ANSES
Automatic News Summarization and Extraction System

Lawrence Wong
MEng Computing
Imperial College Dept. of Computing
http://www.doc.ic.ac.uk/~lkhw98/project
Acknowledgements

I would like to thank Dr. Stefan Rueger for his supervision, his inspirations and support throughout this project. Only with his guidance to keep me focused in the right direction I was able to develop ideas and transform them into practical solutions.

I also like to thank Marcus Pickering, whose project ‘Video Search Engine’ provided the basis of my project. This project was built on the ideas Marcus has developed, aiming to refine and enhance the usability of an Informational Retrieval System. He has given me useful guidance and pointers to information.

Finally, to CSG support team, who helped in setting up the equipments and provide support for all my hardware and software issues.
Abstract

This project proposes to build a system which addresses summarization at a multimedia level. In one short sentence the system could be described as:

“Watch the news and tell me what happened while I was away.”

This project combines a Video scene change algorithm, with the current text segmentation and summarization techniques to build an automatic news summarization and extraction system.

Television broadcast news are captured both in Video/Audio with the accompanying subtitles in text format. News stories and identified, extracted from the video, and summarized in a short paragraph which reduces the amount of information into a manageable size. Individual news video clips can be retrieved effectively by a combination of video and text, providing distilled information such as a summarized version of the original text and highlights important key words in the text.
Table of Contents

ACKNOWLEDGEMENTS .................................................................................................................. 3

ABSTRACT ..................................................................................................................................... 4

1 INTRODUCTION ...................................................................................................................... 8
  1.1 AN "AUTOMATIC NEWS SUMMARIZER" ............................................................................ 10
    1.1.1 Aims of the project ........................................................................................................ 10

2 BACKGROUND ......................................................................................................................... 12
  2.1 TEXT SUMMARIZATION ...................................................................................................... 12
    2.1.1 Definition of text summarization ................................................................................ 12
    2.1.2 Roots of Text Summarization .................................................................................... 12
  2.2 MULTIMEDIA SUMMARIZATION ....................................................................................... 14
    2.2.1 Different types of summaries .................................................................................... 14
    2.2.2 Text Summarization Techniques .............................................................................. 15
  2.3 LEXICAL CHAINS ............................................................................................................... 16
    2.3.1 Why lexical chain method is chosen .......................................................................... 16
    2.3.2 Procedures for constructing Lexical Chains ............................................................... 17
    2.3.3 Relatedness Criterion Defined .................................................................................. 17
    2.3.4 The Lexical Chain Algorithm .................................................................................... 18
    2.3.5 Building Summaries using Lexical Chains ................................................................. 21
  2.4 WORDNET – ONLINE LEXICAL DATABASE ...................................................................... 23
    2.4.1 Background of WordNet ............................................................................................. 23
  2.5 GATE SYSTEM .................................................................................................................... 24
    2.5.1 Named Entity recognition ........................................................................................... 24
    2.5.2 Coreference resolution ............................................................................................... 24
    2.5.3 Tokeniser .................................................................................................................... 25
    2.5.4 Part of Speech Tagger (POS-tagging) ......................................................................... 25
  2.6 VIDEO SEGMENTATION ..................................................................................................... 26
  2.7 STREAMING MEDIA TECHNOLOGIES .............................................................................. 26
  2.8 REAL-TIME STREAMING PROTOCOL (RTSP) ................................................................. 27
  2.9 SMIL FORMAT .................................................................................................................... 27
    2.9.1 Advantages of SMIL .................................................................................................... 27
    2.9.2 Use clips from different locations .............................................................................. 28
    2.9.3 Create interactive SMIL files .................................................................................... 29
  2.10 REALTEXT ........................................................................................................................ 30
    Example RealText file ............................................................................................................ 30
  2.11 INFORMATION RETRIEVAL SYSTEM ............................................................................. 31

3 SYSTEM SPECIFICATION ........................................................................................................... 32
  3.1 HYPOTHESIS ..................................................................................................................... 32
  3.2 DEFINITION OF A NEWS SUMMARIZATION AND RETRIEVAL SYSTEM ... 33
    3.2.1 Source Data Capture .................................................................................................. 33
    3.2.2 Data Analysis ............................................................................................................. 33
    3.2.3 Media Generation ....................................................................................................... 33
    3.2.4 Information Retrieval System .................................................................................... 33
  3.3 OVERALL ARCHITECTURE ............................................................................................... 35
  3.4 COMPONENT SPECIFICATION ........................................................................................... 35
    3.4.1 Video Capture .............................................................................................................. 35
    3.4.2 Close-Caption (Subtitles) capture ............................................................................. 36
    3.4.3 Text Summarization Engine ...................................................................................... 37
    3.4.4 Natural Language Parser ........................................................................................... 37
    3.4.5 Semantic Analysis ....................................................................................................... 38
    3.4.6 Story Segmentation Detection .................................................................................... 38
    3.4.7 Summary Generation .................................................................................................. 38
4 DESIGN AND IMPLEMENTATION ......................................................... 41
  4.1 SYSTEM WORKFLOW ................................................................. 41
  4.2 SYSTEM DESIGN DECISIONS .................................................... 42
  4.3 ANSES COMPONENTS ............................................................... 44
     4.3.1 Video/Audio Capture ............................................................ 44
          Hardware .................................................................................. 44
          Software Tools ........................................................................ 44
     4.3.2 Subtitle Capture ...................................................................... 45
          Software Tools ........................................................................ 45
     4.3.3 Storing Video/Audio & Subtitles ............................................ 45
          Storage format ........................................................................ 45
     4.3.4 Scene Change Detection ....................................................... 46
     4.3.5 Storing 'Video segment' boundaries ...................................... 47
     4.3.6 Generate corresponding 'Text' segments .............................. 47
          Removing meaningless characters .......................................... 47
          Removing duplicates ............................................................... 48
          Solution .................................................................................. 49
          Rearrange data into complete sentences ................................. 49
          Generating Text Segments ...................................................... 50
     4.3.7 Detect Key Entities and Parts of speech ............................... 51
          Introduction ............................................................................ 51
          General Architecture for Text Engineering (GATE) ................ 51
          Detected Key Entities ............................................................. 51
          POSTagging ........................................................................... 52
          Accuracy of GATE ................................................................. 52
     4.3.8 Story Segmentation ............................................................... 53
          Introduction ............................................................................ 53
          Segmentation Algorithm ......................................................... 53
          Merging Criteria ....................................................................... 54
          Similarity Function ................................................................ 55
          Weights of matched word ........................................................ 55
          Merging Algorithm ................................................................. 57
          ANSES Demo Program ........................................................... 58
     4.3.9 Storing 'Story segment' boundaries ...................................... 59
     4.3.10 Summarization by Lexical Chains ...................................... 59
          Introduction ............................................................................ 59
          WordNet 1.6 ........................................................................... 59
          Candidate Words ..................................................................... 60
          Generating Lexical Chains ...................................................... 60
          Selecting Strong Chains .......................................................... 61
          Generating the summary ......................................................... 62
          Limitations – Dealing with search space explosion ................. 62
     4.3.11 Managing Gigabytes Information Retrieval Engine ............ 63
          Managing Gigabytes ............................................................... 63
          Modifying .mg_getrc to index video material .......................... 63
     4.3.12 Encoding into RealMedia format ....................................... 64
     4.3.13 RealText and SMIL file generation ..................................... 64
          RealText ................................................................................... 64
          SMIL file generation .............................................................. 65
     4.3.14 Web-based News Retrieval Interface ................................. 67
          Introduction ............................................................................ 67
          Retrieving News Stories ......................................................... 67
          Viewing full story text ............................................................. 68
          Browsing the Video ................................................................. 69
          Playing the Video .................................................................... 70
5 TESTING AND EVALUATION .......................................................... 71
  5.1 SOURCE DATA CAPTURE ......................................................... 71
  5.2 STORY SEGMENTATION .......................................................... 71
     5.2.1 Test carried out for segmentation algorithm ...................... 71
1 Introduction

With the rapid growth of broadcast systems, the internet and online information services, more and more information is available and accessible. Explosion of information has caused a well recognized information overload problem. There is no time to read everything and yet we have to make critical decisions based on whatever information is available.

The technology of automatic text summarization is indispensable for dealing with this problem. Text Summarization has been identified as a crucial research area by several organizations such as in United States (e.g. DARPA [8]), the European Community and Pacific Rim. It is also been increasingly exploited in the commercial sector, e.g. in the telecommunications industry (BT’s ProSum [4]), in data mining of text databases (Oracle’s Context) and filters for web based information retrieval.

“Text summarization is the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user or task.”

There are many uses of summarization in every day activities, which indicates the type of function that summarization can perform:

- Headlines (from everyday news around the world)
- Outlines (notes for students)
- Minutes (from a meeting)
- Previews (of a movie)
- Synopsis (soap opera listings)
- Digests (TV guide)
- Bulletins (weather forecasts / stock market reports)

In general, humans are extremely capable summarizers. We have an incredible ability to condense vast information down to the critical bit, as revealed in the following quote [2]:

“He said he was against it.”

--Calvin Coolidge, on being asked what a clergyman preaching on sin had said.

However, automatic summarization, done by machine, posts a rather challenging problem to computer scientist, due to the abstractness and complexity of human
languages. This problem was recognized and tackled early in 1950s. Since then several well-known algorithms have been developed, but the achievement was bound by the limitation of the current Natural Language Processing technologies, and until today it still remains an active research topic.

The objective of this project is not to improve on the existing algorithms, but to study and apply these algorithms, combining with other useful techniques which uses other forms of media such as video scene change detection to produce a practical and usable Multimedia Information Retrieval System.
1.1 An “Automatic News Summarizer”

The aim is to build an automatic news summarization system. It records broadcast television news, analysis the content to identify news stories. Content of each story is summarized and important keywords are extracted. This information is to be stored in a central database, and an Information Retrieval system is to be implemented which let users to search for any piece of news in the database. Such a system have advantage over other search engines, as we use summarization techniques to highlight the most important information to the user, enabling him/her to locate the story he/she is looking for in a shorter time, when compared to a ordinary text based search engine.

From here onwards I shall refer the system as ANSES (Automatic News Summarization and Extraction System).

1.1.1 Aims of the project

Apart from an information retrieval system, ANSES was also intended to address 3 important issues:

1. **Story segmentation** – in order to identify story boundaries from 30 minutes worth of video and closed caption (subtitles). (E.g. to find that there are 10 individual pieces of news in a lunch time BBC news). A technique suitable to solve this problem is text segmentation. It identifies the current topic of discussion and detects when/where the topic changes. There are many known methods to carry out text segmentation but all of them suffer from certain limitations, and this problem has been proved to be difficult to solve. For details on existing algorithms refer to background section.

   Instead of traditional methods we try to implement story segmentation with the combination of existing text algorithms and introduce the idea of video segmentation. We make use of observable information from the video such as a scene change. A scene change is defined when there is a large difference in pixel intensities between 2 consecutive video frames, which represents ‘large amount of movement’ or a ‘change of scene’ on the screen.

