A Joint Human/Machine Process for Coding Events and Conflict Drivers

Bradford Heap[1], Alfred Krzywicki[1], Susanne Schmeidl[2], Wayne Wobcke[1] and Michael Bain[1]

[1]School of Computer Science and Engineering
[2]School of Social Sciences
University of New South Wales
Sydney NSW 2052, Australia
{b.heap,alfredk,s.schmeidl,w.wobcke,m.bain}@unsw.edu.au

Abstract. Constructing datasets to analyse the progression of conflicts has been a longstanding objective of peace and conflict studies research. In essence, the problem is to reliably *extract* relevant text snippets and *code* (annotate) them using an ontology that is meaningful to social scientists. Such an ontology usually characterizes either types of violent events (killing, bombing, etc.), and/or the underlying drivers of conflict, themselves hierarchically structured, for example security, governance and economics, subdivided into conflict-specific indicators. Numerous coding approaches have been proposed in the social science literature, ranging from fully automated "machine" coding to human coding. Machine coding is highly error prone, especially for labelling complex drivers, and suffers from extraction of duplicated events, but human coding is expensive, and suffers from inconsistency between annotators; thus hybrid approaches are required. In this paper, we analyse experimentally how human input can most effectively be used in a hybrid system to complement machine coding. Using two newly created real-world datasets, we show that machine learning methods improve on rule-based automated coding for filtering large volumes of input, while human verification of relevant/irrelevant text leads to improved performance of machine learning for predicting multiple labels in the ontology.

1 Introduction

Identifying and tracking drivers of conflict in order to anticipate the outbreak of violence has been an elusive challenge for social science. The field of peace and conflict studies suffers greatly from the lack of reliable quantitative data on societal and political factors and events which are more abstract than the basic information typically collected for armed conflicts, such as individual battle statistics. This limits the ability of analysts to develop effective and general conflict analyses and early warning models [3].

A serious problem is that, even when data is collected, it needs to be stored in a form that analysts can later search and use to analyse the long-term progression of a conflict. The idea is to *code* the data by annotating *relevant* information with concepts from a pre-defined ontology [15, 10]. Abstract concepts in the ontology are designed to capture what analysts consider to be important for making sense of the many individual events or factors involved in an extended conflict. Moreover, useful information can originate from a variety of sources and be of a variety of types. At one extreme, long and complicated formal analyst reports can provide historical context and in-depth analysis, often focusing on underlying *conflict drivers* (structural and causal factors contributing to the progression of a conflict). At another extreme, news and social media can provide timely and useful information (and also misinformation) on a daily basis, typically reporting many isolated events (such as individual battles, attacks, deaths, gains/losses of territory) that need to be classified for future analysis.

During the past two decades, much progress has been made in automating event coding through the development of systems that apply term dictionaries and syntactic rules to automatically filter text content and extract actors, actions, places and times related to an event [4, 3, 7]. However, while this machine extracted information is more suited for studies that involve the consolidation of event statistics, the coding is typically less sophisticated than human analysts require for conflict analysis [10]. Automated event coding methods are error-prone and suffer from the problem of extracting the same event multiple times (duplicated events). On the other hand, human coding is expensive and time-consuming, which means information is not up to date, and datasets suffer from disagreement between coders as to how best to characterize an event or fact (this disagreement is sometimes legitimate, as "coding" is subject to interpretation and made on the basis of background knowledge that may differ amongst experts). Thus hybrids of human and machine coding systems are required to balance the accuracy and complexity of coding with the timeliness and coverage of the events and factors involved in a conflict.

In this paper we empirically analyse a proposed hybrid human/maching coding process. We study two different conflict domains, which, while different in the type of document sources and the information to be extracted and coded, nevertheless are similar in the requirements to: (i) identify the small amount of *relevant* text from a large volume of input, and (ii) *code* (annotate) the relevant text using concepts from a (possibly) domain-specific ontology. As part of our work, we have developed two new datasets that may be of independent interest to social scientists studying those conflicts. The specific conflicts are the long and ongoing "cycles" of violence in the Democratic Republic of the Congo (DRC) during the period 2002–2006 (before the outbreak of violence surrounding the presidential elections), and the long-running war in Afghanistan, focusing on events reported in news sources in 2016 (which we have called the AfPak dataset, following common usage of this term to denote Afghanistan/Pakistan). The basic approach to the experiments is that, given a "ground truth" dataset constructed by human expert coders, we mimic the performance of a hypothetical joint human/machine coding process, to determine where, in the future, human input is best utilized. Considering the two-step process of determining relevance and coding, we simulate the effect of using human input at each step to refine the output of the system at that stage. This enables us to determine the specific strengths and weaknesses of a variety of machine learning methods, on the individual steps and in combination. Our results show that human input at the stage of identifying relevance can greatly improve the overall performance of the system (in terms of precision and recall), and that machine learning is most effective in identifying a small number of concepts from the ontology for using in coding, from which a human can select the best one(s).

