set C was added to increase both available data and the number of contracts on which testing could be carried out.

To simplify processing all contracts were converted into text files as a preprocessing step.[8] Also we removed material that typically does not appear in contracts (primarily guideline notes for drafters).

For purposes of our study each 'line' or 'paragraph' in a contract constituted a data point. Sets A, B and C provided 1825, 2157 and 2231 lines/data points, respectively.

To classify our data for machine learning we first employed our hand-coded tagger to produce a labelled data set constituted of a line and primary and secondary labels according to our classification system (see below). The classifications applied by the hand-coded tagger were then manually corrected to remove any labelling errors. Both for the reason that hand tagging data from scratch is more error prone than correcting previously machine tagged data, and because of the high labour in manual tagging, machine assisted tagging is a standard method used in corpus creation (Biber et al., 1998, p 262), ((ed) Christopher S. Butler, 1992, p 131).

4.3 Feature Selection

Features were then extracted from our labelled lines using a hand-coded feature extractor to extract up to 40 features to be used for machine learning, with the primary class label earlier applied to each line serving as the target for machine learning.[9] Features for machine learning were selected based on assumed relevance to the intended classification.[10] While a bag of word representa-

[3]The search phrase we employed was: "clause party agreement" , limiting the search to "pages from Australia" and the filetype to ".doc". The collection of the corpus was undertaken in the period 6 - 24 December 2009. Each document was visually inspected by one of the authors to verify that it was an Australian contract and documents were added to the corpus in order of their appearance in Google search results. We have made available the list of URLs of the documents that make up the corpus at: `http://cs.anu.edu.au/people/Michael.Curtotti/`. We also make code and data referred to in this paper available at the same location.

[4]Lexical diversity refers to the ratio of the total number of tokens in the corpus to the total number of types of tokens i.e. tokens/vocabulary. The statistics and information provided here were extracted using the NLTK. (Bird et al., 2009).

[5]Contracts numbered in our corpus: 55, 74, 77, 91, 94, 144, 174, 185, 208 and 213

[6]Contracts 11, 12, 164, 193, 196, 199, 249, 59, 64, 9

[7]Contracts 200, 254, 175, 180, 120, 102, 207, 127, 75 and 79

[8]Notably, this step does have the downside of discarding valuable style and layout information found in the word document format.

[9]In some instances the assignment of class is not entirely disjunctive and implies a priority ordering of potential classifications. For instance a clause may contain address details, and be classified as such rather than as a clause. The feature extractor removed secondary labels.

[10]The full list of features is as follows: 1. relative position of a line within the contract; 2. line end punctuation; 3. num-

tion with automatic feature extraction was trialed, this was found to result in lower accuracy than a manually selected (but small) number of features. Traditionally, 'feature engineering' or 'selection' in text classification has focussed on 'bag of words' or proximate features such as n-grams or noun phrases (Scott and Matwin, 1999). Our selection of features might be regarded as 'hybrid' in that they implicitly encode domain knowledge and 'rules'. For example, we expect a definition line to contain the widely used word 'means' as in 'x means y'. We would expect clause headings to display a different pattern of parts of speech occurrence, as compared to clauses themselves. We also consider features such as the relationship between the particular line/datapoint and its neighbouring lines (e.g. is the preceding line long or short), or the document as a whole (e.g. relative placement of the line within the document).

Features used in the related work carried out by others on contracts and legislation include: n-grams (Indukuri), bag of word together with capitalisation and character patterns (Mencia), the use of sequential word and token data as a HMM to predict state where state is the classification of the portion of text (Bacci/Francesconi) and the use of word terms identifying titles and lists of indexed terms (Hasan).

4.4 Classes

We adopted 32 classes for our classification scheme. These classes represented significant structural elements (such as clause headings or content lines) or sometimes key contract meta data (for instance the parties, the date on which an agreement is made, email addresses, ABNs etc).[11] Of particular interest to us, was an ability to accurately identify clausematter (the primary location of legal rules of a contract) and clause headings (which in some contracts effectively mark boundaries between major rule sets).

