Evolved Apache Lucene SpanFirst Queries are Good Text Classifiers

Laurie Hirsch

Abstract—Human readable text classifiers have a number of advantages over classifiers based on complex and opaque mathematical models. For some time now search queries or rules have been used for classification purposes, either constructed manually or automatically. We have performed experiments using genetic algorithms to evolve text classifiers in search query format with the combined objective of classifier accuracy and classifier readability. We have found that a small set of disjunct Lucene SpanFirst queries effectively meet both goals. This kind of query evaluates to true for a document if a particular word occurs within the first N words of a document. Previously researched classifiers based on queries using combinations of words connected with OR, AND and NOT were found to be generally less accurate and (arguably) less readable. The approach is evaluated using standard test sets Reuters-21578 and Ohsuned and compared against several classification algorithms.

I. INTRODUCTION

Automatic text classification is the activity of assigning predefined category labels to natural language texts based on information found in a training set of labelled documents. In recent years it has been recognised as an increasingly important tool for handling the exponential growth in available online texts and we have seen the development of many techniques aimed at the extraction of features from a set of training documents, which may then be used for categorisation purposes. It has also been recognised that knowledge discovery is best served by the construction of predictive models which are both accurate and comprehensible.

In the 1980’s a common approach to text classification involved humans in the construction of a classifier or ‘expert systems’, which could be used to define a particular text category. Such a classifier would typically consist of a set of manually defined logical rules, one per category, of type

if \{DNF formula\} then \{category\}

A DNF (“disjunctive normal form”) formula is a disjunction of conjunctive clauses; the document is classified under a category if it satisfies the formula i.e. if it satisfies at least one of the clauses. An often quoted example of this approach is the CONSTRUE system [1], built by Carnegie Group for the Reuters news agency. A sample rule of the type used in CONSTRUE to classify documents in the ‘wheat’ category of the Reuters dataset is illustrated below.

\[ \text{if } ((\text{wheat} \& \text{farm}) \text{ or} \text{ (wheat} \& \text{commodity}) \text{ or} \text{ (bushels} \& \text{export}) \text{ or} \text{ (wheat} \& \text{tonnes}) \text{ or} \text{ (wheat} \& \text{winter} \& \text{not soft}) \text{) then WHEAT else \sim \text{WHEAT} \]

Such a method, sometimes referred to as ‘knowledge engineering’, provides accurate rules and has the additional benefit of being human understandable. In other words, the definition of the category is meaningful to a human, thus producing additional uses of the rule including verification of the category. However the disadvantage is that the construction of such rules requires significant human input and the human needs some knowledge concerning the details of rule construction as well as domain knowledge [2].

Since the 1990’s the machine learning approach to text categorisation has become the dominant one. In this case the system requires a set of pre-classified training documents and automatically produces a classifier from the documents. The domain expert is needed only to classify a set of existing documents. Such classifiers, usually built using the frequency of particular words in a document (sometimes called ‘bag of words’), are based on two empirical observations regarding text:

1. the more times a word occurs in a document, the more relevant it is to the topic of the document.
2. the more times the word occurs throughout the documents in the collection the more poorly it discriminates between documents.

A well known approach for computing word weights is the term frequency inverse document frequency (tf-idf) weighting which assigns the weight to a word in a document in proportion to the number of occurrences of the word in the document and in inverse proportion to the number of documents in the collection for which the word occurs at least once, i.e.

\[ a_{ik} = f_{ik} \log \left( \frac{N}{n_i} \right) \]

where \( a_{ik} \) is the weight of word \( i \) in document \( k \), \( f_{ik} \) is the frequency of word \( i \) in document \( k \), \( N \) the number of
documents in the collection and $n_i$ equal to the number of documents in which $a_i$ occurs at least once. A classifier can be constructed by mapping a document to a high dimensional feature vector, where each entry of the vector represents the presence or absence of a feature [3], [4]. In this approach, text classification can be viewed as a special case of the more general problem of identifying a category in a space of high dimensions so as to define a given set of points in that space. This is usually accompanied by some form of feature reduction such as the removal of non-informative words (stop words) and by the replacing of words by their stems, so losing inflection information. Such sparse vectors can then be used in conjunction with many learning algorithms for computing the closeness of two documents and quite sophisticated geometric systems have been devised such as [5].