We begin the paper with a more detailed description of coding, including the construction of our ontologies and datasets, then present a formalization of document coding as a machine learning problem, and finally describe our proposed process for joint human/machine coding and the experimental results forming the evaluation of the process.

2 Coding of Events and Conflict Drivers

In essence, *coding* requires both the development of a coding scheme or ontology, and a systematic coding process specifying how to apply coding rules to extracted data for inclusion in a dataset [15]. Note here that the "coding rules" are typically meant for human use, so may be incomplete or open to interpretation.

Historically, the ontologies of the World Event/Interaction Survey (WEIS) [8] and the Conflict and Peace Databank (COPDAB) [1] have been used for large scale human coding in developing conflict datasets. While the largest coding projects, such as the Global Database of Events, Language, and Tone (GDELT) [7] have now become fully automated there are still some projects which are entirely human coder based [6, 11].

2.1 Ontologies for Events and Conflict Drivers

The development of any coding scheme requires expert domain knowledge and careful consideration of what data to code to ensure the coding process results in a high quality and accurate dataset. Numerous coding schemes have been developed, originally focused on international relations [8, 1], and later expanded to include domestic relations and local events [3, 15, 5, 11].

Generally, coding ontologies are hierarchical and information can be classified at different levels of the hierarchy. Ontologies for events are simpler than for conflict drivers (see below) because the ontology takes the form of a type hierarchy. As our objective is to code events relating to the conflict in Afghanistan, we adapted the basic CAMEO ontology [5], focusing on violent events (attacks, killings, etc.) and statements (public announcements, claims of responsibility, etc.). Figure 1 shows part of our event type ontology. Note that with this coding scheme, events can be classified at the "top level" only, e.g. if an event is an attack but not one of the subtypes of attack specified in the hierarchy. Note also that some of the concepts are domain specific, due to the nature of the conflict in Afghanistan (such as suicide bombing), though the aim is make the ontology as domain independent as possible.

– Attack
  - Drone strike
  - Suicide bombing
  - · · ·
– Kill
  - Behead
  - Kill civilians
  - · · ·
– Make a public statement
  - Claim responsibility
  - · · ·

Fig. 1. AfPak Event Type Ontology

The ontology for conflict drivers focuses on factors (which could include events) that contribute causally to an ongoing conflict. One way to characterize conflict drivers is in terms of *structural causes* or *root causes* (pervasive factors or grievances that have become entrenched in a society and create the preconditions for violent conflict), *proximate conditions* or *intervening factors* (factors contributing to a climate conducive to violent conflict or its further escalation that are under more direct control), and *triggers* (single key acts or events that set off or escalate violent conflict).

Many existing machine-based coding ontologies consider only the extraction of events and ignore structural causes and proximate conditions, which, even for an expert human coder, can be difficult to extract [3]. But by coding only events, the resulting datasets may have inherent biases [14]. In contrast, our ontology includes drivers at multiple layers of abstraction.

The drivers for the Democratic Republic of the Congo (DRC) dataset are characterized as a three-level hierarchy of *pillars*, *categories* and *indicators*. Pillars capture the basic objectives of peacebuilding, including *economics*, *governance*, *rule of law* and *security*. The *indicators* cover dynamic conditions such as *economic decline* and *external influence*. Indicators are grouped into categories, such as *human rights* and *internal stability*, which are aspects of the associated pillar. Thus the hierarchy is not a simple type or part-whole hierarchy but a collection of abstract concepts some of which involve causal relations (e.g. *cross border aggression* causes (a decrease in) *internal security* which in turn is an aspect of *security*). Nonetheless, each indicator is associated with exactly one category and each category with exactly one pillar.

2.2 Datasets

In our work we have developed two new datasets. We describe first a dataset of events reported in news sources on Afghanistan and Pakistan (AfPak), then describe that based on conflict drivers extracted from analyst reports on the Democratic Republic of the Congo (DRC).