2. **Story summarization** – provides the user with a short summary of the story. This allows the user to decide whether a video clip returned by the search engine is relevant to the topic he/she is searching for. We apply a text summarization technique called ‘Lexical Chain’ to summarize the text.
Apart from a summary of the original story, we could also try to identify important keywords such as “London”, “Prime Minister” or “tube strike”. Such keywords, if identified correctly, would contribute tremendous amount to indicate what the topic of the story is about. For here onwards these keywords shall be referred as ‘Key Entities’.

For both story segmentation and summarization, we have used a text processing system called GATE [12], developed by Sheffield University to extract these keywords. Accuracy of the system has proved to be high, and it is one of the main components which provide the stability of this segmentation technique.

3. **An automatic system** – A typical user would like to select a program to record from the programme guide, and then leave the system to do the rest of the work. We want this system to be fully automatic, from source data capture, data processing to the generation of the central data store and online web-based retrieval system.

Over time this system can build up a larger online database, which could be used as a backend system which provides the source data to implement data mining applications.
2 Background

2.1 Text Summarization

2.1.1 Definition of text summarization:

“Text summarization is the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user or task.”

2.1.2 Roots of Text Summarization

Early systems were developed in the late 1950’s, characterized by a surface-level approach, for example exploiting thematic features such as term frequency, e.g., Luhn's paper [20], and Rath et al [27]. Whereas the first entity-level approaches based on syntactic analysis appeared in the early 1960s, (Climenson et al. 1961 [6]), the use of location features was not developed until somewhat later (see Edmundson's 1969 paper [10]). In the early 1970s, there was renewed interest in the field, with extensions being developed to the surface-level approach to include the use of cue phrases (bonus versus stigma items), and which resulted in the first commercial applications of automated abstracting, see Pollock and Zamora [25]. The late 1970s saw the emergence of more extensive entity-level approaches (Skorokhodko 1972 [31]) as well as the first discourse-based approaches based on story grammars (vanDijk 1980 [35], Correira 1980 [7]). The 1980s enjoyed an explosion of a variety of different work, especially entity-level approaches based on
artificial intelligence such as the use of scripts (Lehnert 1981 [19]), (DeJong 1982 [9]), logic and production rules (Fum et al. 1985 [11]), semantic networks (Reimer and Hahn 1988 [30]), as well as hybrid approaches, e.g., (Rau et al. 1989 [28]), see also (Aretoulaki 1994 [1]).

The current period (late 1990s) represents a renaissance of the field, with all three types of approaches being explored very aggressively, heightened by government and commercial interest. Recent work has almost exclusively focused on extracts rather than abstracts, along with a renewed interest in earlier surface-level.

Recently, there has been an increase in the research and development budgets devoted to automatic text summarization. The United States (DARPA [8]), the European Community and Pacific Rim countries have identified text summarization as a critical research area, and are beginning to invest in it. Text summarization is also increasingly being exploited in the commercial sector, in the telecommunications industry (e.g., BT's ProSum [4]), in data mining of text databases (e.g., Oracle's Context), in filters for web-based information retrieval (e.g. Inxight's summarizer [18] used in AltaVista Discovery), and in word processing tools (e.g., Microsoft's AutoSummarize). In addition to the traditional focus of automatic indexing and automatic abstracting (of scientific and technical text) to support information retrieval, researchers are investigating the application of this technology to a variety of new and challenging problems, including multilingual summarization ("tell me what the Spanish, Japanese, and Russian press is saying about the Lewinsky affair") (Cowie et al. 1998 [5]), multimedia news broadcasts ("watch the news and tell me what happened while I was away") (e.g., Merlino and Maybury's article [21]), providing physicians with summaries of on-line medical literature related to a patient's medical record ("summarize and compare the recommended treatments for this patient") (McKeown et al. 1998 [22]), and audio scanning services for the blind ("scan in this page and read it quickly") (Grefenstette 1998 [14]). As the information overload problem grows, and people become increasingly mobile and information-hungry, new applications for text summarization can be expected.
2.2 Multimedia Summarization

This project can be classified as a multimedia summarization system, where the input and/or output need not to be text. In this case it is a combination of the video news broadcast combined with the accompanying closed-caption (Subtitles). We use video scene change detection, together with topic detection to identify possible news story boundary, and lexical chains on the closed-caption to summarize the text. A “news story boundaries” is the logical boundary that separates two individual news stories (i.e. a News Article on a particular topic) given a sequence of sentences.

This is a new area with research work in a very early stage, but with the growing availability of multimedia information in our computing environments, likely to be the most important of all.

Two broad cases can be distinguished based on input and output: cases where source and summary are in the same media and cases where the source is in one media, the summary in the other. Techniques may leverage cross-media information in fusing across media during the analysis or transformation phases of summarization, or in integration across media during synthesis.

An interesting example of this leveraging is found in the work of (Takeshita et al. 1995 [33]) who selects representative images from video by analyzing the topic structure (specific to the genre and the Japanese language) of the accompanying closed caption text.

2.2.1 Different types of summaries

Traditionally, generic summaries aimed at a broad readership community and written by authors or professional abstractors served as surrogates for full-text. However, as our computing environments have continued to accommodate full-text searching, browsing, and personalized information filtering, user-focused summaries have assumed increasing importance. Such summaries rely on a specification of a user information need, such as an area of interest, topic, or query. It should be borne in mind that the notion of a truly generic summary is problematic, since some background assumptions of an-audience are involved in every case.

A broad distinction is usually drawn between indicative summaries, which are used to indicate what topics are addressed in the source text, and thus can be used to alert the user as to the source content, and informative summaries, which are intended to cover the concepts in the source text to the extent possible given the compression
rate. These types of summaries are often contrasted with evaluative or critical summaries, which offer a critique of the source.

In this project emphasis was placed on the production of indicative abstracts (i.e. abstracts that allow a searcher to screen a body of literature to decide which documents deserve more detailed attention) rather than on the production of informative abstracts (i.e. abstracts that can serve as substitutes for the document). Producing a quality informative summary of an arbitrary text remains a challenge which requires full understanding of the text. Indicative summaries, which can be used to quickly decide whether a text a worth reading, are naturally easier to produce. Furthermore, it is also hypothesized that an extract of a document (i.e. a selection of “significant” sentences of a document) can serve as an abstract.

2.2.2 Text Summarization Techniques

There are several ways in which one can characterize different approaches to text summarization. One useful way is to examine the level of processing. Based on this, summarization can be characterized as approaching the problem at the surface, entity or discourse levels.

**Surface-level** approaches tend to represent information in terms of shallow features which are then selectively combined together to yield a salience function used to extract information. These features include:

- Thematic features (presence of statistically salient terms, based on term frequencies statistics)
- Location (position in text, position in paragraph, section depth, particular sections)
- Background (presence of terms from the title or headings in the text)
- Cue words and phrases (e.g. in-text summary cues such as “in summary”, “our investigation”, emphasers such as “important”, “in particular” as well as domain-specific ‘bonus’ and ‘stigma’ terms.)

**Entity-level** approaches build an internal representation for text, modeling text entities and their relationships. These approaches tend to represent patterns of connectivity in the text to help determine what is salient. Relationships between entities include:

- Similarity (e.g. Vocabulary overlap)
- Proximity (distance between text units)
- Thesaural relationships among words
- Coreference (i.e. of referring expressions such as noun phrases)
• Syntactic relations (based on parse trees)

**Discourse-level** approaches model the global structure of the text, and its relation to communicative goals. This structure can include:

• Format of the document  
• Threads of topics as they are revealed in the text  
• Rhetorical structure of the text, such as argumentation or narrative structure

**Note:** A discourse-level approach does not apply in this situation because of the informative structure of the captured closed-caption. The text input is in the form of a series of consecutive sentences which may include one or more story boundaries. There is no paragraph structure and the accuracy of the closed caption might be varied by aerial reception, and as a result punctuation might be lost and odd characters occasionally appear. Hence it is unreliable to build the analysis upon the structure of the text.

### 2.3 Lexical Chains

#### 2.3.1 Why lexical chain method is chosen

When deciding which algorithm should be used for this project, a vast amount of different algorithms were studied, ranging from surface-level approaches, entity-level to discourse-level approaches as mentioned earlier:

Summaries can be built on a deep semantic analysis of the text. For example in (McKeown & Radev 1995 [22]), McKeown and Radev investigate ways to produce a coherent summary of several texts describing the same event, when a full semantic representation of the source texts is available. This type of abstraction is the most expressive, yet very domain dependent and computational power is demanding.

On the other hand, summaries can be built from a shallow linguistic analysis of the text. From all journals Barzilay and Elhadad studied, they concluded that all the techniques presented are easily computed and rely on formal clues found in the text (e.g. word frequencies, title words, location, cue phrases, sentence length, etc). As reported in (Paice 1990 [23]), location and cue phrases produced better results then the word frequency method, and can be accurately computed.

However, Barzilay and Elhadad also pointed out that there are limitation on location and cue phrases method; that when the number of rhetorical markers changes critically, there are large difference in accuracy. Techniques relying on formal clues can be seen as a high gamble.
Method that rely more on the content do not suffer from this brittleness, and Lexical Chains is an example of such a method. The method Barzilay and Elhadad presented rely on word distribution and lexical links among them to approximate content in a more robust form. This produces a summary of an original text without requiring its full semantic interpretation, but instead relies on a model of topic progression in the text derived from lexical chains.

This algorithm requires the use of a WordNet Thesaurus and a part-of speech tagger from GATE System [12]. Summarization proceeds in 4 steps: the original text is segmented, lexical chains are constructed, strong chains are identified and significant sentences are extracted.

_Thesaural relationships among words_

By considering the semantic meaning of each verb or noun, we can detect relationships between different words (synonymy, hyponymy, part-of relations). This is a vital part of the process of the constructing a lexical chain.

### 2.3.2 Procedures for constructing Lexical Chains

There are **3 stages** for constructing lexical chains:

1. Select a set of candidate words
2. For each candidate word, find an appropriate chain relying on a relatedness criterion among members of the chain
3. If it is found, insert the word in the chain and update it accordingly

A lexical chain is created by taking a new text word and finding a related chain for it according to the “relatedness criteria”.

### 2.3.3 Relatedness Criterion Defined

When constructing lexical chains, we only consider **nouns** and **noun compounds**. These are referred as ‘candidate words’. For each new candidate words encountered, the meaning of the word is examined. If the meaning is not matched with any of the existing chains, a new chain is created to hold this word, and the meaning is associated with this new chain. Otherwise, the word is connected to the matching chain.
Here we need to be careful when dealing with polysemous words (words that appear in more than one synsets). It means they have more than one meaning and we need to consider all the meaning of the word is a possible branch. This leads to exponential growth of our search space and needs to be addressed. (More details in next section)

Here we define the relatedness criterion that decides whether to include a new word in constructing lexical chains. This algorithm is designed by Barzilay and Elhadad and is extracted from their journal.

Relatedness of words is determined in terms of distance between their occurrences and the shape of the path connecting them in the WordNet thesaurus database. Three kinds of relation are defined: extra-strong (between a word and its repetition), strong (between two words connected by a WordNet relation – Synonymy and Hypernymy) and medium-strong when the link between the synsets of the words is larger than one.

In selecting which chain to insert given a candidate word, extra-strong chains are preferred to strong relations, which itself is similarly preferred to medium-strong relations.

