KNOWLEDGE MEDIA

KMi

INSTITUTE

# Text Mining Methods for Event Recognition in Stories

**Tech Report kmi-05-2**
**April 2005**

**Kelly P. Vincent**

The Open University

# Text mining methods for event recognition in stories

Kelly P. Vincent

Knowledge Media Institute

The Open University

Milton Keynes, MK7 6AA, UK

kpvincent@hotmail.com

Navigating an online story collection requires a system which can make connections between the stories and their elements. One known way of accomplishing this is by annotating the stories, which can be a costly process. Finding methods for providing computer support for this process is a tactic for bringing the cost down. This paper describes several experiments which tested a variety of text mining methods for viability in accurately assisting the classification and annotation of stories in a small document collection. Latent Semantic Indexing is tested for dimension reduction, and decision trees, k nearest neighbor and naïve Bayes are all tested for classification. Additionally, both stemming and removing stopwords are tried. The study shows that all of these methods can be useful, but that there is great variability in their performance even within this small collection.

## 1. Background and previous work

Heritage and cultural institutions are some of the many organizations that are beginning to use the Web as a means of sharing digital artifacts such as stories (Mulholland, Collins, and Zdrahal, 2004). These stories include items like folklore or first-person accounts that pertain to the period or place of interest to the institution. One of the main functions of this kind of repository of stories is exploration and discovery: users should be able to discover new information when exploring the story collection. Providing computer support for these activities is not a trivial issue. In order to provide facilities for the exploration of these digital resources, the resources must be processed and organized in some meaningful way. This is where the greatest challenges for providing computer support lie and it is the focus of this project.

There are several very basic methods of navigating an online story space, which are familiar to any Web user. For instance, a list of story titles linked to the stories could be provided, perhaps with a summary. A slightly more flexible option would be to provide a keyword search facility. Although these methods do certainly allow users to find stories they might be interested in, they are too simple, and do not exploit the fact that the story text holds a great deal of information. Story titles and even summaries can sometimes be misleading, or simply not reveal all of the aspects of a story that a reader might find interesting. Keyword searches are known to be problematic for a variety of reasons, including that they often miss related documents or return irrelevant documents. If a user wants stories about financial institutions, entering a word like ‘bank’ into a keyword search would also pull up documents making reference to a river bank. It could also miss those about credit unions, or even documents about banks which fail to use the word ‘bank’, for instance by referring only to ‘banking’.

These issues must be dealt with in order to obtain improved navigation through a collection of stories. The documents should be annotated in some way. This may be as simple as using some method to identify terms that are semantically similar, such as finding all words that are related to the idea of financial institutions. Ideally, this annotation should allow the documents to be classified into various meaningful categories. The method for navigating the story collection would therefore be to make relevant connections between similar or related categories. The exact approach to this varies widely, but generally a taxonomy or ontology is used to structure the categories. The view taken in this project is that the stories can be regarded as being composed of different events that are each populated by various objects (Mulholland, Collins, and Zdrahal, 2004). Here, an ontology is used to describe the relationships between various events and objects. However, the best method to identify these events and objects is not clear, and both event recognition and object recognition are areas worthy of further

study. The following section will introduce some of the previous work that has been done in this area and discuss the various methods that can be used.

## 1.1. Annotating and classifying digital resources

There are two general classes of approaches to annotating and classifying digital resources: manual and automatic. The basic technique for both requires associating each resource with a semantically significant conceptual structure, which would in turn allow effective classification or searching. In the manual case, a knowledgeable person familiar with the domain would carry out the association of instances to certain classes, for example by generating basic heuristics or designing a complex knowledge model. This is a time-consuming and therefore expensive process. The automatic approach encompasses techniques that would be used to infer the semantics fully automatically. These might be Latent Semantic Indexing (LSI) or any of several statistical or machine learning techniques.

Historically, the successes in this area have been obtained primarily through manual approaches. Cheaper, automatic methods have been explored, but so far fail to achieve the quality results that manual methods can offer. As a consequence, many current solutions are hybrid approaches which strive to semi-automate the process, thus guaranteeing a certain level of quality while keeping the cost down. Current systems naturally vary in complexity, with the simplest systems usually being fully automatic, and the need for manual methods increasing as the complexity increases. This section will describe three main types of approaches, divided as follows: approaches which are based completely on the words in the text; approaches which are also based on the words but which allow for some hierarchical information; and complex approaches which assume the text has a conceptual structure.

### *1.1.1. Fully term-based approaches*

The simplest approaches are fully automatic and rely entirely on the words in the text, without assuming any conceptual or other structure. Several researchers are working on the problem of text classification by trying different statistical and machine learning techniques. Yang and Liu tested five different approaches: neural nets, support vector machines (SVM), naïve Bayes, k-nearest neighbors (kNN) and linear least-squares fit (LLSF) (Yang and Liu, 1999). They found that all of the methods perform similarly when there are many positive training examples, but that naïve Bayes and neural nets perform worse than the others when there are relatively few positive training examples. Apte, Damerau and Weis have used boosting and decision trees to carry out a similar task (1998). In their study, all of the methods were tested on a modified version of the Reuters-21578 collection, and the results are slightly better than reported breakeven points for other methods used on the collection, indicating that this is a method worth pursuing. Nigam, Lafferty and McCallum have used an approach based on maximum entropy techniques for text categorization, finding that it outperformed naïve Bayes in most cases (1999). This sample of studies shows that there is quite a bit of experimentation with such techniques in text classification. It is also clear that these techniques have something to offer, even if the best methods among them are not obvious.

### *1.1.2. Approaches assuming hierarchical term structure*

The second type of approaches to automatic annotation and classification is slightly more complex. It still focuses on the words in the text, but assumes that there is some sort of hierarchy of terms that the words follow. In many search problems, generating a list of keywords that appear in the document can be very informative. Feldman, Daga and Hirsh describe a system that analyzes keyword frequency distributions across several documents in the Reuters-22172 text collection (1998). They used a variety of methods for looking at keyword distributions, including a simple keyword hierarchy, and found that these could aid the discovery of several useful relationships. Collins, Mulholland and Watt explored the possibilities of using the concept of genre to support classification (2001). They used a corpus of departmental news stories written by various members of the department, finding that the stories tended to follow the inverted pyramid structure of traditional news stories, despite the fact that there was no enforced standardized structure. Their system used keywords, but focused on the location of these keywords in the document. Knowledge about the genre of the documents was therefore successfully exploited to better classify them. These approaches can overcome many of the weaknesses of the purely word-based approach. However, even more powerful knowledge modeling may be worthwhile.

### 1.1.3. Approaches assuming high-level conceptual structure

Finally, the third type of approaches assumes both that the data is more complex than simply the words in the text, and that there is a conceptual structure associated with the data. This structure is most commonly modeled with an ontology. Noy and McGuinness pinpoint five major benefits of using ontologies: a common understanding of knowledge structure can be shared; domain knowledge can be reused; domain assumptions can be made explicit; domain knowledge and operational knowledge can be separated; and domain knowledge can be analyzed (2001). Although some standard ontologies already exist, in domain-specific work it is still usually necessary to define the ontology. Traditionally this has been done by hand, using human experts, who can usually produce a quality ontology if sufficient time and money is invested. However, some progress has been made in automatically or semi-automatically extracting ontologies from data. Maedche and Volz used keyword frequency counts to create concepts, and then created an algorithm to carry out hierarchical clustering to actually generate an ontology (2001). Once a usable ontology exists, it can then be used to help populate the knowledge base. This again can be done manually or automatically.

Researchers have used a variety of different techniques to define and then populate an ontology. Vargas-Vera and Celjuska had some success using machine learning and information extraction techniques to identify events in stories, and then semi-automatically populate a specified ontology (2003). Their system used heuristics, partial parsing, a dictionary induction tool and a sentence analyzer tool to provide classification suggestions for a story to a human user. The user would then confirm or reject the suggestions, allowing the ontology to be generated over several iterations. This was found to work for the majority of events, but manual updates were required in cases where the story has implied information which is not explicitly stated, or if none of the heuristics apply.

Anani *et al.* describe a system that generates biographies of artists after automatically extracting knowledge from the Web and populating a knowledge base on artists (2003). It uses an ontology, coupled with WordNet (Fellbaum, 1998) and GATE (Cunningham *et al.*, 2002), to identify relevant documents on the Web. It then processes these documents by dividing them into sentences and phrases, finding grammatically related phrases and identifying named entities. The semantic information on both anticipated and required relations between the entities was inferred from the ontology and augmented by WordNet.

It is obvious that these approaches are the most complex, as they incorporate both an ontology and statistical or machine learning techniques. They also tend to have several distinct phases, such as one where the ontology is first constructed, then where it is populated to create the knowledge base, and finally where the knowledge base is used in some way.