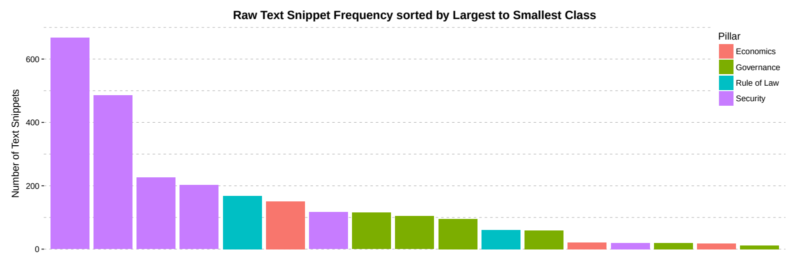

Fig. 2. Frequency distribution by category of text snippets in the ICG DRC dataset

AfPak Events Dataset This dataset focuses purely on single events of the form "who did what to whom and where". The ontology extends CAMEO [5] to include not only event types, as described above, but Afghanistan-specific individuals, organizations, locations, and types of equipment. In total, the ontology contains 441 individuals, 155 organizations, 708 locations and 7 types of equipment. The ontology includes 119 event types, making event coding a difficult problem for human and machine. The experiments in this paper use online news articles in English from local and international media, and NGO and extremist group sources; data is from August, October and November 2016.

To develop the dataset, human coders extracted events that could map to the ontology, and with each event, where possible, identified actors, targets and locations, and a snippet of text specifically expressing the event type. In total, human coders read over 6,500 webpages containing more than 112,000 sentences, and extracted a total of 1,478 events which were mapped to 72 top-level event types (lower level event types are not considered in this paper, due to the lack of instances). The low ratio of extracted events to the number of sentences reflects the amount of filtering required to determine the relevant sentences.

ICG DRC Dataset This dataset contains "text snippets" (short fragments of text) extracted from 15 International Crisis Group (ICG) reports on the DRC during the period 2002–2006, annotated with indicators, categories and pillars from the ontology. Social science domain experts had full control over the creation of the ontology, coding rules and extraction of driver related text snippets included in the dataset. An ontology was defined before coding commenced, with minor adjustments made during the coding process. The final ontology contains 70 indicators, 17 categories and the 4 pillars. To construct the dataset, a domain expert read 8,836 sentences across the 15 reports, extracted 2,541 text snippets, and used 68 of the 70 indicators. The analysis in this paper is at the level of categories, as many indicators have very few examples in the data.

Figure 2 shows the distribution of assigned categories in the ICG DRC dataset, showing (as typical in this research field) the highly imbalanced nature of the dataset. Here the top 4 categories are all related to the *security* pillar, as the data includes much information on fighting between militia groups in various parts of the country over the period.

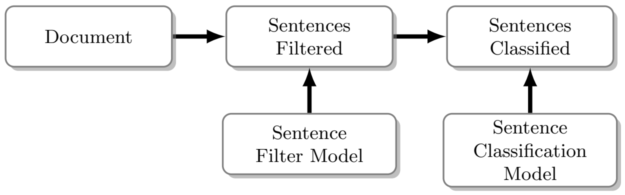

Fig. 3. Automated Machine Coding Process

3 Document Coding as a Machine Learning Problem

As outlined above, document coding is a two step process: (i) identifying relevant information (events or driver related text, for now assumed to be sentences), and (ii) coding the extracted information by annotating it with concepts from the ontology. An overview of an automated coding process is shown in Figure 3, where the process uses two models, a sentence filtering model for determining relevant information, and a sentence classification model for the coding.

Formally, we define a *document* D in its simplest form as simply a sequence of tokens $D = [t_0, t_1, \ldots]$, where a *token* t is defined to be a word (individual or compound) or punctuation symbol. We then define a *text snippet* S as a (not necessarily contiguous) subsequence of tokens $S \sqsubseteq D$, and a *sentence* to be a maximal text snippet that is a contiguous subsequence of D, ends with a punctuation symbol indicating the end of a sentence, such as '.' or '?', and which, apart from this token, is grammatically correct. We can then reasonably assume that a text document D consists of a sequence of sentences (though also contains headings, footnotes, image captions, etc.).

Thus each text snippet S is a subsequence of the document's token sequence $S = [t_0, t_1, \ldots]$, however there is no guarantee that a text snippet forms a fully parsable sentence, and moreover, text snippets can cross sentence boundaries, and may not even be contiguous subsequences of the tokens in the document.

We now define the step of identifying relevant information as classifying a sentence as *relevant* by use of a classification function $f_r : \mathcal{S} \to R$, where $\mathcal{S}$ is the set of all sentences and $R = \{\text{relevant}, \text{not relevant}\}$. If a sentence is classified as relevant, the second step of the process is to assign to it one *or more* concepts from the ontology. Thus importantly, coding is a *multi-label* classification problem. More formally, if the ontology is taken as the set of concepts $C = \{c_0, c_1, \ldots\}$, the classification of a sentence is defined by a function $f_c : \mathcal{S} \to 2^C$, where obviously, if S is not a relevant sentence, f_c assigns S the empty set of concepts.

3.1 Automated Sentence Extraction and Classification

As with the construction of a human coding scheme, an automated text extraction and classification system (Figure 3) requires information about relevant sentences and how to extract them. There is commonality with the human process, since the same classification labels are applied to the extracted sentences.