There are limitations on the maximum length allowed for each type of relation: for extra-strong relations, there is no limit on the distance, for strong relations, it is limited to a window of seven sentences; and for medium strong relations, it is within three sentences back.

2.3.4 The Lexical Chain Algorithm

Let us define the algorithm by the help of a work example and a few illustrative diagrams.

We apply our algorithm for the following passage:

Mr. Kenny is the person that invented an anesthetic machine which uses micro-computers to control the rate at which an anesthetic is pumped into the blood. Such machines are nothing new. But his device uses two micro-computers to achieve much closer monitoring of the pump feeding the anesthetic into the patient.

1. Highlighted in bold are the candidate words. For each candidate word, we consider all the possible meanings of the word. We use a node to store the candidate word, together with its meanings (referred as senses in the notation of WordNet), i.e.

   {Mr., Mister}
2. When a new word $N$ with senses \{n1, n2, ..., nX\} is processed, we try to find all the WordNet relation between $N$ with all the words previously discovered. If a relation is found, a link is established. Like so,

![Diagram showing WordNet relations between Mr., Person, and their senses.]

3. There can be more than one meaning associated with a word, therefore as the process goes on we have a list of interpretations. E.g. the word person has two senses: “human being” (we call this person-1) and “grammatical category of pronouns and verb forms” (person-2). The choice of sense for “person” splits the chain world into 2 different interpretations are shown below:

![Diagram showing two interpretations of person and Mr.]

4. If we consider the word “machine”, which has 5 senses, we found that we have several interpretations:
5. Under the assumption that the text is Cohesive, we define the best interpretation as the one with the most connections (edges in the graph). In this case, the second interpretation at the end of step 3 is chosen. We define score of an interpretation as the sum of its chain scores. A chain score is determined by the number and weight of the relations between chain members.

6. Clearly if we continue this process the number of interpretations greatly increases, and we have limited computational power. To solve this problem, after a certain threshold (no. of steps), we prune the weak interpretations that has the weakest total chain scores. In the end, we select from the pool the strongest interpretation.

This is what the graph would look like after the algorithm:
2.3.5 Building Summaries using Lexical Chains

After processing in the previous stages, we have a list of lexical chains extracted from our original. How do we decide which one to choose to represent the whole document?

- **Scoring Chains**

In order to use lexical chains as outlined above, one must first identify the strongest chains among all those that are produced by our algorithm. As is frequent in summarization, there is no formal way to evaluate chain strength (as there is no formal method to evaluate the quality of a summary).

According to Barzilay and Elhadad’s method, lexical chains are scored based on the following 2 parameters:
**Length:** The number of occurrences of members of the chain.

**Homogeneity Index:** 1 – the number of distinct occurrences divided by the length.

The Score function:

\[
\text{Score(Chain)} = \text{Length} \times \text{Homogeneity Index}
\]

The strong chains are those which satisfy the “Strength Criterion”:

\[
\text{Score(Chain)} > \text{Average(Scores)} + 2 \times \text{StandardDeviation(Scores)}
\]

- **Extracting Significant Sentences**

Once strong chains have been selected, the next step the summarization algorithm is to extract full sentences from the original text based on chain distribution.

For each strong chain, we choose a list of “representative words”. We define “representative word” as *a word that has a frequency in the chain no less then the average word frequency in the chain.*

For each strong chain, we choose the sentence that contains the first appearance of a representative chain member in the text.
2.4 WordNet – Online Lexical Database

WordNet is an online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, and adjectives are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets.

Such a database may be useful to this project when trying to identify the possible list of "key entities". We define a list of key entities to be a list of nouns / noun compounds in a document that are classified by categories. (E.g. a list of persons, a list of locations, etc) Such lists can give useful information as to indicate the topic that is being discussed in a news story.

2.4.1 Background of WordNet

The following information was extracted from a news article [17]:

Nearly two decades ago, the Princeton University psychology professor needed a decent computerized dictionary to help devise experimental tests to determine how children's brains learn language. The major dictionary publishers, however, all wanted several thousand dollars in fees before they would turn over their software.

"I said, 'The hell with you. I'll make my own dictionary,'" recalled Miller, 81. So the Princeton professor got a small government grant and a stack of dictionaries and set out to type in all the nouns. His wife took the adjectives, while a colleague took the verbs. With that, WordNet and the next generation of dictionaries were born.

Instead of just listing the words and their definitions, Miller decided to show how every word is linked to another. Type in the word "tree" and the user gets not only the definition, synonyms and antonyms, but the hypernyms (a tree is a kind of what?), meronyms (what are the parts of a tree?) and more. The user can also find a list of thousands of trees, from yellowwood to the Tree of Knowledge, and even all words that contain the letters t-r-e-e.

At last count, the WordNet had grown into an unprecedented web of 138,838 English words linked hundreds of thousands of different ways. Linguists call Miller's project one of the biggest leaps for dictionaries since scholars sat down to write the epic Oxford English Dictionary in 1879. Online dictionaries modeled after WordNet are being built at universities around the world for more than a dozen languages, from Basque to Bulgarian.

For more information on WordNet, please refer to reference [12]:
GATE is an infrastructure for developing and deploying software components that process human language. GATE helps the developer in three ways:

i. by specifying an architecture, or organizational structure, for language processing software;
ii. by providing a framework, or class library, that implements the architecture and can be used to embed language processing capabilities in diverse applications;
iii. by providing a development environment built on top of the framework made up of convenient graphical tools for developing components.

### 2.5.1 Named Entity recognition

The simplest and most reliable IE technology is *Named Entity recognition* (NE). NE systems identify all the names of people, places, organisations, dates, and amounts of money. This is part of the GATE language engineering architecture and development environment.

NE recognition can be performed at 96% accuracy; the current GATE system performs at 92% accuracy. Given that human annotators do not perform to the 100% level (measured in MUC by inter-annotator comparisons), NE recognition can now be said to function at human performance levels, and applications of the technology are increasing rapidly as a result.

The process is weakly domain dependent, i.e. changing the subject matter of the texts being processed from financial news to other types of news would involve some changes to the system, and changing from news to scientific papers would involve quite large changes.

For more information on NE, see reference [34]:

### 2.5.2 Coreference resolution

Coreference resolution (CO) involves identifying identity relations between entities in texts. These entities are both those identified by NE recognition and anaphoric references to those entities. For example, in

*Alas, poor Yorick, I knew him well.*

Coreference resolution would tie ``Yorick'' with ``him'' (and ``I'' with Hamlet, if that information was present in the surrounding text).
This process is less relevant to users than other IE tasks (i.e. whereas the other tasks produce output that is of obvious utility for the application user, this task is more relevant to the needs of the application developer). For text browsing purposes we might use CO to highlight all occurrences of the same object or provide hypertext links between them. CO technology might also be used to make links between documents, though this is not currently part of the MUC programmed. The main significance of this task, however, is as a building block for TE and ST. CO enables the association of descriptive information scattered across texts with the entities to which it refers. To continue the hackneyed Shakespeare example, Coreference resolution might allow us to situate Yorick in Denmark.

2.5.3 Tokeniser

The tokeniser is responsible for segmenting the input text into words and sentences. More advanced tokenisers (also called preprocessors) attempt to recognize proper names, acronyms, phrasal constructions, etc, as single tokens, usually employing specialized dictionaries and finite-state sub-grammars.

Text tokens segmented by the tokeniser are essentially just strings. After tokenisation, the text tokens are then sent to the morphological classifier which is responsible for classifying string-tokens as word-tokens with sets of morpho-syntactic features. This is usually implemented as lexicon lookup -- words are listed in the lexicon with their morpho-syntactic features and the lookup retrieves all possible readings for a given word.

2.5.4 Part of Speech Tagger (POS-tagging)

The task of POS-tagging is to assign part of speech tags to words reflecting their syntactic category. But often, words can belong to different syntactic categories in different contexts. For instance, the string “books” can have two readings: in the sentence he books tickets the word “books” is a third person singular verb, but in the sentence he reads books it is a plural noun. A POS-tagger should segment a word, determine its possible readings, and assign the right reading given the context.

At the end of the tagging process the input text-string is segmented into words each of which is assigned a single POS-tag, like so:

He/PRP books/VBZ tickets/NNS./SENT

Which identifies that ‘He’ is referring to a personal pronoun, ‘books’ is a verb, ‘tickets’ is a noun plural, and ‘.’ is the end of sentence.
For more information on the notations used here, see reference [13].

2.6 **Video Segmentation**

In this project the text summarization and text segmentation algorithms are coupled with the video segmentation algorithm by Marcus Pickering in his final year project Video Search Engine (Pickering, M. 2000 [24]). The video segmentation algorithm considers the color histogram between successive video frames in order to determine whether there is a “video scene change”. The criterion of this algorithm is whenever a video scene change occurs, there will be a significant change between two video frames, the difference in pixel intensities of the two frames will be great.

We define a “**Video Segment**” to be a video clip that contains no scene change.

This algorithm is combined with the text segmentation algorithms discussed above to facilitate the detection of news story boundaries.

2.7 **Streaming Media Technologies**

The output format used for this system is a streaming media format. Streaming media is being a standard method for delivering real-time video data over the internet. By ‘streaming’ it means that the content is delivered to the client in real-time, which means there are no download wait and no file to take up the space on the hard drive. Streaming media combines the ease and real-time aspect of television and radio with the interactive power of the Web.

The two main stream companies who defines the industrial standard for streaming media technologies are Microsoft and RealNetworks, and RealMedia format is chosen for this project, due to the reasons that RealMedia supports Linux platforms (which is the platform this project is built on) as well as Windows and Macintosh.
2.8 Real-Time Streaming Protocol (RTSP)

RealSystem uses RTSP to control the streaming of video packets from the server to clients. Below is a diagram illustrating what RTSP does:

![Diagram of RealSystem Server and RealPlayer](image)

Designed specifically for streaming, RTSP lets RealSystem Server adjust streaming data to keep clips playing smoothly. When two clips play side-by-side, for example, RealPlayer communicates with RealSystem Server about each clip's progress, indicating how much data it needs to keep playback synchronized. RealSystem Server can then adjust the data flow to compensate for changing network conditions, reducing low priority data if necessary to ensure that crucial data gets through. Communication like this is not possible through HTTP.

2.9 SMIL format

When your streaming presentation contains multiple clips — such as a video and streaming text played together — Synchronized Multimedia Integration Language (SMIL, pronounced "smile") is a simple but powerful markup language that lets you coordinate the clips so that they each play how, when and where you want them to.

**Understanding SMIL**

SMIL is a standard language defined by the World Wide Web Consortium (W3C). It's designed to be the standardized markup language for playing streaming media clips in media players, just as HTML is a standard language for creating Web pages that display in Web browsers.

For example, in its simplest form, a SMIL file lists multiple clips played in sequence:
Once the video clips (e.g., video, text, still images, etc.) are encoded in their streaming formats, they are assembled together using SMIL.

2.9.1 Advantages of SMIL:

Here we justify the reasons that SMIL format is chosen for this project.

Assemble RealText from a video clip

Using SMIL format we can assemble the news clip with the accompanying closed-caption (Subtitles). We can layout the presentation in RealPlayer such that the running closed-caption is synchronized with the video. This ‘running text’ that has time information is stored in RealText format, as described later in the section.

