RC 21838 (98287) 22 September 2000 **Computer Science**

IBM Research Report

Speech Transcript Analysis for Automatic Search

Anni R. Coden, Eric W. Brown
IBM T. J. Watson Research Center
Hawthorne, NY 10532

Research Division
Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich

Speech Transcript Analysis for Automatic Search

Anni R. Coden, Eric W. Brown
IBM T.J. Watson Research Center
Hawthorne, NY 10532
anni@us.ibm.com, ewb@us.ibm.com

Abstract

We address the problem of finding collateral information pertinent to a live television broadcast in real time. The solution starts with a text transcript of the broadcast generated by an automatic speech recognition system. Speaker independent speech recognition technology, even when tailored for a broadcast scenario, generally produces transcripts with relatively low accuracy. Given this limitation, we have developed algorithms that can determine the essence of the broadcast from these transcripts. Specifically, we extract named entities, topics, and sentence types from the transcript and use them to automatically generate both structured and unstructured search queries. A novel distance-ranking algorithm is used to select relevant information from the search results. The whole process is performed on-line and the query results (i.e., the collateral information) are added to the broadcast stream.

1. Introduction

The emergence of the World Wide Web as an information and entertainment media is generating exciting changes in the more traditional media of broadcast television. In particular, broadcasters have begun to link these two media together to create a much richer television viewing experience. The first phase of this linkage is rather loose; television programs routinely display URLs that point to web sites related to the program. The next phase of linkage, however, will be much tighter as set top boxes and TV tuner computer cards become more prevalent. Such devices will allow broadcasters to send Web content with the television broadcast and display the audio/video program in an integrated fashion with the Web content.

This tighter integration presents a number of challenges, with one of the more difficult challenges being how to identify the information that should be broadcast with the television program. Currently, program producers manually identify the information to broadcast. This process may be supported by software that aids in scheduling the data broadcast, or software that automatically accesses databases to obtain, for example, stock quotes. Nevertheless, the overall information seeking and selection process is manual.

This approach has several disadvantages. First, it is slow and expensive. Second, there is no way to tie additional information into a live broadcast, where the time at which a particular topic is discussed is not known beforehand. Currently, if a significant event (e.g., an airplane crash) occurs during a broadcast of the daily news, the producers have a difficult time just reporting the event, and in general have no time to find background information. Third, with the advent of set top boxes, users may want to customize the information displayed on their TV. One person may want to see only sports related information, while another may want to be able to choose news related to a specific geographic location.

To address these problems we have developed a system called WASABI (Watson Automatic Stream Analysis for Broadcast Information). WASABI takes speech audio as input, converts the audio stream into text using a speech recognition system, applies a variety of analyzers to the text stream to identify information elements, automatically generates queries from these information elements, and extracts data from the search results that is relevant to the current program. The data is multiplexed into the broadcast signal and transmitted along with the original audio/video program. The system is fully automatic and operates on-line, allowing broadcasters to add relevant collateral information to live programming in real time.

In the remainder of this paper we discuss related work, describe the overall methodology and architecture, provide a more detailed analysis of the different parts of the system, and conclude with pointers to future work.

2. Related work

The problem we are trying to solve here is most closely related to the work on Topic Detection and Tracking (TDT)[1]. In TDT, the goal is to analyze news broadcasts (text articles or text transcripts generated automatically from audio and video) and identify previously unseen news events, or topics. Topics are then tracked by identifying subsequent news stories covering the same

event. This is accomplished using a variety of off-line text processing, language modeling, and machine learning algorithms. TDT differs from our work in two ways. First, our system must operate in real time in order to annotate a live broadcast with collateral information, while the typical TDT system operates off-line. Second, our system goes beyond topic detection to include automatic query formulation and collateral information discovery.

The techniques we use to accomplish this task are drawn largely from traditional information retrieval [8] and text analysis techniques [6]. Again, we have extended these techniques to support on-line processing of streaming text data. These distinctions will become clear in the remainder of this paper.

We should also mention that a number of commercial systems exist that support the manual addition of data to a broadcast signal (see, for example, Wave Systems Corporation and SkyStream Networks). These systems allow program producers to select, format, and schedule the delivery of data with the broadcast. The key difference with our work is that these systems require manual identification of collateral data, while our system is fully automatic.

