

Investigating Statistical Techniques for Sentence-Level Event Classification

Martina Naughton

School of Computer Science,
University College Dublin,
Belfield, Dublin 4, Ireland

Nicola Stokes

NICTA Victoria Laboratory,
University of Melbourne,
Victoria, Australia

Joe Carthy

School of Computer Science
University College Dublin
Belfield, Dublin 4, Ireland

Abstract

The ability to correctly classify sentences that describe events is an important task for many natural language applications such as Question Answering (QA) and Summarisation. In this paper, we treat event detection as a sentence level text classification problem. We compare the performance of two approaches to this task: a Support Vector Machine (SVM) classifier and a Language Modeling (LM) approach. We also investigate a rule based method that uses hand crafted lists of terms derived from WordNet. These terms are strongly associated with a given event type, and can be used to identify sentences describing instances of that type. We use two datasets in our experiments, and evaluate each technique on six distinct event types. Our results indicate that the SVM consistently outperform the LM technique for this task. More interestingly, we discover that the manual rule based classification system is a very powerful baseline that outperforms the SVM on three of the six event types.

1 Introduction

Event detection is a core Natural Language Processing (NLP) task that focuses on the automatic identification and classification of various event types in text. This task has applications in automatic Text Summarisation and Question Answering (QA). For example, event recognition is a core task in QA since the majority of web user

questions have been found to relate to events and situations in the world (Saurí et al., 2005). For complex questions such as *How many people were killed in Baghdad in March?*, QA systems often rely on event detection systems to identify all relevant events in a set of documents before formulating an answer. More recently, much research in summarisation has focused on the use of phrasal concepts such as events to represent sentences in extractive summarisation systems. Specifically, (Filatova and Hatzivassiloglou, 2004) use event-based features to represent sentences and shows that this approach improves the quality of the final summaries when compared with a baseline bag-of-words approach.

In this paper, we investigate the use of statistical methods for identifying the sentences in a document that describe one or more instances of a specified event type. We treat this task as a text classification problem where each sentence in a given document is either classified as containing an instance of the target event or not. We view this task as a filtering step in a larger pipeline NLP architecture (e.g. a QA system) which helps speed up subsequent processing by removing irrelevant, non-event sentences.

Two event detection approaches are explored in this paper. More specifically, we train a Support Vector Machine (SVM) using a variety of term, lexical and additional event based features to encode each train/test instance. We also adopt a probabilistic language modeling approach that captures how text within sentences that describe event instances is likely to be generated. We estimate a series of models using three well-known smoothing approaches, including LaPlace, Jelinek-Mercer and Absolute Discounting Smoothing. Their overall behavior on classification performance is ex-

amined. One advantage of language modeling for text classification is that instead of explicitly pre-computing features and selecting a subset based on arbitrary decisions (as is often the case with standard classification learning approaches such as an SVM), the language modeling approach simply considers all terms occurring in the text as candidate features, and implicitly considers the contribution of every feature in the final model. Thus, language modeling approaches avoids a potentially error-prone feature selection process.

Event classification at a sentence level is a challenging task. For example, if the target event is “Die”, we want our system to extract sentences like “5 people were killed in the explosion.” and “A young boy and his mother were found dead on Wednesday evening.”. However, it also needs to detect complex cases like: “An ambulance rushed the soldier to hospital, but efforts to save him failed.” and reject instances like “Fragmentation mines have a killing range of 100 feet.”. It seems intuitive that a naïve system that selects only sentences that contain terms with senses connected with death like “kill”, “die” or “execute” as positive instances would catch many positive cases. However, there are instances where this approach would fail. In this work we evaluate the effectiveness of such a shallow NLP approach, by developing a manual rule based system that finds sentences connected to a target event type using a hand crafted list of terms created with senses found in WordNet.

We use two datasets in our experiments. The first is the ACE 2005 Multilingual Training Corpus (Walker et al., 2006) that was annotated for 33 different event types. However, within the ACE data the number of instances referring to each event type is somewhat limited. For this reason, we select the six types with the highest frequency in the data. These include “Die”, “Attack”, “Transport”, “Meet”, “Injure” and “Charge-indict” types. The second corpus is a collection of articles from the Iraq Body Count (IBC) database¹ annotated for the “Die” event. This dataset arose from a larger humanitarian project that focuses on the collection of fatalities statistics from unstructured news data. We use this additional corpus to augment the amount of data used for training and testing the “Die” event type, and to investigate the use of extra training data on overall classification performance.