Here is an example of how SMIL can be used to design the layout and assemble the news with RealText.

<smil>
  <head>
    <meta name="title" content="201010_0000.smil" />
    <layout type="text/smil-basic-layout">
      <region id="VideoChannel" title="VideoChannel" left="0" top="0" height="240" width="320" background-color="#888888" fit="meet" />
      <region id="TextChannel" title="TextChannel" left="0" top="240" height="120" width="320" background-color="#888888" fit="fill" />
    </layout>
  </head>

  <body>
    <par title="multiplexor">
      <video src="201010_0000.rm" id="Video" region="VideoChannel" title="Video" fill="freeze" />
      <textstream src="201010_0000.rt" id="Subtitles" region="TextChannel" title="Titles" fill="freeze" />
    </par>
  </body>
</smil>
2.9.2 Use clips from different locations

Using SMIL format we can assemble news video clips that are located physically on different servers. For example if we have 2 RealMedia files on serverA and serverB respectively, we can concatenate the 2 files like this:

```xml
<smil>
<body>
<video src="rtsp://serverA.com/one.rm"/>
<video src="rtsp://serverB.com/two.rm"/>
</body>
</smil>
```

2.9.3 Create interactive SMIL files

SMIL files can be generated dynamically using scripting languages; therefore we can have a web interface for the News Retrieval System which generates the SMIL file dynamically which points to the video location, stores in the Central Data Store.

For more information on SMIL format, refer to RealNetworks or the following references:


SMIL Specification from W3C:

http://www.w3.org/TR/REC-smil/

RealSystem Production Guide:

2.10 RealText

Also defined by RealNetworks, this is a markup language for streaming text. Using RealText we can set a piece of text to appear at a specific time during the video playback.

**Example RealText file**

```
<window
type="generic"
duration="01:00.0"
bgcolor="#0033cc"
link="#ffffff"
width="176"
height="950"
underline_hyperlinks="true"
/>
<font size=2 face="arial">
Ossifragi pessimus divinus senesceret tremulus quadrupei, etiam matrimoni insectat oratori, utcunque Medusa senesceret chirographi, quod bellus saburre agnascor tremulus chirographi, semper ossifragi vocificat satis utilitas oratori, etiam rures corrumperet fiducias, ut umbraculi locari pretosius oratori, utcunque fiducias neglegenter adquireret Aquae Sulis. Caesar insectat quinquennalis quadrupei.
</font>
</window>
```

For more information on RealText, refer to RealNetworks RealSystem SDK [29].
2.11 Information Retrieval System

A document contains a mixture of multimedia information extracted from the original source which includes the original video and closed-caption, automatically extracted key frames, key sentences (summary passage), the most frequent named key entities. These information combined serves as an indicative multimedia summarization extract.

Using a list of key entities extracted from the text summarization process, an index will be generated. This index can be used to form an index table which tells whether a specific keyword has appeared in a document. This information is stored in a database and forms a search engine. Because it uses indexing table the time required for the search will be very short (< 1 sec).

A web interface shall be implemented, using ASP/JSP/scripting languages. When a user search on a key word, e.g. car, the search engine will return a list of documents, each containing the word ‘car’ in the video somewhere.

Through a web interface, user can search for a piece of news by a keyword. For each requests the Central Data Store returns a list of relevant documents.

With these distilled information the user can quickly locate the piece of multimedia data he is looking for.
3 System Specification

The aim of this project is to implement an automatic News Summarization and Extraction/Retrieval System. Such a system involves a wide range of information extraction and retrieval techniques such as video capture, Subtitles (closed-caption) capture, video processing and text processing technologies, e.g. text segmentation and text summarization.

An automatically summarized paragraph gives a good indication of the general content of a piece of recorded video, and helps a user to decide whether it is relevant to what he is searching for.

3.1 Hypothesis

In this project the following assumptions were made:

i. A news broadcast capture consists of one or more COMPLETE news stories, i.e. we always capture the whole BBC news broadcast program.

ii. News stories are always constructed by complete and disjoint Video Segments, i.e. a story does not contain part of a video segment, and one video segment (2.6) belongs to ONE and only ONE news story.

Below is a diagram to clearly illustrate the understanding of a “News Story”:

![Graphical Representation of captured closed-caption (Teletext)](image)

**Explanation:**

The bottom line represents a list of words (GREEN) captured by our closed-caption (Subtitles) capturer. Logically a sentence (LIGHT BLUE) is constructed by a list of words. A Video Segment (BLUE) consists of one or more sentences and similarly a News Story (MAGENTA) consists of one or more Video Segments.

A list Video Segments will be constructed using a video scene change algorithm (2.6), and the hypothesis (3.1) states that the video will be divided up into fine grains such
that we do not have a video segment that contains the content from more than one news story.

### 3.2 Definition of a News Summarization and Retrieval system

We define the system by identifying the **4 main components** of the system; each of them will be discussed in full details in the following sections:

![Main Parts of the system](image)

Information flows from left to right, in different form or format in each of the above boxes.

#### 3.2.1 Source Data Capture

Input to this system includes video broadcast news (0) and closed-caption (3.4.2) (otherwise known as Teletext Subtitles in the UK); both captured using a PC TV capture card.

#### 3.2.2 Data Analysis

The captured information is analyzed and broken down into logical separated entities and a structural representation for information extraction and segmentation of the video and text into separate *news stories*.

#### 3.2.3 Media Generation

The most vital information stored is then condensed and extracted to produce *news abstracts*, i.e. a summarized passage for each news story. The video and closed-caption is segmented according to the detected news story boundary. For each video clip key frames as extracted out as static pictures. List of *key entities* are also extracted from each news story. This could include a list of locations, persons mentioned in the news, times/dates of events, etc.

#### 3.2.4 Information Retrieval System
All the above mentioned information is stored in a central database, and a web based information retrieval system (3.4.10) can be constructed. Through the use or a search engine data can be located and extracted efficiently.

“Optimal mixture of multimedia information”
Using summarized contents, screen snapshots, video as well as the accompanying subtitles, combined with a reversed indexed keyword search engine and a web interface, this system provides a tool for searching, browsing and summarizing TV news broadcasts.

For each item returned in the search results, this optimal mixture of multimedia information is presented to the user, which helps the users to identify the news story accurately and in minimal time.
3.3 Overall Architecture

3.4 Component Specification

3.4.1 Video Capture

- Real-time news broadcast should be captured using a PC TV capture card connected to a TV aerial.
- The video should be recorded using motion JPEG codec (mjpeg) and stored in AVI (Audio Video Interleaved) format.
• Daily news (from BBC1) should be automatically recorded twice (One o’clock and ten o’clock news) by a scheduler, which sets the channel and initiates the recording at the specific times of the day.

3.4.2 Close-Caption (Subtitles) capture

• Real-time closed caption should be captured using the same TV capture card and TV aerial at the same time as the video recording occurs.

• Subtitles are captured by the subtitles capturer, which does the following:
  o Records the time and the subtitle together, so that the text can be synchronized back together.
  o Stores the output in text (ASCII) format.

• Any illegal characters, meaningless punctuation (e.g..#^&<>) caused by occasional interruption of TV signals should be removed.

• Any repetitions due to the design flaw of subtitles reception should be removed to prevent miscalculations of term frequencies, i.e.

  The following two lines:
  
  line 1: Good morning. More rail strike starts
  line 2: Good morning. More rail strike starts at Paddington Station.

  Should be truncated to:

  line 1: Good morning. More rail strike starts at Paddington Station.
3.4.3 Text Summarization Engine

The captured closed-caption is passed to the text summarization engine for text processing and analysis. This involves several inter-related text analysis processes, and is better explained using the following diagram:

![Text Summarization Engine Diagram]

The following sections give detail Specification of each sub-component.

3.4.4 Natural Language Parser

- Takes the output of the closed-caption capturer (ASCII)
- Use the tokeniser to parse the input text into numbered tokens
• Coreference relations between entities

To increase the accuracy of our summarization algorithms, we can also consider the possibility of the use of Coreference relations (i.e. of referring expressions such as noun phrases), in order to identify whether two sentences are related, thus contributing in the construction of the lexical chain.

*note: This shall be an extension to the system due to time limitation

3.4.5 Semantic Analysis

• The output of the tokeniser is passed into the POS tagger
• Key entities are identified and classified into different categories

3.4.6 Story Segmentation Detection

• The results of video segmentation (3.4.8) are used to extract the part of closed-caption – “text segments” associated with the video segments.
• For each key entities in each text segment, term frequencies (i.e. occurrences of the key entities in that segment) are calculated.
• 2 video segments should join when the key entities mentioned overlaps. This is measured by the following criteria:
  o Consider the 10 key entities that have the high term frequencies in both text segment.
  o 2 text segments should be joined if more than 5 “top-10” key entities appear in both text segments.
• Results are passed to Video Segementer (3.4.8) for video segmentation

3.4.7 Summary Generation

• Summary shall be generated using the Lexical Chain (Barzilay, R., Elhadad, M. 1997 [3]), defined in the Background section (2.3.4).
• The key entities extracted from Semantic Analysis are used for detecting WordNet Relations. This information is used to build the lexical chain (2.3.4)
• Lexical Chains are used to generate the summary (2.3.5)

3.4.8 Video Segmenter

• Takes the (AVI) input format generated by the Video Capturer (3.4.1)
• Scene change algorithm (2.6) shall be used to segment the video (Implemented by M. Pickering 2000 [24]). The results of the algorithm are a list of time values, stored in text (ASCII) format. This is passed to the Story Detection Engine (3.4.6).

• Next it waits for the results to return from the Story Detection Engine (again, a list of time values). This is used to segment the video into video segments.

• Each video segment shall be encoded into RealMedia format (*.rm)

• The associated closed-caption for each video segment shall be encoded in RealText format (*.rt)

• The SMIL file will be generated.

• Relative Path of the SMIL file, RealText and RM file will be stored in the Central Data Store (3.4.9)

3.4.9 Central Data Store

The Central Data Store stores all the analysed results from the previous applied algorithms. This includes video and textual data. Below is a summary of all the information that will be stored in the data store for each news story:

• Video Clip (*.rm) – in RealMedia format

• Segmented Closed Caption (RealText format)

• List of key entities – i.e. people, location, company, time, date (XML or ASCII)

The data store serves as a database to hold a mixture of multimedia information and more importantly as a search engine which provides users with functionality to search through the data store to locate a particular piece of news. In order to achieve this:

• The list of key entities will be used to generate an index to the SMIL files.

3.4.10 The Information Retrieval System

A “document” is an optimal mixture of multimedia information presented to the user. This includes:

• The time and date of the captured news broadcast

• A text summary of the news broadcast

• A list of Persons mentioned in the news

• A list of Locations mentioned

• A list of Organizations or Topics mentioned

• The key frames of the video

• A link to the original full closed-caption
• The video itself in SMIL format

This might involve the following processes:

• A web interface shall be implemented for the user to invoke a search
• Interface should provide a keyword search (refer to key entities)
• Results of search should be return as a list of documents.
4 Design and Implementation

In this section design and implementation of ANSES is broken down into components mentioned in System Specification section.