Although this method has produced accurate classifiers based on the vector of weights, it has been widely noted that a major drawback of such classifiers lies in the fact that such classifiers are not human understandable. In recent years there have been a number of attempts to produce effective classifiers that are human understandable e.g. [6], [7], [8]. The advantages of such a classifier include

1. The classifier may be validated by a human.
2. The classifier may be fine tuned by a human.
3. The classifier can be used for auditing purposes.
4. The classifier may be used for another task such as information extraction or text mining.

As an example Oracle Corporation offer various options for classification in their Oracle Text product\(^1\). Two supervised classifiers are provided using user supplied training documents. The first option uses SVM technology and produces opaque classifiers with high accuracy. The second option produces classification rules which are transparent and can be understood and modified by a human. The second option uses a decision tree algorithm and is recognised as having much lower accuracy. The example clearly indicates that readability and modifiability have recognized value to commercial classification products and that the production of readable rules with high accuracy is a worth while objective in text classification research.

Generally, the attempts to produce classification systems that are human understandable have involved the production of a set of rules which are used for classification purposes [6], [7], [8], [9], [10], [11]. Often, the set of rules is quite large which reduces some of the qualitative advantages because it will be harder for a human to comprehend or modify the classifier. In this paper we describe a method to evolve compact human understandable classifiers using only a set of training documents. Furthermore each category in the dataset requires only one rule. The rule is particularly easy to comprehend because it is in the form of a search query.

The system described here uses a genetic algorithm (GA) to produce a synthesis of machine learning and knowledge engineering with the intention of incorporating advantageous attributes from both. We have tested the system on two standard datasets: Reuters 21578 and Ohsumed. The search queries produced by the GA are in a reasonably readable form and produce competitive levels of classification accuracy, and indeed to our knowledge are the most accurate human understandable classifier that have been evaluated on these datasets.

In the next section, we discuss GA, review previous classification work and introduce Apache Lucene which we use for indexing and searching. We then provide information concerning the implementation of our application and the results we have obtained.

\section*{II. BACKGROUND}

\subsection*{A. Evolving text classification search queries.}

Genetic Methods such as Genetic Programming (GP) and GA can be used to induce rules or queries useful for classifying online text. Both are stochastic search methods inspired by biological evolution. The evolution will require a fitness test based on some measure of classification accuracy [6], [7], [9], [11], [12], [13]. The basic idea we introduce here is that each individual will encode a candidate solution in a search query format. The query will return a set of documents from the dataset and can be given a specific fitness value for a particular category according to the number of correct and incorrect training documents returned by the query.

\subsection*{B. Apache Lucene}

Systems using evolutionary based methods are generally computationally intensive. In our case each individual in the population will produce a search query for each category of the dataset and the fitness is evaluated by applying the search query to a potentially large set of text documents. With a population of a reasonable size (for example 1024 individuals) evolving over 100 or more generations it is critical that such queries can be executed in a timely and efficient manner. For this reason we decided to use Apache Lucene which is an open source high-performance, full-featured text search engine. We use Lucene to build indexes on the text datasets and to evaluate the queries produced by the GA. A full description of the indexing system and query syntax is given at the official Lucene site (http://lucene.apache.org/) together with the Java source code and other useful information concerning Lucene. Lucene provides many other features and in particular a tf-idf based weighting system for search terms. However, in our application this was not used and we only counted the total number of relevant and irrelevant matching documents for each search query.

We have previously described a system whereby GPs were able to produce search queries for classification using a

\footnote{http://www.oracle.com/technology/products/text/index.html}
variety of query operators including AND, OR, NOT and term proximity [11]. In this paper we investigate the use of two query types namely OR and SpanFirst. We have found that reducing the number of functions in this way produces an improvement in classification accuracy together with an improvement in classifier readability.