### 1.1.4. Summary

All of the systems described in this section are exploring techniques that could one day be commonplace in various text classification systems. It is probable that these future systems will resemble those described in the previous paragraphs, since these attempts to exploit multiple sources of information have had some success. It will be interesting to see what further solutions text mining and knowledge modeling will offer to document and story classification.

## 1.2. Data and noise reduction

One aspect that becomes very important when dealing with this type of data is its high dimensionality, which can be both computationally very expensive and a source of noise. Statistics and machine learning techniques have been used to deal with these problems. These techniques are effectively clustering methods. The primary idea behind clustering in the text domain is the concept of grouping similar words into semantically-related clusters, rather than leaving them as distinct words. The notion is relatively simple, and there are several established clustering algorithms; it is deciding on a measure of similarity that is generally the most difficult task. Soft clustering allows words to be placed in more than one cluster, while hard clustering allows each word to be assigned to only one cluster. In natural language processing tasks, soft clustering is often preferred, because it is recognized that language is inherently ambiguous (Manning and Schütze, 2002). One very simple way to perform clustering is to use stemming. Stemming is basically the process of stripping off the affixes from words, effectively reducing words to their semantic class (Manning and Schütze, 2002). For instance, the words ‘laughing’, ‘laughs’ and ‘laugh’ would all reduce to the stem ‘laugh’. This can be useful, but the actual process of stemming is not always simple, as there can be ambiguous words such as ‘lying’, whose stem could be either ‘lie’ or ‘lay’.

There are other, more powerful methods for dealing with high-dimensional data. Two of the most commonly-used algorithms in information extraction currently are Latent Semantic Indexing (LSI) and Principal Components Analysis (PCA). Both have been applied to problems relating to text mining. The main idea behind LSI is a well-known information retrieval concept, co-occurrence, which exploits the situation where words that occur in the same document do so at a greater rate than chance (Manning and Schütze, 2002). LSI considers documents in a space of "latent" semantic dimensions, so the clustering similarity measure is by definition semantic. These dimensions correspond with the axes of greatest variation, so the first dimension explains the greatest variability, and the highest dimension explains what is not explained after all of the other dimensions have been subtracted out (Manning and Schütze, 2002). To actually perform the similarity judgment, LSI uses a common technique from matrix theory, the Singular Value Decomposition (SVD) (Han and Kamber, 2001). The SVD represents the frequency matrix in a lower dimension so that the distance between vectors is minimized.

LSI has been successfully applied to the problem of text classification. When first proposing the method, Deerwester *et al*. found that even in these early stages and despite some known weaknesses, it outperformed or equaled the performance of simple term matching (1990). Zelikovitz and Hirsh have proposed a method for using LSI to help in integrating background (unlabeled) text into a system (2001). In their study, LSI and the use of the background text did helped to improve the accuracy of classification during testing, showing that the level of noise in the data was reduced.

The second major approach, PCA, is closely related to LSI. It identifies the axes of the space to be projected onto by finding the top *n* orthogonal vectors, and combining attributes of the data to fit this space (Han and Kamber, 2001). The words will be clustered into *n* groups. Ke and Shaoping used an algorithm which combined concept indexing and PCA, succeeding in reducing the dimensionality of the data without losing accuracy (Ke and Shaoping, 2002). Clearly, dimension reduction is a technique that should always be considered when working with textual data.

### 1.3. Text mining methods

The main goal of annotating and classifying documents is to allow efficient information retrieval and extraction. Although ontologies are quite good for representing text, the expense required in maintaining them and populating knowledge bases means that they are not usually sufficient on their own. Consequently, numerous word-based text mining methods are being used to counter the cost and even improve the success in this area. Several of the various statistical and machine learning methods used for this type of task will be described in the next several paragraphs.

In text mining, the most basic ingredient in all of the statistical- or machine learning-based methods is the words themselves, which make up the text (i.e. the raw data). When using such methods, the most common way of representing a text is the bag of words approach, traditional in information retrieval. This approach disregards word order, but allows words to be repeated so that frequency information is maintained. The text is generally represented as a feature vector, with each possible word representing a single dimension of the vector, which leads to very high-dimensional vectors. Techniques like PCA and LSI can be used in these instances.

This type of representation often calls for normalization, or smoothing. When working with keywords or words in general, one of the most important features of the set of words is the frequency of certain words. If a word is not present in the set, then there is no knowledge available about that particular word. If the word is truly unrelated, that may be a reasonable scenario, but in many cases it is a misleading representation of the document. The same can be true of words that do appear, but with very low frequencies. The basic approach to many smoothing algorithms is to discount the frequency counts (or probabilities) of more-frequent words, and redistribute the remainder, so that even non-present words will be given a non-zero frequency or probability. One very common way to normalize for very frequent and not useful words is by weighting the word frequency counts. When there is a collection of documents, one of the most popular methods is TFIDF (term frequency - inverse document frequency) weighting (Manning and Schütze, 2002). This involves multiplying the frequency value for a given word by the log of the quotient after dividing the total number of documents by the number of documents containing the word (Manning and Schütze, 2002).

Although pre-processing tasks are certainly important, the most useful text mining methods are the machine learning techniques. Some of the most basic methods include k nearest neighbor (kNN) and regression. kNN involves measuring the Euclidean distance between a given item and comparing it to each of the others. It assigns the same class to a given instance as the most frequently occurring class of

the *k* items that have the smallest distance from the instance (Mitchell, 1997). It is possible to apply distance weighting to this measure when *k* is greater than 1, to handle outliers. LLSF is a particular approach to multivariate regression, which involves performing the linear least-squares fit on the training data to establish a matrix representing the word-class regression coefficients (Yang and Liu, 1999). New items can then be classified by ranking and thresholding on the category weights.

Some of the more advanced methods include the naïve Bayes classifier, decision trees and neural nets. Using the naïve Bayes classifier involves establishing a hypothesis based on the overall probability of each class combined with the probability of each class given each term from the training data. This hypothesis can then be used to classify new instances. The decision tree approach involves constructing a tree with a divide-and-conquer method, and then utilizing it to identify classification rules, which are then used to classify new instances (Han and Kamber, 2001). The tree is constructed by placing all of the training data into a top node and branching nodes off this one. Information gain (an entropy-based metric) is used to find an attribute to best separate the data in a given node into different nodes (Han and Kamber, 2001). If a node contains items all of the same class, it is terminated and considered a "leaf".

Artificial neural networks are a general class of algorithms, with backpropagation and SOM commonly used. The term "neural net" often refers to the backpropagation approach, whereas the other approaches are referred to more specifically by name. Backpropagation involves constructing a network with one input layer and any number of weighted output layers, where the last is the final output layer and all others are considered hidden (Han and Kamber, 2001). The training items are provided to the network one-by-one and the weights are adjusted (backwards, from the final output layer through to the first hidden layer) to minimize the distance between the network's predicted class for the item and the item's actual class. With SOM, the weights of the unit closest to a training example, and its nearest neighbors in the network, are adjusted for each example, allowing a feature map to be constructed (Han and Kamber, 2001). This method assumes that there is some topological structure to the data, which can be iteratively discovered. SOM is effectively another clustering method, and can also be used for dimension reduction.

It is clear that there are very many possible directions for any exploration of text classification. Combining these methods with an ontology appears to offer the most promising solutions, but this requires choosing the right combinations. Currently it seems that the best way to discover the best approaches for particular data sets is by testing several of the methods. Clearly there is a need for a great deal of further research in this area, as ultimately it would be ideal to be able to predict the best method without having to test so many of them.

### 1.4. Project focus

The primary goal of this project was to find a viable approach to implement in a system that will assist the human annotators of a set of stories, which will be implemented at a later date. However, an important secondary goal of the project was to generalize these findings, to better understand some of the text mining approaches that are being used in current systems. For both of these reasons, several different text mining methods were explored. Although the particular domain in this case was that of a heritage center's story collection, it is believed that the findings are general enough to apply to other domains.

## 2. Experimental data

The dataset was composed of 64 documents or stories relating to Bletchley Park, which were written by tour guides working at the Bletchley Park Heritage Centre. Bletchley Park was Britain's codebreaking center during World War II, known as 'Station X' during the war. It is also well known for the people who worked there, like Alan Turing, and for the fact that the first programmable computer was conceptualized there (Bletchley Park – Station X). The tour guides working there are interested in various topics related to Bletchley Park, and often do independent research. In order to share their findings with the other guides, they enter "stories" on their Guides' Web site. The Guides' site is not publicly viewable, but a couple of screen shots are included below, in Figures 2.1 and 2.2. Their research often involves interviewing people who lived or worked at or near Bletchley Park around the time of the war, and consequently the majority of the stories entered are interviews. There is no enforced structure on the stories, and occasionally other personal accounts or news items are shared, however, only the interviews were used in this project. The stories were stored in several formats, including raw text and HTML.