3. Background

The goal of this work is to find collateral information in real time based on the words spoken during a news broadcast (or any other spoken discourse). There are several challenges in this arena. Although voice recognition has improved tremendously over the last few years, it cannot be expected that a voice recognition system will deliver a perfect transcript. Transcript quality is by far the best when the voice recognition system is trained with the voice of the speaker and the recording is made in a quiet environment with appropriate microphones. Unfortunately, in a broadcast setting (and many other similar settings) such optimal circumstances are not available. Here, there are many speakers, some recording from a studio, others from the field. Furthermore, background noise and sub-optimal microphones contribute to the deterioration of the transcript quality.

The quality of the transcript has tremendous implications on the methods that can be applied to analyze it. The effectiveness of traditional text analysis tools decreases as the quality of the transcript decreases. Some of the issues that arise include lack of punctuation, lack of grammatical structure, and mis-recognized words (wrong words added as well as correct words missing). Sentences are "constructed" from the continuous stream of spoken words by setting a pause threshold between words. This and the erroneous recognition of words lead to sentences that are grammatically incorrect. Hence, methods that rely on analyzing the structure of a sentence are bound to give worse results. Erroneous word recognition has a detrimental effect on word statistics, such that relying on these statistics may lead to unintended or unexpected results. Adding to these difficulties is the need to process the text in real-time.

Even when reading a poor transcript, however, a person can usually describe the essence of the discourse. Our goal is to be able to capture this "gist". Once captured, it constitutes metadata for the discourse, which can be stored with the discourse and provide value in its own right. Processing this metadata is at the core of our methodology.

Using the automatically created metadata we show a novel method to perform concept searches that produce the desired collateral information. A new ranking algorithm is described which sorts the results of the concept searches, and could also prove to be quite appropriate in traditional text searches.

4. Architecture

Given the goal of finding collateral information for a live broadcast in real time, each part of the system must perform as close to real time as possible. To facilitate development of real time components, the components are isolated from each other by a modular architecture with clearly defined interfaces. This approach has a further benefit that over time more modules can be developed and integrated seamlessly into the system. Figure 1 gives an overview of the architecture.

The input to the Real Time Feature Extractor is a live television broadcast. A module in the extractor can determine a particular feature in real time from a video/audio signal. In particular, the Speech module transcribes the audio signal into English and the CCText module extracts the closed caption text from the video

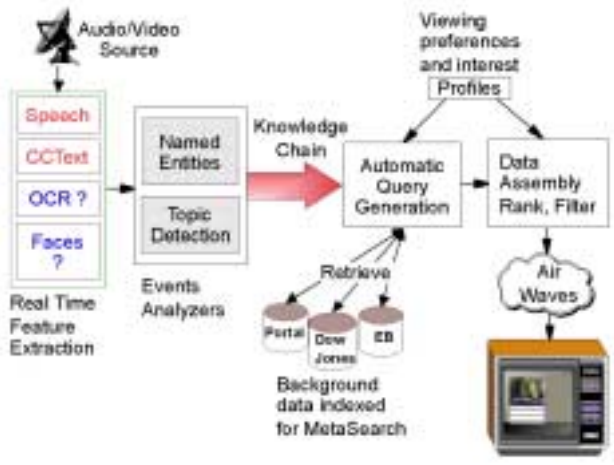

Figure 1. Architecture overview

stream. Both these modules currently operate in real time. The other two modules shown in the architecture are expected to work in real time shortly. The OCR (Optical Character Recognition) module deciphers the text overlaid on a frame. The Face module determines the number of faces and sizes present in any given frame.

Currently the Feature Extractor produces ASCII Text, both in the form of transcribed text and closed caption text (CCText) if it is available. Such text is time stamped and can be stored in conjunction with the source.

The next component of the system is the Event Analyzer subsystem, which represents the most important and innovative part of the system. Each analyzer has a task to perform based on the ASCII text. The output of one analyzer (an event) can be the input to another analyzer. The output of an analyzer is again an ASCII string, which is time stamped (start and end time).