A SpanFirst query restricts a query to search in the first part of a document. This appears to be useful since often the most important information in a document is at the start of the document. For example, to find all documents where "barrel" is within the first 100 words we could use the Lucene query:

\[
\text{SpanFirstQuery.new(SpanTermQuery.new(:content, } \text{"barrel"}, 100))
\]

In this paper we simplify the format and write the above as: (barrel 100). A more complex query might be: (barrel 100) (oil 20) which would retrieve documents where the word “barrel” occurred within the first 100 words of a document OR the word “oil” occurred within the first 20 words.

We summarise the key features below:
• The basic unit we use is a single word (no stemming is used) which occurs in the training documents.
• Lucene search queries are produced for each category in the dataset; thus each search query is a binary classifier.
• Queries are constructed using a set of disjunct SpanFirst queries.
• The terms used and the range (or slop in Lucene terminology) of the SpanFirst queries are determined by the GA individuals.
• Fitness is accrued for individuals producing classification queries which retrieve positive examples of the category but do not retrieve negative examples. Thus the documents in the training set are the fitness cases.

In this paper we refer to the system as GA-SFQ and compare our results with our previously reported GP system (GPTC) [11], and a recently developed GA system which uses a combination of OR and NOT operators [7]. We also include results for a number of alternative rule based and statistical classifiers.

C. Data Sets

The task involved categorising documents from three text datasets extracted from two collections. The first two were selected from the Reuters-21578 test collection which has become a standard benchmark for the text categorisation tasks [14]. Reuters-21578 is a set of 21,578 news stories which appeared in the Reuters newswire in 1987, classified according to 135 thematic categories, mostly concerning business and economy. Generally researchers have split the documents in the collection into a training set used to build a classifier and a test set used to evaluate the effectiveness of that classifier. Several of the categories are related (such as WHEAT and GRAIN) but there is no explicit hierarchy defined on the categories. In our experiments we use the “ModApt’e split”, a partition of the collection into a training set and a test set that has been widely adopted by text categorisation experimenters. The top 10 (R10) categories are most commonly used and we focus our discussion on the results we obtained on this subset. We also generated a classifier for the subset of 90 categories (R90). An in depth discussion concerning the Reuters dataset is given in [15].

The second test collection is taken from the Ohsumed corpus (ftp://medir.ohsu.edu/pub/ohsumed) compiled by William Hersh. From the 50216 documents in 1991 which have abstracts, the first 10000 are used for training and the second 10000 are used for testing. The classification task considered here is to assign the documents to one or multiple categories of the 23 MeSH "diseases" categories (Ohs23) which have been used in [4] and [7] among others.

D. Pre-processing

Before we start the evolution of classification queries a number of pre-processing steps are made.

1. All the text is placed in lower case.
2. A small stop set is used to remove common words with little semantic weight.
3. For each dataset a Lucene index is constructed and each document labelled (using Lucene fields) according to its category(ies) and its test or training status.
4. For each category of the dataset a list of potentially useful words is constructed for use by the GA. Each word found in the training data is given a score according to its effectiveness as a single term classifier for the relevant category. So, for example, if we find the word ‘oil’ in the training data for a particular category, we can construct a query based on the single word which will retrieve all documents containing that word. We give the word a value (F1 score) according to the number of positive and negative examples retrieved by the single term query. We can then put the words in order of their score and select the top N words for use by the GA.

E. Fitness

Individuals are set the task of creating valid Lucene search queries by producing one or more disjunct SpanFirst queries. Such a query must be evolved for each category in the training set. Each query is actually a binary classifier i.e. it will classify any document as either in the category or outside the category.