**Figure 2.1. The Front Page of the Guides' Site**

BLETCHLEY PARK
Tour Guides - Interviews

## BLETCHLEY PARK STABLE YARD- BUDD FAMILY RECOLLECTIONS

Tuesday, 02 December 2003

**Report of Interview with Jean Cheshire (nee Budd) in Jan 2002**

Posted by: John Pettifer

Guides will recall that Mrs Jean Cheshire (nee Budd) wrote to BP in June 2001 to correct some statements made by a guide concerning the Stable Yard. during the War she had lived in cottage number 2.

Sometime later John Pettifer took up Jean's offer of a chat. He visited her at her home in Luton on 8 January 2002, and these notes convey the outcome. She also gave him some written recollections made by her, by her sister Faye and by her brother Neville. These are in now the BP archives.

The Budd Family.

Jean's father (Mr R G Budd) served in the Royal Navy (a photo shows him as a Chief Petty Officer) until 1939; there were four children: Robert born 1925 (died 1999), twin sisters Jean and Faye born 1934 and Neville born 1938 (and named after Neville Chamberlain!).

In 1938 the family lived in Portsmouth. Jean remembers going to stay (together with Faye) with "Aunty Etty" at BP when her mother was giving birth to Neville. Aunty Etty was married to Mr Harry Hymers, who was a friend of her father's from the Navy. At that time (1938) the Hymers lived in cottage number 2, and Jean remembers an elderly couple – Mr and Mrs Sanders living at number 1. Mrs Sanders gave the twins a new laid egg from her chickens and they had half each for breakfast!

Jean has been in touch with the Hymers' daughter Winnie; she is now Mrs Ribchester and lives in London. Jean believes that the Hymers moved out of cottage number 2 in 1940 and the Budd family moved in then. Mr Hymers was Mr Budd's immediate boss at BP and Jean thinks he probably got him the job there. [Comment: in fact the Budds' move to BP was probably very late in 1940 (or even in 1941) as Jean definitely does not recall the bomb falling by the back gate.]

Jean is not sure exactly what her father's job was, but thinks it was to do with transport. She said his office was in the building (now demolished) between the military vehicles and the car park used by staff at weekends. She also remembers him being in charge of stores of furniture at various villages including Mursley and Drayton Parslow, but this may have been after the War. He continued working at BP until 1950; after this he ran the Plough Inn at Simpson for a while but later worked for the Diplomatic Wireless Service at Hanslope Park running the stores. This included a two-year period in Singapore.

Lodgers

Jean remembers two ATS drivers who lodged with her family – Madge and Ann, and has been in touch with Ann (a Scot and now Mrs Witherbed) recently. Presumably this is the same lady whose interview is set out on page 58 of Other People's Stories, Book 2, and whose name is given as Graham. However, the notice alongside the patchwork in the mansion says Ann's surname was MacDonald. The different names were mentioned to Jean, but she was not sure which was correct. Jean does not recall Ann returning later as a civilian under her married name (as per page 59 of Other People's Stories, Book 2).

Jean also remembers a lady staying for a while who had a gentleman caller in the evenings! He came in through the kitchen and said "goodnight" to the kids!

Neighbours

The neighbours Jean recalls were:

At Cottage 1 – elderly couple Mr & Mrs Sanders, then empty for a while – then Mr & Mrs Saunders with children Diane, Janet and a boy - then Mr and Mrs Arkell, two girls (not twins). Jean recently met the Arkell's daughter Betty by chance at BP. She is now Mrs Simes and lives in Egham.

At Cottage 3 – a lady – then Mr & Mrs Jenkins with children Jane & John – then (after Mr Jenkins died) a chef from the canteen with children Roger & Wendy.

Later, in the "Bungalow" opposite, Commander Bradshaw.

After the War, a small flat under the clock tower was renovated for Bob Watson's son when he married.

Jean does not think there were any code-breakers working in Cottage 3 at any time while the Budds lived in number 2. However, she thinks it is just possible that there were some in Cottage 1 in the period between the Sanders and the Saunders. The "turret room" – sometimes said to be where Alan Turing worked, was in fact the stairs and landing of the Budds' cottage.

Jean does not recall any other names and is sure that there was no other pair of twins living in the stable yard. [Comment: this casts doubt on the story in "BP Walk" (Nov 2001 edition) about a naval lieutenant with twin daughters being in cottage number 1]

**Figure 2.2. A Story Example from the Guides' Site**

Each of the stories had between one and five event types associated with it, corresponding with events occurring within the story. There were thirteen different event types in total, which are shown in Table 2.1. The distribution of the events in the stories was quite varied. The most frequent event type, "Bletchley Park Life Experience", was found in 52 of the 64 stories in the database. The others are much less numerous, with the next most frequent appearing in 20 stories. Five event types were found in only one or two stories. Figure 2.3 shows the number of documents assigned to each event type.

| | **Event Type** |
|---|---|
| 1 | Attribute-Assignment |
| 2 | Battle |
| 3 | Birth |
| 4 | Bletchley Park Life Experience |
| 5 | Conceptual Creation |
| 6 | Dissolution |
| 7 | Experiment |
| 8 | Interview |
| 9 | Meeting |
| 10 | Modification |
| 11 | Move |
| 12 | Production |
| 13 | War Life Experience |

**Table 2.1. Event Types**

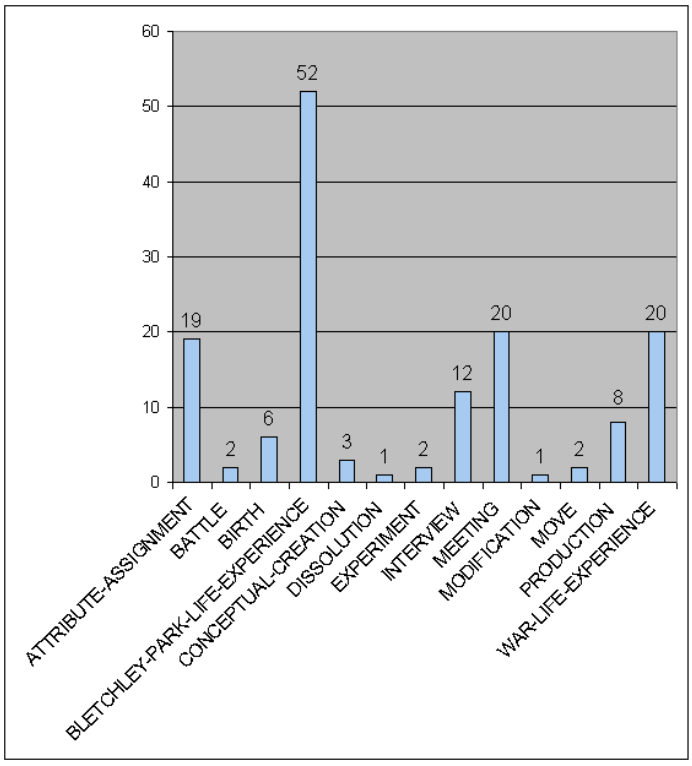


**Figure 2.3. Number of Documents Assigned to Each Event Type**

The figure reveals that there are possibly three distinct bands in terms of document frequency in the database, corresponding to event types with high frequency, medium frequency, and low frequency. The high and medium frequency event types probably would be handled reasonably well by the machine learning methods. It was expected that those of low frequency would probably not have sufficient training examples, and consequently the methods would not do well for these. However, the threshold between low and medium was not clear.

# 3. Method

The intention of this project was to try several different methods, in combination with each other, to discover the best approach for later implementation of a semi-automated annotation and classification system. In order to facilitate the testing of many different algorithms in such a short time as allowed for this project, the Weka machine learning toolkit was used (Witten and Frank, 2000). Much of the data preparation was done using Perl or Python, but all of the classifier tests were done using the Weka Explorer interface. This interface allows for easy set up of experiments with multiple files.

## 3.1. Preparing the data sets

Because the term frequency matrices were expected to be quite large, it was assumed that dimension reduction techniques should be used. LSI and PCA were selected and were to be implemented in Matlab. However, due to problems with memory limitations, it proved impossible to implement PCA, and LSI was only implemented for 75% of the data sets. Ultimately, the largest matrix in this data set was 64 x 5757, and the smallest 64 x 4265. Although these are large matrices, it is believed that it would be possible to implement both LSI and PCA on a machine with more memory. After dimension reduction, decision trees, k nearest neighbor (kNN) and neural nets (backpropagation) were to be used for classification of event type. If there was sufficient time, other methods would be tried. Ultimately, several decision trees, kNN and naïve Bayes were implemented, but backpropagation proved too time-consuming for this limited project time frame.

These methods were to be tested on a total of 21 different data sets. There were twelve sets of "Raw" data, and nine of "LSI" data. The twelve Raw sets were composed as follows: the term frequency (TF), term frequency-inverse document frequency (TFIDF), and normalized version of the TFIDF matrices would each be generated for four different sets of terms (the full set of original terms, the stemmed terms, the stoplisted terms, and the stemmed and stoplisted terms). LSI was then carried out on these, but failed on the full terms, as the full term matrix was too large and the function output a matrix with error values. Consequently, there were only nine LSI data sets. Regardless, it was possible to compare the performance of methods on quite a few different dimensions.

