Final version of benchmarking methods

Deliverable D4.5

TOSCA-MP identifier:	TOSCAMP-D4.5-JRS-BenchmarkingMethods-v10.docx
Deliverable number:	D4.5
Internal reviewers:	R. Cattoni, D. Giuliani (FBK)
Work package / task:	WP4 / T2,3
Document status:	Final
Confidentiality:	Public

Version	Date	Reason of change
1	2013-06-20	Document setup, based on D4.4
2	2013-08-09	Added related work, more experimental results, notes
3	2013-08-22	Details and analysis of diff tools for metadata
4	2013-08-29	Intermediate version distributed at integration meeting
5	2013-12-20	Update on experiments, tools, and benchmarking search
6	2014-02-14	Added description of XML differencing tools and benchmarking service
7	2014-03-07	Added more experimental results
8	2014-03-20	Updated text on simulation, added experimental results
9	2014-04-03	Version for internal review
10	2014-04-14	Final version

Acknowledgement: The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 287532.

Table of Contents

List of Figures

List of Tables

1 Executive Summary

This deliverable discusses how formal task models can be used for the assessment of content annotation and search tools, both on component and system level. It presents an approach for cost assessment and provides experimental results using the proposed methods. It also describes the integration of benchmarking tool into an analysis workflows.

Media production processes can benefit from the use of automatic information extraction tools that analyse multimedia content and provide information for content description, indexing and search. However, it is difficult to assess the impact of a specific information extraction tool (e.g., genre classification) on the overall process in terms of quality improvements or cost savings w.r.t. manual processes. One observation is that existing benchmarks of individual components do not always reflect the applicability of the methods for a certain task in a process. In order to address this issue, we propose to assess tools in the context of a specific task of a real media production workflow, rather than evaluating these tools in an isolated lab setting.

In this document, we discuss issues of specifying and representing ground truth. The basic approach is to consider a set of edit operations between two metadata documents (of which one might be part of ground truth). These edit operations and related costs can be used to model also cases with multiple choices for ground truth or ground truth items with different confidence levels.

We describe the use of task models for benchmarking and simulation. For benchmarking, we model error propagation in content analysis and search tasks. We provide a number of experimental results that show that the approach can be also used in cases where the ground truth is incomplete or no ground truth but only generated results are available for intermediate steps. For simulation, the task model is transformed into a business process, and costs are estimated based on different parameterisation of the process. Finally we discuss the integration of the proposed benchmarking and simulation approaches with the service oriented architecture of TOSCA-MP.

2 Introduction

2.1 Purpose of this Document

This document proposes task-based approaches for the assessment of content annotation and search methods. This includes both benchmarking methods considering the contribution of individual tools to solving an analysis and/or search task in media production as well as methods for estimating the cost efficiency of introducing automatic tools into the process.

2.2 Scope of this Document

This document describes a task-based view on assessment of content annotation and search tools, proposes approaches for task-based benchmarking and cost estimation based on simulation. The proposed methods are validated in experiments. The software tools implementing these approaches are described.

2.3 Status of this Document

Final

2.4 Related Documents

This document is based on the task modelling method and the set of task models described in *D4.3 Task Models*, and is an update and extension of *D4.4 First Version of Benchmarking Methods*.

3 Motivation

This part of the work of TOSCA-MP has originally been motivated by the EBU MIM/SCAIE[1] working group, established 2007, which aims at bringing automatic information extraction tools into media production processes. It is an acquired concept that media production processes can benefit from the use of automatic information extraction tools that analyse multimedia content and provide information for content description, indexing and search. However, it is difficult to assess the impact of a specific information extraction tool (e.g., genre classification) on the overall process in terms of quality improvements or cost savings w.r.t. manual processes. One observation is that existing benchmarks of individual components do not always reflect the applicability of the methods for a certain task in a process. In order to address this issue, we propose to look at evaluation of automatic information extraction methods from a novel perspective. Rather than evaluating these tools in an isolated lab setting, the tools are assessed in the context of a specific task of a real media production workflow.

For example, if we take the benchmark data for a person identification software based on recognition of the speaker voice, typically we get figures related to reference data sets used by the research community to compare results, and we do not have any indication on how errors (e.g., false detections) impact in real usages of the technologies. Here are other examples, where there are mismatches between the commonly used benchmarking approaches of components and their contribution to solving an actual task. Typically, precision and recall of shot boundaries w.r.t. a ground truth is evaluated. However, missed shot boundaries coinciding with scene boundaries will strongly impact the result of a subsequent scene clustering tool, while missed shot boundaries within scenes and several false positives might be tolerable. For person identification, there might be cues in multiple modalities (e.g. face, text insert, name mentioned) that contribute to successful identification. Depending on the structure of the data set the overall result varies with the performance of components working on the different modalities. For example, if persons are identified by text inserts, and video OCR performs very well, this might mask missed detections of a face detector. When performing search for content related to a specific location, video segments might not be retrieved because a certain step in indexing failed. However, mistakes of various analysis tools might impact the result in a very different ways. Visual clustering might assign a clip to an entirely different location, making retrieval of the segment for the correct location impossible. ASR or video OCR might misspell an unknown proper name, but phonetic matching against a place vocabulary might still help to assign it correctly.

The second aspect concerns the cost effectiveness of automation in media production processes. Automatic tools are not perfect, and manual correction and validation might be needed. The effort needed for the manual intervention depends not only on the performance of the automatic process, but also on the user friendliness of the tools used for validation and correction, and on the required quality level of the results for a certain process. For example, for automatic subtitling exactly timed boundaries in the speech transcript are required, while for semantic enrichment based on recognised named entities (e.g., annotating the identifier of a place mentioned in the ASR transcript) a few seconds offset in timing do not matter. Models of tasks in the media production process put the tools into context and provide a holistic picture of the costs involved. Simulations comparing fully manual execution of the tasks with variants in which some subtasks are automated, and considering different performance values of these tasks, can provide information about the cost effectiveness of the use of (specific) automatic tools. This can also help increasing the acceptance of using automatic content annotation tools in the media production process.

This deliverable is structured as follows. Section 4 discusses the needed metadata for the approach proposed in this document, i.e. representations for ground truth and deviations from it, and the tools to determine these edits. In Section 5 we propose an approach for assessing individual analysis components in a task context, and in Section 6 we discuss benchmarking of search and result presentation. Section 7 describes a range of experiments that we have performed using the proposed methods and discusses their results. Section 8 discusses cost assessment using the task context, and Section 9 describes the integration into a service oriented architecture like the one developed by TOSCA-MP.

[1] http://tech.ebu.ch/groups/pscaie

4 Representing Ground Truth and Deviations

The methods described in this document need representations of ground truth as well as of the deviations between ground truth and outputs of certain components and systems.

4.1 Ground truth

The term ground truth denotes the correct output of a component or algorithm for a defined input. While this is straight forward e.g. for a binary classifier, ground truth may have a quite diverse nature depending on the type of the component under evaluation.

In addition to straight forward cases where we have a well-defined and correct output for a defined unit of input, the following issues related to ground truth occur in practical problems.

Incompleteness. In some cases ground truth is not complete due to cost reasons. A typical example is the evaluation of precision and recall in retrieval problems: For example, checking whether all reported examples from a pool of systems are correct (in order to assess precision) is much less effort than checking the whole data set for all relevant items (in order to assess recall). In this example, pooling of results is an approach to get an estimate of recall, but a relevant item found by a system not in the pool will not be counted as correct. Another notion of incompleteness can be found in ground truth for temporal segmentation, where it may be specified that a correct boundary occurs around that time, but it may not be frame precise.

Alternative choices. In some cases, alternative choices – possibly subject to constraints – may exist. One example is temporal segmentation into scenes, where there may be alternative options to place the scene boundaries, with the constraint that only one of a set in a specific time range can be chosen. The Differential Edit Distance (DED) has been proposed as a metric for assessing video segmentation [Sidiropoulos, 2012]. It is a variant of the edit distance agnostic to the specific symbol representing an element and measures differences in segment boundaries. However, based on the application it has been designed for, it makes assumptions about the convexity of segments and does not allow for temporal shifts.

Set of reference examples and subjective scores. In cases where the quality of a result is judged subjectively, a set of reference examples or users' judgments may be provided. Examples include the evaluation of machine translation (using BLEU [Papineni, 2002]), of video summaries (using VERT [Li, 2010]) or of temporal segmentation using a model for tolerance [Piazolla, 2010]. For almost all quality analysis methods the presence of an impairment (e.g. blur, blocking) or the perceived strength of an objectively present impairment (e.g., flicker) depends on statistical comparison to a set of subjective judgements.

Confidence levels. Similar to the case of reference examples, multiple human annotators may not fully agree on the ground truth. However, there may be some items in the ground truth that get higher inter-operator agreement than others (e.g., for scene boundaries). In such a case it is desirable not to treat all reference examples independently, but to use the different levels of confidence in the evaluation.

Different sampling rates. Different sampling intervals for temporally dense data (e.g., sampled data of quality properties such as noise or grain level), it might be required to compare information extracted with different sampling intervals

4.2 Deviation from ground truth

Commonly used performance metrics are aimed at providing a concise representation of the performance of a component, ideally a single number. In contrast, we are also interested here in a fine-grained analysis of different types of errors, not making assumptions about the evaluation metric. Also, we are not only interested in what is different between a result and a ground truth, but also in more details on *how* a result differs from the ground truth. For example, a misplaced boundary in a temporal segmentation should not represented as one missed and one added, but as shifted (if it is still in a certain temporal proximity).