¹<http://www.iraqbodycount.org/>

Overall, our results demonstrate that the trained SVM proves to be more effective than the LM based approach for this task across all event types. We also show that our baseline system, a hand crafted rule-based system, performs surprisingly well. The remainder of this paper is organised as follows. Section 2 covers related work. We continue with details of the datasets used in the experiments in Section 3. Section 4 describes our event classification approaches while Section 5 presents their results. We conclude with a discussion of experimental observations and opportunities for future work in Section 6.

2 Background and Related Work

Event detection, in the context of news stories, has been an active area of research for the best part of ten years. For example, the NIST sponsored Topic Detection and Tracking (TDT) project, which began in 1998 investigated the development of technologies that could detect novel events in segmented or unsegmented news streams, and track the progression of these event over time (Allan et al., 1998). Although this project ended in 2004, event detection is still investigated by more recently established projects such as the Automatic Content Extraction (ACE) program, and in domains outside of news text such as Biomedical Text Processing (Murff et al., 2003).

The aim of the TDT First Story Detection (FSD) or New Event Detection (NED) task was to flag documents that discuss breaking news stories as they arrive on a news stream. Dragon Systems adopted a LM approach to this task (Allan et al., 1998; Yamron et al., 2002) building discriminator topic models from the collection and representing documents using unigram term frequencies. Then they used a single-pass clustering algorithm to determine the documents that describe new events. The overall goal of the TDT Event Tracking task was to track the development of specific events over time. However, these TDT tasks were somewhat restrictive in the sense that detection is carried out at document level. Our work differs from TDT research since event detection is performed at a sentence level where the amount of data to build discriminate models for recognising event instances is far more limited.

The goal of the ACE Event Detection and Recognition task is to identify all event instances (as well as the attributes and participants of each

Table 1: ACE Corpus Statistics

	Die	Injure	Attack	Meet	Transport	Charge-Indict
Number of Documents	154	50	235	84	181	43
Avg. document length	29.19	29.74	29.62	31.41	32.78	14.81
Avg. event instances per document	2.31	1.64	3.55	1.55	2.55	1.72
Avg. event instances per sentence	1.13	1.11	1.12	1.02	1.08	1.03

Table 2: IBC Corpus Statistics

	IBC Corpus
Number of Documents	332
Number of Sources	77
Avg. document length	25.98
Avg. events per document	4.6
Avg. events per sentence	1.14

instance) of a pre-specified set of event types. An ACE event is defined as a specific occurrence involving zero or more ACE entities², values and time expressions. Two spans of text are used to identify each event: the event *trigger* and the event *mention*. An event *trigger* or *anchor* is the word that most clearly expresses its occurrence. In many cases, this will be the main verb in the event mention. It can also appear as a noun (“The **meeting** lasted 5 hours.”) or an adjective (“the **dead** men ...”). The event *mention* is the sentence that describes the event. Even though the task of identifying event mentions is not directly evaluated in ACE, systems still need to identify them so that the various attributes and participants within the mention can be extracted. The algorithms evaluated in this paper can also be applied to the detection of event mentions that contain the ACE events. Overall five sites participated in this task in 2005. The most similar work to that describe in this paper is detailed in (Ahn, 2006), who treats the task of finding all event triggers (used to identify each event) as a word classification task where the task is to classify every term in a document with a label defined by 34 classes. Features used included various lexical, WordNet, dependency and related entity features.

3 Corpora

The ACE 2005 Multilingual Corpus was annotated for Entities, Relations and Events. It consists of articles originating from six different sources including Newswire (20%), Broadcast News (20%), Broadcast Conversation (15%), Weblog (15%),

²An ACE Entity is an entity identified using guidelines outlined by ACE Entity Detection and Recognition task.

Usernet Newsgroups (15%) and Conversational Telephone Speech (15%). Statistics on the documents in this collection are presented in Table 1. We evaluate our methods on the following event types which have a high number of instances in the collection: “Die”, “Attack”, “Transport”, “Meet”, “Injure” and “Charge-indict”.