All of the experiments required data in the form of term frequency matrices. A script already existed for generating three versions of them: TF, TFIDF, and normalized TFIDF. The original version of this script generated these matrices for the stemmed and stoplisted version of the terms. This script was modified in order to generate four different versions of each type of matrix: one with the full set of original terms as-is, another with the terms stemmed, another with the terms stoplisted, and a final one with the terms both stemmed and stoplisted. In all cases, the stemming was carried out using the Porter stemmer (Porter, 1980), and the stoplist used was from the DVL/Verity Stop Word List (Defense Tech. Inf. Ctr., 2000). These were used to generate the matrices for all of the base test sets.

Once the term frequency matrices had been generated, it was possible to run LSI on each of them. Both LSI and PCA were first tried in Matlab, but proved impossible to implement in this way, as there was not enough memory on the PC that was being used for this project. In the case of LSI, the problem was with transposing the matrix *V*, a necessary step in reconstructing the simplified matrix. With PCA, there was not enough memory to produce the singular value decomposition of the covariance matrix. Ultimately, LSI was implemented in Perl using the PDL (Perl Data Language) (Schwebel *et al.*, 2004). However, the full term matrix was still too big, and it was not possible to generate LSI versions of it. This same approach was tried with PCA, but the covariance matrix still proved problematic, and working with the matrix shut down the Perl interpreter. Consequently, the only data reduction used in this study was LSI. However, LSI was used on the majority of the data sets, and generated files in identical format to the term frequency matrix files.

Before any tests could actually be carried out, several more tasks had to be done for each data file. First, the files had to be converted to Weka's required format, ARFF. A script was written to do this. Then each document had to be labeled with its correct event type(s). A list of the documents and their

associated types was provided by another researcher, and a script was written that appended this information in the relevant files. Because multiple event types were allowed for each story, it would be necessary to learn and test for each event type individually, rather than simply learning or choosing one event type for a given story. This meant that several "binary" files had to be created for each data file. Specifically, there were 13 files for each data file, each corresponding to one event type, with the event class labeled for each document as either True (the document has that event type) or False (the document does not have that event type). Then, certain types of tests (specifically ID3) require nominal data, so it was also necessary create additional files with the numerical feature vector values to binary (nominal) values, which corresponded to zero (False) or greater than zero (True).

### 3.2. Testing several classifiers

Tests were carried out on several different classifiers, including k nearest neighbor, decision trees, and naïve Bayes. In all cases, 10-fold cross-validation was the primary test method, and ten runs were carried out. The data files used for all of these tests were the same between tests, with the exception of the ID3 decision tree. ID3 requires nominal values only, so converted files (as described above) were used.

The first step in running the actual experiments was to set up each experiment in Weka's Experimenter interface. This involved creating a new "experiment" file for the particular combination of the following: machine learning algorithm (kNN for each *k*, ID3, C4.5, RandomTree, or Naïve Bayes); data type (Raw or LSI); TF matrix type (TF, TFIDF, or Normalized TFIDF); and term set type (Full terms, Stemmed, Stoplisted, or Stemmed and Stoplisted). These were given informative names so they were distinguishable by filename. Then, for each experiment, an output file with a name corresponding to the experiment filename was specified. The next step was to add the appropriate data files to the experiment setup. For all of the initial experiments, this involved adding 13 files, one for each event type of the corresponding data set (data type, TF matrix type, and term set type). Once the files were appropriately named and the data files added to the experiment file, the actual machine learning methods could be added and specified.

Most of the machine learning methods had parameters that had to be specified. Four kNN tests were set up first. These were chosen in an attempt to isolate the most appropriate value for *k*. Four initial values of *k* were tested: 1, 2, 3, and 5. In these cases, no distance weighting was employed. Three decision tree algorithms were tried: ID3, C4.5, and Weka's "Random Tree". There were no settings that had to be handled for ID3. However, Weka's C4.5 algorithm, J48, required several parameters to be set. For the initial runs, the following values were used: pruning confidence factor = 0.25; minimum number of instances per leaf = 2; C4.5 error pruning = on; unpruned tree = true. The third decision tree to be tried was the RandomTree, which constructs an unpruned tree with *n* randomly selected attributes at each node in the tree. By default, there is 1 attribute at each node, and the minimum total weight of the instances in a leaf is 1.0. Finally, the naïve Bayes classifier was tested. By default, this classifier uses a normal distribution for estimating numeric values, and this was unchanged.

Running the experiments was then as simple as clicking the Run button in the interface. However, these were CPU-demanding processes, and it took several days to run approximately 150 experiments. After all of the experiments had been run, a results script was written to help accumulate all of the data into a readable format. This script took the mean of the 10 runs of each approach, to produce a single vector representing the particular approach.

The best approaches were initially identified by the highest F-measures and percent correct, and the lowest number of false negatives and number of false positives. There were seven event types for which reasonable results were obtained, corresponding to those appearing in six or more documents. The F-measure is a combination of the recall and precision metrics, and is generally regarded as the best measure of the success of this type of task. The three measures are calculated as follows, where 'tp' is the number of true positives, 'fp' the number of false positives, and 'fn' the number of false negatives:

$$Recall = \frac{tp}{tp + fn};$$

$$Precision = \frac{tp}{tp + fp};$$

$$FMeasure = 2 * \frac{Recall * Precision}{Recall + Precision}.$$

However, for this particular task, it was considered desirable to explicitly consider the percent correct (the accuracy) and change the emphasis on the false negatives and positives. False positives were regarded as less important than the other three measures, because in the particular application that will be implemented, it is more important for the system to not fail to suggest the correct class than it is to minimize the number of incorrect classes it suggests. The number of false negatives clearly should not be too high, or the system is missing the correct assignment. Accuracy itself is not particularly useful as a metric, but it can be useful in comparing two algorithms that otherwise perform similarly. Consequently, another metric, "Algorithm Score", was devised. This metric was calculated as follows:

$$AlgorithmScore = 4 * \frac{(1.5*FMeasure) * Accuracy * 0.5 * (1 - fp / test\ set) * (1 - fn / test\ set)}{(1.5*FMeasure) + Accuracy + 0.5 * (1 - fp / test\ set) + (1 - fn / test\ set)}.$$

This value will always be between 0 and 0.75, and lower than the F-measure. It should be clear from the formula that the F-measure was emphasized and the number of false positives deemphasized. The F-measure was regarded as the most important measure, so was emphasized. Accuracy and the proportion of false negatives were somewhat important, and were explicitly considered with this formula.

The algorithm score was then used to rank the top five approaches for each of the seven best event types. Then each of these approaches were run several more times, with various parameters of the methods being changed. This fine-tuning was intended to see if any improvements could be made on the best approaches. Ten runs were again carried out on each, and the mean of these taken. Table 3.1 outlines the changes that were made in each case.

| Method | Changes |
|---|---|
| kNN | Distance weight = 1 / distance |
| | Distance weight = 1 - distance |
| ID3 | None possible |
| C4.5 | Minimum number of instances per leaf = 1 |
| | Confidence factor = 0.5 |
| | Confidence factor = 0.75 |
| | Confidence factor = 2.0 |
| | Use Laplace pruning = True |
| RandomTree | Attributes in each node = 2,<br>Minimum weight of instances in node = 1,<br>Seed for random number generator = 1 |
| | Attributes = 3, Minimum weight = 1, Seed = 1 |
| | Attributes = 1, Minimum weight = 1.5, Seed = 1 |
| | Attributes = 1, Minimum weight = 2, Seed = 1 |
| | Attributes = 1, Minimum weight = 1, Seed = 2 |
| | Attributes = 1, Minimum weight = 1, Seed = 3 |
| | Attributes = 2, Minimum weight = 1.5, Seed = 3 |
| | Attributes = 1, Minimum weight = 0.5, Seed = 1 |
| Naïve Bayes | Use kernel estimator |

**Table 3.1. Changes made for each machine learning approach**

## 4. Results and discussion

This section displays a summary of the major results of the experiments, and discusses their implications. The first part of this section deals with the results for each event type, used in determining the best approach for implementation in the future semi-automatic annotation system. The second part of the section deals with the more generalizable findings.

There was great variability in the performance of the machine learning methods in classifying the different event types. Figure 4.1 and Figure 4.2 show the spread of the mean algorithm score for each method and data set combination, by event type and number of documents with a given document type, respectively. Figure 4.1 makes it clear that the approaches had varying degrees of success in identifying the different event types. The event type that had the most success was Bletchley Park Life Experience, with several values approaching 0.60, and a median value well over 0.40. The only values around zero for this event were outliers. Several other events had some success, but all of the remaining ones had a lower quartile which reached zero, indicating that at least some of the approaches did not do well at all. However, four other event types had non-zero medians, and upper quartiles or outliers that reached 0.30 or higher. These were Attribute Assignment, Interview, Meeting and War-Life Experience. Finally, the Birth and Production event types had zero median scores, but had outliers that did relatively well.