In general, we are interested in a measure of difference between two metadata documents (of which one might describe ground truth). We thus propose using a set of edit operations and an associated edit distance for this purpose. The set of edit operations will of course not be universal, but might be

different depending on the type of metadata we are dealing with. Even if we are not interested in the actual value of the edit distance, i.e., the total cost of edit operations, it still makes sense to assign costs to edit operations, as it can give precedence to operations. For example, if a shift of a boundary with +/- *t* frames has a smaller cost than deleting it, we will choose this edit. In addition, the cost for shifting could not only be a fixed value, but depends on the number of frames the boundary has to be shifted. Costs can be used to support alternative choice of metadata elements (within some constraints, e.g., a time window), as well as confidence levels, by assigning higher costs for performing edits on items with higher confidence. Costs could also vary depending on the context of the component. Depending on the type of metadata we can also make use of constraints introduced in different flavours of edit distances, such as enforcing sequential order, limiting the number of matches of one item or supporting gaps.

Evaluation measures originating from information retrieval (e.g., precision, recall) are also used for evaluating content analysis tools. In turn, the edit distance approach can be applied to representations of search results, such as inserting/deleting from a result list, rank changes etc.

Established evaluation measures are typically based on some (weighted) combination of such edits, so that they can be easily computed from a set of edits.

4.3 Data representation

This section discusses the data representations needed for exchanging and processing data for benchmarking.

4.3.1 Ground truth

Ground truth is from its nature not different than extracted metadata. Thus the same representation as for the corresponding type of metadata should be used, for example MPEG-7 AVDP or ELAN Annotation Format (EAF). Metadata formats support the annotation of confidence, which can be used for representing different confidence of ground truth annotations as well. If alternative choices of ground truth items are needed, multiple descriptions in one metadata document or multiple metadata documents (depending on the capabilities of the respective format) can be used.

For the representation of search results, there are not so obvious metadata representations. Metadata standards such as MPEG-7 or MPEG-21 provide means for representing collections of content, which can be used for result sets or ranked lists.

4.3.2 Edits between metadata documents

There is no common representation for edit operations. As many commonly used metadata formats support XML representations, one possibility is to look at languages that represent differences or modifications of XML documents. One early example is DUL [Mouat, 2002], emerging from writing a Diff tool for XML. XUpdate[2] aimed at creating an update language for XML in a database context. The same goal is shared by the update extension of the W3C XQuery language [Robie, 2011].

While basic insert, delete, move, change etc. operations on XML documents are covered, the semantics are not appropriate for our purpose. For example, a shifted time index will result in a change of the corresponding value in the document, rather than “shift”.

We thus need to extend existing languages in order to precisely represent the semantics of the edits. As DUL is represented using XML document rather than the SQL-like syntax of XQuery, it seems more appropriate as a basis for our purpose.

4.3.3 User logs

The main information needed from user logs is the time needed by a user to perform a certain type of action for modifying/adding metadata or interacting with search results. The type of action should correspond (or unambiguously map) to an edit operation for the corresponding type of metadata.

For Web servers, the Common Log Format[3] is an accepted standard, for which a tool chain exists. However, as the format only gives a time point for an operation, and reports a number of other parameters, it does not seem very well suited for our application.

[2] http://xmldb-org.sourceforge.net/xupdate/

[3] http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format

Thus a proprietary format containing lines with the triple

<start-time> <end-time> <operation>

seems more appropriate for our application.

4.4 Determining differences between metadata documents

In order to obtain a representation of edits between metadata documents, we need to extract this information from a pair of XML documents. In the following, we discuss the assessment of XML Diff tools that may provide a basis for extracting this information. As experiments showed that their output is too noisy on complex metadata documents, we describe the tool we have implemented to solve this problem.

4.4.1 Evaluation of XML Diff Tools

A range of differencing tools for XML exist, many with their own proprietary representation. There are some commercial XML editors or diff tools. Oxygen[4] provides a diff&merge option, but the result cannot be serialised. A previously existing XMLDiff API[5] has been discontinued. Altova DiffDog[6] is a quite powerful tool, that comes with its proprietary serialisation. DeltaXML[7] inserts attributes into a document to markup changes. Other commercial tools are Stylus Studio[8] and Liquid XmlDiff[9].

There is also a number of open source tools, many resulting from research on new XML differencing algorithms. XML-Diff[10] is a tool written in Perl that uses the DIFFGRAM/XCVS notation as output. Another Perl tool is XMLDifferenceMarkup[11], which defines a simple XML output format, supporting copy, insert and delete. XyDiff[12] is a tool written in C/C++ that produces a quite compact, but somewhat cryptic output, that is hard to process further.

diffxml[13] is a tool written in Java, that uses Delta Update Language (DUL) as its output representation. The output is quite cluttered with many inserts and deletes, and XPath expressions with only numbered indices. Although DUL supports updates, it seems they are not generated. The output includes also diffs on character level, and includes whitespace modifications. We have thus implemented a XSLT stylesheet to clean up and add named XPaths. As the trees are incrementally built by insertion, only the XPath for the top node can be retrieved. The tree seems to contain many text elements, that are counted differently by other XML/XSLT processors (e.g., Xalan, Saxon).

DiffMk[14] is a Java tool that outputs changes as attribute markup in the XML document.

xmldiff[15] is a Python tool that provides the option to use XUpdate[16] as output. The tool is however limited to documents with 100 nodes.

X-Diff[17] is a tool available in Java and C++ versions. It supports delete/insert/update, using an unordered node model, i.e., only ancestor relations are significant. The tool inserts its output into the input XML document. We have thus implemented a converter of this output to DUL. Because of the unordered nodes assumption, child position numbers of insert/delete operations are not always correct, and the tool fails on more complex metadata files.

[4] http://www.oxygenxml.com/xml_editor/xml_diff_and_merge.html

[5] http://archives.oxygenxml.com/Oxygen/Diff/InstData4.0/Developer/oxygenXMLDiffSDK.zip

[6] http://www.altova.com/diffdog/xml-diff.html

[7] http://www.deltaxml.com/support/documents/deltav2

[8] http://www.stylusstudio.com/xml_diff.html

[9] http://www.liquid-technologies.com/compare-xml.aspx

[10] http://search.cpan.org/dist/XML-Diff/Diff.pm

[11] http://search.cpan.org/dist/XML-DifferenceMarkup

[12] http://leo.saclay.inria.fr/software/XyDiff/cdrom/www/xydiff/index-eng.htm

[13] http://diffxml.sourceforge.net

[14] http://diffmk.sourceforge.net

[15] http://www.logilab.org/859

[16] http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html

[17] http://pages.cs.wisc.edu/~yuanwang/xdiff.html

XmlDiff[18] is a Java tool, with several limitations w.r.t. size and namespace support. It uses a custom format using attributes attached to elements (changed, new, deleted).

Table 1 provides an overview of XML diff representations of several of the tools considered. Three of the representations have been considered as candidates. The Altova format document is not accessible. The XyDiff output has turned out to be quite cryptic and hard to process. DUL has found to be a useful representation, to which the output of different tools could be successfully converted.

Table 1: Considered XML diff representations.

Representation	Output		Support update/change	Support move	Candidate
	Format	Diff doc/ attributes			
DUL	XML	doc	x	x	Y
XUpdate	XML	doc	x		N
XQuery / Update	function syntax	doc	x		N
DIFFGRAM / XCVS	XML	doc	x	x	Y
XyDiff	XML	doc	x	x	Y
X-Diff	XML	doc	x		N
XMLDifferenceMarkup	XML	doc			N
Altova format	XML	doc	x	x	Y
DeltaXML	XML	attr	x		N
DiffMK	XML	attr	x		N

Large-scale tests have been performed on comparing shot boundary outputs from different analysis tools from 1,150 videos of the TOSCA-MP data set. There are several points that cause problems for the XML Diff tools being tested. The shot boundary detection results have sometimes small overall offsets or small offsets for some segments. This will cause insert/delete operations for all segments with nearly all of the tools. For the segments not only start and end time may differ, but identifiers for segments are assigned differently by the different tools, thus causing differences in values. Further annotations on the shot segments may not be identical (e.g., confidence of detection), causing none of the tools to produce satisfactory output. Even segments that would result in minor diffs if they were in the correct order are reported as complete insertions and deletions when other segments are missing or inserted before. In some cases, the reported sampling rates may be different, even if they refer to the same time point (e.g., 25/50 frames/sec, or 500 milliseconds).

Figure 1 shows an example that causes problems for XML diff tools not aware of the structure of the metadata format. The upper and the lower listing show the description for the same three shots in the video, as produced by two shot boundary detection tools. The durations of the shots are identical, however, there is a constant offset in the timeline of the two videos (e.g., one using the start of the file, the other the start of the programme as zero point). Both tools have their own scheme for assigning IDs to the segments they create, and one tool describes key frames, while the other does not. Applying XML diff tools results in a list consisting of a deletion and an insertion for every segment, rendering the result useless for performance evaluation.

The conclusion is that the output of general purpose XML diff tools is too noisy for our purpose. We have thus decided to implement a dedicated tool, aware of the semantics of time-aligned metadata rather than post-processing the output of another XML diff tool.

