Experiments in Emerging Trend Detection and First Story Detection

Indro De

Monday, April 25th 2005

Submitted to the faculty of Ursinus College in fulfillment of the requirements for Honors in Computer Science

Advisor(s) _____
April Kontostathis

Committee members:

_____
Lynne Edwards

_____
April Kontostathis

_____
Richard Liston

Approved: _____
Roger Coleman

Abstract

Emerging trends are topic areas in textual data that are growing in interest and utility over time. Our Emerging Trend Detection (ETD) application, which is based on the Text Mining Infrastructure (TMI), implements a term clustering approach which relies on dimensionality reduction similar to that used in Latent Semantic Indexing (LSI), a well-known information retrieval application.

Our Emerging Trend Detection (ETD) application previously has been shown to detect 93% of emerging trends in five different collections. The current project involved testing our application on larger, more robust collections. The Linguistic Data Consortium (LDC) has developed data sets of news stories from various sources for the Topic Detection and Tracking (TDT) initiative, an ongoing project for the advancement of text search technologies. We applied our ETD application to several data sets from this collection.

A First Story Detection (FSD) application was applied to several data sets from this collection. This task requires identifying those stories within a large set of data that discuss an event that has not already been reported in earlier stories. In this FSD approach, algorithms look for keywords in a news story and compare the story with earlier stories.

1. Introduction

Textual Data Mining (or "TDM") can be considered a field of its own, containing a number of applications. It has also been also known as text analysis, text mining or knowledge-discovery in text. In general, TDM applications are used to extract non-trivial and useful information from large corpora of text data, which are available in unstructured or structured format. Text mining applications require the use and application of many related fields such as Information Retrieval, Machine Learning, Statistics, and Linguistics. There are various applications of TDM, including in the bioinformatics, market research, consumer trend studies, and scientific research.

We are looking at two specific applications of TDM, namely Emerging Trend Detection and First-Story Detection. Both applications deal with trends or events, and use common Information Retrieval techniques and machine learning tools to analyze text data sets, perform statistical analysis, and make predictions regarding the behavior of topics and trends.

Emerging Trend Detection (ETD) applications are applications that are able to automatically process large amounts of data and identify emerging trends. Emerging trends are topic areas which have recently appeared in a data corpus and since then grown in interest and utility.

Emerging trend analysis is an important aspect of a variety of fields that involve monitoring and analyzing data. ETD applications can detect trends in a constantly growing quantity of available information.

The second Text Mining application used in our work is called "First-Story Detection" (FSD), or Novelty Detection. This is one of the original tasks of the Topic Detection & Tracking Tasks initiative of the National Institute of Standards and Technology (NIST). First-Story Detection is defined to be the process to find all stories within a corpus of text data that are the first stories describing a certain event [1]. An event is a topic that is described or reported in a number of stories. Examples can be governmental elections, natural disasters, sports events, etc. The First-Story Detection process runs sequentially, looking at a time-stamped stream of stories and making the decision based on a comparison of key terms to previous stories.

First-Story Detection and Emerging Trend Detection both involve the extraction and analysis of terms from the documents in a text corpus. However, ETD processes the whole corpus, while FSD examines one document at a time and then generates the results using previous knowledge.

2. Definitions

In Information Retrieval applications, such as ETD and FSD, the two key entities are *terms* and *documents*. A term is a keyword or group of words that can be used to describe a document. In our experiments, we use noun phrases described by a regular expression. These terms are extracted from a document in the early phase of the ETD application. Figure 1 shows typical examples of noun phrases.

Examples of extracted terms from the TDT3 collection:

hurricane george	congolese rebels	nobel laureates
anwar ibrahim	national elections	kyoto accord
disaster areas	linda tripp	drug giant novartis
police trial	anti-communist uprising	swissair flight 111
impeachment talks	weapons programs	hurricane earl

Figure 1: Example of extracted terms from the TDT collection

Documents are stories, articles, events, or other forms of text. In our trend detection experiments the documents used were transcribed news stories from a number of different sources. The collection of all documents is the data set or corpus. In section 3, we will closely analyze the type of documents used for our experiments.

In topic detection and tracking applications we are generally dealing with topics (or events). A *topic* can be defined as a collection of documents related to the same general topical subject. For example, topics can be a longer ongoing story or a short one-time event. Documents of the dataset can be either on-topic or off-topic

with respect to a certain topic area. In the following section, we will discuss some sample topics in our dataset.

In our first experiments, we are looking at extracted terms and trying to classify them as trends or notrends. By picking certain terms related to predefined topics, we will be able to make predictions about emerging topics within out dataset.

At some point, every event in a dataset is "introduced" through a story. In First-Story Detection we are trying to distinguish these stories from the subsequent stories. Our second experiments focus on this task. Figure 2 shows an overview of the definitions introduced in this section.

Overview:
Corpus (dataset) → Documents (stories) → Terms (features, keywords/phrases)
Ex.: All CNN docs → one CNN documents → CNN document broken into phrases

Types of documents (stories):
on-topic, off-topic, first-story

Type of terms:
trends, notrends

Figure 2: Overview of important definitions

3. TDT Corpus

For the ETD experiments a number of different test data samples from the TDT3 corpus were used for the testing of the ETD application.

The TDT3 corpus is a large data set developed for the TDT2000 project by the Linguistic Data Consortium (http://www.ldc.upenn.edu). The Linguistic Data Consortium is an open consortium of universities, companies and government research laboratories. It creates, collects and distributes speech and text databases, lexicons, and other resources for research and development purposes [5]. The TDT3 data corpus includes a large number of news stories from different sources in English and Mandarin language during the period of October – December in the year 1998. The news stories come from sources such as ABC, AP, NBC, CNN, and more. For our research purposes, only the English data was used. Figure 3 shows the news sources and distribution of the TDT3 data:

	ABC	**APW**	**CNN**	**MNB**	**NBC**	**NYT**	**PRI**
199810	334	2535	3000	256	280	2341	515
199811	338	2456	3106	214	290	2281	503
199812	340	2347	2897	213	276	2249	557
Total	**1012**	**7338**	**9003**	**683**	**846**	**6871**	**1575**

Figure 3: News story count by source and month

Data within the TDT3 corpus is structured in Standard General Markup Language (SGML), a commonly used markup language for documents. SGML uses a wider array of markup tags than the Extensible Markup Language (XML), a language derived from SGML. Figure 4 shows the standard header information of TDT3 data.

<DOC>
<DOCNO> APW19981119.0832 </DOCNO>
<DOCTYPE> NEWS </DOCTYPE>
<TXTTYPE> NEWSWIRE </TXTTYPE>
<TEXT>
After years of digging across northern France, war buff Philippe Gorczynski
has found his prize: a rusty but intact British tank from the Great War, with a few surprises inside. On Thursday, a dozen archeological workers used an earth mover to gently remove tons of red earth covering

Figure 4: TDT3 data (clean, few tags and special characters)

For the ETD task, only the text within the <TEXT> tags are relevant, but FSD requires the extraction of the document number <DOCNO>, to identify stories.

For the TDT3 corpus, 60 explicit topic areas were outlined and developed by the Linguistic Data Consortium (University of Pennsylvania). These topics fall into one of 11 categories, including elections, scandals/hearings, legal/criminal cases, natural disasters, accidents, ongoing violence or war, science and discovery news, finances, new laws, sports new, or any other miscellaneous news. Figure 5 shows some of the topics relevant to the project.