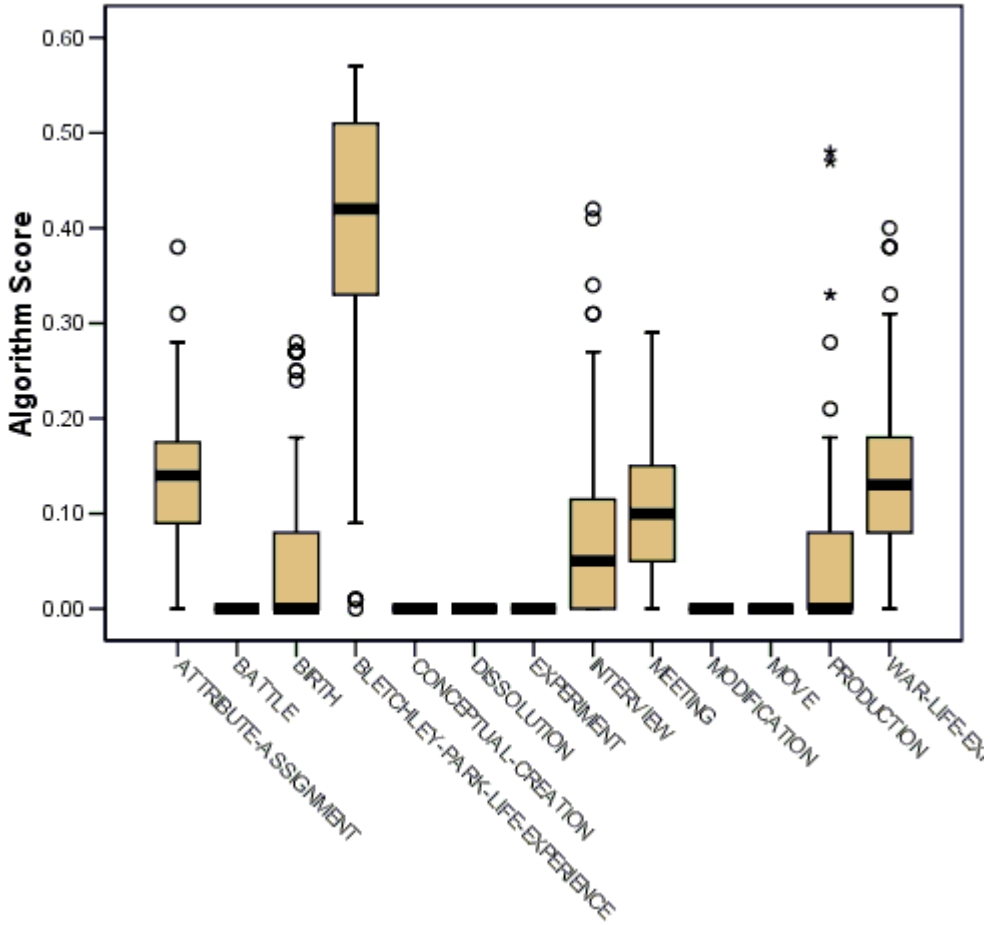


**Figure 4.1. Boxplot of Algorithm Score values for all algorithm and data set combinations, by event type**

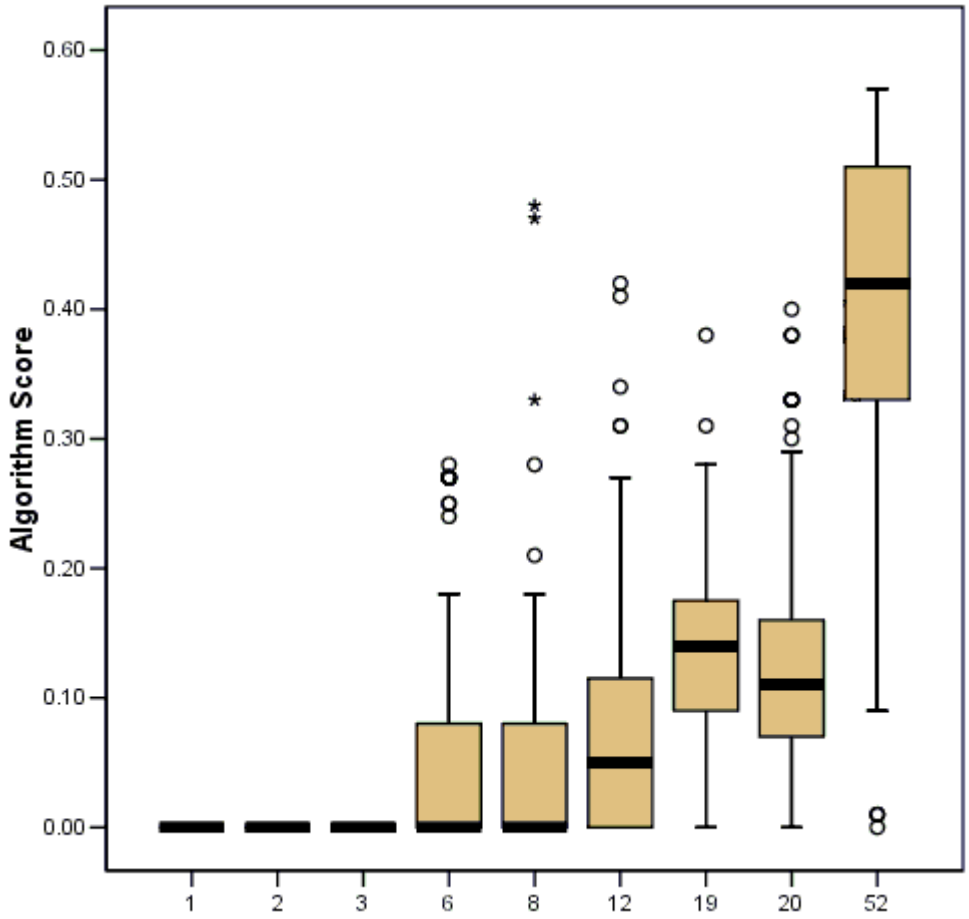


**Figure 4.2. Boxplot of Algorithm Score values for all algorithm and data set combinations, by count of documents for event type**

These findings are more sensible in the context of Figure 4.2, which shows that the performance of the methods overall improves as the number of documents associated with the event increases. This makes sense, as the various methods are supplied with more training data in order to learn the patterns to look for during classification. This figure also provides some support for the idea of three bands in document counts. It is fairly clear that having 52 documents is probably much better than only 20, or less. Although there is greater variability, the median algorithm score for this count is much higher than any of the other counts. It appears that the threshold between high frequency and medium frequency falls somewhere between 20 and 52. The chart clearly shows that having only 1, 2 or 3 documents is insufficient, and these fall in the low frequency category. The medium band itself appears to cover counts down to six or less, with the threshold being between 3 and 6.

### 4.1. The best approaches for particular events

Although Figure 4.1 does not provide any information about which approaches are best for each event type, it does make it clear that some of the event types will need to be dealt with in an entirely different manner. The events Battle, Conceptual Creation, Dissolution, Experiment, Modification, and Move are all found in 3 or fewer stories, and have mean algorithm scores that are all clustered around 0. However, the remaining 7 event types appear to have enough successful approaches that it would be worthwhile to look more closely at them. These will each be inspected individually. However, in each case, the best values were obtained by inspecting and ranking the algorithm scores of all approaches used. Table 4.1 shows a summary of the best approach for each event type.

| Event Type | Machine Learning Method | Data Set |
|---|---|---|
| Attribute-Assignment | KNN, *k = 1* | LSI, stemmed normalized TFIDF |
| Birth | KNN, *k = 2* | LSI, stemmed TF |
| Bletchley Park Life Experience | KNN, *k = 3* | Raw, stemmed and stoplisted TF, TFIDF or normalized TFIDF |
| Interview | C4.5 | LSI, stoplisted TFIDF |
| Meeting | RandomTree | Raw, full terms TF |
| Production | C4.5 | LSI, stemmed TFIDF |
| War Life Experience | Naïve Bayes | LSI, stemmed TF |

**Table 4.1. The Best Approach for Each Event Type**

### *4.1.1. Attribute Assignment*

Figure 4.3 shows the algorithm scores for the best 15 combinations for the Attribute Assignment event type. The figure makes it clear that the best two machine learning algorithms for this particular event type were kNN and naïve Bayes. It appears that for the kNN method, the LSI data is superior to the raw data. However, for the naïve Bayes case, both the raw and LSI data sets can perform well. For the kNN method, it appears that stemming and using a stoplist are both useful, individually and together. For the naïve Bayes classifier, five of the seven instances in the top 15 use a stoplist, although the two other instances (both using the full term set) outperform these. The type of term set does not appear to be very important in this case, as all three types appear four to six times. Although several did well, the best combination for the Attribute Assignment event appears to be kNN with $k = 1$, using the LSI version of the normalized TFIDF representation of the stemmed terms. The mean algorithm score for this combination was 0.38. Attempts at improving this score by changing the algorithm parameters failed, so the default settings are sufficient.