The data we use from the IBC database consists of Newswire articles gathered from 77 different news sources. Statistics describing this dataset are contained in Table 2. To obtain a gold standard set of annotations for articles in the IBC corpus, we asked ten volunteers to mark up all the “Die” event instances. To maintain consistency across both datasets, events in the IBC corpus were identified in a manner that conforms to the ACE annotation guidelines. In order to approximate the level of inter-annotation agreement achieved for the IBC corpus, two annotators were asked to annotate a disjoint set of 250 documents. Inter-rater agreements were calculated using the kappa statistic that was first proposed by (Cohen, 1960). Using the annotated data, a kappa score of 0.67 was obtained, indicating that while the task is difficult for humans the data is still useful for our training and test purposes. Discrepancies were adjudicated and resolved by an independent volunteer.

4 Event Detection as Classification

We treat the task of determining whether a given sentence describes an instance of the target event as a binary text classification task where it is assigned one of the following classes:

- **On-Event Sentence:** a sentence that contains one or more instances of the target event type.
- **Off-Event Sentence:** a sentence that does not contain any instances of the target event type.

4.1 A Machine Learning Approach

In an attempt to develop a gold standard approach for this task we use Support Vector Machines (SVM) to automatically classify each instance as either an “on-event” or “off-event” sentence. SVMs have been shown to be robust in

classification tasks involving text where the dimensionality is high (Joachims, 1998). Each sentence forms a train/test instance for our classifier and is encoded using the following set of features.

Terms: Stemmed terms with a frequency in the training data greater than 2 were used as a term feature. Stopwords were not used as term features.

Noun Chunks: All noun chunks (e.g. “american soldier”) with a frequency greater than 2 in the training data were also used as a feature.

Lexical Information: The presence or absence of each part of speech (POS) tag and chunk tag was used as a feature. We use the Maximum Entropy POS tagger and Chunker that are available with the C&C Toolkit (Curran et al., 2007). The POS Tagger uses the standard set of grammatical categories from the Penn Treebank and the chunker recognises the standard set of grammatical chunk tags: NP, VP, PP, ADJP, ADVP and so on.

Additional Features: We added the following additional features to the feature vector: sentence length, sentence position, presence/absence of negative terms (e.g. no, not, didn’t, don’t, isn’t, hasn’t), presence/absence of a modal terms (e.g. may, might, shall, should, must, will), a look-ahead feature that indicates whether the next sentence is an event sentence, a look-back feature indicating whether or not the previous sentence is an event sentence and the presence/absence of a time-stamp. Time-stamps were identified using in-house software developed by the Language Technology Group at the University of Melbourne³.

In the past, feature selection methods have been found to have a positive effect on classification accuracy of text classification tasks. To examine the effects of such techniques on this task, we use Information Gain (IG) to reduce the number of features used by the classifier by a factor of 2.

4.2 Language Modeling Approaches

The Language modeling approach presented here is based on Bayesian decision theory. Consider the situation where we wish to classify a sentence s_k into a category $c \in C = \{C_1 \dots C_{|C|}\}$. One approach is to choose the category that has the largest posterior probability given the training text:

$$c^* = \arg \max_{c \in C} \{Pr(c|s_k)\} \quad (1)$$

Specifically, we construct a language model $LM(c_i)$ for each class c_i . All models built

are unigram models that use a maximum likelihood estimator to approximate term probabilities. According to this model (built from sentences $\{s_1 \dots s_m\}$ belonging to class c_i in the training data) we can calculate the probability that term w was generated from class c_i as:

$$P(w|LM(c_i)) = \frac{tf(w, c_i)}{|c_i|} \quad (2)$$

where $tf(w, c_i)$ is the term frequency of term w in c_i (that is, $\{s_1 \dots s_m\}$) and $|c_i|$ is the total number of terms in class c_i . We make the usual assumptions that word co-occurrences are independent. As a result, the probability of a sentence is the product of the probabilities of its terms. We calculate the probability that a given test sentence s_k belongs to class c_i as follows:

$$P(s_k|LM(c_i)) = \prod_{w \in s_k} P(w|LM(c_i)) \quad (3)$$

However, this model will generally underestimate the probability of any unseen word in the sentence, that is terms that do not appear in the training data used to build the language model. To combat this, smoothing techniques are used to assign a non-zero probability to the unseen words, which improves the accuracy of the overall term probability estimation. Many smoothing methods have been proposed over the years, and in general, they work by discounting the probabilities of seen terms and assign this extra probability mass to unseen words. In IR, it has been found that the choice of smoothing method significantly affects retrieval performance (Zhai and Lafferty, 2001; Kraaij and Spitters, 2003). For this reason, we experiment with the Laplace, Jelinek-Mercer and Absolute Discounting Smoothing methods, and compare their effects on classification performance in Section 5.