Sample topics from TDT3 (October – December 1998)

Cambodian Government Coalition	Congolese Rebels vs. Pres. Kabila
Hurricane Mitch	Car Bomb in Jerusalem
Pinochet Trial	Anwar Ibrahim Case
Osama bin Laden Indictment	SwissAir111 Crash
Tony Blair Visits China in October	Thai Airbus Crash
Brazilian Presidential Elections	AOL - Netscape Merger
Euro Introduced	Indonesia - East Timor Conflict
Nobel Prizes Awarded	PanAm Lockerbie Bombing Trial
U.S. Mid-Term Elections	China Human Rights Treaty
North Korean Food Shortages	

Figure 5: Sample TDT3 topics [6]

4. Emerging Trend Detection

In this section, we will discuss the methodology for our research, including all necessary steps leading to the detection of emerging trends within selected documents of the TDT3 dataset.

4.1. Text Mining Infrastructure

A Textual Data Mining Infrastructure (TMI) tool, developed at Lehigh University, was also used in our experiments [2]. The TMI tool can be used for research in a number of text mining applications, including emerging trend detection. The TMI is written in C++ and runs under the Visual Studio .Net environment. It relies on a number of outside libraries, such as WEKA, a Machine Learning library, the JNI (Java Native Interface), and Flex (a lexical analysis tool) [2].

4.2. Feature and Term/Doc Generation

Our first step was to build a data repository using TMI. Our data was divided into four time-stamped repositories. These repositories included all English stories within that time frame.

Next, we integrated the data into the TMI application. Only the data within <TEXT> tags had to be indexed. This was done by modifying an existing parsing

function used for previous experiments done at Lehigh University on different datasets.

TMI was used to parse our repositories to generate terms and construct term lists and term/doc matrices.

Terms are words or group of words that can be used to refer to the content of a document [4]. In our case, terms were small noun phrases and the documents were single stories in the data set.

A *term/doc matrix* is a matrix whose rows correspond to documents and whose columns correspond to terms. In general, the number of terms greatly exceeds the number of documents. Additionally, the nature of text data leads to the creation of very sparse matrices, as there are few terms that appear in the majority of the documents. Very common words (known as stop-words) were not extracted.

TMI uses a sparse matrix format to store the terms and document as a doc/term matrix. We used the Parallel General Text Parser (PGTP) to decompose the sparse matrix as described in the next section [3].

4.3. Singular Value Decomposition

Singular Value Decomposition (SVD) was applied to the term-by-document matrices developed by TMI. The SVD process decomposes the matrix into three matrices: T, a term-by-dimension matrix; S, a singular value matrix (dimension-by-dimension); and D, a document-by-dimension matrix. The number of dimensions is

the rank of the term-by-document matrix. The decomposed matrices can be transformed into the original matrix through matrix multiplication [4].

Latent Semantic Indexing (LSI) is a common preprocessing step in information retrieval. LSI truncates the T, S and D matrices into k dimensions (the *k-value*) [4].

The SVD computations were performed on a supercomputer from the National Center for Supercomputing Applications (NCSA).

4.4. Term/Term Similarities

Following the SVD computations, we used a program written in C++ to calculate term/term similarities in order to produce term clusters. Term clustering is a common and effective technique in information retrieval applications. By clustering closely-related terms or documents, it is possible to include a wider range of possible target terms in or results.

The term/term program uses a cosine similarity computation to determine the vector similarity. The cosine similarity is calculated for each pair of rows in the term-by-dimension matrix and results in a value between -1 and +1 [4]. This value describes the strength of the relationship between two terms.

The appropriate cluster generation algorithm takes two main parameters: the truncation value, k, and a threshold, which determines the strength of the relationship.

A sparsification algorithm is used to improve retrieval quality and run time performance by removing a large portion of the entries of the SVD matrices. The algorithm we used is described in [13]. *Sparsification* signifies the percentage of values that can be removed without severely impacting retrieval performance.

4.5. The Truth Set

A truth set is a list of previously classified terms. It is structured as a simple text file listing a number of terms from the collection. Terms classified as emerging trends are marked as "trend", while terms that are not emerging trends are marked as "notrend".

Because a truth set for the TDT3 data was not available, a new truth set had to be created manually. This was done by classifying terms from each data set and applying the appropriate "trend" or "notrend" label. Because of the nature of the data (recorded news stories) the judgment was based on the topic indexes given by the Linguistic Data Consortium. The TDT3 corpus consists of 60 main evaluation topics. In order to create a truth set, the topics present in our data sample were identified by first comparing the appropriate dates of the events/stories and then searching for relevant terms. Once a main evaluation topic was identified, more relevant terms were identified and copied into the truth set. Within a main topic there are many terms, some of which describe new trends or new events and some which can not be classified as trend (e.g. term which mention previous events). In the non-obvious cases, fair judgment was made after reading the evaluation topic descriptions and making comparisons with other terms.

Different truth sets of approximately 160 terms each were compiled by the author for our testing purposes. Furthermore, a more stable truth set was developed using a student survey as described in the next section.

4.6. Truth Set Survey

Due to the subjectivity required when creating truth sets, a trend evaluation was performed. Previously, terms were selected according to their relevance with the standard topics of the TDT3 corpus. Then, using the knowledge of the data set and personal judgments, trends were defined as terms that played an increasing role in the context of the appropriate topic. The number of occurrences of certain terms was not weighted as highly as the author's opinion about whether or not the term describes a trend or a new topic.

The data sets allowed us to only sample a time frame of a number of days (etc. 8, 9, 12). Emerging trends may develop in a shorter time frame, but are harder to detect. An emerging trend in news stories must be defined as an emerging or new event. Taking this into account, it is important to view the results of the system in the context of the time-frame of the data set. A term could be identified as an emerging event in as little as two days, if the occurrences of that term rise continuously. On the other hand, if the time-frame is extended, most events would not qualify as emerging trends, as the number of occurrences will decrease greatly. Examples of the types of topics we want to detect are sudden events, like an accident (for example a plane crash), emerging events (an upcoming presidential election), re-occurring events (weekend sports). All these can be observed in different ways.

For the term evaluation, seven students in a computer science seminar class focusing on information retrieval were asked to analyze a list of terms and make their choice of whether a term is a trend or not. The students had a background in computer science and information retrieval applications and techniques.

Along with the list of 186 terms, the students received the actual data of news stories and a list of on-topic events, as outlined by the TDT3 consortium. The students were then asked to make their own judgments about each term, making a "yes" or "no" vote regarding the classification as an emerging trend. The general procedure was to identify whether the given terms matched one of the on-topic events. Then a student would analyze the occurrences of that term following the first occurrence in the data set. Based on these observations the final judgment would be made.

This approach was a different approach than the first one, but made more sense because the terms and topics were new to the students, and by analyzing the occurrences in the data sets they would use a more methodological approach (similar to the one used by the machine learning algorithm).

The judgments of the seven students were summarized and analyzed. On average, 82% of the total number of terms were classified as non-trends. The range of the portion of terms classified as trends spanned from a low 7% to a high 37%. To identify the final truth set, different types of measurements were evaluated.

a) Rule: at least fie of the same tendency

Taking the eight total truth sets (one previous and the seven student's sets), a term was classified as a trend, if five of the eight classifiers gave it a "yes" vote. The trend was classified as a non-trend, if five of the eight classifiers gave it a "no" vote. Terms yielding four to four votes (equal number of yes and no trends) were classified as "undecided". Using this rule, we obtained only five undecided cases, 162 non-trends, and 19 trends.

Increasing the number of necessary votes to six instead of five, we obtained 29 undecided cases, 147 non-trends, and 12 trends. Figure 6 shows sample terms.