However, where the human process requires the creation of a codebook and minimal set of examples, the machine process either requires a set of structural rules (c.f. [14]) or many examples per class to learn a model (c.f. [10]). This requirement of many training examples is problematic for rare events and events whose interpretation depends on context or background knowledge.

Once a coding scheme has been developed, models can be built to classify sentences. In most existing machine-based text extraction and classification systems for conflict analysis, there is a reliance on the direct matching of the text to the concepts in the coding ontology.

Machine-based sentence extraction and classification approaches are much quicker and cheaper than human systems. Once a model is developed, it can run continuously and will be consistent in its performance [14]. However, these systems are much less flexible than human coding systems. In particular, special cases are hard to handle and classification errors are often quite “far” from the correct classification [15]. Furthermore, many machine-based coding systems are unable to recognise new terms and phenomena [14], and models which are built based on assumptions about the distribution of events may perform poorly if that distribution changes.

The continued reliance on rule-based pattern matching in these systems is contrary to most trends in classification systems. In other domains, the supervised machine learning approaches of Multinomial Naive Bayes (MNB) and Support Vector Machines (SVM) have largely superseded rule-based systems [16].

4 A Joint Human/Machine Coding Process

Human and machine coding processes have different strengths and weaknesses. While machine coding processes are always consistent in their rule and pattern matching, this also forms part of their Achilles’ heel. In contrast, a human coder can easily adapt to changes in language expression, use and terminology. A joint human/machine coding process should be designed to take advantage of the consistency of machine-based coding and also allow a human coder to correct errors or add information missed by the automated process. Although this joint process will be many times slower than a fully automated process, the overall quality of the data extracted should be higher, allowing for much richer analysis. In summary, the key benefit of a hybrid approach is to enable a machine to perform the mundane tasks of identifying possible sentences containing events, but with human verification. This prevents the human from missing events, and also allows the human to help the system classify rare events, or correct misclassifications.

Figure 4 outlines the steps in our joint human/machine process. The key difference between this joint process and the fully automated machine-based process is the inclusion of the human coder after the extraction of relevant sentences. This allows the human coder to discard any incorrectly identified sentences, and to select a shorter text snippet from the sentence containing the the tokens that the human coder judges to be sufficient to determine the correct classification.

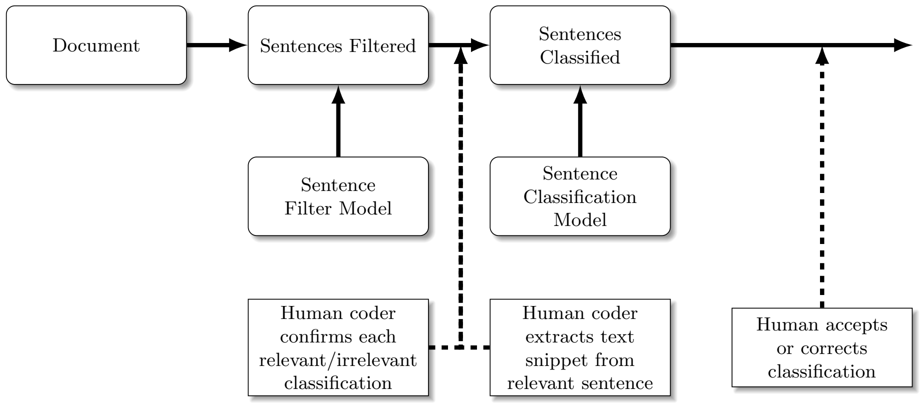
Document
Sentences Filtered
Sentences Classified
Sentence Filter Model
Sentence Classification Model
Human coder confirms each relevant/irrelevant classification
Human coder extracts text snippet from relevant sentence
Human accepts or corrects classification

Fig. 4. A Joint Human/Machine Coding Process

Finally, after the machine classifies the sentence (possibly using only the text snippet), the human coder can accept or change the classification. This optional step allows the system to present to the human coder a list of the most relevant potential classification labels, from which the human selects the correct labels.

4.1 Development and Evaluation of Sentence Extraction Models

Both the automated and joint coding processes involve a common step of extracting the relevant sentences. Although all the above-mentioned event extraction systems approach this problem using patterns based on NLP parsing, we propose using supervised machine learning for this task.