For this classification task, we normalise all numeric references, locations, person names and organisations to “DIGIT”, “LOC”, “PER”, and “ORG” respectively. This helps to reduce the dimensionality of our models, and improve their classification accuracy, particular in cases where unseen instances of these entities occur in the test data.

4.3 Baseline Measures

We compare the performance our ML and LM approaches to the following plausible baseline systems: **Random** assigns each instance (sentence)

³<http://www.cs.mu.oz.au/research/lt/>

randomly to one of the possible classes. While **Majority Class Baseline** assigns each instance to the class that is most frequent in the training data. In our case, this is the “off-event” class.

According to the ACE annotation guidelines⁴ event instances are identified in the text by finding event triggers that explicitly mark the occurrence of the event. As a result, each event instance tagged in our datasets have a corresponding trigger that the annotators used to identify it. For example, terms like “killing”, “death” and “murder” are common triggers used to identify the “Die” event type. Therefore, we expect that a system that selects sentences containing one or more candidate trigger terms as positive “on-event” sentences for a given event type, would be a suitable baseline for this task. To investigate this further we add the following baseline system:

Manual Trigger-Based Classification: For each event type, we use WordNet to manually create a list of terms that are synonyms or hyponyms (is a type of) of the event type. For example, in the case of the “Meet” and “Die” events common trigger terms include {“encounter”, “visit”, “reunite”} and {“die”, “suicide”, “assassination”} respectively. We classify each sentence for a given event type as follows: if a sentence contains one or more terms in the trigger list for that event type then it is assigned to the “on-event” class for that type. Otherwise it is assigned to the “off-event” class. Table 3 contains the number of trigger terms used for each event⁵.

Table 3: Trigger term lists for the six event types used in the experiments.

Event Type	Number of triggers terms
Die	29
Transport	14
Meet	12
Injure	10
Charge-Indict	8
Attack	8

5 Evaluation Methodology & Results

A standard measure for classification performance is classification accuracy. However for corpora where the class distribution is skewed (as is the

⁴Available at <http://projects.ldc.upenn.edu/ace/annotation/>

⁵The lists of the trigger terms used for each event type are available at <http://inismor.ucd.ie/ventype/~martina/>

case in our datasets where approx. 90% of the instances belong to the “off-event” class) this measure can be misleading. So instead we have used precision, recall and F1 to evaluate each technique. If a is the number of sentences correctly classified by a system to class i , b is the total number of sentences classified to class i by a system, and c is the total number of human-annotated sentences in class i . Then the precision and recall for class i can be defined as follows: $Prec_i = \frac{a}{b}$, $Recall_i = \frac{a}{c}$. Finally, F1 (the harmonic mean between precision and recall) for class i is defined as $F1_i = \frac{2 \times Prec_i \times Recall_i}{Prec_i + Recall_i}$. In the results presented in this section we present the precision recall and F1 for each class as well as the overall accuracy score.

Results: In our experiments we use a relatively efficient implementation of an SVM called the Sequential Minimal Optimisation (SMO) algorithm (Platt, 1999) which is provided by the Weka framework (Witten and Frank, 2000). Results presented in this section are divided into two parts. In the first part, all results were obtained using the IBC dataset where the target event type is “Die”. We provide a more detailed comparison of the performance of each algorithm using this type as more data was available for it. In the second section, we examine the effectiveness of each approach for all six event types (listed in Section 3) using the ACE data. All reported scores were generated using 50:50 randomly selected train/test splits averaged over 5 runs.

As part of the first set of results Table 4 shows the precision, recall and F1 achieved for the “on-event” and “off-event” classes as well as the overall classification accuracy obtained by each approach. Two variations of the SVM were built. The first version (denoted in the table by SVM(All Features IG)) was built using all terms, nouns chunks, lexical and additional features to encode each train/test instance where the features were reduced using IG. In the second version, the same features were used but no feature reduction was carried out (denoted in the table by SVM(All Features)). LangModel(JM), LangModel(DS) and LangModel(LP) represent language models smoothed using Jelinek-Mercer, Discount Smoothing and LaPlace techniques respectively. Overall these results suggest that the SVM using IG for feature selection is the most effective method for correctly classifying both “on-event” and “off-event” sentences. Specifically, it

Table 4: % Precision, Recall and F1 for both classes as well as the classification accuracy achieved by all algorithms using a 50:50 train/test split where the target event type is “Die”.