	YES	NO	>= 5	>= 6
budget surplus	8	0	yes	yes
prime minister tony blair	6	2	yes	yes
hong kong	**3**	**5**	**no**	**undec**
premier zhu rongji	1	7	no	no
visit shanghai	1	7	no	no
russian president boris yeltsin	**3**	**5**	**no**	**undec**
russian parliament	2	6	no	no
industrial activity	1	7	no	no
congolese rebels	7	1	yes	yes
two-day visit	**3**	**5**	**no**	**undec**
zagreb friday	0	8	no	no

Figure 6: Examples

Figure 6 shows sample terms and their corresponding "yes" and "no" votes. Applying rule a) we obtain three trends and eight notrends. Increasing the number of necessary votes to six, we can remove three terms that do not qualify. These are shown in bold above.

b) Rule: at least four "yes" votes"

Here, just the number of "yes" votes was counted. Four votes would qualify a term as a trend. This approach yielded 24 trends and 162 non-trends. Figure 7 shows the first two rules applied to our sample terms.

	YES	NO	>= 5	>= 6	>=4 Y
budget surplus	8	0	yes	yes	yes
prime minister tony blair	6	2	yes	yes	yes
hong kong	**3**	**5**	**no**	**undec**	**no**
premier zhu rongji	1	7	no	no	no
visit shanghai	1	7	no	no	no
russian president boris yeltsin	**3**	**5**	**no**	**undec**	**no**
russian parliament	2	6	no	no	no
industrial activity	1	7	no	no	no
congolese rebels	7	1	yes	yes	yes
two-day visit	**3**	**5**	**no**	**undec**	**no**
zagreb friday	0	8	no	no	no

Figure 7: Examples (4 "yes" votes)

c) Dropping minimum and maximum voters

Next, only the data of the middle six classifiers (in terms of percentage of classified trends) was used, which would lead to a more stable result. Again, the previous rule (four "yes" votes) was applied, returning 18 trends and 168 non-trends. As this still included a lot of undecided and borderline cases, especially in the group of non-trends, only unanimous non-trends were included in the final results. This returned 92 non-trends. Figure 8 shows rule c) applied to our terms.

	YES	NO	>=4 Y
budget surplus	6	0	yes
prime minister tony blair	4	2	yes
hong kong	**1**	**5**	**no**
premier zhu rongji	**1**	**5**	**no**
visit shanghai	**1**	**5**	**no**
russian president boris yeltsin	**3**	**3**	**no**
russian parliament	**1**	**5**	**no**
industrial activity	0	6	no
congolese rebels	6	0	yes
two-day visit	**3**	**3**	**no**
zagreb friday	0	6	no

Figure 8: Examples (dropping minimum and maximum voters)

Figure 8 shows rule c) applied to the sample data set. Because only unanimous nontrends were included in the final truth set, a number of terms were removed (in bold). The remaining terms from this sample set included in the truth set would be “budget surplus”, “prime minister tony blair”, “industrial activity”, “congolese rebels”, and “zagreb friday”. This equals three trends and two nontrends.

d) Removing more non-trends

The number of non-trends was reduced to a final number of 75, by using only trends that were unanimously classified as non-trends by all eight (including the two truth sets that were dropped previously) votes. Figure 9 shows a summary of the final truth set used.

In our example, this rule would erase “industrial activity”, as this received seven out of eight notrend votes considering all eight voters.

18 trends	19.30%
75 notrends	80.70%
93 total	**100%**

Figure 9: Breakdown of survey results

As we will see in the next section, this truth set (summarized in figure 9) returned very good results. However, after analyzing the correspondent decision-trees and the non-trend terms, it showed that the non-trends were mostly terms just appearing once in the data set (and therefore clearly identifying them as non-trends). This will be analyzed in the corresponding experimental results section.

4.7. Evaluation

The emerging trends were detected using a machine-learning tool called WEKA. WEKA is an open-source collection of machine learning algorithms that can be applied to datasets or called via an Application Programming Interface (API) [4].

WEKA was used to create decision trees which represent optimal models for each test collection. Additionally, statistical results were collected to measure the performance of our experiment. We used two metrics (precision and recall) which are commonly used in information retrieval applications:

P = True Positives / (True Positives + False Positives)

R = True Positives / (True Positives + False Negatives)

The precision rate evaluates the number of trends mistakenly detected as notrend, and the recall rate evaluates the number of notrends mistakenly detected as trends.

5. Results

In this section, the results of our experiments are presented.

5.1. 9-Day Sample

A small data sample using the first eight days of the TDT3 corpus was used for testing the ETD system (October 1st – October 8th). The following four repositories were created: “latest two days”, “previous two days”, “second previous two days”, “all other previous days”.

The original noun phrase extraction settings according to following the regular expression shown in (1) was used to generate the terms. These settings have previously shown good results.

(1) C?(G|P|J|)*N+(I*D?C?(G|P|J)*N+)*

In this expression, C is a cardinal number, G is a verb (gerund or past participle), P is a verb (past participle), J is an adjective, N is a noun, I is a preposition, D is a determiner, ? indicates 0 or 1 occurrence, * indicated 0 or more occurrences, and + indicates 1 or more occurrences [4].

After generating the term and term/doc files, the SVD was computed and the results of the first two repository files used for the term cluster were generated. The test files of this data set had the properties shown in figure 10.

	Latest two days	Previous two days
# of Items	22591	18223
# of Documents	835	514

Figure 10: Data characteristics

Term clusters were created for k-values of 25, 30, 35, 40, and 45 (see section 4.3). Sparsification levels from 0% to 90% were used (see section 4.4). Finally, a threshold of 1.0 was used. We used different replication factors to develop our models. The replication factor was incremented for each experiment until the performance remained constant [4]. The following table shows results with replication factors 4x and 10x. Using these tests and training sets the emerging trend detection was performed. Figure 11 shows a sample of results.

k	Sparsification	Replication	Recall	Precision
35	60%	4	.72	.29
40	0%	4	.61	.41
40	40%	4	.65	.33
25	80%	10	.74	.25
30	70%	10	.73	.28
35	80%	10	.79	.23

Figure 11: Settings and results

Figure 12 shows the generated decision tree for k-value of 35 (truncation value) with sparsification of 80% (we use different sparsification levels to see how many values of our matrices we can remove and still obtain good retrieval results [4]). This is the best result we obtained in this experiment, with a recall rate of .79 and a precision rate of .23. In later experiments we were able to improve the recall and the precision rate significantly.

```
Occurrences_in_Current_timeframe <= 5
|  Concepts_in_Cluster <= 13
|  |  Occurrences_in_Timeframe_before_Previous_timeframe <= 1
|  |  |  Occurrences_in_Previous_timeframe <= 0: trend (303.0/83.0)
|  |  |  Occurrences_in_Previous_timeframe > 0
|  |  |  |  Occurrences_in_Timeframe_before_Previous_timeframe <= 0: notrend (11.0)
|  |  |  |  Occurrences_in_Timeframe_before_Previous_timeframe > 0: trend (11.0/1.0)
|  |  Occurrences_in_Timeframe_before_Previous_timeframe > 1
|  |  |  Occurrences_in_All_Noncurrent_timeframes <= 4: notrend (3.0)
|  |  |  Occurrences_in_All_Noncurrent_timeframes > 4: trend (20.0)
|  Concepts_in_Cluster > 13
|  |  Occurrences_in_Previous_timeframe <= 2
|  |  |  Occurrences_in_All_Noncurrent_timeframes <= 0: trend (79.0/9.0)
|  |  |  Occurrences_in_All_Noncurrent_timeframes > 0
|  |  |  |  Long_Words_In_Feature <= 2: notrend (5.0)
|  |  |  |  Long_Words_In_Feature > 2: trend (10.0)
|  |  Occurrences_in_Previous_timeframe > 2: trend (10.0)
Occurrences_in_Current_timeframe > 5
|  Occurrences_in_Current_timeframe <= 11: trend (30.0)
|  Occurrences_in_Current_timeframe > 11: notrend (2.0)

Number of Leaves  :        11

Size of the tree :         21
```

Figure 12: Decision tree

5.2. 9-Day Sample (smaller noun phrases) – Threshold of 0.9

The same TDT3 data sample was used in a second test. This time, the feature extraction equation was modified in order to obtain more usable terms. For the clustering calculations, a threshold of 0.9 was used.