5 Experimental Evaluation

Analysis of our data sets was carried out using two major methods: analysis using the training Set A measuring peformance on test Set B (i.e. testing for accuracy of classification of lines in Set B); and analysis using training Set A where each contract in test Sets B and C was used individually as a test set (testing for accuracy of classification of lines of each individual contract). The former testing approach is more widely used but the latter is more consistent with our intended end application: that is, developing an ability to accurately classify lines in a previously unseen contract as an individual document.

Set A consisted of 1825 lines of data, Set B consisted of 2157 lines of data and Set C consisted of 2231 lines of data.

5.1 Tools

The following tools were used to carry out machine learning tasks reported in this paper:

1. the *python* programming language was used to develop a hand coded line tagger, a feature extractor and code for evaluation of the performance of the hand coded tagger;

2. the Natural Language Toolkit (NLTK)(Bird et al., 2009) was used in corpus development and in the hand coded tagger for application of parts of speech tags; and

3. the Weka data mining software(Hall et al., 2009) was used to carry out machine learning.

bering at the beginning of the line; 4. line length in characters; 5. line length in tokens; 6. number of nouns; 7. number of adjectives; 8. number of verbs; 9. number of prepositions; 10. number of coordinating conjunctions and cardinal numbers; 11. number of modal verbs; 12. number of personal pronouns, possessives; 13. number of adjectives; 14. whether must occurs; 15. whether may occurs; 16. whether shall occurs; 17. whether means occurs;18. the position of the word means in the line; 19. whether the word 'include' occurs; 20. the position of the word 'includes' in the line; 21. whether the phase 'has the same meaning' occurs; 22. whether the line begins with a capital letter followed by a stop or a space; 23. do the letters 'abn' or 'acn' appear; 24. does the word address appear near the beginning of the line; 25. does the word 'contact' appear near the beginning of the line; 26. does the word 'email' appear near the beginning of the line; 27. does the word 'fax' appear near the beginning of the line; 28. does the word 'note' appear at the beginning of the line; 29. does the word 'phone' appear at the beginning of the line; 30. do the terms 'web' or 'www' appear near the beginning of the line; 31-36. token lengths of 3 lines before and after the data line; 37. the tag applied by the hand coded tagger to the data line; 38. whether the contract from which the line comes has clause headings; 39. whether the contract from which the line comes has clause sub-headings; 40. whether the contract has a schedule.

[11] The following class labels were applied: CLAUSEMATTER, CLAUSEHEAD, DEF, CONTENTSHEAD, BLANK, DROSS, AND, BETWEEN, DATEMADELINE, PARTIESHEAD, PARTYLINE, RECITALHEAD, RECITALLINE, HEAD, CONTENTLINE, PRELIM, OPERATIVEPROVISIONLINE, NUMBEREDLINE, SCHEDULEHEAD, SCHEDULEITEM, EXECUTIONBLOCK, NOTE, ABNLINE, ADDRESSLINE, EMAILLINE, WEBLINE, FAXLINE, PH-LINE, REF-LINE, SCHOTHERMATTER, CONTACTOFFICER, TITLELINE.

5.2 Hand-Coded Tagger

Our hand-coded tagger follows a pipeline methodology. A contract is separated into a list of lines. In the first phase, each line is passed through a series of "if-then-else" routines (essentially utilising regular expressions) to assign primary and secondary labels to the line. This produces a contract with enhanced information content, in the form of a tuple of labels and line text. This output is then fed through a series of methods, progressively bootstrapping improvements to labelling, based on the information available when the method is applied. For instance content lines will be identified, these are then used in later methods to seek to identify clause headings. In addition, the tagger extracts contract meta data such as: a list of definitions, a list of clause headings, where the beginning of operative provisions occurs, where the schedule begins. Such features may or may not occur in a contract. In addition the tagger also provided a method for hand correcting tagging which was used to create our data set for supervised machine learning purposes.

Development of the tagger relied heavily on domain knowledge to identify logical tests for label application. For instance the occurrence of numbering at the beginning a line was used to identify clause headings, or the presence of a common text pattern which occur in definitions i.e. (the pattern *["word[1-3]] means [...]"* was used as a criterion for identifying a line as a definition).