In information retrieval and text categorisation the break-even-point (BEP) statistic is a commonly used measure of classification accuracy. BEP finds the point where precision and recall are equal. Since this is hard to achieve in practice, a common approach is to use the average of recall and precision as an approximation:

\[
\text{BEP} = \frac{p + r}{2}
\]

where: p = precision and r = recall.
\[ \text{BEP} = \frac{p + r}{2} \]

Recall (r) = the number of relevant documents returned/the total number of relevant documents in the collection and precision (p) = the number of relevant documents returned/the number of documents returned.

The F1 measure is also commonly used for determining classification effectiveness and has the advantage of giving equal weight to precision and recall [16]. F1 is given by

\[ F1 = \frac{2pr}{p + r} \]

F1 also gives a natural fitness measure for an evolving classifier since BEP may favour trivial results, for example, if no data is correctly categorized then r=0 and p=1 so their average is 0.5 instead of 0 when using the harmonic average. Such classifiers are actually likely to be the norm in the early generations of a GA run, therefore, the fitness of an individual is assigned by calculating F1 for the generated query. This approach is also taken in the Olex-GA and GPTC systems [7], [11].

A few examples may be useful at this point. If we are evolving a classifier for the Reuters 'crude' category a GA might produce the following query:

(barr 25) (bbl 200)

By default the elements of Lucene queries are disjunct i.e. there is an implicit OR between elements of a query and the above query would retrieve any document containing either the word 'barrel' in the first 25 words or the word 'bbl' in the first 200 words. In fact, such a query is quite an effective classifier for the crude category and has F1 of 0.693.

The GA structure we have produced leaves room for redundancy, for example the query

(corn 252) (corn 7)

is equivalent to the query (corn 252). We remove the redundant entries in the results reported here.

The integer part of the SpanFirst query are (initially random) numbers between 1 and 300. This means that the classifiers reported here are entirely based on the first 300 words of any document.

F. GA Parameters

We used a fixed set of GA parameters in all our experiments which are summarised in Table 1 For those not familiar with these it is worth noting the following points:

- An individual is selected according to fitness and can be simply copied into the then next generation (reproduction) or part of the chromosome may be randomly changed (mutation) or most commonly parts of the chromosome are exchanged with another selected individual to create two new individuals (crossover). The probabilities of these 3 possibilities are determined by the parameters in Table 1.

- Subpopulations can be used as a method of increasing diversity in the GA population. Only limited communication (immigration/emigration) is allowed between subpopulations. In our case we exchanged 3 individuals between the two subpopulations every 20 generations.

A maximum of 20 SpanFirst queries are produced by each chromosome.

III. Experiments

A. Objectives

The objectives of our experiments were twofold:

1. To evolve effective classifiers against the text datasets.
2. To automatically produce compact and human understandable classifiers in search query format.

B. Evolution

In all the experiments reported here the GA system only had access to the training data. The final result was determined by applying the best queries evolved to the test data. The evolution for each dataset was repeated 5 times.

C. Performance

Queries must be evolved for each category of the document set and each individual in the evolving population must fire a Lucene query to obtain its fitness. All the experiments were run on an Intel Core 2 processor with 2G of memory. Such a system was able to produce the classification queries for R10 listed in Table 2 in just over 14 minutes. Considering the number of queries and the number of documents in the set we would suggest that this result is a testament to the efficiency of ECJ and Lucene. It is worth noting that we used the Lucene instantiated package which significantly improved performance using an all in memory index.

The result of all the training work is a search query. To test the R10 classifier requires the execution of 10 search queries and the result will occur in a time frame well below human perception. The fact that search queries will scale up to large text databases, such as the Internet, is well known.

D. Results

The best queries evolved on the training documents were applied to the test set to produce the final result. The query produced is an important part of our system since we are
emphasizing the qualitative difference of this particular classifier, and so we give the complete set of classification search queries for the R10 in Table 2 together with the F1 test result.