According to the new feature set, a truth file was compiled and used for the model building process. Figure 13 shows the results for this experiment.

k	Sparsification	Replication	Recall	Precision
5	0%	10	.80	.42
5	70%	10	.75	.37
5	9%	10	.85	.38
10	90%	10	.85	.38

Figure 13: Settings and results

The results show better recall- and precision rates. Figure 14 shows the generated decision tree for k-value of 5 with no sparsification:

```
Occurrences_in_Current_timeframe <= 0
|  Concepts_in_Cluster_Previous_timeframe <= 2938
|  |  Occurrences_in_All_Noncurrent_timeframes <= 5: notrend (25.0)
|  |  Occurrences_in_All_Noncurrent_timeframes > 5: trend (11.0/1.0)
|  Concepts_in_Cluster_Previous_timeframe > 2938
|  |  Long_Words_In_Feature <= 2: trend (66.0/6.0)
|  |  Long_Words_In_Feature > 2
|  |  |  Long_Words_In_Feature <= 3: trend (21.0/1.0)
|  |  |  Long_Words_In_Feature > 3: notrend (4.0)
Occurrences_in_Current_timeframe > 0
|  Occurrences_in_Previous_timeframe <= 1
|  |  Occurrences_in_All_Noncurrent_timeframes <= 0
|  |  |  Long_Words_In_Feature <= 2
|  |  |  |  Concepts_in_Cluster <= 1810
|  |  |  |  |  Concepts_in_Cluster <= 1611
|  |  |  |  |  |  Concepts_in_Cluster <= 1420: trend (11.0/1.0)
|  |  |  |  |  |  Concepts_in_Cluster > 1420: notrend (3.0)
|  |  |  |  |  Concepts_in_Cluster > 1611: trend (30.0)
|  |  |  |  Concepts_in_Cluster > 1810: notrend (2.0)
|  |  |  Long_Words_In_Feature > 2: trend (31.0/1.0)
|  |  Occurrences_in_All_Noncurrent_timeframes > 0: notrend (4.0)
```

```
|  Occurrences_in_Previous_timeframe > 1
|  |  Occurrences_in_Previous_timeframe <= 7: trend (40.0)
|  |  Occurrences_in_Previous_timeframe > 7
|  |  |  Occurrences_in_Previous_timeframe <= 17: trend (10.0)
|  |  |  Occurrences_in_Previous_timeframe > 17: notrend (2.0)

Number of Leaves  :     14

Size of the tree :      27
```

Figure 14: Decision tree

Figure 15 shows the generated decision tree for k-value of 5 with sparsification of 70%:

```
Occurrences_in_Current_timeframe <= 0
|  Concepts_in_Cluster_Previous_timeframe <= 354
|  |  Occurrences_in_Previous_timeframe <= 3
|  |  |  Long_Words_In_Feature <= 2: notrend (19.0)
|  |  |  Long_Words_In_Feature > 2
|  |  |  |  Occurrences_in_All_Noncurrent_timeframes <= 1: trend (11.0/1.0)
|  |  |  |  Occurrences_in_All_Noncurrent_timeframes > 1: notrend (6.0)
|  |  Occurrences_in_Previous_timeframe > 3: trend (11.0/1.0)
|  Concepts_in_Cluster_Previous_timeframe > 354
|  |  Long_Words_In_Feature <= 2: trend (65.0/5.0)
|  |  Long_Words_In_Feature > 2
|  |  |  Long_Words_In_Feature <= 3: trend (12.0/2.0)
|  |  |  Long_Words_In_Feature > 3: notrend (3.0)
Occurrences_in_Current_timeframe > 0
|  Occurrences_in_Previous_timeframe <= 1
|  |  Occurrences_in_All_Noncurrent_timeframes <= 0: trend (77.0/7.0)
|  |  Occurrences_in_All_Noncurrent_timeframes > 0: notrend (4.0)
|  Occurrences_in_Previous_timeframe > 1
|  |  Occurrences_in_Previous_timeframe <= 7: trend (40.0)
|  |  Occurrences_in_Previous_timeframe > 7
|  |  |  Occurrences_in_Previous_timeframe <= 17: trend (10.0)
|  |  |  Occurrences_in_Previous_timeframe > 17: notrend (2.0)

Number of Leaves  :     12

Size of the tree :      23
```

Figure 15: Decision tree

These decision trees show the steps of the machine learning algorithm in determining whether a term is an emerging trend. Decisions are made according to the number of occurrences in the different timeframes. For example, if the term occurs in the current timeframe ("occurrences_in_current_timeframe > 0") then the number of occurrences in the previous timeframe is evaluated. If this number is lower or equal to 7, the term is declared a trend, otherwise more evaluations are being made.

"Concepts_in_cluster" indicates that our term/term clusters are being used in the detection process.

5.3. 9-Day Sample (smaller noun phrases) – Threshold of 0.7

We used the same data set but used a threshold of 0.7. Figure 16 shows the results for this experiment.

k	Sparsification	Replication	Recall	Precision
5	0%	10	.77	.41
5	10%	10	.75	.36
5	90%	10	.85	.38
10	80%	10	.65	.38
10	90%	10	.85	.38

Figure 16: Settings and results

The results are slightly worse than those of experiment (3.2). Figure 17 shows the generated decision tree for k-value of 5 with sparsification of 10%:

```
Occurrences_in_Current_timeframe <= 0
|  Concepts_in_Cluster_Previous_timeframe <= 2873
|  |  Occurrences_in_Previous_timeframe <= 3: notrend (24.0)
|  |  Occurrences_in_Previous_timeframe > 3: trend (10.0)
|  Concepts_in_Cluster_Previous_timeframe > 2873
|  |  Long_Words_In_Feature <= 2: trend (68.0/8.0)
|  |  Long_Words_In_Feature > 2
|  |  |  Long_Words_In_Feature <= 3: trend (21.0/1.0)
|  |  |  Long_Words_In_Feature > 3: notrend (4.0)
Occurrences_in_Current_timeframe > 0
|  Occurrences_in_Previous_timeframe <= 1
|  |  Occurrences_in_All_Noncurrent_timeframes <= 0: trend (77.0/7.0)
|  |  Occurrences_in_All_Noncurrent_timeframes > 0: notrend (4.0)
|  Occurrences_in_Previous_timeframe > 1
|  |  Occurrences_in_Previous_timeframe <= 7: trend (40.0)
|  |  Occurrences_in_Previous_timeframe > 7
|  |  |  Occurrences_in_Previous_timeframe <= 17: trend (10.0)
|  |  |  Occurrences_in_Previous_timeframe > 17: notrend (2.0)

Number of Leaves  :         10

Size of the tree :          19
```