The two human-annotated datasets described in Section 2.2 give us a way to evaluate approaches for the extraction of relevant sentences. Our first evaluation involves identifying if a sentence is relevant in the AfPak dataset, and our second on the ICG DRC dataset. As the AfPak dataset is primarily based on news reports and is coded according to an extension of the CAMEO coding scheme, we are also able to compare supervised machine learning models to the PETRARCH Coding System.[1]

In comparing supervised machine learning approaches to PETRARCH, we expect some differences in performance. First, we expect similar or better levels of recall in the supervised machine learning models compared to PETRARCH, because the supervised machine learning models operate independently of parsing rules and, given that the data were coded by human coders, we expect that the human coders did not consciously follow a subject-verb-object pattern of text extraction. Second, we expect that PETRARCH will identify many non-relevant events, since it is designed to extract events related to many areas of the world, not specifically Afghanistan/Pakistan.

[1] PETRARCH2, the most recent release, was used in our experiments (http://github.com/openeventdata/petrarch2).

Evaluation Methodology In our analysis, we use the standard metrics of recall and precision to measure model performance. These metrics are defined as:

$$\text{recall} = \frac{tp}{tp + fn} \qquad\qquad \text{precision} = \frac{tp}{tp + fp}$$

where tp is the number of true positives, i.e., the number of sentences correctly classified as relevant, fn the number of misclassifications of relevant sentences as irrelevant, fp the number of irrelevant sentences incorrectly classified as relevant, and tn the number of irrelevant sentences correctly classified as irrelevant.

Ideally, a sentence extraction system should produce results which have both high recall and high precision. However, in practice models with high recall are also likely to produce many irrelevant sentences; for instance, a model coul achieve perfect recall if it marks every sentence as relevant. In contrast, models with high precision are likely to have lower recall, since they only suggest relevant sentences in which they have high confidence.

For the joint human/machine coding system, we focus primarily on obtaining good recall, with a secondary focus on precision. This is warranted since, in validating the output of sentence extraction, when a human is shown an incorrect label for a sentence they can easily reject it, but when shown a sentence missing a correct label, they are likely to take longer to identify the relevant content and determine its correct classification.[2] Additionally, to capture variation between methods in the balance of the number of irrelevant sentences classified as relevant (false positives) and the number of relevant sentences not marked as relevant (false negatives), we also calculate the F1 score (the harmonic mean of recall and precision).

For evaluation, we split the AfPak dataset into a training set containing 65% of the articles and a test set the remaining 35%. Articles were presented to our human coders in chronological order, so the training/test split followed the same ordering. The training set contains the first 4,892 articles annotated by our human coders, of which 1,004 sentences (1.26%) are marked as relevant and 78,503 sentences as irrelevant. The test set contains 2,625 articles of which 362 sentences (0.84%) are relevant and 42,640 sentences are irrelevant. We followed the approach of Bagozzi and Schrodt [2] and reduced our dataset to contain only the first 6 sentences of each article. After this reduction, the training set contained 931 relevant sentences (3.95%) and 22,657 irrelevant sentences, and the test set 352 relevant sentences (2.92%) and 12,075 irrelevant sentences.

We compare four approaches for identifying relevant sentences. The first, as a baseline, is Schrodt's PETRARCH system in its default form. The second and third are variants on Multinomial Naive Bayes (MNB). The fourth is a widely-used implementation of the Support Vector Machine (SVM) algorithm. For the ICG DRC dataset, we cannot apply PETRARCH, but we used the same three learning algorithms, and applied 10-fold cross validation.

Multinomial Naive Bayes The MNB classifier [9] is a simple probabilistic classifier in which each token in the text is treated as conditionally independent,

[2] This appears in timings in our log files and was explicitly stated by one of our coders.

Table 1. Sentence Extraction - AfPak Dataset (Time-ordered Train/Test Split)

	tp	fn	fp	tn	Recall	Precision	F1
PETRARCH	141	211	1,559	10,498	0.401	0.083	0.138
MNB	**213**	139	766	11,309	0.605	0.218	0.321
MNB-UP	**252**	100	1,303	10,772	**0.716**	0.162	0.264
SVM	156	196	172	11,903	0.443	**0.476**	**0.459**

Table 2. Sentence Extraction - ICG DRC Dataset (Mean 10-fold Cross Validation)

	tp	fn	fp	tn	Recall	Precision	F1
MNB	124.8	92.4	113.0	553.4	0.575	0.525	0.549
MNB-UP	**155.0**	62.2	188.3	477.7	**0.714**	0.452	**0.554**
SVM	98.4	118.8	93.2	572.8	0.453	**0.514**	0.482

given the class. The conditional probability of a given token for a particular class is calculated using Laplace smoothing. We also use a version of MNB where we assume uniform priors (MNB-UP). This allows us to account for the heavy skew towards sentences being marked as irrelevant (c.f. [12, 13]). In both models, before training and testing we apply the common pre-processing steps of removing stop words, stemming words using the Porter stemmer and removing words which only occur once in the training set.

Support Vector Machines SVM classifiers are widely-used as an alternative to Naive Bayes for classifying text documents [16]. We build SVM models using Weka's[3] implementation of the Sequential Minimal Optimization (SMO) algorithm and our best models were produced without any text pre-processing.