[18] https://github.com/AlinaIoanaFlorea/XmlDiff


```
<AudioVisualSegment id="ID-SHOT-4e42b94004sk">
  <StructuralUnit href="http://www.ebu.ch/metadata/cs/mpeg/avdp/StructuralUnitCS#10"/>
  <MediaTime>
    <MediaRelIncrTimePoint>428</MediaRelIncrTimePoint>
    <MediaIncrDuration>84</MediaIncrDuration>
  </MediaTime>
</AudioVisualSegment>
<AudioVisualSegment id="ID-SHOT-4e42b94328sk">
  <StructuralUnit href="http://www.ebu.ch/metadata/cs/mpeg/avdp/StructuralUnitCS#10"/>
  <MediaTime>
    <MediaRelIncrTimePoint>512</MediaRelIncrTimePoint>
    <MediaIncrDuration>116</MediaIncrDuration>
  </MediaTime>
</AudioVisualSegment>
<AudioVisualSegment id="ID-SHOT-4e42b94804sk">
  <StructuralUnit href="http://www.ebu.ch/metadata/cs/mpeg/avdp/StructuralUnitCS#10"/>
  <MediaTime>
    <MediaRelIncrTimePoint>628</MediaRelIncrTimePoint>
    <MediaIncrDuration>12</MediaIncrDuration>
  </MediaTime>
</AudioVisualSegment>
```

```
<AudioVisualSegment id="TRID_6_AV">
  <StructuralUnit href="http://www.ebu.ch/metadata/cs/mpeg/avdp/StructuralUnitCS#10"/>
  <MediaTime>
    <MediaRelIncrTimePoint>309</MediaRelIncrTimePoint>
    <MediaIncrDuration>84</MediaIncrDuration>
  </MediaTime>
  <MediaSourceDecomposition criteria="..." id="TRID_6_MSD_0" /> <!-- key frames (omitted) -->
</AudioVisualSegment>
<AudioVisualSegment id="TRID_7_AV">
  <StructuralUnit href="http://www.ebu.ch/metadata/cs/mpeg/avdp/StructuralUnitCS#10"/>
  <MediaTime>
    <MediaRelIncrTimePoint>393</MediaRelIncrTimePoint>
    <MediaIncrDuration>116</MediaIncrDuration>
  </MediaTime>
  <MediaSourceDecomposition criteria="..." id="TRID_7_MSD_0" /> <!-- key frames (omitted) -->
</AudioVisualSegment>
<AudioVisualSegment id="TRID_8_AV">
  <StructuralUnit href="http://www.ebu.ch/metadata/cs/mpeg/avdp/StructuralUnitCS#10"/>
  <MediaTime>
  <MediaRelIncrTimePoint>509</MediaRelIncrTimePoint>
  <MediaIncrDuration>12</MediaIncrDuration>
  </MediaTime>
  <MediaSourceDecomposition criteria="..." id="TRID_8_MSD_0" /> <!-- key frames (omitted) -->
</AudioVisualSegment>
```

Figure 1: MPEG-7 AVDP descriptions of the same three shots from two shot boundary detection tools.

4.4.2 Time-aligned Metadata Diff (TAME-Diff)

We have implemented a tool to determine differences between metadata documents, focusing on time-aligned metadata of audiovisual content. The input to the tool are two metadata documents (one of them being the reference document, i.e., often the ground truth). In case the documents contain multiple metadata time lines, the time line to be compared can be specified (using XPath expressions). The tool then performs longest common subsequence alignment of the segments in the time line, allowing for insertions and deletions. Then a further diff is performed on each of the aligned segments, comparing the values of elements and attributes of the segment. Filters for certain elements/attributes can be specified (e.g., to ignore identifiers).

Adapted LCS Algorithm

Similar to the problem of matching spatial coordinates [Vlachos, 2002] or feature sequences of videos [Bailer, 2009], there is no binary match between two metadata segments, but similarity based on their time and content distance. We define two parameters, which constrain whether segments are considered matching: θ defines the minimum overlap of two segments to be counted as matching, δ defines the maximum time distance between start/end times of segments, in case they are non-overlapping. The algorithm can be configured to apply one of the two constraints only, or link them with AND or OR. Based on these parameters, we can determine matching segments in the initial step of the LCS algorithm. In the backtracking step of the algorithm, we consider not just a single longest

sequence, but consider a set of long sequences (e.g., top *k*). We perform the backtracking step for each of these (possibly partly overlapping) sequences, and determine its similarity score as follows.

The time distance d_t of the segments S_1 and S_2 is defined as

$$d_t(S_1, S_2) = abs\big(start(S_1) - start(S_2)\big) + abs(end(S_1) - end(S_2)).$$

All times are normalised to seconds (as floating point numbers) before distance calculation.

In addition, a content distance can be defined over a set of values in the segment, specified by a set of relative XPath expressions *P*, and a type of value distance d_v (absolute difference, binary equality, string edit distance). The content distance d_c is then defined as

$$d_c(S_1, S_2) = \sum_{p \in P} d_p\big(xpath(S_1, P), xpath(S_2, P)\big),$$

with *xpath*(*S*,*P*) being a function retrieving the value from a segment *S* using the XPath expression *P*, and d_p the choice of d_v for *p*. The segment distance d_S is then a weighted combination of time and content distance, with a user defined parameter α:

$$d_s(S_1, S_2) = d_t + \alpha d_c$$

In the backtracking step of the LCS algorithm, the per-segment distances $d_S(S_x,S_y)$ are summed for all segments S_x, S_y that have been determined as matching, yielding an overall distance D_S for the two documents. We select the matching sequence with minimum D_S.

Determine Segment Differences

We can directly report unmatched segments in the reference document as deletions and unmatched segments in the other input document as insertions, if they do not overlap with any segment in the other document.

In case that they overlap with a segment in the other document, unmatched segments in the reference document constitute a split of segments, unmatched segments in the other input document constitute a merge of segments.

Segments that have been matched, but differ in their start and/or end time, and/or values in the set *P* will be reported as updated. In case that the values in *P* match (i.e., $d_c(S_1,S_2)$=0), segments will be reported as shifted (a special case of update).

Result Representation

As discussed above, we use Delta Update Language (DUL) as for the representation of the matching result. Two elements are added to DUL:

- Split: Specifies that a segment in the reference document has been split. The split descriptors references the segment in the reference document and contains the new segments that have been created.
- Merge: Specifies that segments of the reference document have been merged. The merge descriptors link to the segments in the reference document and contains the segment into which they have been merged.

The DUL output for the example of Figure 1 is shown in Figure 2. It contains the updates of IDs and time points, as well as the insertion of the key frames. For the performance evaluation of shot boundary detection, these ID changes and insertions can of course be ignored.

In addition, the tool produces statistics for matches and each type of edit between the two documents.

```
<update node="AudioVisualSegment[1]/@id">TRID_6_AV</update>
<update node="AudioVisualSegment[1]/MediaTime[1]/MediaRelIncrTimePoint[1]">309</update>
<insert parent="AudioVisualSegment[1]" childno="3" nodetype="1"
        name="MediaSourceDecomposition" />
<update node="AudioVisualSegment[2]/@id">TRID_7_AV</update>
<update node="AudioVisualSegment[2]/MediaTime[1]/MediaRelIncrTimePoint[1]">393</update>
<insert parent="AudioVisualSegment[2]" childno="3" nodetype="1"
        name="MediaSourceDecomposition" />
<update node="AudioVisualSegment[3]/@id">TRID_8_AV</update>
<update node="AudioVisualSegment[3]/MediaTime[1]/MediaRelIncrTimePoint[1]">509</update>
<insert parent="AudioVisualSegment[3]" childno="3" nodetype="1"
        name="MediaSourceDecomposition" />
```

Figure 2: DUL output for the example description fragments shown in Figure 1.

Implementation

The tool has been implemented to handle metadata documents conforming to the MPEG-7 Detailed Audiovisual Profile (AVDP). AVDP groups results of automatic metadata extraction tools into separate temporal decompositions of the root audiovisual segment. TAME-Diff can thus work on one of these decompositions from two documents. As AVDP also supports the representation of multiple instances of analysis results from different tools, TAME-Diff can also be applied to determine differences between two temporal decompositions in the same document.

The implementation of TAME-Diff is general enough to be easily adapted to other metadata formats supporting the representation of time aligned metadata.

4.5 Differences for different types of metadata

The following table summarises the types of differences considered for analysis and search tool developed in TOSCA-MP.

Table 2: Types of metadata edits.

Analysis/search tool	granularity	Type of information	Edits
Action, event, concept detection	segment	class	insert, delete, substitute
		location	symmetric difference of region
Logo Detection	segment	class	insert, delete, substitute
		location	symmetric difference of region
Sport Camera View Classification	segment	type	substitute
Specific content types	segment	type/class	insert, delete, substitute
Genre classification	global	genre	substitute
	segment	genre	insert, delete, substitute
Quality Analysis	segment	defect occurrence	insert, delete
		classification/value	substitute
Highlight Detection	segment	type	insert, delete, substitute
		rating/score	substitute
Near duplicate detection	segment	set of links	insert, delete
		similarity score	substitute
Video segment clustering	segment		
Shot clustering service	segment	cluster ID	inserted to cluster, deleted from cluster, shifted, inserted, deleted
Temporal segmentation	time stamp/range	type	insert, delete, shift
Player Detection	s/t region		insert, delete, symmetric difference of region
Face detection	s/t region		insert, delete, symmetric difference of region
Player Identification	s/t region	identifier	insert, delete, substitute
Automatic Speech	segment	Transcript + time points	insert, delete, substitute

Analysis/search tool	granularity	Type of information	Edits
Recognition			
Machine Translation	segment	translated text(s)	insert, delete, substitute
Named entity recognition	segment	entity	insert, delete, substitute
Spoken Language Identification	segment	language ID	substitute
Audio Segmentation And Speaker Clustering	segment	speaker ID	insert, delete, substitute
Search Service	global	ranked result set, global annotations	insert, delete, rerank
	segment	ranked result set	insert, delete, rerank

5 Assessing components in task context

5.1 Task-based benchmarking

The advantage of benchmarking components as part of a system rather than stand-alone is that the performance for solving the task of the overall system is measured. This means that only those aspects of the individual components' performance contribute to the results that are actually relevant in the system context, while individual evaluation cannot make this distinction. In many cases, the overall system task is on higher level in terms of both semantics and granularity, making it less effort to generate ground truth for the entire system result than for those of individual components. The drawback is of course, that it is difficult to draw conclusions about the impact of a single component on the system performance and assess the quality of results of an individual component.