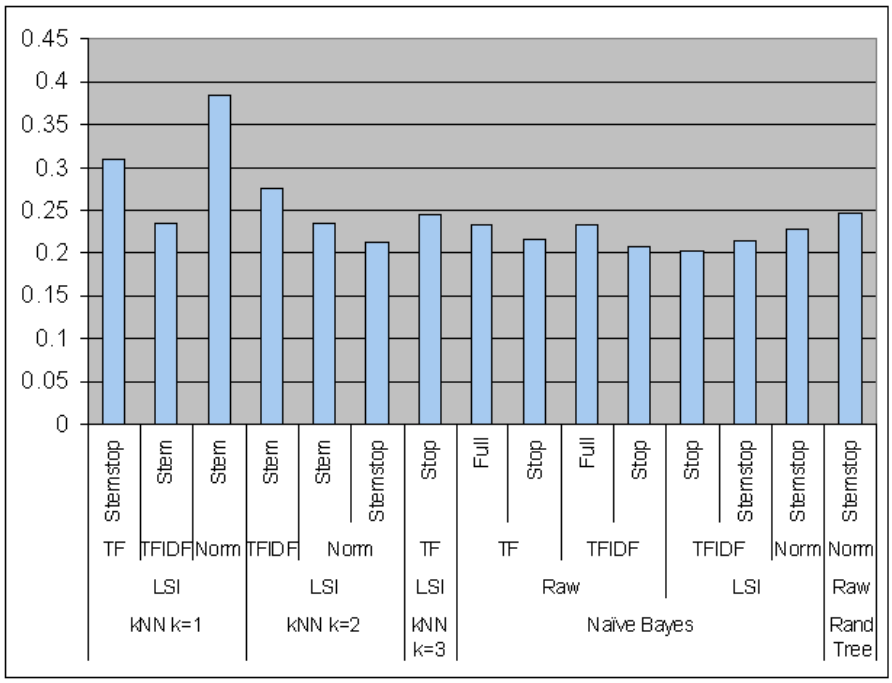


**Figure 4.3. Top 15 Algorithm Scores for Attribute Assignment**

### *4.1.2. Birth*

The algorithm scores for the top 15 combinations for the Birth event type can be seen in Figure 4.4. Like for Attribute Assignment, naïve Bayes and kNN dominate. The naïve Bayes results are particularly interesting, because they show that the type of term frequency matrix is irrelevant in this case, as is the choice of term set modification. Although the full term set does well, it is not as

successful as stemming, using a stoplist, and doing both, which all perform equally well. This is a clear instance where removing some of the data actually improves performance, showing that both stemming and stoplisting remove data which amounts to noise for this event type. However, despite the fact that naïve Bayes does very well, the best combination again involves the kNN algorithm. In this case, $k = 2$, the data is again stemmed, and the LSI version of the TF matrix is used. Like for Attribute Assignment, the results could not be improved by tweaking the algorithm parameters.

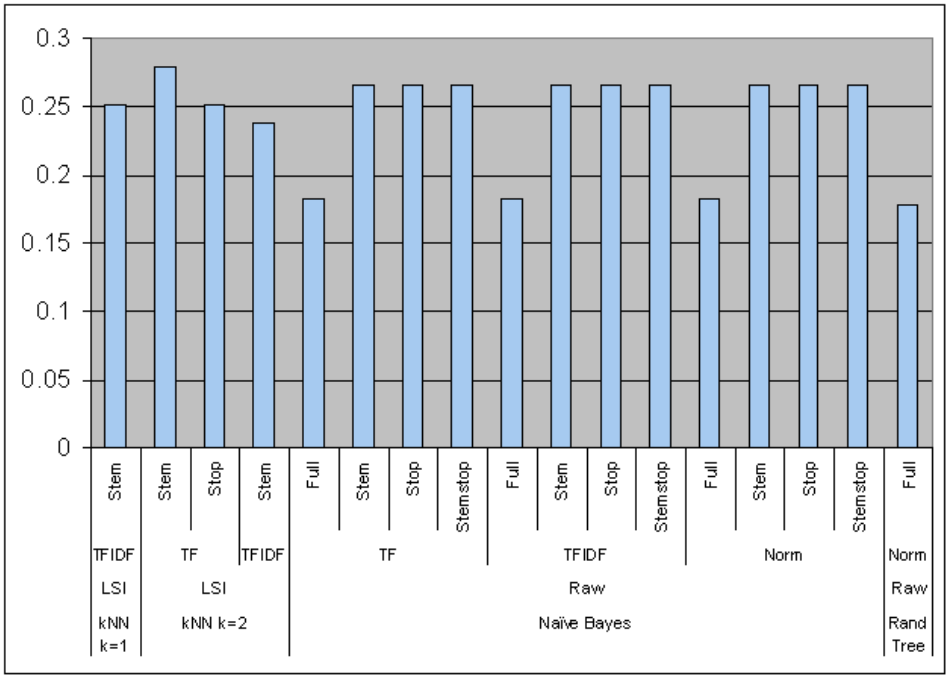


**Figure 4.4. Top 15 Algorithm Scores for Birth**

### *4.1.3. Bletchley Park Life Experience*

Like both previous event types, the two best algorithms for Bletchley Park Life Experience are naïve Bayes and kNN. The top results can be seen in Figure 4.5. However, in this case, kNN with $k = 3$ clearly outperforms naïve Bayes. The algorithm scores for this method for the corresponding term sets are the same, regardless of the term frequency representation used. Stemming and stoplisting are both good, but doing both is better. Interestingly, these all do much better than the full terms, which actually has one of the lowest scores among all of the methods. This is further evidence that stemming and stoplisting benefit the procedure by removing noisy data. For the naïve Bayes approach, the full terms do still produce good results, as do the stoplisted terms. What is noise to the kNN approach appears to be better handled by naïve Bayes, but then to bring down the overall performance. But again, the term frequency matrix used does not appear to affect the results. Ultimately, the best method is to use kNN with $k = 3$ on the raw, stemmed and stoplisted data. Any term frequency matrix could be used. In this case, the algorithm score is 0.53. Again, this score could not be improved with any algorithm parameter adjustments.

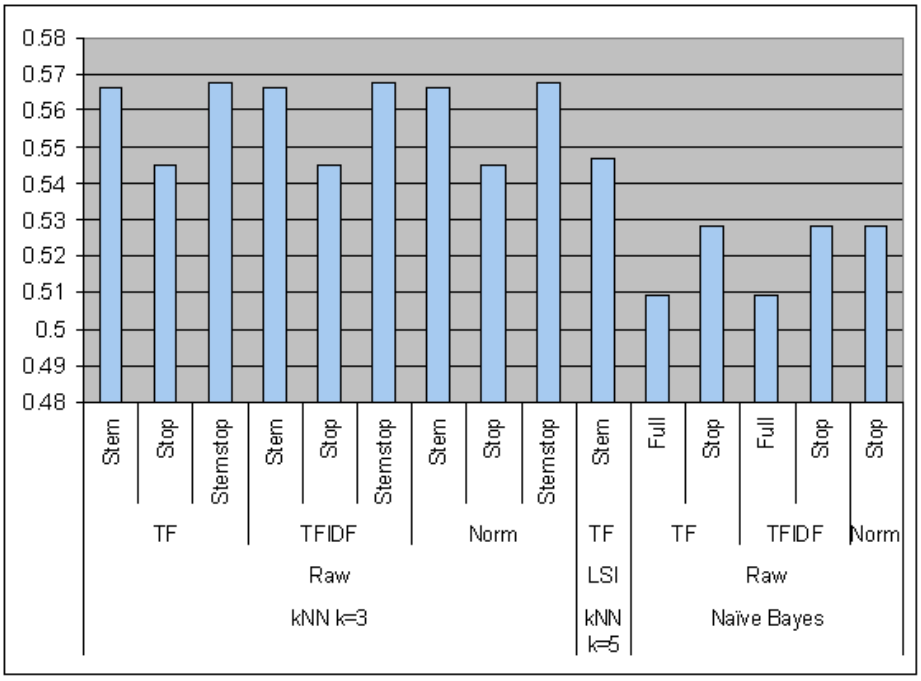


**Figure 4.5. Top 15 Algorithm Scores for Bletchley Park Life Experience**

### *4.1.4. Interview*

The top algorithm scores for the Interview event type can be seen in Figure 4.6. This is the first event type for which decision trees produced some of the best results, and in fact the top two were generated by them. Specifically, the top score of 0.42 was produced by the C4.5 algorithm, on the LSI version of the stoplisted TFIDF data. C4.5 and ID3 produced several other good scores, in all but one case using a stoplist. In fact, using a stoplist appears to be particularly beneficial for this event type, as only one of the top 15 algorithms does not use one. Besides decision trees, the other dominant algorithm in this group is kNN. This algorithm does well with $k$ = 1, 2, and 3, and in all cases the LSI version of the data is used. Finally, although attempts were made to better the score of 0.42, this proved impossible.