Our previously reported GPTC system [11] was able to use a variety of Lucene query operators but the results were less accurate. Also, the queries were not as readable, for example the GPTC query produced for the Crude category is show below:

\[
\left(\left(\text{barrel AND barrel AND NOT acquistis} \right) \cdot \text{stake} \cdot \text{stake} \cdot \text{trade} \cdot \text{stake} \cdot \text{wheat} \cdot \text{"compani share"} \cdot 10 \cdot \left(+\text{oil NOT compani} \right) \cdot \text{NOT acquistis} \cdot \text{"crop crude "} \cdot \text{oil petroleum} \cdot 20 \right) \cdot \text{NOT "march loss"} \cdot 10\cdot \text{NOT net} \cdot \left(+\text{barrel NOT merger} \right) \cdot \text{NOT dollar} \cdot \text{"march loss"} \cdot 10 \cdot \left(\text{"opec oil"} \cdot 5 \cdot \left(\text{ga AND import} \right) \cdot \text{"rate bank"} \cdot 10 \cdot \left(\text{refineri AND oil} \right) \cdot \text{"corn corn"} \cdot 10 \cdot \text{"oil energi"} \cdot 20 \cdot \left(\text{barrel AND barrel AND NOT acquistis} \right) \cdot \text{oil petroleum} \cdot 20 \cdot \text{"oil explor"} \cdot 5 \cdot \left(\text{ga AND opec} \right) \cdot \text{vs}
\]

The GA-SFQ system produces the following, more readable and slightly more accurate query:

\[
\text{(oil 35) (crude 46) (distillate 48) (refinery 54)} \\
\text{(iranian 87) (opec 98) (refineries 111)} \\
\text{(barrel 185) (barrels 205) (bbl 298)}
\]

Table 3 shows the results for GA-SFQ in comparison to other classifiers for the R10 dataset. In this case we show the BEP as in the past this result has been the most widely used and is therefore useful for comparison purposes. We are particularly interested in the other rule based classifiers which are at least partly human understandable. These are Olex-GA [7], TRIPPER [8], RIPPER [10], ARC-BC (3072 rules) [17], C4.5 [18] and bimgrams [19]. We also note that the query evolved for the wheat category, which scores F1 against the test set of 0.90 is both more compact and more accurate than the rule constructed by a human expert discussed in the introduction which has an F1 of 0.84.

The results for the R10 set show that GA-SFQ produces rules of higher accuracy than any other rule based system overall and in almost every category. To further test the statistical significance of the results, we applied the paired t-test to the results obtained on the R10 dataset similar to [20]. With a confidence level of 95% we found that the performance of GA-SFQ was significantly greater than all other methods with the exception of SVM and GPTC where there was no significant difference at this level.

Table 4 and Table 5 shows the results for GA-SFQ in comparison to the older GPTC classifier together with the results of the widely reported survey of 40 classifiers applied to the Reuters set [15]. The micro-average is a global calculation of F1 regardless of category and the macro-average is the average on F1 scores for all the categories. The results show that GA-SFQ to be well above average in the task of classifying both R10 and R90 and an improvement on the GPTC system. The results also indicate that the GA-SFQ system produces results of high accuracy on the R90 set.

It is important to compare results on at least two dataset so we present our results for the Ohsu(23) in Table 6. Again we can see results of high accuracy for GA-SFQ which again is the most accurate of the rule based systems.

IV. DISCUSSION

We argue that the queries produced might be used for auditing, verification, labelling and other purposes due to their readability. For example, for the Ohsumed category 4 (Cardiovascular) the following query was generated and obtained an F1 value of 0.789 on the test set.

\[
\left(\text{angina 139} \right) (\text{angioplasty 281}) (\text{anti hypertensive 135}) \\
\left(\text{aortic 84} \right) (\text{arteries 23}) (\text{cardiac 23}) (\text{coronary 266}) \\
\left(\text{diastolic 95} \right) (\text{doppler 68}) (\text{echocardiography 291}) \\
\left(\text{heart 19} \right) (\text{hypertension 32}) (\text{hypertensive 109}) \\
\left(\text{ischemia 21} \right) (\text{myocardial 70}) (\text{valve 280})
\]

Each term in a GA-SFQ query is given a SpanFirst query slop number and this may be some indication of that terms importance. A lower number means that the word must occur nearer to the start of the category if the document is to be returned, so, for example, we would suggest from the above query that the most significant terms for the category would be ‘heart’, ‘ischemia’, ‘arteries’ and ‘cardiac’. Similarly, if we look at the terms with the lowest slop factor used in the query evolved for the R10 acquisitions category (see Table 2 for the full query) we get:

\[
\left(\text{buy 10} \right) (\text{company 11}) (\text{bid 13}) (\text{offer 15})
\]

It is worth noting that these words are common terms which might return many irrelevant documents if the query did not restrict the search to the first few words of the document.