One approach to address this issue is comparative evaluation of components, i.e. benchmarking the system with two different implementations of a component. The SOA based architecture of TOSCA-MP facilitates this greatly and enables automation, as the process model derived from the task model enables resolving any dependencies resulting from the exchange of a component (e.g. further feature extraction services needed by one of the implementations). The task model provides information of the dependencies of subtasks, and thus about which components can be evaluated independently. This helps reducing the number of combinations of components that need to be evaluated. Based on the task models, subtask structures that are reused in different system configurations can be identified, so that comparative evaluation can also be done on that coarser level of granularity. The same approach can of course also be applied to evaluating parameterisations of components, which is specifically relevant for TOSCA-MP in order to automatically optimize processing chains to certain types of material.

Figure 3 shows a schematic diagram of a system implementing the automatic steps of a task model. All components depend on a set of parameters θ_i. Some components (A1) depend only on the input audiovisual content (typically performing low-level feature extraction), others depend on both the content and outputs of other components (A2), or only on outputs of components earlier in the task (A3, A4).

In order to benchmark an individual component (or parameterization) in the task context, one would have to run the system for the entire task for every choice of component and parameters, and using different types of content. Doing this on a large scale for a complex system is too time consuming. Thus, we aim at breaking the problem into assessing the impact of different contents, inputs and parameters to the output and measure these effects per component.

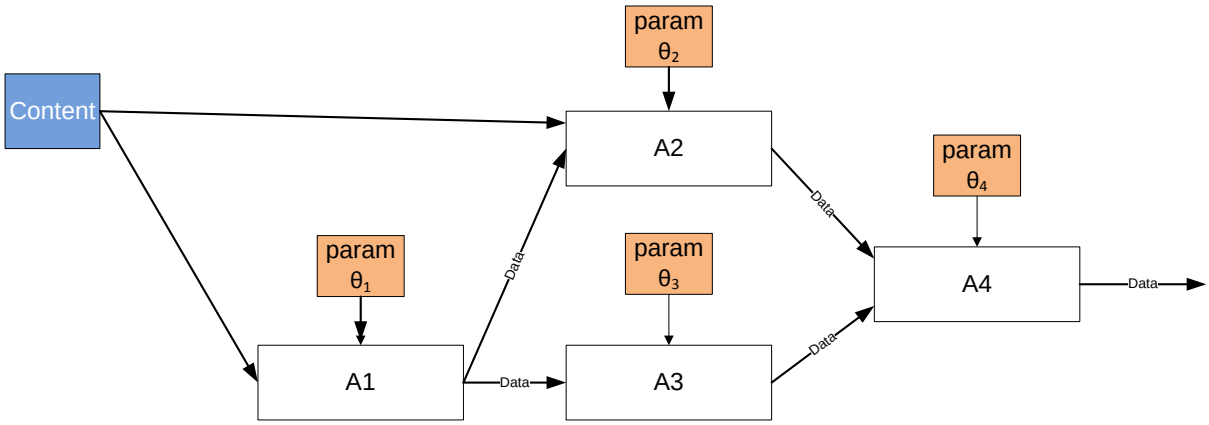

Figure 3: Schematic visualisation of a system for executing a content analysis task.

5.2 Modelling error propagation

In order to benchmark a component in the task context, we aim at modelling the impact of content properties, parameters and errors in the results from previous components on errors in the component's output.

The task structure of a system can be considered as a directed graph (an acyclic one, if feedback is not included). This does not only model the flow of information, but can also be used to build a model of the

propagation of different types of errors to subsequent components (if there is sufficient ground truth to assess the performance a component individually). Clearly, the influence will differ for each error types, and depend on the type of content. Once the model parameters have been estimated for a certain type of content, the model can be used to assess the impact on the system performance based on individual evaluation results of a component.

In order to derive such a model of error propagation from a task model, we use the following approach:

- Each data flow between components is replaced by a set of error nodes. This set consists of the types of ground truth edits that are of relevance for the type of metadata exchanged between these components.
- We introduce a node for the parameters of the component, and connect it to its output error nodes.
- If the component accesses the content, we introduce a node for the content properties, and connect it to the component's output error nodes.
- We connect every input error node of the component (note that they may stem from different input error nodes) to every output error node of the component.
- An output error node represents the probability of this type of ground truth edit, given the content properties, the parameters of the component and the probability distributions of input errors.

As a representation for this model we use a Bayesian network. A part of the network resulting from the system above is shown in Figure 4. We model the error probabilities as conditional probabilities, conditioned on the set of error probabilities in the input (for simplicity, we omit parameters and content properties here). For example, output error probabilities of A2 are now given as $p(e^{'}_{A2}|e^{1}_{A1},e^{2}_{A1})p(e^{1}_{A1})p(e^{2}_{A1})$. We make the assumption, that for the components under test the types of input errors are independent. If we test a single component by a simulation that introduces different samples of errors this is in fact the case. In the task context, errors may be originally caused by the same content property or previous error, but this dependency is actually modelled in links between previous nodes in the network.

The reasons for choosing a Bayesian network as a representation are the following. The Bayesian network can be easily derived from the task model, and model error probabilities with various distributions and of various origins. The notion of output errors conditioned on input errors and priors (e.g. content properties) is quite intuitive. For example, one could as well include the error rates of human annotators. Bayesian networks are flexible in terms of structure and types of nodes, which enables not only modelling of arbitrary tasks, but also the inclusion of hidden layers in order to model more complex relations between input and output errors in components. Typically, the exact distributions of the error types are unknown, and can only be estimated from sampled data. Also, inference algorithms for Bayesian networks are able to deal with incomplete data, which is a requirement in practical applications.

Bayesian networks have also been proposed in [Bai, 2005] to model software failures. Like in our problem, the exact distributions of the error's probabilities are unknown. An alternative representation is a neural network. However, as we do not know the function of each node analytically, estimating weights using the traditional back-propagation can only be done with estimating the gradient from samples (cf. [Oohori, 2007]). Compared to Bayesian networks, it is more difficult to model arbitrary structure, as well as errors that do not only have a probability, but an associated value (e.g. the amount of shifting a boundary).

The model benefits from a modular representation of errors that are ideally as independent as possible. Thus, we argue for a representation of errors based on individual edits to the ground truth as proposed in Section 4.2, rather than using common measures such as precision or recall, that include various types of errors.

In practical cases, we might experience cases where certain deviations from the input ground truth (which are errors) cause improvements of the output. We treat those as errors as well, as they are deviations from a target result.

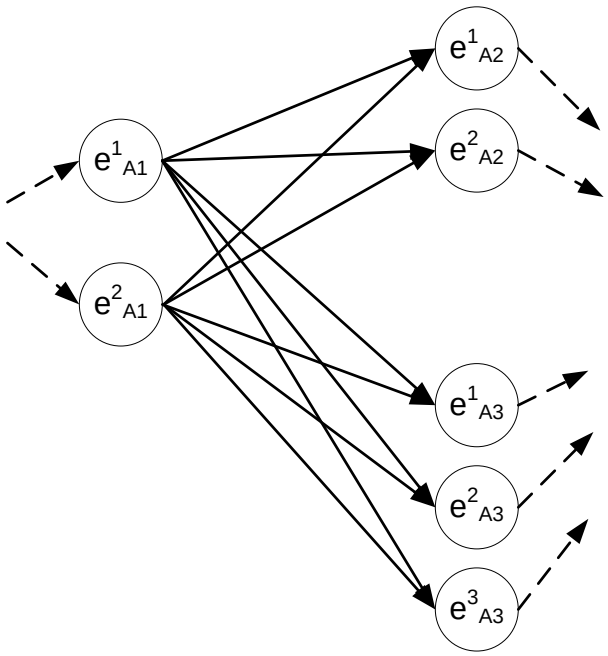

Figure 4: Bayesian network model corresponding to a part of Figure 3.

A model for a complete network can be assembled from the building blocks for individual components, making use of the model parameters estimated in smaller experiments individually for each of the components.

We represent all nodes as univariate Gaussians. We use the EM algorithm for learning the network parameters, as implemented in the Bayes Net Toolbox [Murphy, 2001].

5.3 Component parameters

So far, we have only considered the influence of input errors on the output. We can model parameters of one component as a node or a set of nodes. Every output error variable is in addition to the input errors also conditioned on the parameters. The parameters are tied conditions for the output errors, i.e. all output error nodes depend on the same set of parameters.

As described above for the input errors, we can sample different sets of parameters and assess their impact on the output errors. Different implementations of a component can be thought of as entirely different parameter sets (with the practical drawback that they cannot be so easily varied). This a typical procedure for parameter optimisation and evaluations such as ROC curves. However, the use of the results is different, as in the end we may not choose the optimal parameters concerning the output errors for the component itself, but the errors of a later component in the process.

5.4 Content dependency

The properties of the actual content to be processed have direct impact on components that access the content, and not only results of previous components. In this context, low-level features are treated like the content itself, as the extracted feature can be assumed to be “correct”, if the extractor has been tested. We do not have an intermediate result that abstracts the dependency of the content properties in this case, thus any component using low-level features is considered depending on content properties.

For these components we introduce nodes that represent content-specific information. The probabilities output error nodes of a component are conditioned on these nodes. We can consider two options for modelling these dependencies.

Error priors. We can measure output error probabilities for a specific content item (or better, a set of content items sharing similar characteristics), using only ground truth for the other inputs (if any). The output errors are then conditioned on the respective type of input error, measured on the same content (class) using ground truth inputs. The drawback is that we need specific priors for each output error type and that the number of measurements we can obtain is typically small.