4.1 System Workflow

The following diagram gives a ground-level view of ANSES, showing the data flow in the complete system.
4.2 System Design Decisions

ANSES is a heterogeneous system in the sense that more than one component in the system relies on other 3rd-party software. These 3rd-party tools were developed in different programming languages. As a result, ANSES was written in more than one language. Below gives the technical background and design decisions of ANSES:

<table>
<thead>
<tr>
<th>Component</th>
<th>Sub-</th>
<th>Language(s)</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed-Caption Capture</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hauppauge WinTV Capture Card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store Video/Audio &amp; Closed Caption</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate corresponding ‘Text’ segments</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parse Video for scene change detection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store ‘Video’ segment boundaries</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect Key Entities and Parts of speech</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect Story Boundaries</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store ‘Story’ segment boundaries</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Summarize stories by Lexical Chains</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Segment and compress Video to RealMedia format and generate key frames.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build reversed index search engine</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate RealText file and SMIL files for presentation.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyword retrieval of documents via web based interface</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Color coding indicates which component each activity belongs to in Overview diagram.
<table>
<thead>
<tr>
<th>Component</th>
<th>com pone nlt</th>
<th>Language</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video/Audio Capture</td>
<td>1,3</td>
<td>C++</td>
<td>Free, open source software ‘streamer’ was used for video capturing.</td>
</tr>
<tr>
<td>Subtitles Capture</td>
<td>2,3</td>
<td>C++</td>
<td>Free, open source software.</td>
</tr>
<tr>
<td>Video Segmenter</td>
<td>4,5,12,13</td>
<td>C++</td>
<td>Tool was provided by Marcus Pickering, which was implemented in C++ and runs on Linux. This tool is adopted and modified to suit system requirements</td>
</tr>
<tr>
<td>Text Summarization Engine</td>
<td>6-10</td>
<td>Perl, Java</td>
<td>Main program was written in Perl, which is best for pattern matching using regular expressions, and string manipulation routines. For detecting key entities, a java system, developed by Sheffield University called GATE is used. Java application was invoked from within Perl code.</td>
</tr>
<tr>
<td>Search Engine for retrieval</td>
<td>11</td>
<td>C</td>
<td>A fast, text based search engine, developed in C was used. Reversed index files are generated for all text in the stories to enable fast retrieval.</td>
</tr>
<tr>
<td>Web Retrieval System</td>
<td>14</td>
<td>Perl CGI</td>
<td>An apache web server, runs on the Linux machine provides a web-based news retrieval interface. Perl CGI scripts were used to query the search engine and return the results to the web page.</td>
</tr>
</tbody>
</table>

Components can be treated as a separate program, and communication between components was done using pipes and files. Although this incurs extra communication overhead, we gain the advantage of the ability to choose the best language/tool available for each task. Moreover, components in ANSES are more modular and can easily be replaced or reused by other systems.

Design and implementation of each component should be considered as one individual unit, which are discussed in details below.
4.3 ANSES Components

4.3.1 Video/Audio Capture

It was required to capture Video/Audio and closed-caption in synchronization. A system was developed to capture automatically each day’s news with its associated closed caption.

![Video capture in Linux](image.jpg)

**Figure 4.** Video capture in Linux

**Hardware**

The Video/Audio was captured using a Hauppauge WinTV capture card, installed on a PC. The TV card was capable of capturing Video and subtitles at the same time, therefore only one TV capture card was needed.

**Software Tools**

A Linux operating system was chosen for developing ANSES because of the availability of open source software and drivers. A major factor for this decision is due to the fact that the Video Scene Change algorithm, implemented by Marcus Pickering and runs on Linux systems. Therefore it follows that the video capture should be done in the same platform. The fact that the retrieval search engine ‘Managing Gigabytes’ was Linux dependent was also a major factor.
4.3.2 Subtitle Capture

The closed-caption was captured by modified open source software that captures closed caption using standard video drivers in Linux. The tool was modified to record the times as well as the subtitles, which allows the subtitles to be synchronized with the video.

Software Tools

Cron was used to schedule the start/stop of both video/audio capture and subtitle capture via a shell script.

Below is a sample of the data captured:

![Sample captured subtitles](image)

Note: the black blocks in between the text are control characters, which were used to record colour information of the subtitles.

4.3.3 Storing Video/Audio & Subtitles

Storage format

At this stage the video/audio was captured in raw format as it is required by the Video Segmenter.

Subtitles are stored in text format, as shown in Figure 5.
4.3.4 Scene Change Detection

After the video is recorded, it is parsed by the Video Segmenter for scene change detection.

![Figure 6. Three sequential frames of a crowd scene, each indicated as shot changes by the pairwise comparison method. The camera is panned to the right.](image-url)

This algorithm detects scene change by applying a colour histogram method and motion vectors in calculating the difference between frames in video, in order to decide whether there is a significant change in the screen. This tool can detect immediate scene change as well as gradual scene change. For more details on the design and implementation of this tool, follow this link: [http://km.doc.ic.ac.uk/video-se/](http://km.doc.ic.ac.uk/video-se/)

Based on this algorithm, we make use of the video scene boundaries as a factor for discovering the true story boundaries in the video. More details are included in the Text Summarization Engine section.

Modifications on this tool were made to suit the needs of ANSES. This tool was based on a linux media player called ‘mplayer’ and was developed in C++. RealProducer SDK from RealNetworks was also used to compress the video and text into suitable format that can be read by RealPlayer. The tool was divided up into 2 parts (corresponds to boxes highlighted in green in System Workflow diagram):

The first part of Video Segmenter is scene change detection. This divides the video into ‘Video segments’ – where each segment represents a single scene. The times of scene change were recorded as they correspond to the start and stop time of a ‘Video segment’. Below is a sample output of Video segment boundaries:
4.3.5 Storing ‘Video segment’ boundaries

The results generated above were stored in a text file and the Text Summarization Engine is invoked.

4.3.6 Generate corresponding ‘Text’ segments

Due to the imperfect nature of terrestrial reception, subtitles captured were not always perfect. These include misspelling of characters and illegal characters caused by noise. In addition, Teletext subtitles were designed for the sole purpose – to be displayed on a TV screen. A design flaw appears as the text is saved to a file. As shown below there are lots of repetitions in the content. (e.g. ‘police and customs officer’ are repeated more than once) This is due to the reason that words are appended onto screen instead of starting with a new line.

Duplicates cause errors in the story boundary detection algorithm, because word frequencies are inaccurate when there are duplications in words. Therefore, after the subtitles were acquired, some data pre-processing were required:

**Removing meaningless characters**

Block character highlighted in circle above are control characters recorded from the subtitle broadcast. They represent color information of the subtitles. These characters,
along with other illegal characters caused by improper reception, should be removed as they bare no meaningful information to our story detection algorithm and key entities detection process.

The following perl code eliminates illegal and CTRL chars:
\$str =~ s/[ \t\n\r[:punct:]]//g;

**Removing duplicates**

There are different situations where duplicates can occur.

(a) **Case 1 – Current line is subset of next line**

The next line is a repetition of the current line, plus a few new words append, as shown above. In the case the 3 lines should truncate into 1, keeping the first timestamp.

(b) **Case 2 – Next line is a subset of current line**

Here at the last 2 lines, we can see another trivial error that the whole of last line has repeated. Hence we can discard the last line.

(c) **Case 3 – Partial duplicates**

After processing, subtitles become:
The problem here (highlighted in the diagram), however, is to more difficult to tackle. This partial match cannot be easily fixed, as it would require a regular expression like so:

```perl
if ($thisline =~ /^(.*)(.+)$/ and $nextline =~ /^$1(.+)/){
    # Partial duplicate found!
}
```

This states that a partial duplicate is found if $thisline can be split into 2 parts, and the second part match the beginning of next line. This does not work at the mechanism of pattern matching in perl as it will try to match as many characters as possible for the first term (.*). As a result the term (.* has the same effect as (.*)(.+).

**Solution:**

A character-by-character regular expression matching was used to solve this problem. Below is a short snippet of code that performs partial match:

```perl
# Split the line into characters
my @chars = split ///, $text;
my $count = length($text);
my $subStr1 = $text;
my $subStr2 = "";
while ($count > 1){
    $subStr1 = substr($text, 0, $count);
    $subStr2 = substr($text, $count, length($text)-$count);
    out("DEBUG," - #$subStr1##$subStr2#); 
    if ($lastline =~ /^(.*)\s\Q$subStr1\E$/){
        my $discard = pop @$new;
        push @$new, [$discard->[0], $lastline.$subStr2];
        $lastline = $lastline.$subStr2;
        next LINE;
    }
    $count--;
}

# If nothing matches, then push this line in
push @$new, [$time, $text];
$lastline = $text;
```

It is also apparent the generic type of partial duplicates cannot be matched (highlighted below). These errors generated by the design flaw of Teletext cannot be removed.

**Rearrange data into complete sentences**

For detection of key entities using GATE, the text should be arranged in complete sentences. This is because POS tagging analysis the structure of a sentence in order to determine what part of speech a word belongs to in the context:
For example,
1: This is a book
2: Tom wants to book tickets for the movie.
In this example the word book belongs to different part of speech. In sentence 1 it is a Noun, whereas in sentence 2 it is a Verb.
However, it was discovered that this method was inappropriate in this situation. As the subtitles captured was noisy, very often the end of line punctuation marks (.,?!?) were not captured. Rearranging the lines into complete sentence would result in destroying the structure of the sentence. Hence this was not included in the final implementation. To get around this problem the sentence splitter in GATE was used instead.

**Generating Text Segments**

After the input data has been cleaned, the subtitles are divided up into Video segments, according to the boundaries detected by Video Segmenter.

Each Text segment consists of one or several lines of text with the timestamp of each line also recorded.
4.3.7 Detect Key Entities and Parts of speech

Introduction
In order to improve the accuracy of the story segmentation algorithm, some means of extracting important keywords from the original text is required. These keywords serve as a good indication to the topic in the context.

General Architecture for Text Engineering (GATE)
Main steps in extracting key entities and POS tagging:
- Initialize GATE
- Create an instance of the following processing resources (Tokeniser class, SentenceSplitter class, POSTagger class, Gazetteer class, Transducer class, Orthomatcher class)
- Load the text document and run each processing resource in order
- Call doc.getAnnotations() to get the list of tokens detected
- From the type of token, determine whether it is Key Entity or POS Tag

Schoolchildren had been given the morning off to With the Jubilee weekend still several days away, parts of Scotland, it seems, are already It wasn’t all good news for the Queen. a report out today Queen, a report out today suggests she should abdicate in favour of the parliament building today on her way to Dundee, by coming here and saying what and saying what she did, shows she’s open to the she’s open to the change in the modern day modern day world I don’t think that is the case when it comes to that is the case when it comes to her own role She has no intention of of abdicating She has made had a – that clear. she – that clear. she has said this is a job for life I don’t think a job for life.

Figure 9. Sample Text Segment

Detected Key Entities

Figure 10. Key Entities detected by GATE, from the sample text segment
**POSTagging**

![Figure 11. Sample results showing the execution of GATE](image)

Shown above are key entities and POS tags extracted from the original passage. For implementation details see source code.