Algorithm	On-Event Class			Off-Event Class			Accuracy
	Precision	Recall	F1	Precision	Recall	F1	
SVM(All Features IG)	90.61	89.87	90.23	96.15	97.26	96.70	94.60
SVM(All Features)	89.63	88.52	89.06	96.08	96.49	96.28	94.45
Trigger-Based Classification	83.10	93.34	87.92	97.25	93.24	95.20	93.09
LangModel(DS)	63.11	82.4	71.46	93.13	83.16	87.86	82.98
LangModel(JM)	59.46	86.01	70.31	94.22	79.53	86.25	81.22
LangModel(LP)	59.22	79.56	67.89	91.89	80.88	86.03	80.54
Majority Class (“off-event”)	0.0	0.0	0.0	74.50	100.00	85.38	74.17
Random	26.57	51.73	35.10	75.58	50.0	60.18	50.34

Table 5: % F1 for both classes achieved by the SVM using different combinations of features.

Features	F1(On-Event)	F1(Off-Event)
terms	89.52	96.43
terms + nc	89.58	96.31
terms + nc + lex	89.62	96.44
All Features	90.23	96.70

achieves 90.23% and 96.70% F1 score for these classes respectively. When IG is not used we see a marginal decrease of approx. 1% in these scores. The fact that both versions of the SVM obtain approx. 90% F1 scores for the “on-event” class is extremely encouraging when you consider the large skew in class distribution that is present here (i.e., the majority of training instances belong to the “off-event” class).

To examine the effects of the various features on overall performance, we evaluated the SVM using different feature combinations. These results are shown in Table 5 where “terms”, “nc”, and “lex” denote the terms, noun chunks and lexical feature sets respectively. “All Features” includes these features and the “Additional Features” described in Section 4.1. One obvious conclusion from this table is that terms alone prove to be the most valuable features for this task. Only a little increase in performance is achieved by adding the other feature sets.

The graphs in Figure 1 shows the % F1 of both classes achieved by all methods using varying levels of training data. From these graphs we see that the SVM obtains over 80% F1 for the “on-event” class and over 90% F1 for the “off-event” class when only 10% of the training data is used. These results increase gradually when the amount of training data increases. For levels of training data greater than 30% the SVM consistency achieves higher F1 scores for both classes than all other methods for this task.

In general, the language modeling based techniques are not as effective as the SVM approach for this classification task. However, from Table 4 we see that all language models achieve approx. 70% F1 for the “on-event” class and approx. 86% F1 for the “off-event” class when only 50% of the IBC data is used to build the models. This is encouraging since they require little or no feature engineering and less time to train. Models smoothed with the Laplace method tend to have the least impact out of the three model variations. This is due to the fact that this method assigns the same probability to all unseen terms. Thus, a term like “professor” that may only occur once in the dataset has the same likelihood of occurring in an “on-event” sentence as a term like “kill” that has a very high frequency in the dataset. In contrast, the Jelinek-Mercer and Absolute Discounting smoothing methods estimate the probability of unseen terms according to a background model built using the entire collection. Therefore, the probabilities assigned to unseen words is proportional to their global distribution in the entire corpus. Consequently, the probabilities assigned to unseen terms tend to be more reliable approximations of true term probabilities.

Overall, the trigger-based classification baseline approach performs very well achieving similar scores to the SVM. This suggests that selecting sentences with terms associated with the target event is an effective way of solving this problem. That said, it still makes mistakes that the SVM and language models have the ability to correct. For example, many sentences that contain terms like “suicide” and “killing” as part of a noun phase (e.g. “suicide driver” or “killing range”) do not report a death. The trigger classification baseline will classify these as an “on-event” instances whereas the SVM correctly places them in the “off-event” category. More interesting are the cases missed by