Currently we have several analyzers. The simplest analyzer is the SentenceSegmenter. It takes the raw transcript, determines sentence boundaries and reformats the string for display. The NamedEntitiesRecognizer identifies named entities like the names of people, places, organizations and companies, and other types of terms, like financial terms, legal terms, or educational terms, to name a few.

The TopicAnalyzer determines which topic from a taxonomy of topics is being discussed. A number of analyzers look for specific types of sentences, including questions, demands, and requests, and return the sentence and its type (e.g., it's a question). The CalendarAnalyzer determines the time and date if it is mentioned and translates it into a standard format.

Although each analyzer will be described in more detail in subsequent sections, there are some common features that we would like to discuss here. Each analyzer performs its task in real time and adheres to a predefined interface. The real-time aspect made it necessary to invent new algorithms for some of the tasks. Another challenge is that the tasks use word sequences that are (in general) grammatically incorrect as input. Another issue to consider is that spoken words during a news broadcast or a meeting have quite different characteristics than, for instance, written text. There are many "filler" sentences or phrases that are not pertinent to the primary conversation thrown in between the primary topic discussed.

The output events from the analyzers are stored on a linked list data structure, called the Knowledge Chain. Briefly, all events are assembled on a timeline, with a start-event token and an end-event token. The precedence of events that start at the same time is immaterial. The events themselves can then be stored in a database or a program can create a XML document based on the events for any time segment desired.

Once the Knowledge Chain has been created, the next step is to find the collateral information that will be broadcast with the program. This is done by automatically generating queries based on the events recorded in the KnowledgeChain. Profiles (either personal or application specific) could be used to guide the query generation. The results from these queries are then assembled, ranked and sent to the multiplexer, which inserts the results into the broadcast stream.

4.1. Speech transcript

Our system uses the IBM's ViaVoice [7] product to transcribe the audio signal into ASCII text. A special acoustic model was developed by IBM to handle broadcast news. This model is speaker independent and compensates for the background noise inherent in a broadcast news program. This custom acoustic model is combined with the standard business language model included with the product. The accuracy varies with speaker and recording conditions, but the transcribed text conveys the general gist of the broadcast.

The following example is taken from a transcript of an evening news broadcast. The original text is:

This is World News Tonight with Peter Jennings. Good evening. We begin tonight with the Presidential campaign. The Republican Party got its ducks in a row today. Senator John McCain who almost derailed the best financed campaign in history a few months ago, that of George W. Bush, took his medicine today and said I do. Here is ABC's Aaron Brown. John McCain arrived for the meeting exactly two months after he gave up his campaign against Governor Bush and after the 90 minute private meeting Governor Bush got exactly what he wanted. I look forward to enthusiastically campaigning for Governor Bush. Not good enough for reporters who wanted to hear the e word I endorse Governor Bush.

The transcript generated by IBM's ViaVoice is:

Is World News Tonight with Peter Jennings the the the we begin tonight with the presidential campaign of Republican Party got its ducks in a row today. Senator John McCain who almost beat Ray killed the best finance campaign in history few months ago.That of George W. Bush. Took his medicine today and said I do. Is ABC's Aaron Brock. McCain wrought with the make that the two months after he gave up its campaign the next governor of boys. The 90 minute private meeting. Which got exactly what he bought it. Fourth to use yesterday became the main. That and wish. Not good enough for re-

porters. Who wanted to hear the eve workers discovered bullish and bearish data to push ahead.

It is straightforward to add words or phrases into the vocabulary of ViaVoice using the product version, and therefore advisable to add current names or phrases into the system. Examples of words to add include names of politicians (both domestic and international) or phenomena (e.g., La Nina), which a standard business language vocabulary would not contain. If, for instance, the name McCain was not added, the system would pick a name that sounds similar to this politician's name. However, the gist of a reported story would change if a crucial name was not recognized. Such vocabulary can be built up with time.

Currently, several other components for speech recognition are being developed, including speaker and gender identification and the filtering of music. As these components become available, they will be incorporated into the system, improving the quality of the results produced by the analyzers.

4.2. CC-text

We use one of several products available that can extract closed caption text from a video signal. Again, each sentence becomes an event that is inserted into the Knowledge Chain. Clearly, the accuracy is quite high in this case and one could opt to use only CCText for subsequent analysis. However, CCText may not be available in all broadcast programs and CCText does not contain any capitalization, which is quite useful for some analyzers. Moreover, the speech recognition system will provide other information (like speaker identification) in the near future, knowledge that cannot be deduced from CCText alone.