AfPak Dataset Evaluation Table 1 shows the results of our comparison of the models at identifying relevant sentences. The results show the limitation of the rule and pattern based approach as used by PETRARCH compared to machine learning. More than half the relevant sentences are not extracted by the PETRARCH system, and the best machine learning method showed a more than 3-fold improvement on its F1 score. Since we are using PETRARCH in its default form, there are several possible reasons for this. First, it may be discarding potentially relevant sentences as they do not match to known actors and locations in its dictionaries. Second, the form of the human annotated sentences may not match the forms it is configured to detect. Third, it may be discarding sentences which it is not able to classify according to the expanded form of the CAMEO ontology we used for annotatation. Although these issues could potentially be addressed by adding more rules and knowledge to the system, this illustrates the limitations of applying PETRARCH in domains differing from its "who did what to whom" definition of events.

[3] http://www.cs.waikato.ac.nz/ml/weka/

In comparison, the supervised machine learning methods obtained higher levels of recall and precision. The MNB models had the best recall, and the SVM achieved highest precision. This is interesting as their performance in other domains is generally very evenly matched [16, 12]. Furthermore, despite the low precision obtained by the MNB-UP classifier, it still correctly eliminated 89% of all irrelevant sentences which is a large aid to a human analyst.

ICG DRC Dataset Evaluation Table 2 presents the results of our evaluation of the various models for identifying relevant sentences in the ICG DRC dataset. In this dataset, we used the models to classify *all* sentences, as relevant sentences were spread throughout the reports. However, we do not compare the machine learning models to the PETRARCH system as this dataset is not coded according to a CAMEO based ontology. These results show that recall for each of the classifiers in identifying relevant sentences is very similar to the results on the AfPak dataset. For the MNB models, precision is much higher than the AfPak dataset, which is possibly a consequence of the dataset containing a much higher proportion of relevant sentences.

4.2 Human Filtering of Relevant Sentences

After sentences have been classified as relevant or irrelevant, as in Figure 4, a human coder is able to validate the output. To evaluate the relative contribution of a "human-in-the-loop" versus the automated process in removing irrelevant sentences, we simply assume the human coder eliminates any false positive classifications, resulting in maximum precision. Furthermore, the human coder may be able to select an important text snippet, that is, a specific portion of the sentence that more precisely expresses the event type or driver, and which is judged sufficient to determine the classification of the sentence. This text snippet could be a single word (such as 'attack') or a phrase (such as 'killed in a drone strike'). Again, in our evaluation of machine learning for sentence classification, we focus on recall and matching the classifications made by the human coder over precision, as this avoids the assumption that the human made a deliberate decision not to label a particular sentence in a particular manner. We note that if a human coder did not filter out the false positives, these would be classified in the next stage of the process which produces ontology labels for each sentence, resulting in a higher number of incorrectly labelled sentences.

4.3 Ontology Classification

The classification of sentences can be made at either the sentence level or the text snippet level. If a human in the process of filtering relevant sentences only accepts or rejects the sentences and provides no more filtering then the system can only operate at the sentence level. However, if a user is able to filter the sentence to the relevant text snippet then our classifier should have a higher recall as irrelevant tokens are manually filtered out.

Table 3. Sentence Classification - AfPak Dataset

	Sentences Input		Relevant Recall				Fully Misclassified Sentences				
	Machine	Joint	Correct Labels		Missed Labels		Machine		Joint		Error Reduction
MNB	979	213	260	72.2%	100	27.8%	811	82.8%	45	21.1%	74.5%
MNB-UP	1,555	252	**291**	71.5%	116	28.5%	1,362	87.6%	59	23.4%	73.3%
SVM	328	156	210	**78.1%**	59	21.9%	196	59.8%	24	15.4%	74.2%
MNB-UP + SVM	1,555	252	**323**	**79.4%**	84	20.6%	1,342	86.3%	39	15.5%	82.0%

Table 4. Sentence Classification - ICG DRC Dataset (Mean 10-fold Cross Validation)

	Sentences Input		Relevant Recall				Fully Misclassified Sentences				
	Machine	Joint	Correct Labels		Missed Labels		Machine		Joint		Error Reduction
MNB	237.8	124.8	87.8	54.0%	74.6	46.0%	159.9	67.2%	46.9	37.6%	44.0%
MNB-UP	343.3	155.0	**109.1**	**56.0%**	85.8	44.0%	244.4	71.2%	56.1	36.2%	49.2%
SVM	191.6	98.4	71.0	53.9%	60.8	46.1%	129.4	67.5%	36.2	36.8%	45.5%