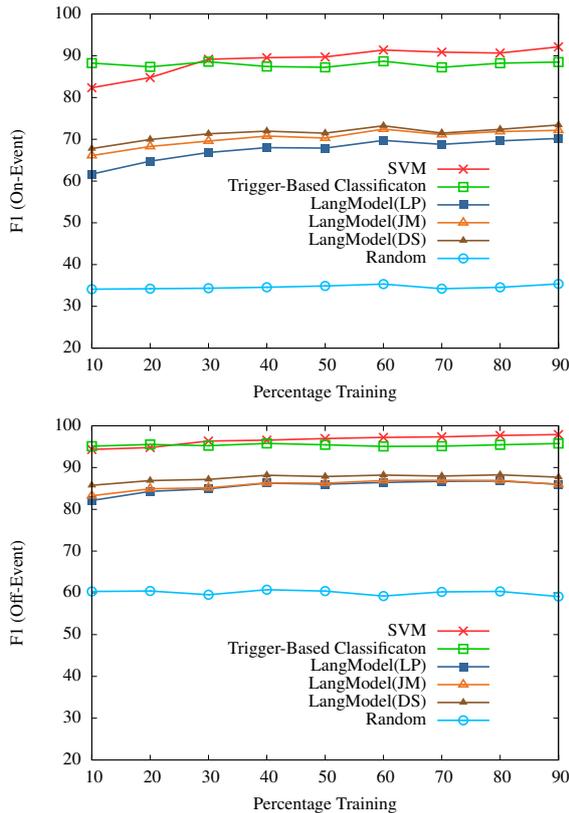


Figure 1: % F1 for the on-event (top) and off-event (bottom) classes for all methods using varying levels of training data where the target event is “Die”.

the trigger classification baseline and SVM that are corrected by the language models. These include sentences like “*Three bodies were found yesterday in central Baghdad.*” and “*If the Americans have killed them, then why dont they show the tape.*”. In fact, it turns out that over 50% of the errors produced by the SVM and manual trigger-based approach when the target event is “Die” are classified correctly by the language models. Although this is encouraging, the overall error rate of the language modeling approach is too high to rely on it alone. However, this evidence suggests that it may prove useful in the future to somehow combine the predictions of all three approaches in a way that improves overall classification performance.

We now move on to the second part of the experiments. Here, we present the results for six event types (as listed in Section 3) using only data from ACE corpus. Figure 2 shows the % F1 of the “on-event” class achieved by all approaches for each event type. We have omitted the % F1 scores for the “off-event” class as they do not vary significantly across event types and are similar to those reported in Table 4. The SVM, language mod-

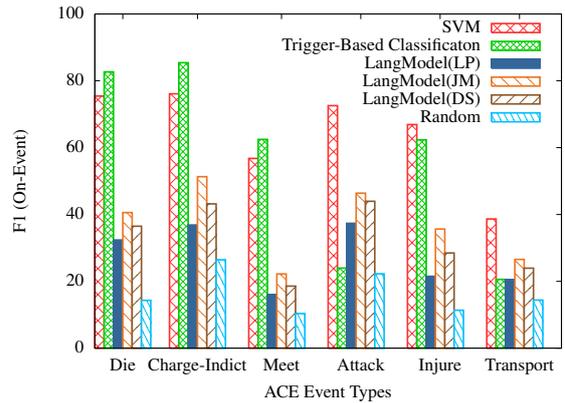


Figure 2: % F1 of the “on-event” class achieved by all methods for the six ACE event types.

els, trigger-based and random baselines achieve approx. 96%, 95%, 86% and 60% “off-event” F1 scores respectively across all event types.

On the other hand, Figure 2 demonstrates that the performance of each approach for the “on-event” class varies considerably across the event types. For instance, the trigger-based classification baseline out-performs all other approaches achieving over 60% F1 score for the “Meet”, “Die” and “Charge-Indict” types. However for events like “Attack” and “Transport” this baseline F1 score drops to approx. 20% thus achieving scores that are only marginally above the random baseline. Interestingly, we notice that although it performs better for events like “Meet” and “Charge-Indict”, the number of trigger terms used to detect these types is much smaller than the number used for the “Attack” and “Transport” types (see Table 3). This indicates that event types where this simple baseline performs well are those where the vocabulary used to describe them is small. Event types where it achieves poor results are broader types like “Transport” and “Attack” that cover a larger spectrum of event instances from heterogeneous contexts and situations. However, we see from Figure 2 that the SVM performs well on such event types and as a result out-performs the trigger-based selection process by approximately a factor of 4 for the “Attack” event and a factor of 2 for the “Transport” event.