4.3. Analyzers

To date we have developed several analyzers that all adhere to the same interface and produce their respective output events in real-time. The most basic analyzer is the SentenceSegmenter. It takes the raw transcript and outputs formatted sentences. Sentence boundaries are deduced based on the length of the pause between words. Formatting consists of adding appropriate capitalization and punctuation. Each recognized sentence is inserted into the Knowledge Chain. We are currently exploring the use of speaker identification and gender identification to improve the accuracy of sentence boundary recognition. We are also considering techniques that can improve grammatical correctness. The more accurate and grammatically correct the sentences are, the better other downstream analyzers will perform.

The output of the SentenceSegmenter is input to the other analyzers described in this section, including the NamedEntitiesRecognizer or the TopicRecognizer. Each of these analyzers adds its output to the KnowledgeChain, which is described in the next section.

An important analyzer is the NamedEntitiesRecognizer, which discovers named entities such as names of people, places, organizations, and companies, and other specific terms belonging to a particular subject, such as financial, banking, or educational terms. The algorithms used are derived from the ones used in IBM's Intelligent Miner for Text product [6]. In particular, they have been modified to perform in real-time. To identify a named entity, the capitalized words are looked up in several dictionaries that list proper names, places, organizations, etc. If a word is a first name, and the subsequent word is capitalized, the analyzer puts them together to form a complete name. The analyzer continues to examine subsequent capitalized words to form the longest possible name. There are additional algorithms to recognize middle initials and titles. No disambiguation is done, as in general there is not enough information to do that (e.g., Tijuana can be a place or a person). It will be shown in subsequent sections that an erroneous classification of a named entity (person or place for example) may not affect the final outcome, which is to find appropriate collateral information.

If a capitalized word is found in a dictionary with a specific type declaration (place or financial term, for example) it is classified as such. Capitalized words for which there is no type declaration are put in the miscellaneous category, as it is believed that capitalized words convey in general more information than verbs or adjectives. There are various algorithms for each type of term. For instance, suppose the name of a town or city is discovered, then the subsequent state word should be treated as a clarification of where the city is and not as an event in its own right.

For example: ***Peter Jennings, Senator John McCain, Ray, Aaron Brock, George W. Bush*** are recognized as Names in our example

The TopicAnalyzer determines which topic is being discussed. It assumes that a taxonomy was specified ahead of time. Here we use the KitKat rule-based system developed by IBM [5], which can be trained with a set of documents and has a user interface to specify rules manually. Clearly, any given sentence could describe more than one topic. A confidence value is associated with each recognized topic, which describes how sure the system is that a particular topic applies. Here the taxonomy is adapted from the Dow Jones set of publications augmented by us to fit the broadcast news scenario. The advantage of using their taxonomy is that any background data drawn from Dow Jones sources is already classified according to the taxonomy, which aids in producing and

processing results from automatically generated queries. Here the output of the SentenceSegmenter is used as input for the TopicRecognizer. However, it is also possible to string several sentences together to form an input. We plan to evaluate how varying the size of the input to the TopicRecognizer changes the performance of this analyzer. In particular we will evaluate the influence of using "paragraphs" (i.e., several sequential sentences) or overlapping paragraphs. However, the extreme run-time requirement of this application restricts the length of the input.

The StructureAnalyzer is a new type of analyzer, although it is based on some ideas developed in the Question-Answering system built by Prager et al. [9] [10]. The idea is that certain sentence structures suggest an action that is different from performing a search based solely on the words in the sentence. For example, the question, "Who discovered Penicillin?" suggests that the person is interested in a name and not in a sentence containing the words of the question (which could be paraphrased in the text). Similarly, the request, "Please show me the full text of the State of the Union Address" requires the system to find a piece of text. The StructureAnalyzer identifies and labels the structure of each sentence. For instance the QuestionRecognizer is a type of StructureAnalyzer. In its simplest form it checks whether a sentence starts with a "question-word" like who, when and how to name a few examples. Having identified a sentence as a question, it replaces parts of it with Question-Answering Tokens [9] [10]. This transformed sentence becomes now a query against a corpus that has been indexed taking Question-Answering Tokens into account. Such a search returns answers in its hitlist, which becomes the desired collateral information.