GA-SFQ produces only one rule per category as opposed to hundreds or thousands using some of the other rule based methods such as [17]. The comprehensibility of the GA-SFQ queries is an interesting question. For example, compare the query for the R10 category ‘corn’ with that produced by a recent alternative GA rule based system known as Olex GA [7]

\[
\begin{align*}
\text{Pos} &= \{\text{“corn”}, \text{“maize”}, \text{“tonnes maize”}, \text{“tonnes corn”}\} \\
\text{Neg} &= \{\text{“jan”}, \text{“qtr”}, \text{“central bank”}, \text{“profit”}, \text{“4th”}, \\
&\text{“bonds”}, \text{“pact”}, \text{“offering”}, \text{“monetary”}, \text{“international”}, \\
&\text{“money”}, \text{“petroleum”}\}
\end{align*}
\]

In Olex-GA we see that there are two sets of terms and a document is considered to be in the category if it does contain any of the positive terms and does not contain any of the negative terms. The authors report that the above query produced a BEP on the test set of 91.1
The GA-SFQ query for the same category is shown below:

(corn 219) (farmer 72) (maize 234) (pik 17)

This uses only 4 terms and achieves a higher BEP on the test set (93.8). It does contain an integer component for each term which we might argue is off putting to human classifiers. However, an advantage is that the human never needs to examine more than the first 300 words of any document whereas with rules similar to that produced by Olex-GA or GPTC the entire document must be scanned for the presence of each term in the rule.

V. Future work

The current GA-SFQ system described here uses the first 300 words of a document for classification purposes. We would like to identify if using other parts of a document (such as the last part) might add to classification accuracy.

We are investigating the usefulness of generating queries for classification based on the proximity of two or more words i.e. if word X occurs within N words of word Y.

Reducing the maximum number of words available to each individual, generally reduces accuracy but maybe useful in generating more comprehensible labels and we hope to investigate this aspect further.

We are also investigating the possibility of generalizing the applicability of such queries, for example, such that you could provide an initial set of training documents for one category only and the queries would retrieve very similar documents when used, for example, as Internet searches.

VI. Conclusion

We have produced a system capable of generating classification queries with no human input beyond the identification of training documents. The classifier makes use of the first part of any document and is able to set the distance in number of words within which a word must occur for a document to be classified under any particular category. We are emphasizing the qualitative advantages of this classifier and believe this new format for a text classifier has important advantages stemming from its compactness, its comprehensibility to humans and its search query format. We suggest that there may be a number of areas within automatic text analysis where the technology described here may be of use.

REFERENCES


Table 1: GA Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population</td>
<td>1024</td>
</tr>
<tr>
<td>Generations</td>
<td>100</td>
</tr>
<tr>
<td>Selection type</td>
<td>Tournament</td>
</tr>
<tr>
<td>Tournament size</td>
<td>5</td>
</tr>
<tr>
<td>Termination</td>
<td>(F1=1) or max generations</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.1</td>
</tr>
<tr>
<td>Reproduction probability</td>
<td>0.1</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>0.8</td>
</tr>
<tr>
<td>Elitism</td>
<td>No</td>
</tr>
<tr>
<td>Subpopulations</td>
<td>2 (exchange 3 individuals every 20 generations)</td>
</tr>
<tr>
<td>Chromosome length</td>
<td>40 (fixed: 20 words with values)</td>
</tr>
<tr>
<td>Word list length</td>
<td>200</td>
</tr>
<tr>
<td>Engine</td>
<td>ECJ 19: <a href="http://cs.gmu.edu/~eclab/projects/ecj/">http://cs.gmu.edu/~eclab/projects/ecj/</a></td>
</tr>
</tbody>
</table>