**Accuracy of GATE**

From the results obtained GATE has achieved a very high successful hit rate. More than 90% of the key entities were matched correctly. However, the accuracy of the key entities extraction depends on having a large enough database which stores this list of words. As new word / phrases appear GATE would need to update the database in order to provide a high accuracy.

GATE can also be custom configured by defining new grammar rules. New grammar rules can be added to recognise custom patterns. E.g. to recognise cue words of the news broadcast.
4.3.8 Story Segmentation

Introduction
Current technology makes the automated capture, storage, indexing, and categorization of broadcast news feasible allowing for the development of computational systems that provide for the intelligent browsing and retrieval of News stories (Maybury & Merlino ’97 [21], Kubula, et al., ’00 [16]). To be effective, such systems must be able to partition the undifferentiated input signal into the appropriate sequence of news-story segments.

Many efforts have gone into research in this area, and many solutions exist on text based summarization. Work done based on cues words alone has been moderately successful in the past. Criticism on this approach is that accuracy of algorithm depends heavily on the structure of the text.(Greiff et al, ’00 [15]) attempted using Hidden Markov Model (Rabiner, ’89 [26]) to model the generation of the words produced during a news program, which achieved promising results, but parameters need to be tuned and refined through a optimization problem.

Segmentation Algorithm
In this project we try to achieve story segmentation from a different prospective. Based on the hypothesis made in specification, we assume that a ‘Video segment’ does not go over story boundaries. In other words, true story boundaries coincide with the Video segment boundaries.
Base on this assumption, story segmentation becomes a problem of merging ‘Video segments’ back together to form a complete story.
**Merging Criteria**

In order to distinguish Text segments that belong to different stories, we need to have a **Similarity function**, which calculates how similar 2 Text segments are. This function returns a 'Similarity score'. Comparisons are made between all the video segments, and the Similarity scores were used to decide which Text segments to join and form complete news stories. To illustrate this idea more clearly, consider the following example:

Below is 3 Text segments extracted from the subtitles. These 3 segments all belong to the story about the Queen's Golden Jubilee celebrations and her tour around London:

<table>
<thead>
<tr>
<th>Text Segment 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1483.78 - The <strong>Queen</strong> made history <strong>today</strong> she addressed the <strong>Scottish Parliament</strong> for the first time.</td>
</tr>
<tr>
<td>1483.78 - Her speech to <strong>MSPs</strong> was part of her <strong>Golden Jubilee</strong> tour of the UK.</td>
</tr>
<tr>
<td>1487.06 - Earlier, the Queen was greeted by <strong>thousands</strong> of schoolchildren.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text Segment 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>* 1490.74 - The Royal Party has spent the <strong>morning</strong> in <strong>Aberdeen</strong>.</td>
</tr>
<tr>
<td>* 1492.70 - <strong>Jennie Bond</strong> reports.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text Segment 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>* 1502.62 - It was as big a <strong>welcome</strong> as she's had anywhere so far had anywhere so far in Scotland.</td>
</tr>
<tr>
<td>* 1502.62 - <strong>Aberdeen</strong>'s main street was lined with well-wishers with well-wishers as the Queen visited the <strong>Scottish Parliament's</strong> Parliament's</td>
</tr>
</tbody>
</table>
The key entities are highlighted in different colors. Key Entities play an important role in story segmentation. As shown above they highlight the most important keywords and could indicate the subject of the current story. In the above example it is apparent that the person ‘Queen’ was mentioned in both segment 1 and segment 3. Also the organization ‘Scottish Parliament’ was mentioned in both segments 1 and 3. This gives strong indications that segments 1 and 3 are of the same topic; hence they should be merged together.

Apart from key entities, nouns and stopped words were also taken into account by the similarity function. Nouns (labeled grey in the above example) are extracted by POS (Part-of-speech) tagging in GATE. Stopped words are the list of words which does not contain any stop words – ‘this’, ‘is’, ‘because’, ‘hence’ are such examples. For more examples, a list of stop words is included in the Appendix section.

We measure similarity of 2 Text segments by counting the frequencies of matched Key Entities, nouns and stopped words. Each of the above have a different weighting to the contribution of the final similarity score.

**Similarity Function**

\[
\text{Similarity score} = \sum \frac{\text{freq}(\text{matched word}) \times \text{weight}(\text{matched word})}{\text{distance}}
\]

freq(matched_word) : frequency of matched_word in the text segment

weight(matched_word) : weight of matched_word is derived according to type of word. i.e. location, organisation, ...noun, stopped word

distance : the no. of sentence in between the 2 occurrences of the matched_word.

**Weights of matched word**

Different types of words and key entities, when matched, will generate a different score depending on the type of Key Entity. After tuning and readjusting the following optimum settings were found to produce the best results:

<table>
<thead>
<tr>
<th>Type of Entity Word</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locations</td>
<td>40</td>
</tr>
<tr>
<td>Person</td>
<td>60</td>
</tr>
<tr>
<td>First Person</td>
<td>40</td>
</tr>
<tr>
<td>Organizations</td>
<td>60</td>
</tr>
<tr>
<td>Date</td>
<td>30</td>
</tr>
<tr>
<td>Noun</td>
<td>10</td>
</tr>
<tr>
<td>Stopped Word</td>
<td>5</td>
</tr>
<tr>
<td>---------------</td>
<td>---</td>
</tr>
<tr>
<td>Capital Word</td>
<td>10</td>
</tr>
<tr>
<td><strong>Merge Score</strong></td>
<td><strong>5</strong></td>
</tr>
</tbody>
</table>

The similarity function uses this table to generate a score when comparing 2 segments. A higher score means the entity word higher importance.

Please note this score will be normalized by the distance. When the normalized score is greater than the Merge Score than the 2 segments and all the segments in between will be merged.
Merging Algorithm

Shown below is the simplified, pseudo version of the merging algorithm code:

```perl
sub detect_story_boundaries(){
    @test_window = (-5..-1,1..5);

    # Loop through text segments
    while ($current_index < $total_no_of_segments){
        # Compare text segments within the defined range (@test_window)
        # Finds the highest score segment and merge
        foreach $offset in (@test_window){
            my $target = $current_index + $offset;
            $score = compare_segments($current_index,$offset);
        }
    }

    # Loop through @test_window and check for segments to merge
    # Merge text segments with $score > $threshold
    ...
    $index=join_text_segments($merge_target,$current_index);
    ...
    # Update $current_index;
}
```

The algorithm calls `compare_segments()` on between the current segment and each of the segments in the test_window. The size of test_window defines the search space this algorithm operates on.

This imposes a localization effect, which states that a segment should only merge to neighbouring segments. Also the similarity function states that as the distance between the current segment and the segment in comparison increases, the score should be normalized according.

After merging, text segment transforms to ‘Story segments’ – each text segment represents a complete news story. The boundaries times are the start and stop time of the video clip.
**ANSES Demo Program**

Halfway through development, a demo program was written in C#, which provides a visual interface with the perl script to perform Key entities extraction and testing of story segmentation. Shown below is the screen capture of the demo:

![Screen capture of the demo program](image)

Figure 12. DEMO Program written in C#, using Active Perl Plug-in for .NET IDE

Source code of demo can also be found in public submission directory.
4.3.9 Storing ‘Story segment’ boundaries

After running Story Segmentation algorithm, detected ‘Story segment’ boundaries are recorded to a file. These are in the format of [ start_time, end_time ] of the news story, similar to those of a ‘Video segment’.

Figure 13. Start and stop times of news stories. The first line denotes the total no. of stories detected by Text Summarization Engine.

This list is then processed by the Video Segmenter to encode the news stories into video clips.

4.3.10 Summarization by Lexical Chains

Introduction

At this stage complete news stories have been extracted from the subtitles. Each segment contains the full text of the stories. Given a list of such stories, without any means summarization user might find difficult and inconvenient to read the whole content of each story, in order to locate a particular piece of news.

We implement summarization by the use of Lexical Chains.

There are 3 stages for constructing lexical chains:
1. Select a set of candidate words
2. For each candidate word, find an appropriate chain relying on a relatedness criterion among members of the chain
3. If it is found, insert the word in the chain and update it accordingly

A lexical chain is created by taking a new text word and finding a related chain for it according to the “relatedness criteria”.

Full details about the algorithm and the theory behind are described in Background section.

WordNet 1.6

In order to be able to measure the relatedness criteria of 2 words, i.e.
whether a
word is related to each other even they are different words, a synonymy dictionary
was required. This was provided by WordNet 1.6. A perl module (QueryData.pm) was used to interface with the flat file database system on WordNet.

**Candidate Words**

All nouns in the story were chosen as candidate words for lexical chains. Noun compounds were not implemented due to time limitation.

**Generating Lexical Chains**

Pseudo code of summarization algorithm:

```plaintext
# Lexical Chain Algorithm (implemented by summarize_story())
# Maintains a list of interpretations.
# Each interpretation consists of a list of lexical chains
# Each Chain is a list of pair of nodes...
# Each pair of node represents a link, and is in the form:
# [$word1,$line1,$word2,$line2]
# When a link is detected, check existing chains, and
# possible append onto chain
# Otherwise, Create new chain with the new pair.
# Loop until reach end_line_index
# At the end of each loop prunes the weak interpretations

my @interpretations;
# Each interpretation consists of a list of Lexical Chains
# Each Lexical Chain consists of a pair of nodes, like this:
# [ [$noun1,$sentence_no1,$sense1,$noun2,$sentence_no2,$sense2], ..... ]

# Step 1
# Consider all the senses of a new candidate word
# Each sense can be a list of words with same meaning, like so,
# Example:
# 5 senses of "car"
# Sense 1
# car, auto, automobile, machine, motorcar
# Sense 2
# car, railcar, railway car, railroad car
# Sense 3
# car, gondola
# Sense 4
# car, elevator car
# Sense 5
# cable car, car
#

# Step 2
# For each noun in the story:
# 1. Multiply the no. of interpretations by no. of senses for
```
Notes
Details of the Lexical Chain theory are included in Background section. Above is an outline of the algorithm implemented in Perl.

Selecting Strong Chains
After all the nouns in the story have been considered, the interpretation with the highest score was chosen to represent the story. Then the 3 highest scoring chains were chosen to be the Strong chains.

\[
\text{chain score} = \sum \text{score generated by each link in the chain}
\]

Scores generated by each link depends on the type of link:

There are 3 types of links: extra-strong (between a word and its repetition), strong (between two words connected by a WordNet relation – Synonymy and Hypernymy) and medium-strong when the link between the synsets of the words is larger than one.

In selecting which chain to insert given a candidate word, extra-strong chains are preferred to strong relations, which itself is similarly preferred to medium-strong relations.
Generating the summary

For summary generation, an approach which makes use of the detected key entities was used. This approach was different to the method mentioned in Background Section.

Two sentences were extracted from each strong chain to be the summary of the story. Each strong chain has a list of nodes. Each node consists of i) the word and ii) the sentence index from which the word appears in.

Below were the steps used for extracting summary from Strong Chains:
   1. Select the representative word of the chain
   2. Extract important sentences to be the summary

Representative word selected

In each chain, the word with the highest occurrence was chosen to be the representative word.

Extract important sentences.