Feature statistics. We can model a set of feature properties as nodes in the Bayesian network, and estimate their parameters based on input data. These measurements could be made on smaller temporal granularity in order to obtain more and more precise measurements. If we use sufficient data, we can expect to obtain a model with good generalisation. While these nodes are tied parameters for all output error nodes of all components, the influence on the specific output errors needs to be trained separately. [Valdes, 2012] propose a system to predict different evaluation scores of video summaries (inclusion of ground truth elements, pleasantness of tempo, low level of redundancy) based on low-level features extracted from the video. They manually pre-select appropriate features for each of the criteria first, and then train a classifier on this data. They tested neural networks and SVMs, and finally used regression trees[19].

We represent content properties in a similar way as errors of earlier components in the workflow, and model output errors conditioned on values/value ranges of the content properties.

5.5 Example

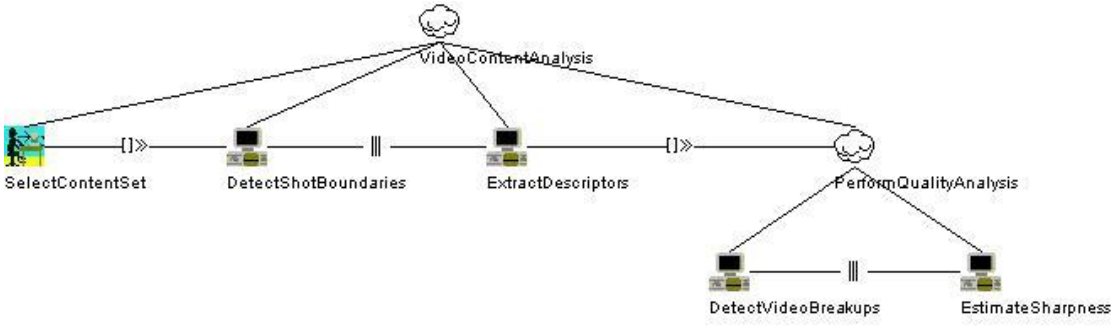

Figure 5: Example excerpt of a video analysis task model.

Figure 5 shows an example excerpt of a task model of a video analysis process. Tasks are either automatic or interaction tasks. The tree structure defines decomposition of tasks into subtasks, while the links on the same level define temporal dependences and information flow.

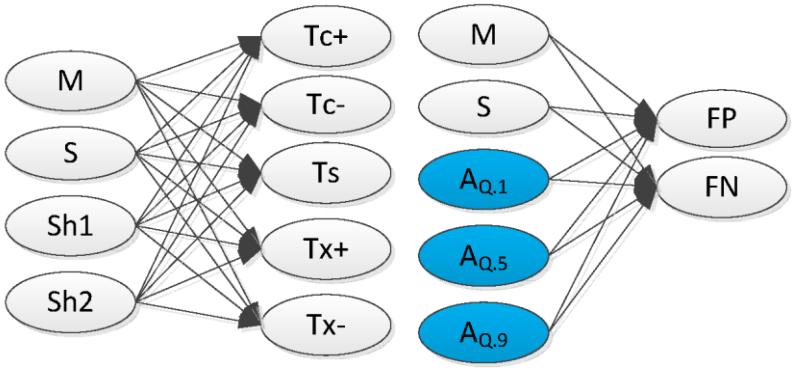

Figure 6: Left: Bayesian network modeling dependencies between shot boundary detection and clustering errors. Right: Bayesian network modeling dependencies between shot boundary detection errors, visual activity and video breakup detection errors.

The Bayesian networks for modelling the errors of two components from the task model in Figure 5 are shown in Figure 6: near-duplicate shot clustering and video breakup detection. Near-duplicate shot clustering groups shots that show visually similar action, such as repeated takes of the same scene. We model the dependency on the errors of the input shot boundary detection w.r.t. a ground truth (see Figure 6 left): merge of two segments (M), split of two segments (S), small shift (<=5 frames) of segment boundaries (Sh1), and large shift (>5 frames) of segment boundaries (Sh2). All input values are expressed in terms of fractions, normalized by the number of segments. For the output of the

[19] Note: regression tree has only discrete set of output values

clustering algorithm, we consider the following edit operations: fraction of takes added to assigned clusters (Tc+), fraction of takes missed from assigned clusters (Tc-), fraction of shifts of boundaries of all assigned takes (Ts), fraction of added unassigned takes (Tx+) and fraction of missed unassigned takes (Tx-).

Video breakup detection aims at identifying major image disruptions, for example caused by head clogging, assemble edits, lost lock, recorded serious digital error corrections, severe TBC hits and damaged tapes. Both shot boundaries and some erratic motions (e.g., water surfaces, flames) may cause false positives of video breakup detection. We thus model the dependencies on both shot boundary detection and the visual activity of the video (see Figure 6 right): the merge of two segments (M), the split of two segments (S) and the 0.1, 0.5 and 0.9 quantiles of the visual activity (nodes $A_{Q.1}$, $A_{Q.5}$, $A_{Q.9}$). The outputs of the video breakup detector are modelled by false positive (FP) and false negative (FN) detections.

6 Benchmarking Search and Result Presentation

We consider a similar approach as for chains of analysis models. A search component is considered a component in the processing flow, which depends on previous analysis modules, each producing metadata. Like analysis component, the search component depends on the output of previous components and its parameters. Instead of depending on the content, the search component depends on the provided query. The output is a specific type of metadata, i.e. a result set. A simple schematic example is shown in Figure 7.

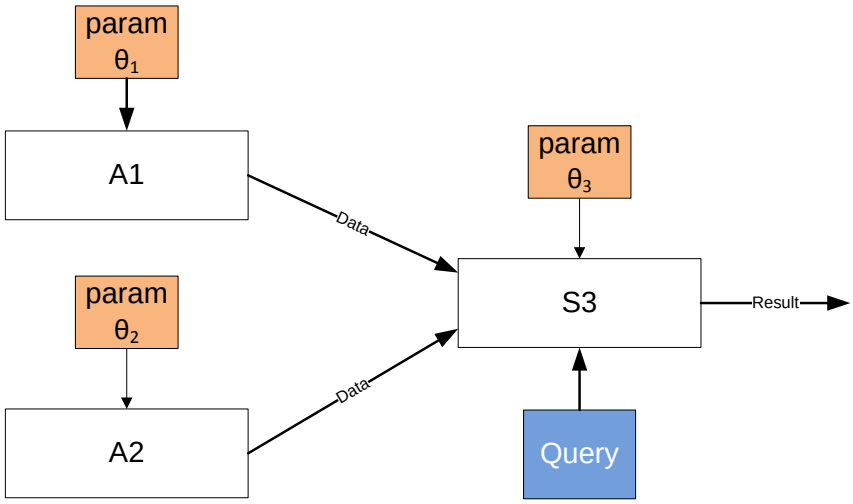

Figure 7: Schematic visualisation of a system performing search based on analysis results.

We can thus apply the same approach for turning the graph into a Bayesian network of edits to metadata, as described in Section 5.2. At the input side, these are the edits to the respective metadata items from the analysis. At the output sides, these are edits to the result list, which may include ranking changes, modification of relevance scores or changes to clustering (in case that results are grouped).

6.1 Search types and properties

Using information from the task descriptions collected in D4.1, we classify different types of search tasks, and discuss their properties and constraints. These properties define how specific aspects of the search result are weighted in the assessment process.

Table 3: Properties of different types of search/browsing tasks.

	Exhaustive search	**Most/few relevant items**	**Known item(s)**	**Stock shot**	**Browsing, exploration**
Use cases	BG1.1 BG1.2	BG2 BG8	BG1.2 BG6	BG7	BG1.1 BG1.2
Tasks (from D4.1)	Search for coverage on news topics Search for multilingual news material	Material for specific news item, Highlight summary	Material for mobile/web news story, near duplicate search		Gathering material for general purpose programmes, documentaries Search for multilingual news material
Exhaustiveness	Yes	Completeness of highlights	No	Yes	Not mandatory

	Exhaustive search	Most/few relevant items	Known item(s)	Stock shot	Browsing, exploration
Diversity	Yes, different viewpoints, different languages	No		Yes, different viewpoints	Yes, different viewpoints, different languages
Inclusion of known items			Yes, from set of broadcast items		
Ranking	Yes, Results ranked w.r.t. the query	Yes, Results ranked w.r.t. the query			Yes, Results ranked w.r.t. the query
Importance of persons/events/ visual properties	Persons, events, places	Persons, events, places	Persons, events, places	Persons, events, places	Specific requests on topics, persons, locations, buildings
Result visualisations	Yes, Advanced features (how elements of the domain are related)			List of stock shot. Additional information (e.g. licence)	Yes, Advanced features (how elements of the domain are related)
Quality				Important	Important
Frequency	>5x / month	1-9 x / day			1-9 x / day
Available time	4-24 hrs	< 0.25 hrs			1-40 hrs

6.2 Evaluating search

[Thomas, 2006] surveyed different methods for comparing result sets, including the Cranfield paradigm (cf. TREC), the analysis of search logs (mentioning the drawbacks trust bias, quality bias and the lack of sufficient user interactions during judging results) and users tests (with the drawback of lab or natural settings). They argue that it is possible to predict real-world performance from lab experiments, and propose side by side presentation of two results, where users are explicitly asked questions. [Sakai, 2007] surveyed different measures and found (average) normalised discounted cumulative gain ((A)nDCG) to be best rank-based measure, and Q-measure to be the best recall based. A drawback of the cumulative gain type measures is the need for graded relevance judgements.