To evaluate sentence classification, we construct models based on the two MNB approaches and the SVM approach. We then input into each model its corresponding true positive classifications from the previous stage of the process, assuming false positives are removed by a human coder. As each sentence may have multiple labels, our ontology classifiers are configured to produce the 3 most relevant classification labels, and use a modified definition of recall more suited to the multi-label setting. For each model, we define *tc* to be the number of correct labels within these top 3 labels and *fc* to be the number of correct labels that were not predicted by the model to be within the top 3 labels, summing over all sentences. We define the *relevant recall* of a model as the overall proportion of the correct labels that are found by the model:

$$\text{relevant recall} = \frac{tc}{tc + fc}$$

Let *fs* be the number of sentences for which the model produces no correct labels within the top 3. We define the *full sentence misclassification rate* as:

$$\text{full sentence misclassification rate} = \frac{fs}{|\text{Sentences Input}|}$$

The aim is high relevant recall and low full sentence misclassification rate.

Sentence Classification Table 3 shows the results of this evaluation applied to the AfPak dataset in classifying the relevant sentences into our enhanced CAMEO ontology. These results show that the MNB-UP model produces the highest raw number of correct label classifications, although it has the lowest relevant recall. This first result is due to the model having the largest number of true positive relevant sentences supplied to it from the previous stage and the latter result suggests that the MNB-UP classifier is not as good in this second stage. In contrast, the SVM model has the highest relevant recall and the lowest number of full sentence misclassifications. Using this result we form

Table 5. Text Snippet Classification - AfPak Dataset

	Sentences Input	Relevant Recall					Fully Misclassified Sentences	
	Joint	Correct Labels		Gain	Missed Labels		Joint	
MNB	213	279	77.5%	5.3%	81	22.5%	54	25.4%
MNB-UP	252	**333**	**81.8%**	**10.3%**	74	18.2%	60	23.8%
SVM	156	214	79.5%	1.4%	55	20.4%	39	25.0%

Table 6. Text Snippet Classification - ICG DRC Dataset (Mean 10-fold Cross Validation)

	Sentences Input	Relevant Recall					Fully Misclassified Sentences	
	Joint	Correct Labels		Gain	Missed Labels		Joint	
MNB	124.8	90.8	57.2%	3.2%	68	42.8%	44.2	34.5%
MNB-UP	155.0	**114.5**	**58.7%**	2.7%	80.4	41.3%	49.4	31.8%
SVM	98.4	74.5	56.5%	2.6%	57.3	43.5%	31.4	31.9%

the hypothesis that the MNB-UP classifier is the best at identifying possible relevant sentences in the first stage of the process and the SVM classifier is the best performer at classifying relevant sentences into the ontology (on the AfPak dataset). To test this hypothesis we feed the MNB-UP's true positive result from the first stage into the SVM ontology classification model. This result is in the last row of Table 3 and shows that this approach produces the highest raw and relative number of correct labels. Importantly, we can see the effect of the human coder in filtering out irrelevant sentences prior to machine learning, which for MNB-UP + SVM gives an 82% reduction in full sentence misclassifications.

However, these results are not as pronounced on the ICG DRC dataset (shown in Table 4). On this dataset the MNB-UP classifier is the best performer in both stages of the classification process. Furthermore, the relevant recall rate in this dataset is much lower than the AfPak dataset evaluation; there is also a reduction in full sentence misclassifications, but it is not as large.

Text Snippet Classification Our final evaluation considered whether the relevant recall rate would improve if a human annotator was able to extract only the relevant text snippet for classification into the ontology. Table 5 shows this evaluation on the AfPak dataset. This shows that for all models there is an increase in the number of correct labels predicted, with the largest increase in performance in the MNB-UP classifier, but at the expense of an increase in the full misclassified sentences rate. This suggests that although the relevant recall increases, the process by which the text snippet is selected can eliminate information from other sentences which were previously classified correctly. A similar but smaller increase in relevant recall is seen in the evaluation on the ICG DRC dataset in Table 6. In this dataset the number of fully misclassified sentences decreases across all models. We conclude that with text snippets, classification is improved, but selecting only very specific words can result in a loss of contextual information and increase the number of fully misclassified sentences.

5 Related Work

With respect to existing approaches for this type of application, the Social, Political and Economic Event Database (SPEED) system [10] is the most similar. In this system human coders are presented with input data (documents) that has been automatically pre-processed and classified as relevant. It is then claimed that humans "perform only the most difficult coding decisions". The SPEED system is developed with a Naive Bayes classifier trained initially on 33,000 training documents. Relevant documents are then passed through a NLP pipeline which extracts people, locations and organisations. Human coders then check all the machine outputs. One of the key arguments of Nardulli *et al.* [10] for their approach is that menial work is handled by the machine and the cognitively challenging tasks are handled by the human. In developing their Naive Bayes model for selecting relevant documents their original document level true positive rate was 33% and was improved to 87% after an additional 60,000 training documents were added to the model. In our datasets, which trained relevant/irrelevant classification models at the sentence level we achieved a recall of 71% for both datasets with a MNB variant trained on a much smaller dataset. The finding that MNB is the best classifier for deciding relevance is consistent with the resuits of Nardulli *et al.* [10].