When we compare both datasets we find that the ACE data is made up of newswire articles, Broadcast news, broadcast conversational texts, weblogs, usenet newsgroup texts and conversational telephone speech that has been transcribed, whereas the IBC corpus consists mainly of newswire arti-

cles reporting fatalities during the Iraqi War. As a result, event instances in the ACE data describing the “Die” event type are likely to report fatalities not only from Iraq but also from more diverse contexts and situations. To investigate how performance differs for the “Die” event type across these datasets, we compare the IBC results in Table 4 with the ACE results in Figure 2. We find that the F1 scores of the “off-event” class are not affected much. However, the F1 scores for the “on-event” class for the SVM and trigger-based baseline are reduced by margins of approx. 12% and 5% respectively. We also notice that the performance of the unigram language models are reduced significantly by a factor of 2 indicating that they struggle to approximate accurate term probabilities when the vocabulary is more diverse and the amount of training data is limited.

6 Discussion

Sentence level event classification is an important first step for many NLP applications such as QA and summarisation systems. For each event type used in our experiments we treated this as a binary classification task and compared a variety of approaches for identifying sentences that described instances of that type. The results showed that the trained SVM was more effective than the language modeling approaches across all event types. Another interesting contribution of this paper is that the trigger-based classification baseline performed better than expected. Specifically, for three of the six event types it out-performed the trained SVM. This suggests that although there are cases where such terms appear in sentences that do not describe instances of a given type (for instance, “*The boy was nearly killed.*”), these cases are in the minority. However, the success of this baseline is somewhat dependent on the nature of the event in question. For broader events like “Transport” and “Attack” where the trigger terms can be harder to predict, it performs quite poorly. Therefore, as part of future work, we hope to investigate ways of automating the creation of these term lists for a specified event type as this proved to be an effective approach to this task.

Acknowledgements. This research was supported by the Irish Research Council for Science, Engineering & Technology (IRCSET) and IBM under grant RS/2004/IBM/1. The authors also wishes to thank the members of the Language

Technology Research Group at the University of Melbourne and NICTA for their helpful discussions regarding this research.

References

- Ahn, David. 2006. The stages of event extraction. In *Proceedings of the ACL Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, Sydney, Australia, July.
- Allan, James, Jaime Carbonell, George Doddington, Jonathon Yamron, and Yiming Yang. 1998. Topic detection and tracking pilot study. final report.
- Cohen, Jacob. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.
- Curran, James, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale nlp with c & c and boxer. In *Proceedings of the ACL 2007 Demonstrations Session (ACL-07 demo)*, pages 29–32.
- Filatova, Elena and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In *Proceedings of ACL Workshop on Summarization*, pages 104 – 111.
- Joachims, Thorsten. 1998. Text categorization with support vector machines: learning with many relevant features. In Nédellec, Claire and Céline Rouveirol, editors, *Proceedings of the 10th ECML*, pages 137–142, Chemnitz, DE. Springer Verlag, Heidelberg, DE.
- Kraaij, Wessel and Martijn Spitters. 2003. Language models for topic tracking. In Croft, Bruce and John Lafferty, editors, *Language Models for Information Retrieval*. Kluwer Academic Publishers.
- Murff, Harvey, Vimla Patel, George Hripcsak, and David Bates. 2003. Detecting adverse events for patient safety research: a review of current methodologies. *Journal of Biomedical Informatics*, 36(1/2):131–143.
- Platt, John. 1999. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods: support vector learning*, pages 185–208.
- Saurí, Roser, Robert Knippen, Marc Verhagen, and James Pustejovsky. 2005. Evita: a robust event recognizer for qa systems. In *HLT*, pages 700–707.
- Walker, Christopher., Stephanie. Strassel, Julie Medero, and Linguistic Data Consortium. 2006. *ACE 2005 Multilingual Training Corpus*. Linguistic Data Consortium, University of Pennsylvania.
- Witten, Ian and Eibe Frank. 2000. *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann Publishers Inc.
- Yamron, JP, L. Gillick, P. van Mulbregt, and S. Knecht. 2002. Statistical models of topical content. *The Kluwer International Series on Information Retrieval*, pages 115–134.
- Zhai, Chengxiang and John Lafferty. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Research and Development in Information Retrieval*, pages 334–342.