Figure 17: Decision tree

Figure 18 shows the generated decision tree for k-value of 10 with sparsification of 100%.

```
Occurrences_in_Current_timeframe <= 0
|  Long_Words_In_Feature <= 3
|  |  Occurrences_in_Previous_timeframe <= 1
|  |  |  Occurrences_in_All_Noncurrent_timeframes <= 1
|  |  |  |  Long_Words_In_Feature <= 2: notrend (22.0/10.0)
|  |  |  |  Long_Words_In_Feature > 2: trend (11.0/1.0)
|  |  |  Occurrences_in_All_Noncurrent_timeframes > 1: notrend (3.0)
|  |  Occurrences_in_Previous_timeframe > 1: trend (85.0/15.0)
|  Long_Words_In_Feature > 3: notrend (6.0)
Occurrences_in_Current_timeframe > 0
|  Occurrences_in_Previous_timeframe <= 1
```

```
|   |   Occurrences_in_All_Noncurrent_timeframes <= 0: trend (77.0/7.0)
|   |   Occurrences_in_All_Noncurrent_timeframes > 0: notrend (4.0)
|   Occurrences_in_Previous_timeframe > 1
|   |   Occurrences_in_Previous_timeframe <= 7: trend (40.0)
|   |   Occurrences_in_Previous_timeframe > 7
|   |   |   Occurrences_in_Previous_timeframe <= 17: trend (10.0)
|   |   |   Occurrences_in_Previous_timeframe > 17: notrend (2.0)

Number of Leaves  :     10

Size of the tree :      19
```

Figure 18: Decision tree

5.4. 10-Day Sample (updated Truth Set)

Using the same parameters as in experiment 3.1 (but using smaller noun phrases), we applied the newly obtained truth set resulting from a student survey. This truth set consisted of 93 terms of which 18 were classified as trends and 75 as non-trend. Figure 19 summarizes the results, using precision and recall rates. Selected results are shown:

k	Sparsification	Replication	Recall	Precision
5	60%	1	.86	.85
10	0%	1	.86	.85
5	30%	10	.90	.75
10	10%	10	.86	.80
10	40%	10	.90	.80

Figure 19: Settings and results

The best result was obtained a k=10 with sparsification of 40%. Figure 20 shows the corresponding decision-tree:

```
Occurrences_in_All_Noncurrent_timeframes <= 0: notrend (33.0)
Occurrences_in_All_Noncurrent_timeframes > 0
|   Concepts_in_Cluster_Previous_timeframe <= 1: trend (161.0/1.0)
|   Concepts_in_Cluster_Previous_timeframe > 1
|   |   Long_Words_In_Feature <= 2: notrend (4.0)
|   |   Long_Words_In_Feature > 2: trend (10.0)

Number of Leaves  :         4

Size of the tree :          7
```

Figure 20: Decision tree

The decision tree in figure 21 clarifies problems occurred with the new truth set:

```
Occurrences_in_All_Noncurrent_timeframes <= 0: notrend (33.0)
Occurrences_in_All_Noncurrent_timeframes > 0
|   Occurrences_in_Current_timeframe <= 2
|   |   Concepts_in_Cluster_Previous_timeframe <= 0
|   |   |   Occurrences_in_Current_timeframe <= 1: trend (21.0/1.0)
|   |   |   Occurrences_in_Current_timeframe > 1: notrend (2.0)
|   |   Concepts_in_Cluster_Previous_timeframe > 0: notrend (2.0)
|   Occurrences_in_Current_timeframe > 2: trend (150.0)
```

Figure 21: Decision tree

This is the tree for a k-value of 10 with sparsification of 60%, which returned a recall rate of .82 and a precision rate of .80.

Several points in the decision trees raise further questions. Looking at the first decision tree, zero occurrences in all non-current years leads to classification as

non-trend. An analysis of the actual occurrences of the non-trend terms of our truth set came to the conclusion that the non-trend terms were mainly terms that had just a single occurrence in the data set. Consequently, they received unanimous “no” votes in the term evaluation. However, because of the majority of non-trends being these terms, a good decision model cannot be created.

In order to test the machine learning algorithm again, the truth set was modified to include more borderline-cases (i.e. terms which did receive a majority “no” vote, but which were not unanimous).

5.5. Repeated experiment with updated truth set

After modifying the truth set to include more borderline-cases, and removing some of the single-occurrence terms, new experiments were made, which yielded very good results. The results using the same settings are summarized in the figure 22:

k	Sparsification	Replication	Recall	Precision
5	30%	10	.90	.80
10	10%	10	.85	.80
10	40%	10	.90	.82
15	0%	10	.95	.80
15	30%	10	.95	.88

Figure 22: Settings and results

The first three rows include the same settings as in the previous experiment are shows for comparison. The 4th and 5th row show the best results received, for a k-value of 15 with sparsification of 0% and 30%, respectively. Figure 23 shows the decision tree for a k-value of 15 with sparsification of 30%.

```
Occurrences_in_All_Noncurrent_timeframes <= 0: notrend (27.0)
Occurrences_in_All_Noncurrent_timeframes > 0
|   Occurrences_in_All_Noncurrent_timeframes <= 2
|   |   Occurrences_in_Current_timeframe <= 3
|   |   |   Concepts_in_Cluster <= 3: notrend (8.0)
|   |   |   Concepts_in_Cluster > 3: trend (10.0)
|   |   Occurrences_in_Current_timeframe > 3: trend (20.0)
|   Occurrences_in_All_Noncurrent_timeframes > 2: trend (160.0)

Number of Leaves  :         5

Size of the tree :          9
```

Figure 23: Decision tree

5.5. Short Conclusion

Our ETD experiments showed the difficulties existing in the nature of the information available and in the different of opinions in the selection of emerging trends. The nature of our experiments relies on a very stable truth set. The results of our tests dramatically improved with our new truth sets.

Our experiments also showed the efficacy of the term/term cluster generation. Clusters were used in most of the decision trees generated to find the emerging trends.

6. First-Story Detection

In this section, we will discuss the methodology of our research involving First-Story Detection (FSD). The First-Story Detection task is one of the original tasks of the TDT (Topic Detection and Tracking) research initiative, the others being Story Segmentation, Topic Tracking, Topic Detection, and Link Detection. FSD is closely linked to Topic Detection (see figure 24), a process that builds clusters of stories that discuss the same topic area or event [6].

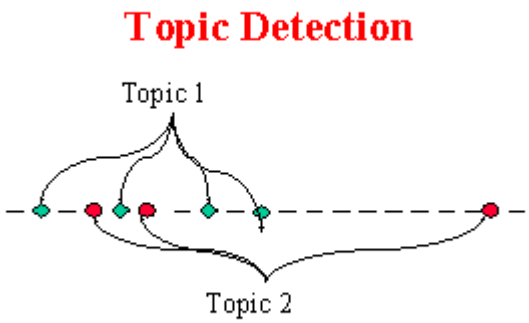

Figure 24: Topic Detection [6] of stories on a timeline

Comparable to the topic detection task, FSD evaluates the corpus and finds stories that are discussing a new event. Therefore, FSD is a more specialized version of topic detection, because in Topic Detection the system has to determine when a new topic is being discussed. The resulting stories are the “first-stories” we want to retrieve. Figure 25 shows the FSD process.