For example: The announcer says: ***Where did George W. Bush and John McCain meet today? It was at the place where they had their last controversy.*** Based on the first sentence, the system creates a query "$place George W. Bush John McCain meet today" and return with ***"Manchester, New Hampshire".***

Another analyzer further assists in the previous example. The DateAnalyzer identifies absolute dates as well as indirect date references (like "today") and calculates an absolute date in a standard format. It is always important to establish a reference date, which is simple in a live broadcast setting or in analyzing meetings. Dates are quite important in finding appropriate collateral information – referring to the previous example, George W. Bush and John McCain had several meetings. The DataAnalyzer is rule-based using the same system as the TopicAnalyzer.

The rule-based system is applied to find other types of structures, such as requests ***"Please show the agenda"*** or to perform a task. ***"Next slide please"***. The discovered events are added to the KnowledgeChain.

4.4. Knowledge chain

In the previous section we discussed several analyzers. Each one of them creates as output an event, e.g., a recognized sentence, topic, name, etc. The idea is that for each discovered event we create an Object, called a KnowledgeBead, which contains the following information: type of event, start time, end time, description of event, an assigned unique id and an optional object. The description of an event could be an object in itself with its own access methods, a simple ASCII text, or in future systems that discover multimedia events; it could be an image or a video clip. Additional pertinent information can also be included, like the confidence value of a discovered topic. Each KnowledgeBead has a corresponding MirrorKnowledgeBead that acts as an end tag for the event in the KnowledgeChain. Each Knowledge Bead is inserted into the KnowledgeChain at its start time. The Mirror Knowledge Bead is inserted at the end time. The precedence of KnowledgeBeads with same start times is immaterial, as is the precedence of the Mirror KnowledgeBeads with same end times. At any given point in time, examining the KnowledgeChain gives a description of the broadcast. The KnowledgeChain has a set of methods associated with it to help its manipulation. Here are a few suggestions for such methods:

1. Insert a KnowledgeBead at time t
2. Delete a KnowledgeBead at time t
3. Find the Mirror to a KnowledgeBead
4. Find all the KnowledgeBeads between a KnowledgeBead and its Mirror
5. Find all the KnowledgeBeads starting at time t
6. Find all the KnowledgeBeads ending at time t
7.

Furthermore, there are access methods for the KnowledgeBeads themselves:

1. Get start time
2. Get end time
3. Get type
4. Get description

The description can be a structure appropriate for the type of KnowledgeBead.

Figure 2 depicts a KnowledgeChain. For simplicity we omit the MirrorKnowledgeBeads for the Named Entities in this example. Note that the KnowledgeBeads are not necessarily inserted in a time-sequential manner. However, the KnowledgeChain gives a time-sorted synopsis of the broadcast.

Figure 2. KnowledgeChain

4.5. Automatic Query Generation

Once the events extracted by the analyzers are assembled on a time line in the KnowledgeChain, the next step is to automatically generate queries that will find collateral information. The first issue we must address is *when* to issue a query. A simple approach would be to pose a query whenever an event occurs. We believe that sending queries at the discovery of every event would both create a performance problem and generate a lot of irrelevant information. For instance, a query based solely on a name would typically return too many hits to send over the broadcast signal, and moreover, no meaningful ranking could be applied to determine the most relevant hits.

We suggest instead that queries be posed whenever a topic is detected. At this point all of the Knowledge Beads between the start of the topic and the end of the topic are assembled into a Query Object, representing the basis for query formulation

Once a Query Object has been created, the next step is to look at the identified sentence structure. If the structure is such that the query should be handled by a specific action subsystem, the query is sent to that subsystem for processing. For instance, the query might request the invocation of a specific program. A request to show a person's presentation for today's meeting requires the following steps:

1. Open the program capable of showing a presentation
2. Find the KnowledgeBeads specifying the name and date of the desired presentation
3. Query the database to find the actual presentation

We assume that there are rules in place describing how presentations are stored, but again these rules can be incorporated into the rule-based system handling requests.