Table 2: R10 results with Evolved SpanFirst queries

<table>
<thead>
<tr>
<th>Category</th>
<th>F1 Test</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acq</td>
<td>0.899</td>
<td>(acquire 212) (acquired 44) (acquisition 117) (bid 13) (buy 10) (buyout 255) (company 11) (completes 16) (definitive 229) (merger 63) (offer 15) (sell 32) (sells 65) (stake 43) (takeover 269) (transaction 278) (undisclosed 111)</td>
</tr>
<tr>
<td>Corn</td>
<td>0.933</td>
<td>(corn 219) (farmer 72) (maize 234) (pik 17)</td>
</tr>
<tr>
<td>Crude</td>
<td>0.872</td>
<td>(barrel 185) (barrels 205) (bbl 298) (crude 46) (distillate 48) (iranian 87) (oil 35) (opec 98) (refineries 111) (refinery 54)</td>
</tr>
<tr>
<td>Earn</td>
<td>0.968</td>
<td>(3rd 126) (declared 23) (dividend 277) (dividends 47) (earnings 37) (gain 24) (loss 69) (losses 8) (net 20) (payout 47) (profit 36) (profits 29) (results 54) (split 46) (turnover 23) (vs 130)</td>
</tr>
<tr>
<td>Grain</td>
<td>0.957</td>
<td>(barley 236) (bushel 222) (ccc 131) (cereals 105) (corn 64) (crop 166) (crops 172) (grain 43) (grains 143) (maize 264) (rice 15) (soybean 202) (wheat 275)</td>
</tr>
<tr>
<td>Interest</td>
<td>0.775</td>
<td>(7-3/4 182) (bills 10) (deposit 14) (indirectly 273) (maturity 176) (money 6) (outright 65) (rate 27) (rates 23) (repurchase 40)</td>
</tr>
<tr>
<td>Money-fx</td>
<td>0.801</td>
<td>(bundesbank 25) (cooperate 70) (currencies 246) (currency 54) (dollar 28) (fed 76) (intervene 226) (miyazawa 125) (monetary 37) (money 29) (stability 17) (yen 13)</td>
</tr>
<tr>
<td>ship</td>
<td>0.805</td>
<td>(7 6) (ferry 68) (freight 13) (loading 76) (port 49) (seamen 79) (shipping 57) (ships 64) (strike 19) (tanker 157) (tankers 35) (tonnage 24) (vessel 161) (vessels 191) (warships 256) (waterway 228)</td>
</tr>
<tr>
<td>trade</td>
<td>0.756</td>
<td>(account 13) (developing 22) (gatt 63) (growing 9) (practices 25) (retaliation 295) (sanctions 201) (taupo 72) (trade 32) (uruguay 5)</td>
</tr>
<tr>
<td>wheat</td>
<td>0.902</td>
<td>(durum 109) (egypt 7) (wheat 172)</td>
</tr>
</tbody>
</table>
### Table 3: R10 Results classifier comparison