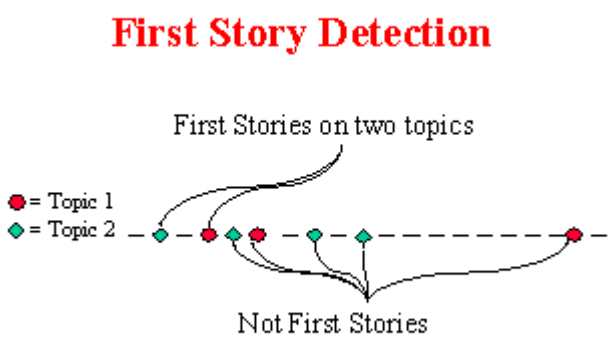

Figure 25: First-Story Detection [6]

A First-Story Detection system runs sequentially and creates immediate results while running through the dataset. In our research, we want to apply some of our current methods and techniques used in ETD to the First-Story Detection task.

The current ETD system is not applicable for FSD computations, so the experimental methodology is very different. For the TDT3 corpus, we have a listing of on-topic stories for each of the 60 outlined topics. Using this truth file, we are able to generate a truth file for the FSD task, and directly implement it into our FSD system. Using this method, we are able to generate immediate results and statistical analysis. Additionally, due to the reliability of the file our results will easier to compare to others, unlike the ETD task, where the truth set was developed manually.

In the following sections we will discuss some of the approaches made in FSD and our own approach, which includes a truth set evaluation, the experimentation itself, and the statistical analysis of the results.

6.1. Previous Approaches

Research in Topic Detection and Tracking started in 1996 with a pilot study (DARPA, University of Massachusetts). In the following, we will discuss several different approaches.

The UMass (University of Massachusetts) approach is based on a clustering-approach of the streaming documents that returns the first document in each cluster as result [10]. Document clusters are groups of documents that appear to be similar in content. Using this approach, and combining it with previously-known solutions to clustering they implemented a modified version of the single-pass (making just one pass through the data) clustering algorithm for the new-story detection. By using a single-pass clustering algorithm, it is possible to run through the stories sequentially, as it is necessary for the FSD task.

In addition to this type of implementation, a rank-retrieval mechanisms, a feature-extraction and selection process based on relevance feedback and a special routing architecture was combined in the FSD process [10].

The UPenn (University of Pennsylvania) approach uses the “single-link” or “nearest-neighbor” technique. This technique stores all stories in clusters of size one, and then merges clusters, if similarities between two clusters are higher than a certain threshold. For the clustering process, a deferral period is defined to be the number of files (including a number of stories) the system is allowed before it relates an event with the stories of that file. An inverted index is then created. After that, all stories are compared to the preceding ones, including stories from a previous deferral period. When the similarity is high enough, their clusters are merged. If a story can

not be combined with any other existing cluster, it becomes a new cluster. These new clusters can be considered new events, thus the story is a first-story [11].

The CMU (Carnegie Mellon University) approach uses the vector-space model to represent each story. It then uses traditional clustering techniques to represent the events. A story is stored as a vector whose dimensions are unique terms from the dataset, and whose elements are the term weights in the story. Term weighting occurs according to simple rules where for example high-frequency terms (which appear in many stories) receive lower weights than terms that seem to have a higher importance in a particular story but not for the rest of the dataset [8, 11]. For the clustering and First-Story Detection, a single-pass algorithm was used. The CMU approach is very similar to our approach, as it uses the same term-weighting system.

6.2. Our Approach

We use a term-weighting system called TF-IDF (Term Frequency times Inverse Document Frequency). Using this technique, each term is assigned with a weight. Our current system uses a two-pass algorithm to assign weights to the extracted terms and store each term and story in a document by term matrix. This is done for programming efficiency and does not violate the spirit of the FSD task. The markup algorithm uses a single-pass algorithm. Several pre-processing steps are performed.

Stop words, words that frequently appear in the English language) are excluded (see example in figure 26). No additional term-filtering or optimization techniques are applied.

Sample stop words		
B	be	became
Because	become	becomes
Becoming	been	before
Beforehand	behind	being
Believe	below	beside

Figure 26: Excerpt of the stop word list

In our FSD system, we look at one source of news stories at a time (for example NBC). The appropriate files are extracted from the whole file list and then combined into a single input file for our program. This input file contains all news stories of the source in a time-sequential order. Additionally, all data is converted to lowercase characters, and special characters are removed to allow for faster and better analysis.

Terms are extracted from the each story (only terms in the title or body of the story were used). Using the number of occurrences of each term within a story and within the corpus, TF-IDF weighting is computed to the term frequency (see figure 27) using a global term-weight. Results are stored in a document by term matrix.

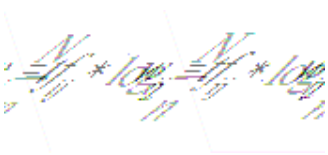

Figure 27: TF-IDF weighting formula [12]

Using the formula from figure 27, we can assign weights to our terms:

w_{ij} = weight of Term T_j in Document D_i

tf_{ij} = frequency of Term T_j in Document D_i

N = number of Documents in collection

n = number of Documents where term T_j occurs at least once

The algorithm assigns a value to each story (the "FSD-value"). The lower the value, the more likely it is that the story is a first-story.

Several methods are used in the acquisition of the FSD-value. As a general rule, the more new high-weighted terms in a story, the more likely it is that the story is a first-story.

The system runs and evaluates every story sequentially:

- The first story in a collection is always a first-story (FSD-value 0).

- The second story is evaluated by calculating the occurrences of terms that were in the previous story, thus calculating a measurement of similarity. In our current approach, this story will most likely be a first-story too.

- We continue these steps for each subsequent story. If a story contains a high number of previously unknown terms, the FSD-value will be lower. If the FSD-value is under a determined value, the story is identified as a first-story.

Results are stored to a file after the evaluation of each story. After passing all stories in our collection, statistical analysis is performed to measure the performance of our system.

6.3. Truth Set Evaluation

Unlike in the ETD task, we already have a stable truth set of valid on-topic stories for each of the 120 test topics. Thus it is very easy to produce a specialized truth file for each of the news sources by extracting the appropriate first-stories from the list. However, the news stories contain a lot more events than the 60 test events laid out in the TDT3 study; therefore it is only possible to evaluate our results for first-stories for these events and not for every event.

Figure 28 shows the raw truth file including a list of all relevant stories for a certain topic in the collection.

<ONTOPIC topicid=30001 level=BRIEF docno=NYT19981228.0443
fileid=19981228_2055_2252_NYT_NYT comments="NO">
<ONTOPIC topicid=30001 level=BRIEF docno=NYT19981229.0004
fileid=19981229_0020_1816_NYT_NYT comments="New in v2.0">
<ONTOPIC topicid=30002 level=YES
docno=CNN19981023.0130.0446
fileid=19981023_0130_0200_CNN_HDL comments="NO">
<ONTOPIC topicid=30002 level=YES docno=NYT19981023.0231
fileid=19981023_0041_1801_NYT_NYT comments="NO">
<ONTOPIC topicid=30002 level=YES docno=CNN19981023.1130.0323
fileid=19981023_1130_1200_CNN_HDL comments="NO">
<ONTOPIC topicid=30002 level=YES docno=CNN19981023.1600.0351
fileid=19981023_1600_1630_CNN_HDL comments="NO">

Figure 28: TDT3 relevant documents file

In this excerpt, the 3rd document (in bold) is a first-story, as it is the first story that is relevant for topic ID 30002. All following stories associated with this topic ID are still on-topic, but not of importance in our FSD research.

Note that the TDT truth file contains all relevant stories, regardless of source. In the example above, the first-story is a CNN document. However, we are running our FSD experiments for one news source at a time, therefore the first-story for a New York Times experiment would be NYT19981023.0231.