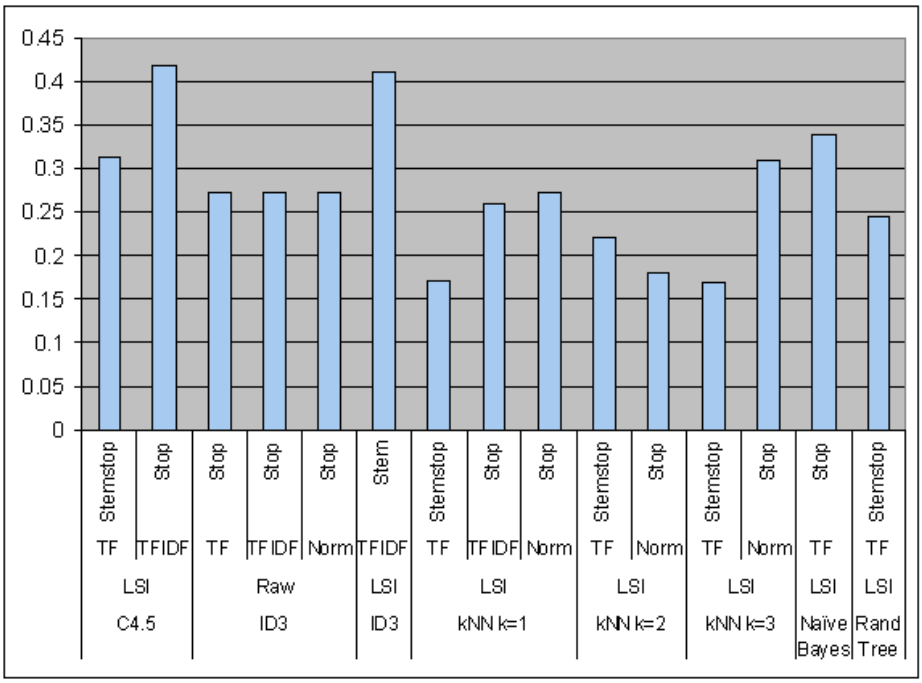


**Figure 4.6. Top 15 Algorithm Scores for Interview**

### 4.1.5. Meeting

The results for the event type Meeting are interesting because there does not appear to be any particular pattern. They can be seen in 4.7. Every algorithm is represented, and kNN appears with two different values of *k*. Overall, stemming and stoplisting appear to be beneficial, but the best algorithm uses the full terms. Similarly, more than half of the term frequency representations use the normalized form, but the best combination uses the TF representation. However, the most interesting fact about these results is that four of the top six algorithm scores were produced by the Random Tree algorithm, including the overall best one. This is surprising, since the algorithm selects attributes randomly. It would be expected that the other techniques, which have some methodical approach to handling the text, would do better. However, a possible explanation is that there are some inconsistencies in the data which amount to noise. A story might be mislabeled, or perhaps there is great variety in the content of the stories. There are twenty stories with this event type, so it does not appear to be an issue of insufficient training instances. The likelihood of randomly selected terms being present across more than a handful of stories seems like it should be low. But if the stories vary widely, it could be that there are only accidental commonalities. Consequently, the Random Tree approach does still seem to be reasonable. The specific data set on which it produces the best algorithm score is the raw TF matrix with full terms. Adjustments made to the algorithm settings did not yield improved performance.

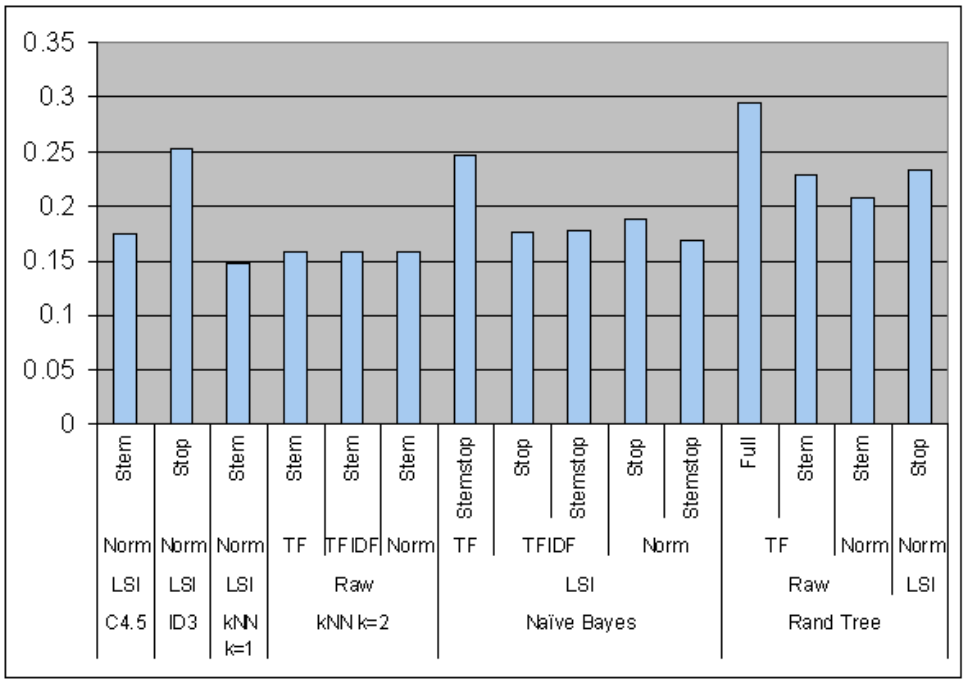


**Figure 4.7. Top 15 Algorithm Scores for Meeting**

### 4.1.6. Production

There are several interesting patterns which can be seen from the data for the Production event type, displayed in Figure 4.8. In this case, it appears that decision trees do very well, producing the top eight algorithm scores. This is primarily through ID3 and C4.5, although Random Tree is also represented in the top 15. For the decision trees, stemming seems to be very useful for LSI data, while the full terms seem better for the raw data. However, the decision trees using the stemmed data do significantly better than all of the other approaches. For the non-decision tree approaches, naïve Bayes and kNN with $k$ = 2, the best data sets appear to be the raw version of the stoplisted data, although the term frequency matrix again does not matter.

Ultimately, the best method was C4.5 on the LSI version of the TFIDF representation of the stemmed terms. The score obtained by this approach was 0.48. However, changing the algorithm parameters did finally yield an improvement in this case. The algorithm score rose to 0.51 when the minimum number of instances per leaf was changed from 2 to 1.

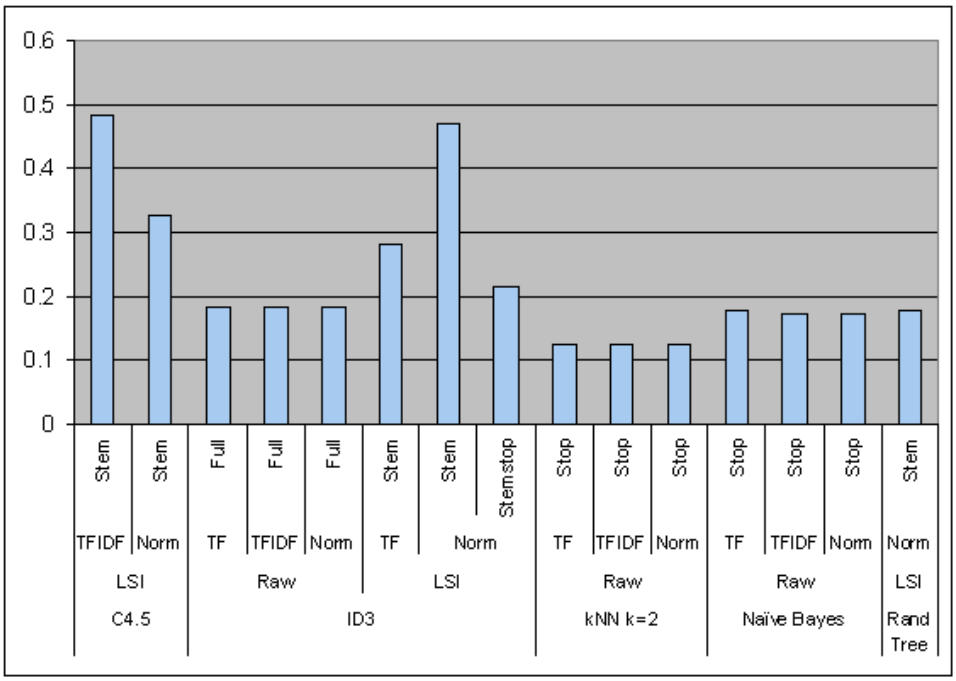


**Figure 4.8. Top 15 Algorithm Scores for Production**

### *4.1.7. War Life Experience*

The results for the War Life Experience event type show that while the C4.5 algorithm dominates in the top 15, the single best algorithm is naïve Bayes. The chart can be seen in Figure 4.9. The C4.5 algorithm worked best on the raw data, with the best results being obtained for the stoplisted data, and performed consistently across the three term frequency matrices. The C4.5 algorithm is less successful on the LSI version of the data, so it is interesting that the best algorithm is naïve Bayes performed on the LSI TF matrix of the stemmed data. However, one consistent point across the top 15 is that both stemming and using a stoplist are beneficial.

