A Two-step Approach to Video Retrieval based on ASR transcriptions

Ken Schmidt, Thomas Körner, Stephan Heinich, Thomas Wilhelm
Chemnitz University of Technology
Department of Computer Science
Straße der Nationen 62, 09111 Chemnitz, Germany
{sken, koert, heist, wilt}@hrz.tu-chemnitz.de

ABSTRACT

In this paper, we describe our experiments for the Rich Speech Retrieval Task at the MediaEval Benchmark Initiative 2011. We start with a brief overview on the used framework and its structure. Our experiments indicate that a two-step retrieval approach and applying a spell checker can improve the quality of retrieval results in the given scenario. Finally, we discuss other techniques that may further improve the quality of the results.

Categories and Subject Descriptors

H.3.1 [Content Analysis and Indexing] and H.3.3 [Information Search and Retrieval]

General Terms

Measurement, Experimentation, Languages

Keywords

Information retrieval, automatic speech recognition, multimedia retrieval

1. INTRODUCTION

The aim of Rich Speech Retrieval Task at MediaEval 2011 [1] was to provide jump-in points for videos given a user query. We had 350 hours of internet video material from blip.tv shows, mostly in English language. Automatic speech recognition transcripts in two versions one from 2010 and another from 2011 served as basic metadata. We used the transcripts from 2010 in our experiments. Additional metadata, such as tags and shot segmentation were supplied in the form of XML documents. Along with the shot segmentation, key frames of every scene were also provided.

2. SYSTEM DESCRIPTION

For the participation at MediaEval 2011, we based our experiments on the information retrieval framework *Xtrieval* (eXtensible reTRIeval and EVALuation) [2]. Xtrieval is being developed at Chemnitz University of Technology since 2005. The idea behind this framework was to create a flexible and adjustable framework with state-of-art retrieval-techniques. Xtrieval provides several Java-based object-orientated API's for different retrieval tasks. There are four main components: indexing, retrieval, evaluation and the user interface. The first three are the main components; we did not use the UI component.

The Xtrieval framework itself works in the following way. The textual data from various sources is captured by the indexer. The indexing process is based on a data collection concept that abstracts the actual collection under investigation. The indexing itself is done by the Indexer class. The resulting index is used for searching. The topics (or queries), index and retrieval parts are necessary to run an evaluation experiment. Using the experiment data structure, we can evaluate various indexing and search approaches.

Xtrieval is capable of providing different retrieval API's, such as Apache Lucene[1], Terrier[2] and Lemur[3], through a common programming interface. We used Lucene as retrieval core for the present experiments. There are many possibilities to tune components and parameters in Xtrieval. Here, we opted to weight a part of the query to emphasize its impact on the result.

We were required to submit one specifically configured run and up to four arbitrary runs. The respective data was extracted from the XML documents using XPath. Before indexing we applied some standard token filters, such as lowercase transformation, Porter stemming, and a standard list of stopwords for English[4]. We built several indexes using the development data set. Various system configurations were tested to investigate the specific characteristics of the data and the respective retrieval problem.

The initial idea of preferring a *two-step* approach over standard retrieval is based on the following thoughts. If the search terms do not appear all together in a single segment, it is hard to define the most relevant segment. We supposed that results can be improved by first identifying a possibly relevant document (based on its full transcript) and determining the most relevant segment in a second step. Thus, we created *two* basic indexes. In the first, we indexed the ASR transcripts and used the *documents* as identifiers to create a (preliminary) ranking. For the second index we treated the *speech segments* as documents in order to create a ranking on segment level.

In the first step of retrieval, we identify the most relevant document, weight it and add its ID to the query. This modified query is submitted to the second index in order to identify the most relevant segment, related to the first document. Here, a weight of 0.0 for the document ID in the modified query means no effect (or modification). In contrast to that, assigning a weight of 1.0 to the ID results in retrieving only segments of documents that were already identified before, i.e. it defaults to standard retrieval.

[1] http://lucene.apache.org/
[2] http://terrier.org/
[3] http://www.lemurproject.org/
[4] http://members.unine.ch/jacques.savoy/clef/index.html

Because the documents that were retrieved from the first index were not always relevant, we decided to give them only a partial influence on the results for the search in the second index. Obviously, using different weights produced different results, which are reported in more detail in the next section. We decided to use this *two-step* retrieval approach, because we observed better results than with a standard search when we were experimenting with the development set.

3. CONFIGURATIONS AND RESULTS

In this section we report the results that were obtained by alternating parameters of our system configuration. We provide a brief discussion of our results for the official evaluation along with some observations on additional experiments.