In addition to the evaluation of an initial result set returned, we consider browsing and exploratory search. This can be done by evaluating individual steps of an interaction flow, e.g., by considering the initial result and proposed links based on a result item as separate steps (cf. the MediaEval Search & Hyperlinking task [Eskevich, 2013]). Alternative options are evaluation based on a task goal, e.g. the set of content items collected after interactive search process, or by questionnaires (cf. [Bailer, 2009]). The disadvantage of the second option is that it cannot be automated, while the first one can be represented in our framework by determining differences to one more user defined ground truth result sets. [Wilson, 2009] proposed a framework for evaluation of exploratory search considering stratified (components of the system) vs. episodic (interaction flow) models, information seeking models (cf. [Belkin, 1995], [Belkin, 2000]) and strategic models. The method measures for each feature and tactic the number of moves required. However, it may not always clear which interaction corresponds to the application of which tactic, and the costs for interactions may differ. Thus it is hard to automate this approach.

For some of the cases above diversity is an important issue. [Radlinski, 2009] observes that the need for diversity may have two reasons: It may in fact be part of the information need (e.g., a query for “Olympics games” should not return only results from one year), or it may be the result of the user’s uncertainty about the query. They propose to deal with redundant results by assigning the cost of the user to skip them.

[Clarke, 2008] propose a model of novelty and diversity based on “information nuggets” and integrate it into a cumulative gain measure. They determine an “ideal gain”, i.e. the optimal cumulative gain over all ranks, and evaluate this method on a question answering task. [Zhai, 2003] consider the contextual dependence between items in the result set, focusing on grouping into subtopics. The evaluation is then done using a combination of subtopic novelty and relevance. When items are grouped by subtopics, this representation can also be treated as a clustering problem and evaluated accordingly. Depending on the specific problem, it may be sufficient to include any representative item from a cluster in the initial result, and leave it to the user to further explore specific topics.

Another discriminating aspect of the types of searches listed above is the number of relevant documents expected. In some cases a few (or even single) relevant document(s) may be sufficient, while in other cases all relevant documents must be returned. [Chen, 2006] proposed an evaluation model for cases where a single or few relevant documents may answer the query. If a topic has subtopic/facets, then one/few for each of them should be returned.

[DeVries, 2004] discusses dealing with varying granularity of results, rather than countable items with binary relevance. They propose a measure called tolerance to irrelevance, defined as the time a user would watch an irrelevant video segment before moving away. On the proposal of the TOSCA-MP team, this measure was also included in the evaluation of the MediaEval S&H task [Aly, 2013].

The methods proposed by [Radlinski, 2009] and [DeVries, 2004] are interesting in the context of TOSCA-MP, as they include the notion of users’ costs for dealing with imperfect results. This is well in line with the work on cost simulation described in Section 7.6 and enables applying this approach to assessing search results.

7 Benchmarking experiments

In this section we describe a number of experiments that we have performed using Bayesian networks as proposed in the previous section. These experiments cover both content analysis, search and result presentation. In particular, we assess task-based evaluation of temporal segmentation, clustering of video segments, quality analysis and ASR and NER, as well as evaluation of search, linking of related search results and content visualisation.

7.1 Clustering similar video segments

We model a network that represents the inputs and outputs of a clustering component for video segments, and we are interested in the following aspects:

- predicting output error probabilities based on knowledge about input error probabilities
- including a simple prior of content properties
- dealing with incomplete data
- level of detail of deviations from ground truth
- lack of ground truth for intermediate steps

7.1.1 Experiment setup

The component under test is the algorithm for clustering similar takes to scene clusters described in [Bailer, 2009]. It has been chosen due to the availability of ground truth from the TRECVID BBC Rushes 2008 Task [Over, 2008]. We use six videos from this set.

The component has the following inputs:

- The content itself (via color, texture and motion features)
- A temporal segmentation of the video (shots)

The component has the following outputs:

- A set of clusters, with assigned take segments
- Start and end times for the take segments (which may be subshots of the input shots in case of partial matches)

For the input segmentation, we consider the following edit operations (corresponding to possible error types) of the input segment w.r.t. a ground truth:

- Merge of two segments (M)
- Split of two segments (S)
- Small shift of segment boundaries (Sh1), (at most 5 frames)
- Large shift of segment boundaries (Sh2), (more than 5 frames)

All input values are expressed in terms of fractions, using the number of segments as denominator.

We consider the following edit operations of the output w.r.t. a ground truth (respect to the processing result obtained for a ground truth input, see experiment descriptions below for details)

- Fraction of takes added to assigned clusters (Tc+)
- Fraction of takes missed from assigned clusters (Tc-)
- Fraction of shifts of boundaries of all assigned takes (Ts)
- Fraction of added unassigned takes (Tx+)
- Fraction of missed unassigned takes (Tx-)

The resulting Bayesian network is shown in Figure 6 (left).

7.1.2 Data

Starting from a ground truth temporal segmentation for each of the videos, we generate a set of 100 segmentations for each video, where we introduce random errors.

The parameters of the errors are as follows (normally distributed):

- Fraction of merged segments: *N*(0.1;0.05)
- Fraction of split segments: *N*(0.1;0.05), the relative split position is *N*(0.5;0.2), i.e., around the center of the segment
- Fraction of shifted segments: *N*(0.2;0.05), the shift in number of frames is *N*(7;3)

We run the clustering component for each of these input segmentations. Figure 8 shows the output errors w.r.t. the results obtained from using the ground truth as input. This excludes the errors occurring *only* based on the content properties. The results show that errors of the types *Tc+*, *Tc-* and *Ts* (i.e., those concerning changes in the result clusters) only appear in a small fraction of the result videos, and if they occur, only a small fraction of error occurs.

Figure 9 shows the output errors w.r.t. the actual output ground truth, i.e., including errors independent of input errors. The situation is now different, especially concerning *Ts*, *Tx+* and *Tx-* which have modal values larger than 0 and higher variance as before.

As a baseline, we have generated samples from a univariate Gaussian with mean and standard deviation of the measured output errors. The results are shown in Figure 10 through Figure 13. Figure 10 and Figure 11 show the distribution of the different types of errors, determined against a result using the segmentation ground truth and against the actual ground truth, respectively. Figure 12 and Figure 13 show the errors obtained from a random (per video and across all videos).For the random experiment and the other experiments we report the absolute error of the predicted fraction or errors vs. the measured fraction of errors. In the plots, the top of the blue box represents the 75 percentile, i.e. 88.5% of the errors can be expected to be below this value.

In each of the experiments, we report per video and across videos results. Per video uses 80% of the samples of a video for training, and 20% of the same video for prediction. Across videos is leave-one-out cross-validation, i.e. uses all samples of all but one video for training, and the samples of the remaining video for prediction.

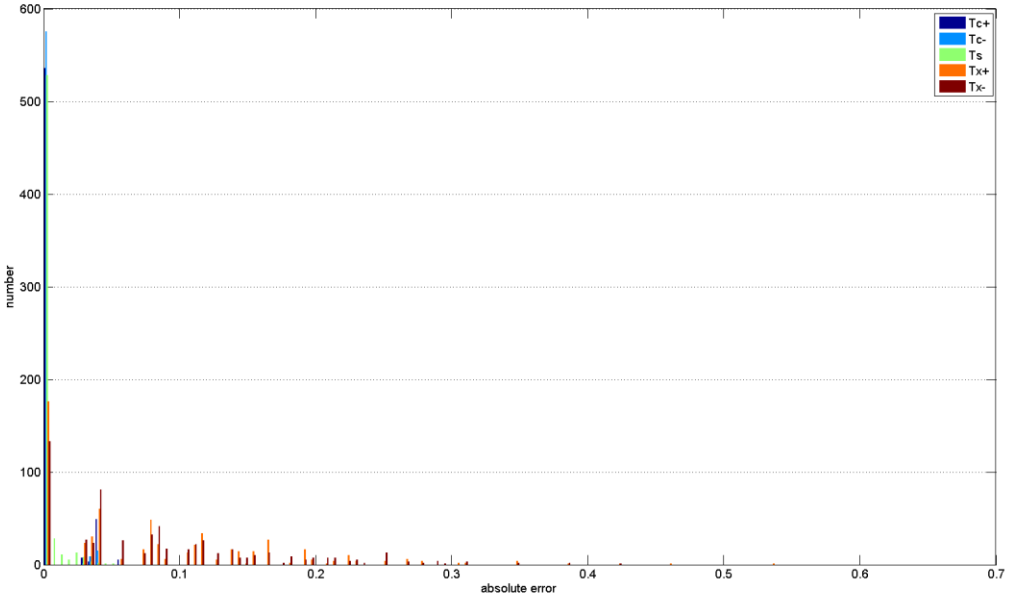

Figure 8: Distribution of errors (against result obtained from using the ground truth as input).