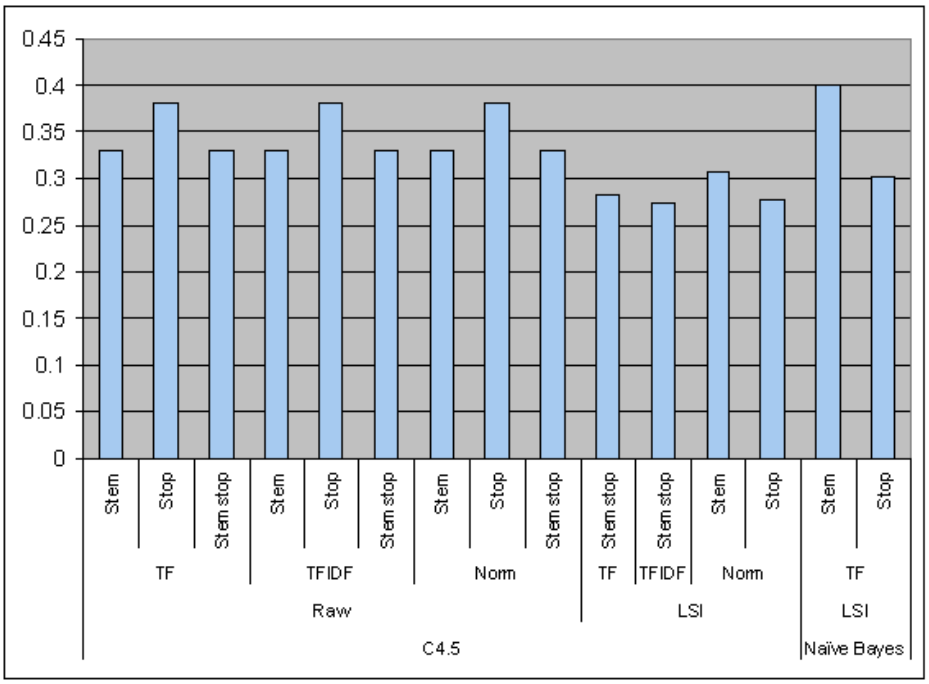


**Figure 4.9. Top 15 Algorithm Scores for War Life Experience**

### 4.2. Generalizing

It is difficult to draw general conclusions based on these tests, because there were many variables, and the results varied widely across all factors. There were too many conditions to allow inferential statistics to be used effectively, so the judgments are slightly subjective.

Despite this, there are a few trends that appear to be present across the data. The first interesting point is that it seems that the term frequency matrix choice has little consistent impact on the algorithm score. In each of the charts showing the top 15 algorithms, it was clear that there was not a dominant matrix type. It was often the case that there were identical scores across the different matrix types, such as in Figures 4.7 and 4.9. Similarly, it is not possible to say whether the raw or LSI data was better, as it varied among the event types. The same is true for the term set, although it does appear that the stemmed or stoplisted versions may have performed slightly better. There were, however, some notable exceptions to this, such as in Figure 4.7. One tendency that did seem to be present was for stemming to be more successful in conjunction with the LSI version of the data, rather than the raw data, which often seemed to benefit more from having stopwords removed.

There were also a few notable points regarding the machine learning methods themselves. First, all of the methods tested did well at least some of the time. One trend that appears to be present across the data is that when kNN is successful, it seemed usually to be with the LSI version of the data. Another aspect of kNN relates to the value of *k*. The only event type for which $k = 5$ produced one of the top algorithm scores was Bletchley Park Life Experience. This is not surprising, since it is the type with the most stories. The other cases where kNN was successful, *k* was 1 or 2. Setting it higher than that for events with few stories creates a risk of missing important aspects of the data. For instance, while kNN with $k = 2$ was the best algorithm for the Birth event type, the algorithm score was 0 for it on all data sets when kNN was used with $k = 3$ and 5.

### 4.3. Future Work

There are several areas of this research that deserve further study. The most obvious work that should be carried out next is to complete some of the original plan: first, the memory problems with carrying out PCA and for LSI with the full terms should be solved, and these should be implemented. The experiments should be carried out for these versions of the data, and compared to the original findings. The experiments carried out during this project should be done again, using more than 10 runs. Based on findings in this project, it appears to be worthwhile to abandon the term frequency dimension, so that more in depth statistical analysis can be carried out on the results of these experiments. The TF matrix should most likely be selected for these experiments, to maintain the purest form of the data. These combinations should yield more information which can help to determine the best approaches to take in this test classification task.

## 5. Acknowledgement

This research was conducted within the European Union Information Society and Technology funded CIPHER project, IST-2001-32559. I would like to thank the Bletchley Park Trust and tour guides for their continued involvement with this work, and the Knowledge Media Institute for giving me the chance to participate in their intern program.

## References

H. Alani, S. Kim, D. Millard, M. Weal, W. Hall, P. Lewis, and N. Shadbolt. Automatic Ontology-Based Knowledge Extraction from Web Documents, *IEEE Intelligent Systems* 18 (2003), no. 1: 14-21.

C. Apte, F. Damerau and S. Weiss. Text Mining with Decision Rules and Decision Trees. In *Proceedings Conference on Automated Learning and Discovery*, Carnegie-Mellon University, June (1998).

Bletchley Park - Station X. http://www.bletchleypark.org.uk/.

CIPHER: Communities of Interest Promoting Heritage of European Regions. http://cipherweb.open.ac.uk/.

T. Collins, P. Mullholland and S. Watt. Using genre to support active participation in learning communities. In *Proceedings of the First European Conference on Computer-Supported Collaborative Learning*, Maastricht, The Netherlands (2001), 324-331.

H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proc. 40th Anniversary Meeting Assoc. for Computational Linguistics (ACL 2002)*. Assoc. for Computational Linguistics, East Stroudsburg, PA, (2002).

Defense Technical Information Center. DVL/Verity Stop Word List. http://dvl.dtic.mil/stop_list.html. (2000).

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing By Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41 (1990): 391-407.

R. Feldman, I. Dagan and H. Hirsh. Mining Text Using Keyword Distributions. *Journal of Intelligent Information Systems*, 10 (1998), no. 3: 281-300.

J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. London: Academic Press (2001).

H. Ke and M. Shaoping. Text categorization based on concept indexing and principal component analysis. In *Proceedings. IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering (TENCON '02)*, 1 (2002): 28-31.

A. Maedche and R. Volz. The Text-To-Onto Ontology Extraction & Maintenance System. In *Workshop on Integrating Data Mining and Knowledge Management co-located with the 1st International Conference on Data Mining*, San Jose, California, USA, November (2001).

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. London: MIT Press, 2002.

T. M. Mitchell. 1997. *Machine Learning*. London: McGraw Hill.

P. Mulholland, T. Collins and Z. Zdrahal. Story Fountain: Intelligent support for story research and exploration. In *Proceedings of the ACM Conference on Intelligent User Interfaces (IUI'2004)*. Madeira, Portugal. January (2004).

K. Nigam, J. Lafferty and A. McCallum. Using Maximum Entropy for Text Classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*. (1999), 61-67.

N. F. Noy and D. L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. Technical report SMI-2001-0880, Stanford Medical Informatics (2001).

M. F. Porter. 1980. An algorithm for suffix stripping, *Program*, 14 (1980), no. 3: 130-137. [Code available on the Web at http://www.tartarus.org/~martin/PorterStemmer/.]

R. Schwebel *et al*. 2004. PDL - PDL - The Perl Data Language. http://pdl.perl.org/.

M. Vargas-Vega and D. Celjuska. Ontology-driven Event Recognition on Stories. Technical report KMI-TR-135, Knowledge Media Institute, Milton Keynes, England (2003).

I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco (2000). [Software available on the Web at http://www.cs.waikato.ac.nz/~ml/weka/].

C. Fellbaum, ed. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA (1998). [Information available on the Web at http://www.cogsci.princeton.edu/~wn/.]

Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*. (1999), 42-49.

S. Zelikovitz and H. Hirsh. Using LSI for Text Classification in the Presence of Background Text. In *Proceedings of the tenth international conference on Information and knowledge management*. Atlanta, Georgia (2001), 113-118.