Once the representative word was chosen, for each sentence this word appears in, we calculate the score of that function using the weighting function defined in section 4.3.8.

\[
\text{sentence score} = \sum \left( \text{no. of key entity(i) detected} \times \text{weight(type of key entity(i))} \right)
\]

Limitations – Dealing with search space explosion

During execution a list of interpretations were kept. In each interpretation each noun considered so far has a specific meaning. The algorithm considers each noun in each line of the story in turn. Every time a new noun is considered we look up all the meanings (senses) of that noun. Then each meaning of the noun is considered when updating the Lexical Chains. This effectively multiplies the no. of interpretations by the no. of senses found in WordNet for this noun.

This leads to a search space explosion. On average one noun has 8 different meanings (senses). Therefore for a typical story containing 30 nouns, the no. of interpretations are:

\[8^{30} = 1.238 \times 10^{27}\] interpretations
This requirement exceeds the current resources available, both in computational time and memory resources.

To deal with this problem, the size of the search space had to be reduced. After every noun is added, the list of interpretations were sorted in descending order of the interpretation scores (interpretation score == sum of scores of lexical chains). Then the top 20 interpretations were kept, and the others discarded.

Pruning the search space means it is not guaranteed that the optimum interpretation can be found, but at the current level of resources available, this is inevitable. (Lexical chain algorithm on top 20 interpretations on a 500 MHz machine with 256 Mb of memory took 45 minutes to run.)

4.3.11 Managing Gigabytes Information Retrieval Engine

**Managing Gigabytes**

Managing Gigabytes is a Linux based, full-text query and retrieval system. A database can be created from a given set of documents which can then be queried to retrieve the documents. Every word in the collection of documents is indexed. Its main components are:

- **mgbuild** – a csh script that executes all the appropriate programs in the correct order to completely build an MG system database ready for queries to be made by mgquery
- **mg_get** – a script that is used to extract text from documents when building the database
- **mgquery** – program which queries the index in response to user requests. This program is run from the CGI script that powers the web interface.

**Modifying .mg_getrc to index video material**

.mg_getrc is a configuration file storied in the user directory. It instructs mgbuild where to get the data for building MG. The following new entry was added:

anses PARA /home/public/anses_out

mgbuild looks for all the files in this directory and calls mg_get to extract documents from each file. Documents are separated by CTRL-B characters.
4.3.12 Encoding into RealMedia format

The second part of the Video Segmenter deals with generation of the news clips in RealMedia format. The start and stop time for each story segment would be transformed into start and stop times of the video clips.

The original video and subtitles were partitioned according to these times and were compressed into RealMedia format and RealText format. Both the text and video were kept synchronized.

The main steps in encoding the video and audio to the RealMedia format using the RealProducer SDK are as follows:

- Create the RealMedia Build Engine and set up its properties.
- Set up the clip properties.
- Call `PrepareToEncode`.
- Call `Encode` on each media sample (audio and video) in any order.
- Call `DoneEncoding`.

4.3.13 RealText and SMIL file generation

RealText

![Sample RealText file](image)

The above diagram shows the format RealText file generated. Note that attributes on the `<window>` tag sets the properties of the text area. Each line of text requires a
<time> tag which specifies when to display the text on the screen. The default color is white, but whenever there is a colour change in the subtitles, an extra <font> tag will be added to instruct the colour change in the RealText clip.

**SMIL file generation**

![Figure 15. An example SMIL file](image)

SMIL files were generated to link the RealMedia file together with RealText. It describes the layout of the presentation to RealPlayer, and serves as an assembler which combines all the media resources in one file, and sets the order of playback for each resource in the video clip.

More details of how to encode RealMedia, RealText and SMIL files can be found in Real SDK documentation.

Shown below is the result of the generated output:
Figure 16. A SMIL presentation playing using RealPlayer
4.3.14 Web-based News Retrieval Interface

Introduction

The retrieval interface provides a text based search engine to retrieve news stories. CGI scripts were used to interact with user inputs and execute searches to query the Managing Gigabytes database.

Retrieving News Stories

The search engine is shown in the above diagram. User enters the query through the textbox. The CGI script passes the query to the mgquery program, as mentioned in the previous section. The results from mgquery was formatted by the CGI script and displayed on the web page.

It is possible for thousands of results to be returned, and a user is not likely to want to wait while all the JPEGs are downloaded, so they are presented in pages of 10, with the option to move to the next page, or to a page of the user’s choice.

Narrowing down the search
Given that a list of documents returned by the search engine, a typical user would then need to narrow down the search to a particular piece of news. He/she can do so by examining the information highlighted within each document:

A document consist multiple extracts from the news story, including:

- key frame of video clip
- date of news story
- length of video clip
- full story subtitles
- a short summary of the news story
- detected Key Entities (Locations, Organisation, Person, First Person, Dates)

![Figure 18. Drill down to see full story](image)

**Viewing full story text**

Having identified the document of interest, the user can drill down to see the complete subtitles for the news video clip. The keyword used in the query is also highlighted in red.
Browsing the Video

Having found a clip of interest, the user may want to browse the video around that point, perhaps due to a miss in story segmentation algorithm which caused a story to be split over 2 video clips, or to find other current news. Functionality has been added to allow browsing backwards and forwards from any of the clips retrieved. By clicking the previous and next button, the user is taken to a second screen.

An example is given in the above diagram, where the key frame for the first result in Figure 18 has been clicked.

The clip that was clicked is highlighted in the centre of the screen, and next and previous clips are shown. Clicking the next or previous button again enables the user to browse though the whole video.
Playing the Video

When the user click on the key frame, RealPlayer will be launched to show the SMIL presentation, as shown in Figure 20. However in order to play the video RealPlayer is required to be installed. The user will be informed of this and a link is provided for the user to download RealPlayer if necessary.
5 Testing and Evaluation

This section details on testing on individual components of ANSES, followed by the testing on the automated batch process for the overall system. Finally, an evaluation on the usability of the system is provided.

5.1 Source Data Capture

The result of video performance was satisfactory at a capture frame rate of 10 fps using motion JPEG codec. Typically for motion pictures and films, a frame rate of 25 fps would be used to ensure smooth motion detectable by human eye. However in this case, there were limited motion in the course of a news broadcast, and a frame rate of 10 fps provided satisfactory results. A frame rate of 15 fps was experimented but due to the limit of processing resources on a 450 MHz machine with 128 Mb of memory, the audio and video were slightly out of sync.

Closed caption capture provided satisfactory results in terms of accuracy to synchronize with the video. However due to the design flaw of Teletext subtitles, mentioned in section 4.3.6, numerous duplicates appeared in captured text. As a result extra text manipulating routines were written to clean up the duplicates.

Due to noise in reception, punctuations in the subtitles were not accurate. As a result the subtitles were not rearranged to form complete sentences, which were proposed in the original specification.

5.2 Story Segmentation

Much of the effort were put into optimising the segmentation algorithm, as it would greatly affects the accuracy of the summarization algorithm, and moreover the usability of the whole system.

5.2.1 Test carried out for segmentation algorithm

5 news recordings were used for training data, and another 5 were manually processed for comparison. Each news recording was 30 minutes of news broadcast. The ground truth was identified by manually viewing the video and the accompanying subtitles, and these were compared to the boundaries detected by the segmentation algorithm.

5.2.2 Test Results
Below are the tabularized results. Details for each test are included in the appendix B section.

Test 1 – 15/01/2002 BBC news at one o’clock

<table>
<thead>
<tr>
<th>Actual Figures</th>
<th>Not Boundary</th>
<th>Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Boundary</td>
<td>78</td>
<td>14</td>
</tr>
<tr>
<td>Boundary</td>
<td>1</td>
<td>26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Percentages</th>
<th>Not Boundary</th>
<th>Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Boundary</td>
<td>0.848</td>
<td>0.118</td>
</tr>
<tr>
<td>Boundary</td>
<td>0.037</td>
<td>0.963</td>
</tr>
</tbody>
</table>

Percentage of decision correctly made : \(1 - \frac{15}{119} = 87.4\%\)

Test 2 – 04/03/2002 BBC news at one o’clock

<table>
<thead>
<tr>
<th>Actual Figures</th>
<th>Not Boundary</th>
<th>Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Boundary</td>
<td>77</td>
<td>2</td>
</tr>
<tr>
<td>Boundary</td>
<td>1</td>
<td>18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Percentages</th>
<th>Not Boundary</th>
<th>Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Boundary</td>
<td>0.975</td>
<td>0.025</td>
</tr>
<tr>
<td>Boundary</td>
<td>0.053</td>
<td>0.947</td>
</tr>
</tbody>
</table>

Percentage of decision correctly made : \(1 - \frac{3}{97} = 96.9\%\)

Test 3 – 10/01/2002 BBC news at ten o’clock

<table>
<thead>
<tr>
<th>Actual Figures</th>
<th>Not Boundary</th>
<th>Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Boundary</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>Boundary</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Percentages</th>
<th>Not Boundary</th>
<th>Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Boundary</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Boundary</td>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td>Not Boundary</td>
<td>34</td>
<td>14</td>
</tr>
<tr>
<td>-------------</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>Boundary</td>
<td>0</td>
<td>19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Not Boundary</th>
<th>0.708</th>
<th>0.292</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary</td>
<td>0.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Percentage of decision correctly made: \(1 - \frac{14}{63} = 77.8\%\)

---

**Test 4 – 04/02/2002 BBC news at one o’clock**

No. of stories detected: 40  
No. of real stories: 18  
No. of Video Segments detected: 115

False Positive: 22  
False Negative: 4

<table>
<thead>
<tr>
<th>Actual Figures</th>
<th>Not Boundary</th>
<th>Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Boundary</td>
<td>75</td>
<td>22</td>
</tr>
<tr>
<td>Boundary</td>
<td>4</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Percentages</th>
<th>Not Boundary</th>
<th>Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Boundary</td>
<td>0.773</td>
<td>0.226</td>
</tr>
<tr>
<td>Boundary</td>
<td>0.222</td>
<td>0.778</td>
</tr>
</tbody>
</table>

Percentage of decision correctly made: \(1 - \frac{18}{115} = 84.3\%\)

---

**Test 5 – 22/05/2002 BBC news at one o’clock**

No. of stories detected: 19  
No. of real stories: 15  
No. of Video Segments detected: 127

False Positive: 6  
False Negative: 3

<table>
<thead>
<tr>
<th>Actual Figures</th>
<th>Ground Truth</th>
<th>Not Boundary</th>
<th>Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Boundary</td>
<td>106</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Boundary</td>
<td>3</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Percentages</th>
<th>Ground Truth</th>
<th>Not Boundary</th>
<th>Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Boundary</td>
<td>0.946</td>
<td>0.054</td>
<td></td>
</tr>
<tr>
<td>Boundary</td>
<td>0.2</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

Percentage of decision correctly made: \(1 - \frac{9}{127} = 92.9\%\)

---

**SUMMARY**
Accuracy of segmentation algorithm = 1 - 67/521 = 87.1%

**Comments**

The above table shows the overall performance of the story segmentation algorithm. The testing was done based on the idea that ANSES makes a decision on whether a video segment boundary is a story boundary, for every ‘Video segment’ detected by scene change detection.

86.4% of non-story boundaries (TRUE NEGATIVE) were correctly identified by ANSES.

90.8% of story boundaries (TRUE POSITIVE) were correctly identified by ANSES.