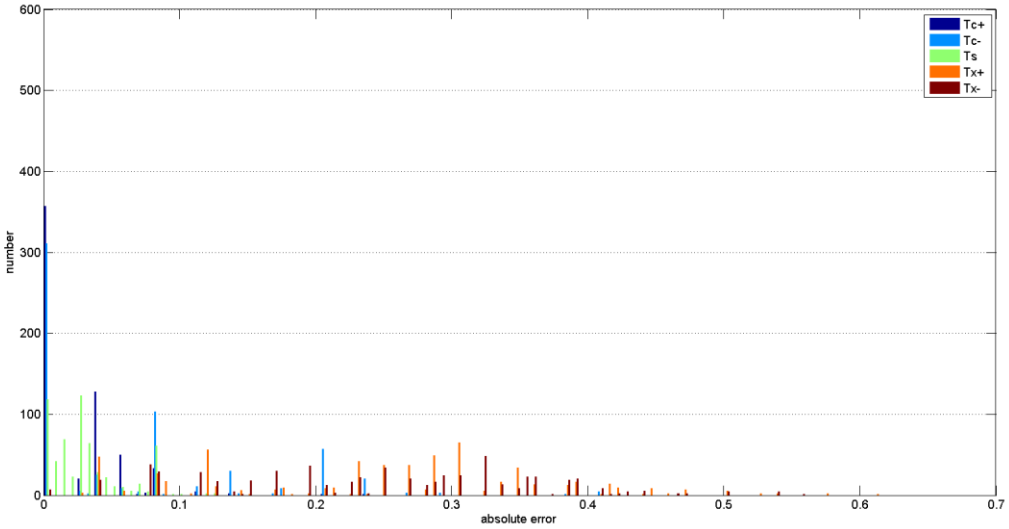

Figure 9: Distribution of errors (against actual ground truth).

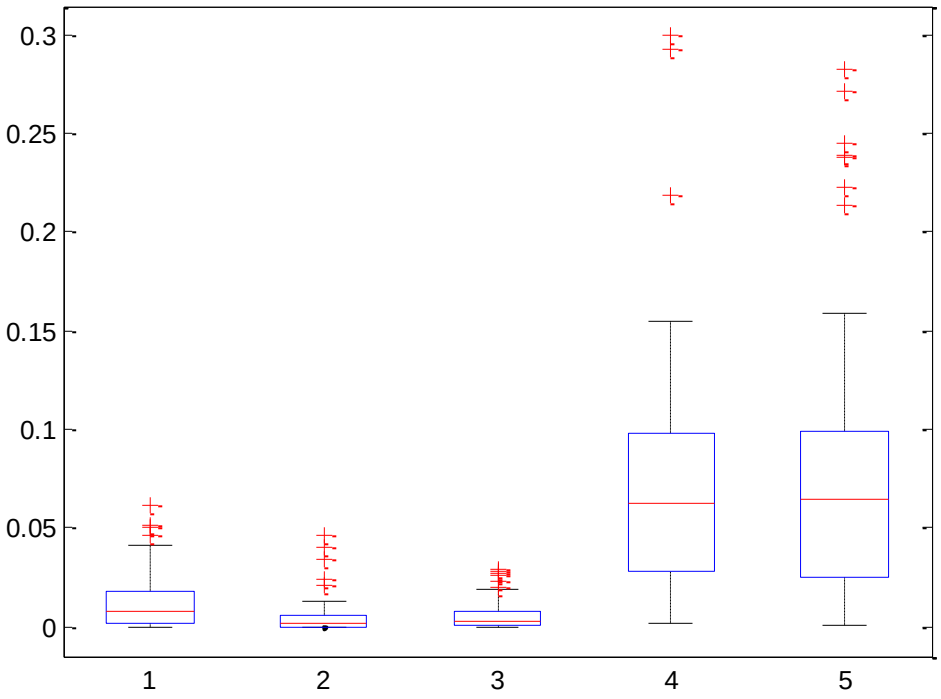

Figure 10: Random prediction (against result from ground truth input), absolute fraction of output errors (per video) for the different types of errors (1=Tc+, 2=Tc-, 3=Ts, 4=Tx+, 5=Tx-).

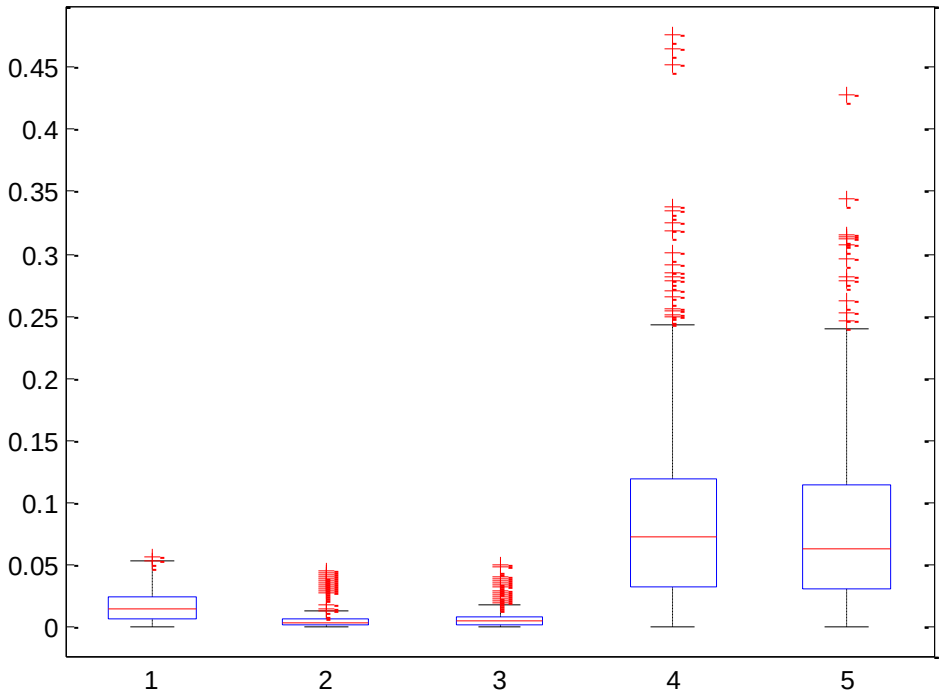

Figure 11: Random prediction (against result from ground truth input), absolute error of fraction of output errors (across videos) for the different types of errors (1=Tc+, 2=Tc-, 3=Ts,4=Tx+,5=Tx-).

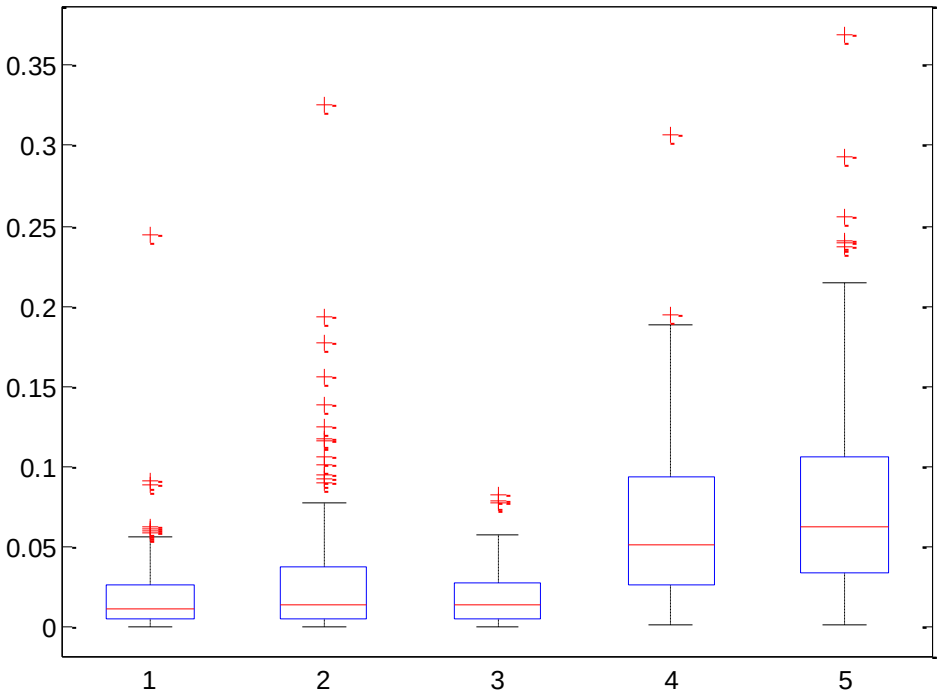

Figure 12: Random prediction (against ground truth), absolute fraction of output errors (per video) for the different types of errors (1=Tc+, 2=Tc-, 3=Ts, 4=Tx+, 5=Tx-).

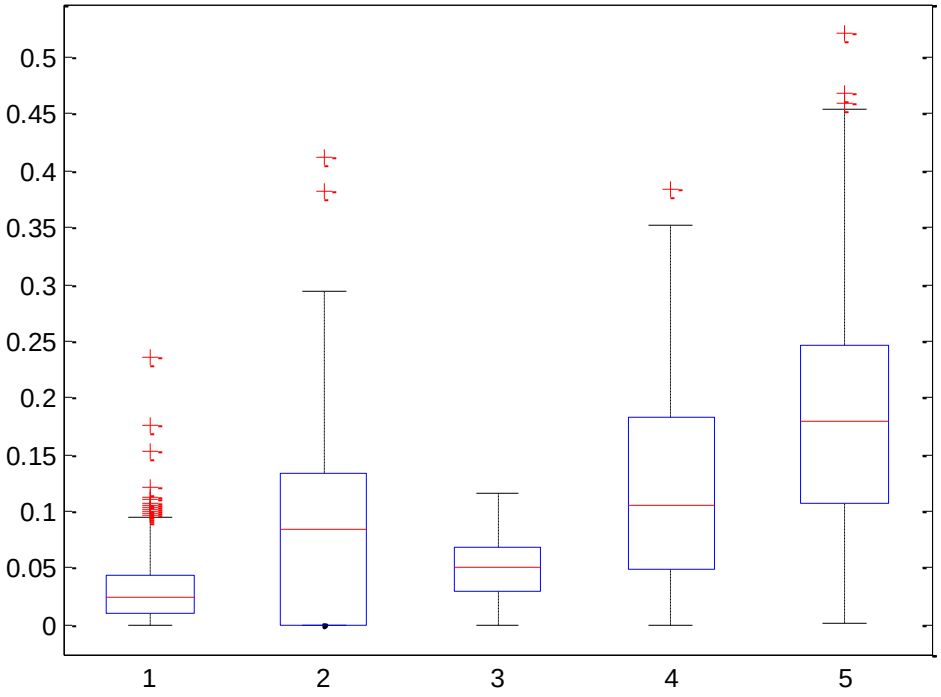

Figure 13: Random prediction (against ground truth), absolute fraction of output errors (across videos) for the different types of errors (1=Tc+, 2=Tc-, 3=Ts, 4=Tx+, 5=Tx-).