	$\overline{x}$	s
On Set B Contracts	86.27%	NA
Contract x Contract	82.84%	10.85%

Table 1: Accuracy of Line Tagger.

The first row of Table 1 reports accuracy of the hand coded tagger when applied to Set B as the test set (the % accuracy of tagging of individual lines in Set B is reported). The second row of Table 1 reports average accuracy with which the lines in each of 20 contracts in Sets B and C are tagged (standard deviation is also reported). The high variance of the results at a contract by contract level suggests a need for future work to identify if issues such as contract 'type' may account for the differences (the lowest level of accuracy was around 50% and the highest around 95%).

Table 2 provides measures of precision, recall and F-measure attained by the hand coded tagger on Set B for key data items: clause headings, clauses themselves and definitions.[12] These results on key measures are sufficient for our initial purposes and will allow us to focus subsequent work on addressing ambiguity in the substantive rules found in a contract.

	Precision	Recall	F-Measure
CL.HEAD	0.98	0.83	0.89
CL.MATTER	0.91	0.91	0.91
DEF	1.00	0.66	0.80

Table 2: Performance results for Hand-Coded Line Tagger

5.3 Supervised Machine Learning

Column 1 of Table 3 reports the accuracy of various supervised machine learning algorithms with Set A as training set and Set B as test set. Of the algorithms applied, Random Forest (implemented using 100 trees and 30 random features) performed best. No ML method performed better than the hand coded tagger tested on the same test set (see row 1 of Table 1).

5.4 Machine Learning vs Hand-coded methods

Given our ultimate focus on developing applied tools, we are primarily concerned to identify the best methods to maximise the accuracy of our classification task. Whether this proves to be machine learning methods or hand-coded methods or a combination of the two is immaterial from this viewpoint (other than in regard of development costs).

Also it is useful to consider whether methods and insights from one method may assist in the other. For instance, hand-coded rules that successfully labelled lines directly suggested 'features' to be extracted for machine learning purposes. Conversely the identification of 'features' for machine learning can be regarded as a programming abstraction where the algorithm is 'under the hood' but human intervention is required in the form of feature selection for 'learning' to occur (implicitly encoding of rules in the feature set). In deriving our results we used a confusion matrix as a tool to assist in identifying areas where the hand-

[12] Precision is $\frac{TP}{TP+FP}$, Recall is $\frac{TP}{TP+FN}$, and F-Measure is $\frac{2PR}{P+R}$, where TP is true positives, FP is false positives, FN is false negatives, P is precision, and R is recall.

coded tagger required improvement and as a common baseline for performance. In certain forms of machine learning, the resulting machine learning model can be represented as code in "if, then, else" form (as in the output of a decision list or decision tree learner). We may thus conceive of machine learning and hand coding of rules as different 'programming paradigms' - a perspective that encourages us to apply insights across the boundary between them.

Further, we wished to explore the relative accuracy of machine learning vs hand coding for this particular task. Such an exploration is valuable from a cost benefit point of view. Supervised machine learning (where we have focused our attention) requires the production of human annotated data (which is costly to produce) as well as feature testing and refinement. Also machine learning may be computationally intensive. Production of an accurate hand-coded tagger requires expert knowledge and considerable testing and code refinement in order to achieve a high level of accuracy (which again is resource intensive). We did not undertake quantitative data collection in respect of time employed in the two methods, particularly as we followed an iterative development model that switched between rule based and machine learning focussed work and as we were searching for optimal outcomes. Nonetheless we anecdotally report that each method presented significant demands in terms of development time.

In the results reported above we note that no machine learning approach alone exceeded the performance of the hand coded tagger when considering performance on test Set B as a whole. This is the result also found at contract by contract level (see below Table 4).

5.5 Leveraging Hand-Coded Labels for Machine Learning

As noted above we also wished to explore whether utilising the output from the hand coded tagger as input features for machine learning might improve the accuracy of machine learning alone. A converse question that could be framed is whether the rule based tagger's performance could be improved by combination with machine learning.