We first extract all relevant stories of the source we are using for our experimentation from the truth file (for example: CNN, see figure 29).

CNN TruthSet

30002 CNN19981023.0130.0446
30002 CNN19981023.1130.0323
30002 CNN19981023.1600.0351
30002 CNN19981023.2130.0265
30002 CNN19981024.1000.0402
30002 CNN19981024.1130.0461
30002 CNN19981024.1300.0378

Figure 29: Extracted truth set for the CNN collection

This list includes all on-topic documents for each of the 120 standard topics (if such a document exists in this collection). To obtain a list of first-stories, we are extracting all the first stories related to a topic and obtain our final truth-file, such as in Figure 30.

CNN TruthSet (only first stories)

30002 CNN19981023.0130.0446
30003 CNN19981017.1000.0129
30004 CNN19981221.1130.0541
30005 CNN19981020.2130.0238
30006 CNN19981005.1600.1233
30007 CNN19981010.1600.0457

Figure 30: CNN FSD truth-set

We use this file to evaluate our algorithm.

6.3. Experimentation and Statistical Analysis

Because we only have a fixed number of first-stories available, our aim is to produce a system that creates a high recall rate. That is, we want to be able to detect all (or a high number) of first-stories from our truth set.

We use the same measurement as we did for the ETD task (P = precision, R = recall).

(1) P = True Positives / (True Positives + False Positives)

(2) R = True Positives / (True Positives + False Negatives)

In the following section, we will discuss the results of our experiments.

7. Results

7.1 MS-NBC Test

For this experiment, we used all MS-NBC stories from the TDT3 collection. The data was preprocessed to eliminate special characters, and converted to lowercase. Figure 31 contains information about the MS-NBC collection.

MS-NBC collection overview

Number of stories:	683
Filesize:	2.04 MB
Lines of data:	37292
Timeframe:	10/01 - 12/31
First stories in truth file:	39

Figure 31: Collection overview

The truth set evaluation returned 39 first stories for the MS-NBC data of the TDT3 collection. This means that from the 120 standard topics, 39 were included in this collection. Our application returned 124 first-stories, and was able to detect 13 stories included in the MS-NBC truth set for a recall rate of .33.

Sample truth set evaluation: MS-NBC test

```
30002  MNB19981023.2100.1558    no
30003  MNB19981028.2100.3203    ok
30004  MNB19981218.2100.1558    no
30006  MNB19981013.2100.1687    ok
30012  MNB19981111.2100.2168    no
30015  MNB19981005.2100.2319    ok
30016  MNB19981015.2100.0266    ok
30017  MNB19981120.2100.1804    no
30020  MNB19981112.2100.3485    no
30023  MNB19981123.2100.0069    no
30024  MNB19981103.2100.0941    no
30026  MNB19981123.2100.1331    no
30027  MNB19981013.2100.0063    ok
```

Figure 32: MS-NBC sample results

Figure 32 shows a sample of the results retrieved by our application. The first number in each row represents the topic identification number (e.g. 30002). Because there are 120 standard topics, these topic IDs start with 30001 and go to 30060, then start from 31001 through 31060 (there are two sets of 60 topics each). Looking at the sample above, we can tell that the MS-NBC collection does not contain official first stories for a large number of these standard topics.

The 2nd column in each row represents the story ID, including collection name, date, time, and identification number. This ID represents the first story for the appropriate topic. The third column signifies if this story was detected by our FSD system (ok = story detected correctly / no = story not in our results list).

7.2 NBC Test

We combined all stories from the NBC (National Broadcasting Company) dataset.

NBC collection overview	
Number of stories:	846
Filesize:	1.91 MB
Lines of data:	36953
Timeframe:	10/01 - 12/31
First Stories in truth file:	59

Figure 33: Collection overview

The truth set evaluation yielded 59 first stories for the NBC data of the TDT3 collection. Compared to the previous test, even though the file size was about equal, the NBC collection included a higher number of stories and the truth file had a higher number of first stories, which would improve the significance of our tests.

Our application was able to detect 29 stories included in the NBC truth set for a recall rate of .491. Figure 34 shows an extract of our truth set evaluation file:

Sample truth set evaluation: NBC test

30059	NBC19981121.1830.0637	no
31001	NBC19981029.1830.1391	ok
31013	NBC19981024.1830.0061	ok
31022	NBC19981030.1830.0385	no
31026	NBC19981017.1830.1375	ok
31031	NBC19981008.1830.0969	ok
31033	NBC19981018.1830.0645	ok
31034	NBC19981108.1830.0898	no
31035	NBC19981003.1830.0062	ok
31036	NBC19981125.1830.0634	ok
31044	NBC19981112.1830.0596	no
31003	NBC19981204.1830.0723	ok
31007	NBC19981021.1830.1238	ok
31008	NBC19981009.1830.1238	ok
31028	NBC19981021.1830.0927	no
31030	NBC19981104.1830.1016	ok
31032	NBC19981012.1830.1224	ok
31038	NBC19981010.1830.1030	ok
31039	NBC19981002.1830.1273	ok
31047	NBC19981207.1830.1595	no

Figure 34: NBC results sample

The table above shows a particularly good sample of the results retrieved by our application, with 14 out of 20 correctly identified first-stories. Figure 35 shows an extract of the actual output file created by our system, which included the detected first stories. All retrieved stories from between November 10th and November 25th are shown.

Sample output: NBC test

nbc19981110.1830.1244	nbc19981110.1830.1579
nbc19981111.1830.1406	nbc19981112.1830.1285
nbc19981113.1830.0765	nbc19981113.1830.1345
nbc19981113.1830.1565	nbc19981115.1830.0704
nbc19981115.1830.0968	nbc19981115.1830.1156
nbc19981115.1830.1480	nbc19981115.1830.1624
nbc19981116.1830.0758	nbc19981117.1830.0816
nbc19981118.1830.1195	nbc19981119.1830.0738
nbc19981119.1830.1238	nbc19981119.1830.1592
nbc19981120.1830.1367	nbc19981121.1830.1204
nbc19981121.1830.1599	nbc19981123.1830.0485
nbc19981124.1830.1725	nbc19981125.1830.0634

Figure 35: Sample output for NBC test

There are a number of days within this timeframe that have no or just one first-story returned, which is an unusually low number. This is reasonable, as our system takes into account which terms have already occurred in previous stories. The later a story appears in the data set timeframe, the less probable is it that this story is being detected as a first story.

However, our system shows some good results for first stories in November and December. In several cases the application retrieved just one single story for a day, but this story was indeed a first-story.

A remaining problem was the improvement of the precision of our system. As a general tendency, towards the beginning of the collection, more stories are retrieved. In this time period the application was able to predict almost all the first-

stories, however, the precision rate was very low. After a number of processed stories, he number of stories retrieved per day gradually decreases.

It makes sense that more stories will be labeled first stories in the first few days of the system run, as there is no "history" of stories before the beginning of our timeframe (October 1st). However, in the following test, the threshold was raised as more stories in the database are processed to obtain a higher-quality results set but with an equally strong recall rate.

7.3 ABC Test

Figure 36 shows the statistics for the ABC (American Broadcasting Company) collection of the TDT3 data set.

ABC collection overview	
Number of stories:	1481
Filesize:	1.65 MB
Lines of data:	33559
Timeframe:	10/01 - 12/31
First stories in truth file:	61

Figure 36: Collection overview

This test was performed using a modified algorithm, which took into account the location of the story to be processed:

- Our goal is to retrieve fewer stories from the beginning of our dataset and more from the end.
- Each story in the collection is numbered. According to their number, a different rule is applied in order to make the FSD decision.
- Start out with stricter rules to qualify as first story. This means that more new, previously unknown terms are required to qualify as a first story.
- The higher the story number (i.e. the longer the application is running), the fewer new terms are necessary to qualify as a first story.