The segmentation algorithm can be said to be 87.1% accurate on making decision on whether a given video segment is a story segment or not.

However, this figure is not directly related to the usability of the system. We need to consider each of the error cases (FALSE POSITIVE) and (FALSE NEGATIVE) in assessing how much impact each kind of error will have effect on result.

1. **FALSE POSITIVES** (13.6%) – these are cases where a video segment was incorrectly identified as story boundaries. As a result the complete story is divided into smaller video segments. This is acceptable as all the video fragments will be successfully retrieved by the search engine, and the user will not spend any additional time when searching for a particular piece of news.

2. **FALSE NEGATIVES** (9.2%) – cases where a story boundary is missed. This causes 2 news stories to be merged in one video clip. As a result the user would need to read through unrelated information in the video and subtitles before locating the new story. Clearly this has a great impact on the usability of the system as it affects the turnaround time of each search. From the test results it was shown that 9.2% of the story boundaries were missed, which is a relatively low figure, and it means that ANSES is 91.8% efficient.
The accuracy of segmentation relies on the hypothesis that a video segment does not go over a story boundary. There are cases where this hypothesis was broken (e.g. Test 1 – line 69, Test 3 – line 54, Test 4 – line 122). This directly caused the story boundary to be missed (FALSE NEGATIVE) as those video segments would contain keywords from BOTH news stories.

An interesting observation is the grain size of the detected video segments. The segment boundaries detected by the scene change algorithm were generally very close together. This means that time between each scene change on average is short. It is apparent that very often a video segment only consists of one line. Therefore one way to improve on the algorithm could be to discard video segments completely and extract the news stories directly from comparison between each line of subtitles. However this has the trade off that the story boundaries detected in this way might not synchronize with the video.

It is also worth mentioning that 2 different versions of GATE were experimented during development. It was found that using the later version (build 825) more key entities were detected, which increased the accuracy of the segmentation algorithm. It is an example whereby the underlying technologies improved and in turn improved the overall result of the end product.

5.3 Story Summarization

There are no formal methods to evaluate a summarization algorithm, as 2 human generated summaries from the same passage could be very different. Therefore an intrinsic method was used to evaluate the Lexical Chain algorithm used in this project.

5.3.1 Outline of the experiment

The approach was to create an ideal summary, written by merging summaries provided by multiple human subjects using methods such as majority opinion, union or intersection. The output of the summarization algorithm was then compared with the ideal summary.

10 summaries were compared. 5 were short stories and 5 were long stories. Only complete stories were considered i.e. story segmentation was accurate. A sample test result highlights the difficulty in evaluating the summarization algorithm:

**Story 1 – Machine Generated results**

By failing to take action against Zimbabwe before the election there, **Tony Blair said the Commonwealth had struck the lowest common denominator deal**. Bush hats for the Commonwealth leaders tonight, a leaders tonight, a sort of togetherness at a barbecue. But Tony Blair, bare-headed, wasn't really joining when he lacerated the compromise deal on Zimbabwe: a fudge to hold together a fractured club. There is no point in diplomatic language
about this. The statement that has appeared is the lowest common denominator. We've postponed the day of judgment on Zimbabwe. I think that was the wrong thing to do. We should have taken action now but we suspend Zimbabwe, to take tough action if Mugabe ends up the victor in a rigged election through violence and intimidation. It took the Commonwealth leaders three days of plain speaking to break the Zimbabwe deadlock. Just three leaders will decide after the elections if the country must be kicked out. The two Africans on the panel, South Africa and Nigeria, have always fought against sanctions but now they do seem ready to Whatever has to be done must be done and it is not a question of whether I am comfortable or not. What has to be done must be done. In Zimbabwe, President Mugabe is fighting hard to prolong his fighting hard to prolong his rule, blaming whites for oppressing blacks with Tony Blair's blessing. But his challenger Morgan Tsvangirai is drawing the opposition urging voters to be brave while observers have seen the results of attacks on opposition homes. Their interim report on the intimidation is said to be searing. Back in Australia, Zimbabwe's opposition phoned the Commonwealth outcome to their leader. The They've let us down, we can't be angry, we don't have time for that, we simply have to keep going. So no-one is happy, as a bruising Commonwealth encounter draws close. For most, the fudge here leaves a sour taste. But at least the Commonwealth has armed itself for possible action to help the people of Zimbabwe if crisis there moves to meltdown. Then the Commonwealth really will face

Story 1 – Human selected summary

By failing to take action against Zimbabwe before the election there, Tony Blair said the Commonwealth had struck the lowest common denominator deal. Bush hats for the Commonwealth leaders tonight, a leaders tonight, a sort of togetherness at a barbecue. But Tony Blair, bare-headed, wasn't really joining when he lacerated the compromise deal on Zimbabwe: a fudge to hold together a fractured club. There is no point in diplomatic language about this. The statement that has appeared is the lowest common denominator. We've postponed the day of judgment on Zimbabwe. I think that was the wrong thing to do. We should have taken action now but we suspend Zimbabwe, to take tough action if Mugabe ends up the victor in a rigged election through violence and intimidation. It took the Commonwealth leaders three days of plain speaking to break the Zimbabwe deadlock. Just three leaders will decide after the elections if the country must be kicked out. The two Africans on the panel, South Africa and Nigeria, have always fought against sanctions but now they do seem ready to Whatever has to be done must be done and it is not a question of whether I am comfortable or not. What has to be done must be done. In Zimbabwe, President Mugabe is fighting hard to prolong his fighting hard to prolong his rule, blaming whites for oppressing blacks with Tony Blair's blessing. But his challenger Morgan Tsvangirai is drawing the opposition urging voters to be brave while observers have seen the results of attacks on opposition homes. Their interim report on the intimidation is said to be searing. Back in Australia, Zimbabwe's opposition phoned the Commonwealth outcome to their leader. The They've let us down, we can't be angry, we don't have time for that, we simply have to keep going. So no-one is happy, as a bruising Commonwealth encounter draws close. For most, the fudge here leaves a sour taste. But at least the Commonwealth has armed itself for possible action to help the people of Zimbabwe if crisis there moves to meltdown. Then the Commonwealth really will face

In this above example the extracted summary was highlighted in yellow, and the ideal summary is highlighted in green. From the above results, 3 sentence out of 5 were matched by the generated summary. However, as mentioned before, the ideal summary chosen cannot be proven to be a perfect summary, therefore the match ratio might not have any meaning to the accuracy of the system.

On the other hand, since the summary generated is an extraction summary, i.e. extracts from original content as summary, it is arguable that an accurate summary can be provided by a few sentences of the original content. From the above example it is apparent that important key entities were spreaded out over more than 5 lines. The algorithm selects only 2 sentences from each strong chain. Since there were no limitation on the length of the sentences extracted, the accuracy of the results varies.
Nevertheless, extracts from the original content will have some indication to the topic of the news story, which combined together with the key entities detected, should serve as a reasonable summary for a news story.

Due to time limitation, the lexical chain algorithm was ran under a small scope size of 20. It is very likely that when this limit is increased the accuracy of the results would increases as well.
5.4 Web-based News Retrieval Interface

Managing Gigabytes provided a satisfactory search engine interface, with an average query turnaround time of 2 seconds.

The web page was designed to be a user friendly interface, which presents all the information on a maximum no. of 2 pages at a time. The control of the interface coincides with the most popular search engines (e.g. Yahoo, Google), which reduces users’ learning time required to adjust to the interface.

Detected Key Entities was proven to be most effective in indicating the topic of the news story. With identified Locations, organisation, persons and dates in the story, it greatly reduces the search time. E.g. (To list all the news stories of ‘Tony Blair’).

5.4.1 Limitations

(a) Presentation of clips
The key frame shown on web page is the 64th frame into the video. Although in theory this would enable us to story content it was impossible to guarantee that the important person/scene to be captured always at this particular moment. In particular
when there is a missed story boundary the key frame contributes very little in indicating the topic of the video clip.

(b) **Advance Search**

Due to the fact that there were no advance search features in this interface, which lets user to perform another search within the current search results.
6  Conclusions and Future Work

6.1 Achievements

In this project a fully automated News extraction and summarization system have been successfully implemented. From the stage of source data capturing, video analysis, text analysis to media generation and refreshing of the search engine database, no human intervention was required.

We successfully designed a segmentation algorithm which uses both Video information as well as Closed Caption (subtitles) to identify news story boundaries with satisfactory accuracy. The algorithm does not rely on the structure of the text or cue words, therefore ANSES does not suffer the brittleness of other news systems that rely on cue words.

Lexical Chain summarization was implemented to provide summaries of news stories. Third party tools such as GATE and WordNet, provided text recognition abilities and vital sources of information which enabled the implementation of such algorithms.

Each component was implemented in a modular fashion, which enables one to easily replace/reuse any components of ANSES. In particular, the segmentation algorithm can be reused to segment any generic type of text.

ANSES can be seen as a pre-processing engine, which automates the collection of video and subtitle information, and enable further data-mining applications built on top.

During development, due to the unfamiliarity of Linux operating system, coding and test was carried out in Windows environment. As a result, the whole system, apart from source data capturing, was made platform independent. Combined with any AVI and closed caption capturer, ANSES can be run on any platform.
6.2 Future Work

Due to the large scope of the system, many areas have not been fully explored and there are limitations on each of the component developed. Limitations on each individual component are mentioned at the end of each section. The following summarizes those into highlighted areas of improvement for the whole system:

6.2.1 Word Compounds

In the Lexical Chain summarization algorithm, only individual nouns were considered. This algorithm shall be extended to include noun compounds. Noun compounds provide the algorithm more information on the context of the text. Consider the compound ‘Football stadium’, it captures the meaning of a sport stadium, which otherwise would not be captured using the individual nouns ‘football’ and ‘stadium’. This could improve the accuracy by generating more links in the lexical chains.

6.2.2 Coreferences

As mentioned in the specification. Coreference of expressions was not considered. With coreferencing one would be able to link the following two sentences.

“Ladies and gentlemen, let me introduce to you, Mr. Tony Blair.”
“He is currently the Prime Minister of the United Kingdom.”

Using coreference we could detect that the word ‘He’ on the second line was referring to the person ‘Tony Blair’. This feature could be included in future work.

6.2.3 Partial Merging

From the results of the segmentation algorithm, it was discovered that occasionally, a story boundary was missed due to the fact that a video segment go over a story boundary. This could be resolved by splitting the video segment into 2 parts and perform partial merging. This way in principal perfect story boundaries could be extracted.

6.2.4 Improvement on Summarization

The quality of the news summaries have to be improved. Investigations could be carried out on either in searching for another generic text summarization algorithm, or to improve the current algorithm. The algorithm search space can be increased as memory becomes cheaper and greater processing power becomes available. Also the criteria in strong chain selection should be optimized to extract more meaningful sentences.
7 Source Code

Complete source code together with configuration instructions can be found in the public submission directory.

http://www.doc.ic.ac.uk/~lkhw98/project/
Appendix A - References


[31] **Skorokhodko, E.F. (1972).** Adaptive Method of Automatic Abstracting and Indexing


[34] **User guide to Information Extraction -** http://www.dcs.shef.ac.uk/~hamish/IE/userguide/main.html


Appendix B - Test Results

Test Scripts on segmentation algorithm can be found in this section.