We carried out two trials that bear on these questions:

1. machine learning alone and machine learning enhanced with rule based input on an entire test set (Set B); and
2. machine learning alone, rule alone and machine learning enhanced with rule based input at contract by contract level.

ML Algorithm	ML Alone	ML + Rule
Naive Bayes	82.38%	84.75%
SMO (SVM)	82.80%	85.40%
Cl. via Regression	83.91%	87.07%
Decision Tree	82.01%	87.02%
Bagging	83.54%	87.76%
Majority Vote	84.42%	87.11%
Random Forest	85.12%	86.69%

Table 3: Comparative Performance ML alone and ML + Rule on train and test set

Column 2 of Table 3 shows the improvement in performance of various machine learning methods when tags applied by the hand coded tagger were provided as input features. All methods show improvement with addition of the rule derived feature on Set B as a whole.[13]

We report below a comparison of accuracy of machine learning, rule based tagging and a combination of both in respect of Decision Trees. Decision trees offer the advantage that the output is easily interpretable as a rule set for tagging.

Method	$\overline{x}$	s
ML Alone	79.12%	10.71%
Rule Alone	82.84%	10.85%
ML + Rule	83.48%	9.83%

Table 4: Tagging Accuracy Mean and Standard Deviation - Contract by Contract

Although the results reported in Table 4 are different (as different contracts are included in the tested contracts)[14] average accuracy continued to show a differential between machine learning alone, and machine learning leveraged with hand rules, as described above. On this combination ML + Rule also outperformed the rule based tagger alone, but by a very narrow margin. In 17 of 20 cases ML + Rule improved or did not worsen performance of ML alone. In 15 of 20 cases, ML +

[13]The training set for the results reported in Table 3 was Set A. The test set was Set B.

[14]The training set was Set A. Sets B and C provided 20 individual contracts (effectively 20 test sets) for testing.

rule was no worse or outperformed the hand coded tagger alone. Notably variation is quite high as compared to the mean. Testing for statistical significance we can reject a null hypothesis that the mean accuracy for ML + Rule is the same as for ML alone. We cannot reject a null hypothesis that ML + Rule is the same as Rule alone.[15] We note also that this investigation was only carried out in respect of the decision tree learner.

5.5.1 Is it more effective to increase data or use rule based features?

To further explore the question of how much effort might be saved by combining rule based approaches with machine learning alone we tested the effect of adding random noise to the feature set provided by the hand coded tagger. We found that we had to randomly flip over 45% of these tags before they ceased to impact positively on classification accuracy. This suggests that even poorly prepared hand coded rules can improve performance.[16]

We also tested the effect of incrementally increasing the amount of available training data to explore any decrease in differential in performance. Testing using Set C as the test set and taking training data from Sets A and B, we progressively increased training data by increments of 500 instances of data. Differential performance decreases progressively, starting around 5% and reducing to 2.5% as more training data is made available.[17] (See Table 5.) Extrapolating from these figures we would expect at a minimum that training data would have to be doubled to remove differential accuracy. Relatively little effort in developing rule based features may substitute for considerable work in creating additional supervised data.

[15]We gratefully acknowledge the assistance of Bob Forrester of the ANU Statistical Consulting Unit in deriving this result, undertaken utilising a generalised linear model to undertake model fitting adjusting for different file (contract) types. Predictions of estimated mean proportions were: ML Alone 0.7760 (s.e. 0.007181), Rule alone 0.8106 (s.e. 0.006775), ML+ Rule 0.8181 (s.e. 0.006677).

[16]Using a Decision Tree, ML + Rule accuracy on test Set B with training Set A, was 87.0% with zero noise, 86.9% with 5% noise, 84.7% with 10% noise, 85.2% with 20% noise, 84.4% with 30% noise, 83.9% with 40% noise, 82.8% with 45% noise, 82.4% with 50% noise. Decision Tree accuracy on ML alone on this training and test set was 82.0%.