<table>
<thead>
<tr>
<th>BEP</th>
<th>GA-SFQ</th>
<th>GP-TC</th>
<th>Olex-GA</th>
<th>trip</th>
<th>rip</th>
<th>ARC-BC</th>
<th>C4.5</th>
<th>Bayes</th>
<th>SVM (rbf)</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq</td>
<td>89.9</td>
<td>91.3</td>
<td>87.5</td>
<td>86.3</td>
<td>85.3</td>
<td>89.9</td>
<td>85.3</td>
<td>91.5</td>
<td>95.2</td>
</tr>
<tr>
<td>corn</td>
<td>93.8</td>
<td>90.6</td>
<td>91.1</td>
<td>85.7</td>
<td>83.9</td>
<td>82.3</td>
<td>87.7</td>
<td>47.3</td>
<td>85.2</td>
</tr>
<tr>
<td>crude</td>
<td>87.8</td>
<td>87.3</td>
<td>77.2</td>
<td>82.5</td>
<td>79.3</td>
<td>77.0</td>
<td>75.5</td>
<td>81.0</td>
<td>88.7</td>
</tr>
<tr>
<td>earn</td>
<td>96.8</td>
<td>95.9</td>
<td>95.3</td>
<td>95.1</td>
<td>94.0</td>
<td>89.2</td>
<td>96.1</td>
<td>95.9</td>
<td>98.4</td>
</tr>
<tr>
<td>grain</td>
<td>95.7</td>
<td>93.3</td>
<td>91.2</td>
<td>87.9</td>
<td>90.6</td>
<td>72.1</td>
<td>89.1</td>
<td>72.5</td>
<td>91.8</td>
</tr>
<tr>
<td>interest</td>
<td>77.7</td>
<td>72.3</td>
<td>64.6</td>
<td>71.5</td>
<td>58.7</td>
<td>70.1</td>
<td>49.1</td>
<td>58.0</td>
<td>75.4</td>
</tr>
<tr>
<td>money</td>
<td>80.6</td>
<td>73.4</td>
<td>66.7</td>
<td>70.4</td>
<td>65.3</td>
<td>72.4</td>
<td>69.4</td>
<td>62.9</td>
<td>75.4</td>
</tr>
<tr>
<td>ship</td>
<td>80.7</td>
<td>84.3</td>
<td>74.9</td>
<td>80.9</td>
<td>73.0</td>
<td>73.2</td>
<td>80.9</td>
<td>78.7</td>
<td>86.6</td>
</tr>
<tr>
<td>trade</td>
<td>76.4</td>
<td>79.5</td>
<td>61.8</td>
<td>58.9</td>
<td>68.3</td>
<td>69.7</td>
<td>59.2</td>
<td>50.0</td>
<td>77.3</td>
</tr>
<tr>
<td>wheat</td>
<td>90.0</td>
<td>90.1</td>
<td>89.9</td>
<td>84.5</td>
<td>83.0</td>
<td>86.5</td>
<td>85.5</td>
<td>60.6</td>
<td>85.7</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>macro-avg</td>
<td>86.9</td>
<td>85.8</td>
<td>80.2</td>
<td>80.4</td>
<td>78.1</td>
<td>78.2</td>
<td>77.8</td>
<td>69.8</td>
<td>86.0</td>
</tr>
</tbody>
</table>

### Table 4: Reuters Micro Average F1

<table>
<thead>
<tr>
<th></th>
<th>GA-SFQ</th>
<th>GPTC</th>
<th>Survey Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>R(10)</td>
<td>0.902</td>
<td>0.891</td>
<td>0.852</td>
</tr>
<tr>
<td>R(90)</td>
<td>0.809</td>
<td>0.772</td>
<td>0.787</td>
</tr>
</tbody>
</table>

### Table 5: Reuters Macro Average F1

<table>
<thead>
<tr>
<th></th>
<th>GA-SFQ</th>
<th>GPTC</th>
<th>Survey Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>R(10)</td>
<td>0.862</td>
<td>0.847</td>
<td>0.715</td>
</tr>
<tr>
<td>R(90)</td>
<td>0.535</td>
<td>0.418</td>
<td>0.468</td>
</tr>
</tbody>
</table>

### Table 6: Ohs23 Micro Average BEP

<table>
<thead>
<tr>
<th></th>
<th>GA-SFQ</th>
<th>GPTC</th>
<th>Olex-GA</th>
<th>C4.5</th>
<th>NB</th>
<th>Ripper</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ohs23</td>
<td>63.1</td>
<td>53.4</td>
<td>62.3</td>
<td>55.1</td>
<td>57.8</td>
<td>59.7</td>
<td>65.7</td>
</tr>
</tbody>
</table>