3.1 Submitted Runs

We submitted five different configurations of our experiments. For the required run (*run1*) we indexed the complete ASR information from the XML transcripts of 2010 in the first index. Our second index contained only the segments of speech detected by ASR. So we did for all our runs. In our retrieval experiments we used the *title* and the *short title* from the queries. The field *act* was discarded, because it significantly decreased the result quality in preliminary experiments. The starting point of the identified speech segment was returned as the required jump-in point.

For our arbitrary configurable runs, we also included the metadata information in the first index. This increased the result quality on the development data. During our experiments we observed, that some of the queries did contain spelling errors. Since these mistakes were not intended, they reflected a realistic use case. Thus, we decided to use a spellchecker and we filtered the queries with it. We used the Google spell check API[5], which returned a weighted list of suggested words. The first word from this list was added to the search query. Applying this technique resulted in a slight improvement of quality.

The weighting of the first identified document was varied from 0.1 to 0.9. We recognized that in case of a non-relevant document on the top of the ranking, giving a high weight to this document ID in the modified query significantly decreased the result quality. A reason could be the fact that higher weights pushed down alternative results in the result list. Our results are listed in detail in the following table.

Table 1: Experiments and results for test and development set

experiment id	*mGAP**		
test set	window 10s	window 30s	window 60s
run1 0.1 TT ASR	**0.1432**	**0.2420**	0.3051
run2 0.1 TT	**0.1432**	0.2419	0.3051
run3 0.1 FCT	**0.1432**	**0.2420**	**0.3052**
run4 0.6 FCFC	0.1164	0.2039	0.2557
run5 0.8 FCFC	0.1140	0.2004	0.2511
development set			
run1 0.1 TT ASR	0.2325	0.2886	0.3228
run2 0.1 TT	**0.2926**	0.3585	0.3946
run3 0.1 FCT	0.2925	**0.3620**	0.4017
run4 0.6 FCFC	0.2863	0.3609	**0.4071**
run5 0.8 FCFC	0.2863	0.3609	**0.4071**

*mGAP: mean generalized average precision

Run1 is the required and restricted experiment. Hence, it is based on the 2010 ASR transcripts only. Our second run (and also third, fourth and fifth) includes additional metadata in the first index. In *run2* we searched with the raw (original) query. The weighting for the first document was 0.1. For the third experiment (*run3*) we filtered the query using the Google spellchecker in the first search step (FC). The weighting remained at 0.1. For the last two experiments *run4* and *run5* we increased the weighting to 0.6 and 0.8. Additionally, we used the spellchecker twice (FCFC), for document and segment search step.

Our experiments with the development data achieved much higher mGAP values than the test set results. This may be due to the smaller size of the development set.

3.2 Additional Experiments

In addition to the submitted runs we followed some other approaches. We also tried searching in one index that contained all available information. But there were no improvements. We decided to explore the possibilities of text recognition in the key frame pictures. But the recognition results were very bad, because of the moderate image quality. Only in some cases we could extract words which were ready to use.

We see some other problems with short queries. Queries that contain for example the name “Hillary Clinton” and only a few general terms deliver bad results. This may be due to the fact that the term “Clinton” appears more often alone or in conjunction with “Bill” and refers to the former president of the USA.

4. CONCLUSIONS AND FUTURE WORK

Our tests showed that a two-step retrieval approach works well for the present scenario. But only in the case that the first identified document is not overrated to avoid excluding alternative solutions. A spelling checker works well in cases of misspelled names. If there are several variations for spelling a name like Denis (or also Dennis), the spell checker adds the most common notation. We suppose a case where the ASR system returns a misspelled name (Denis vs. Dennis). The user knows the person is spelled with two “n” but it was transcribed only with one. So the spell checker adds “Denis” to the query and the document could be found by the user.

Future work could focus on improving quality of the first document. Another possibility is to take various languages of the documents into account. We disregarded this, because there are only few videos with other languages than English. Furthermore, there are prospects to work with the shot segmentation: combining the shot time with the speech segment time to improve the jump-in point. Another disregarded option is to make use of the provided tags. It might be possible to categorize the query in such a way that the search heaps only videos with tags fitting the query.

5. ACKNOWLEDGMENTS

This publication was prepared as a part of the research initiative sachsMedia (http://sachsmedia.tv), which is funded by the German Federal Ministry of Education and Research under the grant reference number 03IP608. The authors take sole responsibility for the contents of this publication.

6. REFERENCES

[1] Martha Larson, Maria Eskevich, Roeland Ordelman, Christoph Kofler, Sebastian Schmeideke, Gareth J. F. Jones: Overview of MediaEval 2011 Rich Speech Retrieval Task and Genre Tagging Task, MediaEval 2011 Workshop.

[2] Jens Kürsten, Thomas Wilhelm, Maximilian Eibl: Extensible Retrieval and Evaluation Framework: Xtrieval. In Proceedings of the LWA – Workshop FGIR, p. 107-110.

[5] http://code.google.com/p/google-api-spelling-java/