If the identified structure of the Query Object does not direct the query to a specific action subsystem, then processing can proceed in one of two ways, both of which are currently under investigation. In the first approach, the query is processed with the following steps:

1. Identify the topic of the Query Object
2. Search for named entities constrained by the topic
3. Perform a free-text search constrained by the topic

We assume that the background data has been categorized using the same taxonomy as that used by the TopicAnalyzer. For any given data item, the assigned topic is stored as metadata in a database. Hence the first step is a database query to identify items with matching topics and constrain the scope of the next two searches. If the initial topic search should return an empty set, the query is repeated with the parent node in the topic taxonomy.

The second query (named entity search) is only applied to data items found by the first query. Named entities can be augmented with variants of the name and then used to search databases specific to the type of named entity. The results from these searches should be high quality data items that can immediately be added to the collateral information set.

The third query is sent to a free text search engine, again constrained to the set of items found in the first step. The input to the free text search engine is the transcript, with stop words removed and lemmatization applied. The free text search engine typically returns a ranked list of items and the top ranked items are selected as collateral information for the broadcast. Note that although this process is described as a sequence of three steps, for better performance the entire procedure can be accomplished with a single "multi-search" query [4].

This first approach is appealing because it is straightforward. Unfortunately, it naively relies on the rank scores returned by the text search engine to select collateral information, potentially resulting in the inclusion of irrelevant data. The second approach to query formulation addresses this problem with a more sophisticated data ranking and selection procedure. We explain this procedure with an example.

Imagine that the news broadcast just mentioned that Putin, Jaspin, and Clinton attended a summit meeting in Berlin. From the topic taxonomy we know that a summit meeting is a type of political meeting, which in turn is a type of meeting. Figure 3 shows a part of this taxonomy, where each node is a topic, the taxonomy is hierarchical with more general topics at the root and more specific topics at the leaves, and the letters inside the nodes indi-

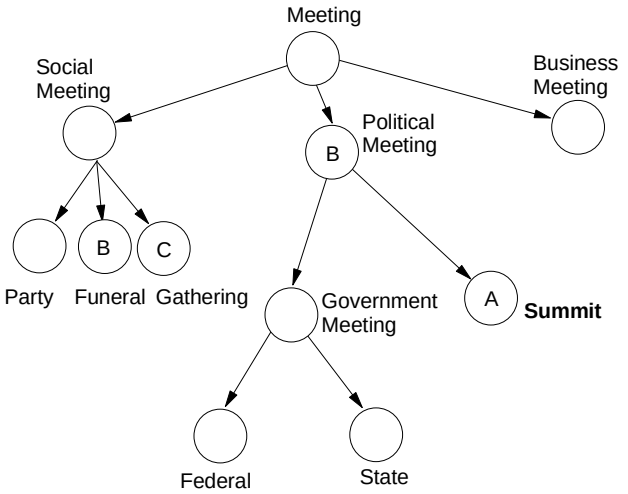

Figure 3. Part of a topic taxonomy

cate how topics have been assigned to three particular documents in the background data collection (documents A, B, and C). It might be interesting to find out what other meetings (maybe within a particular time frame) these three world leaders also attended.

To find documents about such meetings, the system first executes a free text search using all of the terms in the Query Object. The top *n* documents returned by the free text search are then scored by the system using the following formula:

$$S_i = R_i + E_i + P_i$$

where

- S_i is the score of document i
- R_i is the rank number of document i — the top scoring document returned by the free text search has rank 0, the next best scoring document has rank 1, etc.
- E_i is the number of named entities in the Query Object that *do not* appear in document i
- P_i is the taxonomy path score of document i, described below

The taxonomy path score of a document is calculated by locating the document's topic in the taxonomy tree and traversing the shortest path from that topic to the topic assigned to the Query Object. For each edge traversed upward (i.e., from child to parent), 1 is added to the path score, and for each edge traversed downward (i.e., from parent to child), 2 is added to the path score. In Figure 3, the Query Object topic is "Summit". The path score for document A (assigned to topic "Summit") is 0, the path score for document B (assigned to topics "Political Meeting" and "Funeral") is 2, and the path score for document C (assigned to topic "Gathering") is 6. Note that although document B is assigned to two topics in the taxonomy, we calculate the path score using the topic closest to the Query Object topic. This path scoring scheme favors documents whose topic is the same as or a sub-topic of the Query Object topic, and penalizes documents whose topic is a super topic of the Query Object topic or in a different branch of the topic taxonomy.