[17]The data has a negative correlation of -0.70

Data	0.5	1K	1.5	2K	2.5	3K	3.5	4K
Diff.	5.1	3.4	5	4.2	3.3	3.1	3.9	2.6

Table 5: Difference in % accuracy of ML alone and ML with rule based feature with increasing training data.

5.5.2 Decision tree output

An examination of the decision tree produced after adding the output of the hand-coded tagger to machine learning is of interest.

```
LTlabel = #CLAUSEMATTER#
| tknLngth <= 4
|| nounNum <= 1: #CLAUSEMATTER# (6.0/2.0)
|| nounNum > 1
||| linePos. <= 0.383929: #PRELIM# (2.0)
||| linePos. > 0.383929: #CLAUSEHEAD# (2.0)
| tknLngth > 4: #CLAUSEMATTER# (508.0/7.0)
: :
: :
LTlabel = #PH-LINE#
| prepNum <= 1: #PH-LINE# (5.0)
| prepNum > 1: #CLAUSEMATTER# (2.0)
LTlabel = #EMAILLINE#: #EMAILLINE# (3.0)
LTlabel = #ABNLINE#
| preLine2 <= 0: #TITLELINE# (2.0/1.0)
| preLine2 > 0: #ABNLINE# (3.0)
```

Figure 1: Part of the decision tree learnt when the labels from the hand-coded are provided as an attribute.

Using only the machine learning feature set, the size of the pruned tree is 199 with 108 leaves. After addition of labels provided by our hand-coded tagger the size of the decision tree is reduced to 105 with 69 leaves. Figure 1 shows part of this decision tree and as illustrated, the decision tree generally begins with the label assigned by the hand-coded tagger 'LTlabel' and amends the label (where necessary) utilizing other features available to the machine learner. Some labels (e.g. EMAILLINE) were simply adopted without change. In the case of the CLAUSEMATTER label, the machine learner used the number of tokens in the line ('tknLngth'), number of nouns in the line ('nounNum') and the relative position of the line in the contract ('linePos.') to re-assign the class of the line. The significance of the use of labels assigned by the hand-coded tagger is that the decision-tree identifies that attribute as providing the most significant information gain (that is reduction in entropy) and therefore partitions on that feature (Callan, 2003, p245). This remains true in this example irrespective of the class to which

the data is being assigned.[18] It is obvious that the best feature for assigning the correct label for an item of data is the correct or probably correct label (if one can obtain it without undue endeavour). From a practical viewpoint some effort directing to hand-crafting code to extract appropriate class labels may thus assist machine learning.

6 Conclusions and Future Work

In terms of the overall performance of the methods described in this paper the closest point of comparison is the work reported by Indukuri and Krishna (Indukuri and Krishna, 2010). While comparisons using different methods, software and data are of dubious validity, we may note their broad similarity. With a training set of 10 contracts and a test set of 20 contracts, the contract-by-contract average F-measure for the clausematter class was 0.8632. Also in 14 contracts of the 20 tested the F-measure for the clausematter class was above 0.85. With a training set of 10 contracts (Set A) and a test set of 10 contracts (Set B) (as a single test set) the F-measure for the clausematter class was 0.921.[19] Using 4-grams as features, Indukuri and Krishna report an accuracy of 83.48% on the task of classification clauses from non-clauses, applied to 73 sentences to be so classified. Whereas the task in that case was a binary classification task over one contract, we report results on multi-class classification over test sets of up to 20 contracts.[20]

The work we have undertaken and related work reviewed in this paper (Section 3) suggests the potentially broad utility of combining rule based and machine learning methods. One potential hybrid classification method in its simplest formulation may look something like this:

1. determine the required or anticipated level of accuracy for the intended application;

2. develop a simple code rule set for the classification task (i.e. with minimal expenditure of resources);

3. if classification accuracy is sufficient machine learning would not be required, if not proceed as usual in development of machine learning using the output of hand coded rules as an input to machine learning.

Such a development model does not avoid the development of a supervised training set, as such a set is required both to assess the accuracy of the rule based tagger as well as the machine learner (should development proceed to that stage). However it may reduce the amount of data required to attain a desired level of accuracy. The method could of course be applied iteratively, if the accuracy level is insufficient after a first iteration.