The truth set evaluation yielded 61 first stories. Our application was able to detect 24 stories included in the ABC truth set for a recall rate of .393.Figure 37 shows an extract of our truth set evaluation file:

Sample truth set evaluation: ABC test

30012	ABC19981117.1830.0825	no
30013	ABC19981110.1830.0311	ok
30014	ABC19981018.1830.0414	ok
30015	ABC19981005.1830.0602	no
30016	ABC19981001.1830.0750	ok
30021	ABC19981211.1830.0819	ok
30022	ABC19981202.1830.0580	no
30023	ABC19981123.1830.0810	ok
30024	ABC19981105.1830.0064	no
30025	ABC19981004.1830.0594	ok
30026	ABC19981123.1830.0189	no
30027	ABC19981007.1830.1038	no
30029	ABC19981228.1830.0455	ok
30031	ABC19981120.1830.1325	no
30033	ABC19981218.1830.0300	no
30036	ABC19981012.1830.0653	ok
30037	ABC19981009.1830.0996	ok

Figure 37: ABC results sample

Figure 37 a sample of the results retrieved by our application. The recall rate of .393 was slightly lower than in our NBC test, but was still very reasonable. Figure 38 shows an extract of the actual output file created by our system, which included the detected first stories.

Sample output: ABC test

abc19981223.1830.0221	abc19981223.1830.0438
abc19981223.1830.1042	abc19981223.1830.1559
abc19981224.1830.0314	abc19981224.1830.0709
abc19981226.1830.0161	abc19981226.1830.0708
abc19981226.1830.1424	abc19981226.1830.1537
abc19981227.1830.0397	abc19981227.1830.1057
abc19981227.1830.1074	abc19981228.1830.0000
abc19981228.1830.0322	abc19981228.1830.0342
abc19981228.1830.0455	abc19981228.1830.0676
abc19981228.1830.0832	abc19981228.1830.1637
abc19981229.1830.0544	abc19981229.1830.1102
abc19981229.1830.1470	abc19981229.1830.1606
abc19981230.1830.0684	abc19981231.1830.0070

Figure 38: Sample output for ABC test

This sample output shows some major differences from the previous tests. First, the total number of retrieved first stories was only 279, which are 18% of the 1481 total number of stories. The precision rate is still very low at about 1%; however it must be noted that our system retrieves all first-stories, not just the first-stories for the 120 TDT3 standard topics.

Additionally, the stories were a lot more evenly distributed with less first stories in the first part of our collection and more in the later parts (as compared to previous tests).

7.4. Short Conclusion

Our FSD system was able to detect a relatively high number of the first stories included in our truth files. Our experiments show that our simple approach can be very effective.

8. Future Work and Conclusion

The use of term clusters for ETD has been proven to be a very effective technique. The term clusters were widely used in the machine learning process of our ETD application and produced very good results.

The experiments with optimized truth sets generated very good results, with recall rates up to 95% and precision rates up to 88%. Optimization of the ETD results could be obtained by improving the feature extraction tool to generate better terms and changing parameters for the term cluster generation (e.g. the threshold) and the machine learning tool WEKA.

However, to be able to test the ETD application more reliably, it is necessary to develop stable data collections and truth sets. It was shown that minor differences in the definition of emerging trends and in the choice of terms for the truth set, had a great impact on the results of our experiments.

It must be noted, that the results of our ETD applications vary greatly by collection. In our experiments we used news stories from broadcasting sources. These type of stories will naturally produce different sorts of trends than for example a collection of scientific papers and articles.

For the First Story Detection task there exist a number of possible improvements or enhancement to optimize the system and create better results. Because of the limitation of a single-pass algorithm (the FSD task requires the sequential processing of time-ordered stories) it is very hard to implement effective

weighting techniques. Future work must include optimizing these term weighting algorithms and apply them to our current system.

Possible optimizations:

- Elimination of very common terms which are not stop words, but typical to the collection used
- Extraction of noun phrases or n-grams instead of single words
- Algorithms that take into account the total number of stories processed so far (optimization of the technique used in experiment 7.3)

To improve performance evaluations, it may be necessary to create more standard topics (i.e. more than the 120 topics of TDT3) and test the system on smaller data sets. While our experiments were able to detect a certain number of first stories from the official TDT3 truth sets, it would be interesting to see how many first stories the application was able to retrieve in total.

9. Acknowledgments

This work is supported in part by the National Computational Science Alliance under IRI030006 (we utilized the IBM pSeries 690 cluster).

I would like to thank my adviser and mentor April Kontostathis for her great support and effort during the last two years.

Also, I would like to thank the faculty and staff of Ursinus College to allow and assist me pursuing this research project, including Dean Levy (for providing the funding for the TDT3 data set), Dean Lucas, and the Honors committee (Dr. Liston, Dr. Edwards, and Dr. Kontostathis).

Thanks to the Howard Hughes Medical Institute for their generous scholarship enabling me to conduct research during the summer of 2004, the National Center for Supercomputing Applications (NCSA) for the use of their supercomputer systems, and the team behind the Text Mining Infrastructure (TMI) from Lehigh University.

10. Bibliography

1. The Linguistic Data Consortium, "The Year 2000 Topic Detection and Tracking TDT2000 Task Definition and Evaluation Plan", version 1.4, August 2000.

2. L. Holzman, C. Spencer, T. Cunningham, W. Pottenger, "TMI – Text Mining Infrastructure and Library, A Turorial", Distributed Textual Data Mining Lab, Department of Computer Science and Engineering, Lehigh University, 2003. Available at: http://hddi.cse.lehigh.edu/

3. L. Holzman, A. Fisher, L. Galitsky, A. Kontostathis, W. Pottenger, "A Software Infrastructure for Research in Textual Data Mining", Distributed Textual Data Mining Lab, Department of Computer Science and Engineering, Lehigh University, 2003.

4. A. Kontostathis, I. De, L. Holzman, W. Pottenger, "Use of Term Clusters for Emerging Trend Detection", Distributed Textual Data Mining Lab, Department of Computer Science and Engineering, Lehigh University

5. The Linguistic Data Consortium, University of Pennsylvania, May 2004, http://www.ldc.upenn.edu/

6. National Institute of Standards and Technology, http://www.nist.gov/speech/tests/tdt/

7. James Allan, Victor Lavrenko, Hubert Jin, “First-Story Detection in TDT is Hard”, Center for Intelligent Information Retrieval, University of Massachusetts

8. Salton, G., “Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer”, Reading, MA, 1989.

9. Papka, R., “On-line New Event Detection, Clustering, and Tracking”, Ph. D. Thesis, University of Massachusetts, September 1999.

10. Allan, J., “Incremental Relevance Feedback for Information Filtering”, Proceedings of ACM SIGIR, 1996.

11. Ahmet Vural, "On-Line New Event Detection and Clustering using the concepts of the Cover Coefficient-Based Clustering Methodology”, Masters Thesis, Bilkent University, August 2002

12. Gerard Salton, “Automatic Text Processing”, Chapter 9, Addison-Wesley, 1989.

13. Kontostathis, April, William M. Pottenger, and Brian D. Davison. (2004) Assessing the impact of sparsification on LSI Performance. Proceedings of the 2004 Grace Hopper Celebration of Women in Computing Conference. Oct 6-9, 2004. Chicago, IL.