This document scoring formula produces scores such that better documents will have lower scores. It factors in the scores returned by the free text search engine, exploits the fact that named entities tend to be more precise query terms, and uses the query topic and the topics assigned to the documents to further refine the search. If the formula assigns the same score to two or more documents, the documents may be further ordered by considering the frequency of occurrence of the named entities in the documents.

4.6. Background data preparation

The quality of the collateral data found by the system depends directly on the databases available to the system for searching. In our implementation data obtained from Dow Jones is used, which includes sources such as the New York Times, Wall Street Journal, and Newswires. Our other sources include the World Wide Web and Lotus Notes databases. These sources where chosen both for their appropriateness and availability to us. However, what is more noteworthy is how the data is viewed and prepared for the search. It is crucial to remember that the whole process from transcription to analysis, query preparation, search and data assembly has to be done in real time. Currently, searches against the World Wide Web are quite slow and are not appropriate for this application. Therefore we choose to store all the data on a local database with all the associated metadata and create a single index for all the data. However, additional specialized indices could be available for subsystems like the Question Answering system.

The Dow Jones data contains embedded metatags drawn from their taxonomy. Our data preprocessor parses the data and stores the metatags in the database to be used for fast queries. For instance, the data contains geographical information, which is ordered in relevance to the article. We store all this information in a specially designed database system based on DB2. Suppose a relevant article for a broadcast segment is found. Not only can we show the article but also the other areas affected. A good example is trade and company information, which span sometimes several countries. General concept queries about "what events happened in a certain country" can be easily answered and rank ordered by the rank ordering in the geographical data.

4.7. Profiles

Throughout this paper we have described choices an application of our system can make. It starts with which analyzers should be used. The SentenceSegmenter is essential to deal with the transcript, but the rest of the analyzers, the dictionaries used, the rules governing the topics, and the subsystems processing structured requests are application specific. Our system is flexible and one should be able to "mix and match" the various components. However, even for a given application, a user may have specific preferences – seeing (or not seeing) biographical or geographical information, the type of source one is interested in, and the date range, to mention a few examples. The rule-based system used throughout can easily accommodate profiles that are expressed in terms of rules and can be incrementally added to the system.

5. Evaluation

There are a number of performance questions related to the system described here, ranging from the speed and effectiveness of individual analyzers, to the overall usefulness of the system measured by end-user satisfaction. We have a number of evaluations planned to address these questions. System usefulness and end-user satisfaction are best measured by a carefully designed user study. We plan to conduct at least one summative evaluation, which will involve users interacting with our prototype in a controlled fashion. The performance of the system will be measured via observation of the users and survey feedback from the users.

Evaluating the individual analyzers is somewhat more straightforward. Techniques from traditional information retrieval and natural language processing evaluation may be used, including standard precision and recall measures for named entity recognition and text search effectiveness.

Performing any of these evaluations is dependent on building a working prototype of the system, which we have done. The WASABI system was built on a Windows platform with a Java Client front-end. Figure 4 shows a screenshot of the working prototype. To date, we have evaluated the effectiveness of the named entity recognizer.

For our experiments we used the world news by a national television network. We digitized the taped broadcast and transcribed it manually. We first listened to the tape to determine the number and type of named entities present. There were 67 distinct named entities in a half hour broadcast (excluding the commercials). They were divided into four classes: people, places, financial terms, and miscellaneous other terms. Half of the named entities were people and 23% were places. Roughly 20% of the people were names of reporters in the field or people interviewed in the field, names one cannot expect a speech recognition system to recognize. Furthermore, these names are generally not relevant to the type of searches our system performs, where the goal is to find more information about an event. As such, we concluded that only 55 of the named entities mentioned were relevant for the system.

In the transcript generated by our IBM ViaVoice recognition engine, 34 relevant named entities were detected, a 62% recognition rate, which is much higher than the recognition rate for the entire document. WASABI recognized 98% of the named entities in the transcript. WASABI typed the named entities correctly 97% of the time, the errors coming from inherent ambiguities in the names, for instance Tijuana being both a name and a place. To correctly type such words a more extensive analysis of the context has to be performed which cannot be done in a real-time system.