Future work to be explored includes the effective "typing" of contracts (particularly in light of the high variance of performance on individual contracts). For instance, some contracts use clause headings, others do not. Some use schedules, others do not, etc. Such type information may further assist in the classification task (the accuracy of which varies considerably depending on the contract to which it is applied). In typical preprocessing, documents in corpora are converted to plain text. This results in the loss of layout and hierarchical information found in word documents. The preservation of such information for use in classification tasks may improve accuracy.

The preliminary work here is also a precursor to work on identification of ambiguity and the development of practical tools to assist in contract drafting. In separate work, we intend also to describe and analyse the contract corpus that is referred to here as the basis of this work.

[18] The Weka decision tree implementation, "J.48", is based on the Quinlan's C4.5(Quinlan, 1986).

[19] The results reported here are derived using a random forest algorithm and using the hand coded tag as an input feature for classification.

[20] Coincidentally the 83.48% figure is the same as our average result over all classes for 20 contracts.

References

J. K. Aitken and P. Butt. 2004. *Piesse The Elements of Drafting*. Lawbook Co, 10th edition.

L. Bacci, P. Spinosa, C. Marchetti, R. Battistoni, I. Florence, I. Senate, and I. Rome. 2009. Automatic mark-up of legislative documents and its application to parallel text generation. In *Proc. of LOAIT Workshop*, pages 45–54.

D. Biber, S. Conrad, and R. Reppen. 1998. *Corpus linguistics: Investigating language structure and use*. Cambridge University Press.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Rob Callan. 2003. *Artificial Intelligence*. Palgrave Macmillan.

(ed) Christopher S. Butler. 1992. *Computers and written texts*. Blackwell.

E. Francesconi. 2006. The Norme in Rete Project: Standards and Tools for Italian Legislation. *International Journal Legal Information*, 34:358.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software. *SIGKDD Explorations*, 11(1).

I. Hasan, J. Parapar, and R. Blanco. 2008. Segmentation of legislative documents using a domain-specific lexicon. In *Proceedings of the 19th International Conference on Database and Expert Systems Application*, pages 665–669.

Kishore Varma Indukuri and P. Radha Krishna. 2010. Mining e-contract documents to classify clauses. In *COMPUTE '10: Proceedings of the Third Annual ACM Bangalore Conference*, pages 1–5, New York, NY, USA. ACM.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, 2nd edition.

D. Khoury and Y. S. Yamouni. 2007. *Understanding Contract Law*. Lexis Nexis Butterworths, 7th edition.

M. Kipp. 2006. Creativity meets automation: Combining nonverbal action authoring with rules and machine learning. In *Intelligent Virtual Agents*, pages 230–242. Springer.

Eric McCreath and Judy Kay. 2003. Iems: Helping Users Manage Email. *User Modeling 2003*, pages 146–146.

Eneldo Loza Mencía. 2009. Segmentation of legal documents. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 88–97. ACM.

Seong-Bae Park and Byoung-Tak Zhang. 2003. Text chunking by combining hand-crafted rules and memory-based learning. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 497–504.

J.R. Quinlan. 1986. Induction of decision trees. *Machine Learning*, 1:81–106.

M. Rochery, R. Schapire, M. Rahim, N. Gupta, G. Riccardi, S. Bangalore, H. Alshawi, and S. Douglas. 2002. Combining prior knowledge and boosting for call classification in spoken language dialogue. In *Proceedings of the IEEE International Conference of Acoustics and Speech and Signal Processing*.

S. Scott and S. Matwin. 1999. Feature Engineering for Text Classification. In *Machine learning: proceedings of the Sixteenth International Conference (ICML'99), Bled, Slovenia, June 27-30, 1999*, page 379. Morgan Kaufmann Pub.

K. Takahashi, H. Takamura, and M. Okumura. 2005. Automatic occupation coding with combination of machine learning and hand-crafted rules. *Advances in Knowledge Discovery and Data Mining*, pages 269–279.