6 Conclusion

We have demonstrated that standard machine learning techniques can be applied to the problem extracting and classifying events and conflict drivers from news and NGO reports. This approach differs from previous work on event extraction in the social sciences that has focused on the use of rules for matching large dictionaries of actors, locations and specific verb phrases.

Recognizing the importance of human input to the coding process, we proposed a two-step "joint" process and showed experimentally that human input is effective when used to: (i) filter out irrelevant sentences from amongst those classified as relevant by an automated method, and (ii) select the correct classification(s) from a small number of suggestions given by a learning model. The models were tested using event data focusing on violent events in Afghanistan, and a dataset of conflict drivers in the Democratic Republic of the Congo.

Future work involves incorporating the process into a complete "pipeline" for ingestion of news, social media feeds and NGO reports, and storing events and drivers in a searchable database. Research will also address extracting the components of events (actors, targets, etc.), and the use of stream mining methods to extract information in real time, exploiting the temporal nature of the data.

Acknowledgements

This work was supported by Data to Decisions Cooperative Research Centre. We are grateful to Josie Gardner for labelling the ICG DRC dataset, and to Michael Burnside and Kaitlyn Hedditch for coding the AfPak event data.

References

1. Azar, E.E.: The Conflict and Peace Data Bank (COPDAB) Project. Journal of Conflict Resolution 24, 143–152 (1980)
2. Bagozzi, B.E., Schrodt, P.A.: The Dimensionality of Political News Reports. Paper presented at the Second Annual General Conference of the European Political Science Association, Berlin (2012)
3. Bond, D., Bond, J., Oh, C., Jenkins, J.C., Taylor, C.L.: Integrated Data for Events Analysis (IDEA): An Event Typology for Automated Events Data Development. Journal of Peace Research 40, 733–745 (2003)
4. Bond, D., Jenkins, J.C., Taylor, C.L., Schock, K.: Mapping Mass Political Conflict and Civil Society: Issues and Prospects for the Automated Development of Event Data. Journal of Conflict Resolution 41, 553–579 (1997)
5. Gerner, D.J., Schrodt, P.A., Yilmaz, O., Abu-Jabr, R.: Conflict and Mediation Event Observations (CAMEO): A New Event Data Framework for the Analysis of Foreign Policy Interactions. Paper presented at the Annual Meetings of the International Studies Association, New Orleans, LA (2002)
6. LaFree, G., Dugan, L.: Introducing the Global Terrorism Database. Terrorism and Political Violence 19, 181–204 (2007)
7. Leetaru, K., Schrodt, P.A.: GDELT: Global Data on Events, Location, and Tone, 1979–2012. Paper presented at the Annual Meetings of the International Studies Association, San Francisco, CA (2013)
8. McClelland, C.: World Event/Interaction Survey (WEIS) Project 1966-1978. Inter-University Consortium for Political and Social Research (1978)
9. Murphy, K.: Machine Learning: A Probabilistic Perspective. MIT Press, Cambridge, MA (2012)
10. Nardulli, P.F., Althaus, S.L., Hayes, M.: A Progressive Supervised-learning Approach to Generating Rich Civil Strife Data. Sociological Methodology 45, 148–183 (2015)
11. Raleigh, C., Linke, A., Hegre, H., Karlsen, J.: Introducing ACLED: An Armed Conflict Location and Event Dataset Special Data Feature. Journal of Peace Research 47, 651–660 (2010)
12. Rennie, J.D., Shih, L., Teevan, J., Karger, D.R.: Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In: Proceedings of the Twentieth International Conference on Machine Learning, pp. 616–623 (2003)
13. Schneider, K.-M.: Techniques for Improving the Performance of Naive Bayes for Text Classification. In: Gelbukh, A. (ed.) Computational Linguistics and Intelligent Text Processing, pp. 682–693. Springer-Verlag, Berlin (2005)
14. Schrodt, P.A., Davis, S.G., Weddle, J.L.: Political Science: KEDS—A Program for the Machine Coding of Event Data. Social Science Computer Review 12, 561–587 (1994)
15. Schrodt, P.A., Yonamine, J.E.: A Guide to Event Data: Past, Present, and Future. All Azimuth 2(2), 5–22 (2013)
16. Wang, S., Manning, C.D.: Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2, pp. 90–94 (2012)