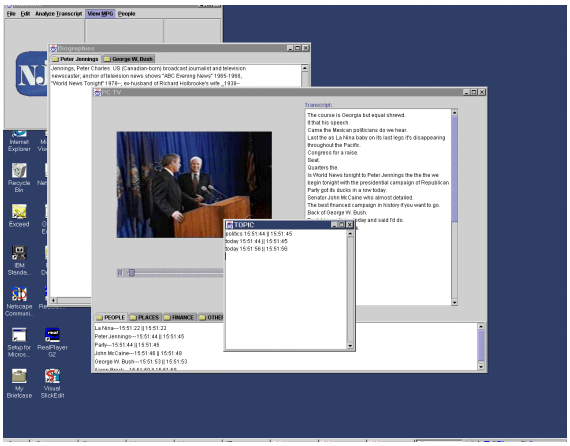

Figure 4. WASABI screen shot

In spite of being a real-time system, WASABI can infer that different references of a name could all point to the same person, for instance “George W Bush”, “Governor Bush” and “Gov. Bush”. Timing information and topic shifts are used to identify these equivalences. In a Data Broadcast application it is quite important to determine equivalent named entities and avoid sending out the same collateral information more than once over the limited bandwidth broadcast channel.

6. Conclusions

The addition of arbitrary data to broadcast television presents exciting possibilities for creating a whole new television viewing experience. This opportunity, however, also presents a number of challenges. In this paper we have addressed the problem of how to automatically determine what data to send along with the audio/video program. Our solution is based on WASABI, a system for analyzing spoken discourse and automatically finding collateral information.

The challenge is to carry out this process in real time using a text transcript generated by an automatic speech recognition system. Traditional information retrieval methods are not always applicable due to the type of discourse and the grammatical mistakes in the transcript. We have developed new algorithms to address these issues. We are currently in the process of developing a rigorous evaluation methodology to compare our algorithms against more standard information retrieval methods. Furthermore, due to the real-time processing constraints, many other text analysis tools are not applicable. In future research, we plan to explore the same problem without the real time constraints to discover whether more sophisticated text analysis tools can improve the quality of the retrieved material.

7. Acknowledgments

We would like to thank Robert Mack, Alan Marwick, and the anonymous referees for their suggestions on ways to improve this paper. We would also like to acknowledge James Janniello and his team for their collaboration on this project.

8. References

[1] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang, "Topic Detection and Tracking Pilot Study: Final Report". *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pp. 194-218.

[2] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, ACM Press: New York, 1999.

[3] E.W. Brown and H.A. Chong. The Guru System in TREC-6. *Proceedings of TREC6*, Gaithersburg, MD, 1998.

[4] A. Coden, J. Brereton and Michael Schwartz, "System and Method for Performing Complex Heterogeneous DataBase Queries using a Single Expression", Patent Application filed 1998

[5] IBM Mail Analyzer, v. 6.2. 1999

[6] IBM Intelligent Miner for Text, "www-4.ibm.com/software/data/iminer/fortext/"

[7] IBM ViaVoice Millennium Pro. 2000

[8] C.D. Manning and H.Schutze, "Foundations of Statistical Natural Language Processing", *MIT Press*, 1999

[9] John Prager, Eric Brown, Anni Coden and D. Radev "Question-Answering by Predictive Annotation", *Proceedings of SIGIR 2000* (to appear)

[10] J.M. Prager, E.W. Brown, A.R. Coden and D. Radev. "The Use of Predictive Annotation for Question-Answering in TREC8", *Proceedings of TREC8*, Gaithersburg, MD, 2000.

[11] Yael Ravin and Claudia Leacock, "Polysemy. Theoretical and Computational Approaches", *Oxford University Press*, 2000

[12] Yael Ravin, Nina Wacholder and Misook Choi. "Disambiguation of Names in Text", *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pp. 202-208, Washington D.C., March 1997.

[13] Yael Ravin, Nina Wacholder and Roy Byrd. "Retrieving Information from Full Text Using Linguistic Knowledge", *Proceedings of the Fifteenth National Online Meeting*, pp. 441-447, New York, May 1994.