We investigate the correlation between the performance of shot boundary detection (measured by precision, recall and F-measure) and the F-measure of the subsequent clustering step. Figure 14 shows the relation between the F-measure of the temporal segmentations and the F-measure of the clustering results obtained using these segmentations. As is apparent from this figure there is actually no correlation when considering F-measure values above 0.5. The correlation coefficient is 0.0036 and the rank correlation coefficient is 0.0028. One reason for this poor correlation is that the metric used for the temporal segmentation does not take all types of errors into account. Shifted shot boundaries create both a true and a false positive, thus reducing both precision and recall. In addition, the different types of errors impact the subsequent clustering differently, which is not taken into account by a generic metric. We can conclude that the traditional evaluation of temporal segmentation does not help us in assessing whether a specific segmentation algorithm (or a specific parameterization) is better for the subsequent analysis step than another one.

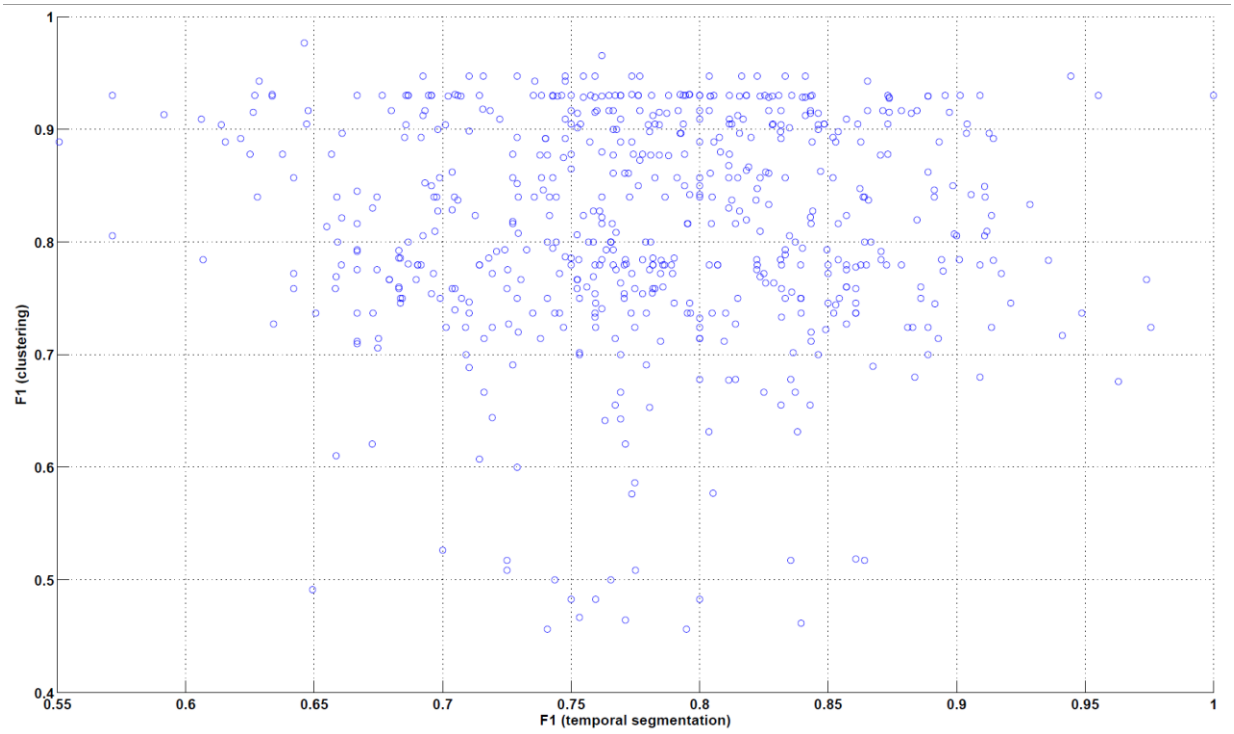

Figure 14: Scatter plot of the F-measure obtained from clustering results against the F-measure of the temporal segmentation.

7.1.3 Experiment 1: basic prediction of output errors

In this experiment we predict only the errors w.r.t. a result generated from the ground truth as input. The content properties of the video are not taken into account. Figure 15 and Figure 16 show the plots for the errors per video and across the videos.

In general, the prediction works quite well. Across the videos, most of the results are only slightly worse than those per video, but with a larger number of outliers.

Table 4: Difference of root median square error between prediction and random prediction.

	Tc+	Tc-	Ts	Tx+	Tx-
per video	0.0032	0.0017	0.0017	0.0069	0.0218
across videos	0.0089	0.0018	0.0018	0.0112	0.0188

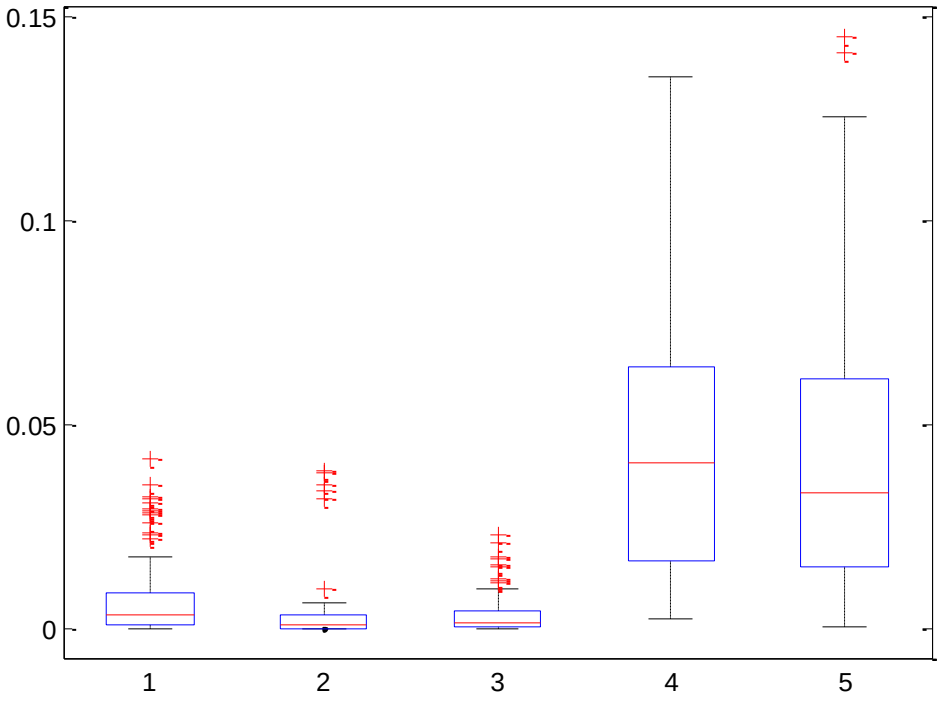

Figure 15: Prediction (against result from ground truth input), absolute fraction of output errors (per video) for the different types of errors (1=Tc+, 2=Tc-, 3=Ts, 4=Tx+, 5=Tx-).

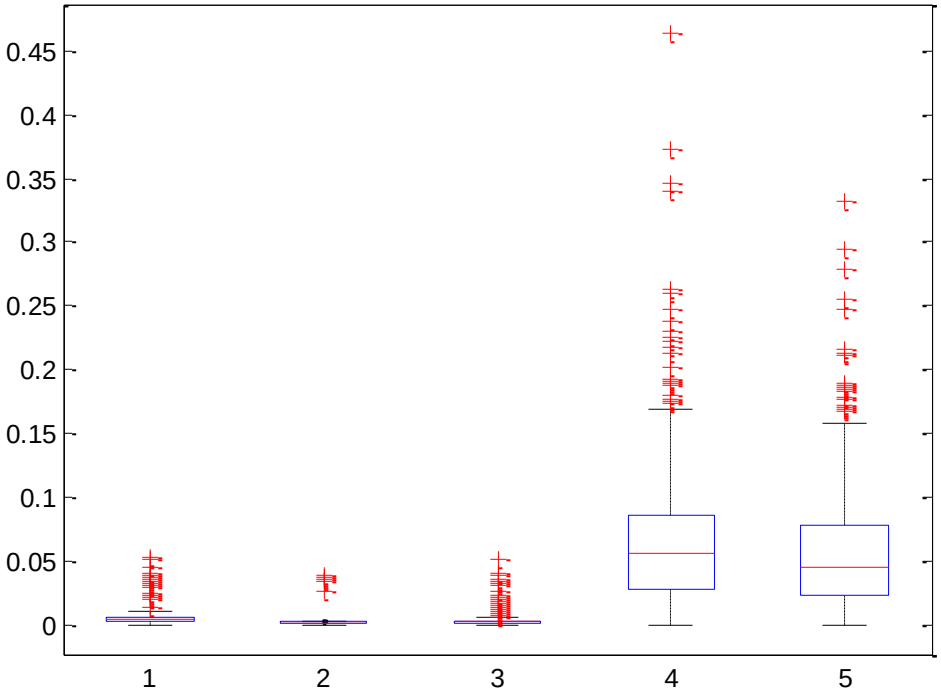

Figure 16: Prediction (against result from ground truth input), absolute fraction of output errors (across videos) for the different types of errors (1=Tc+, 2=Tc-, 3=Ts, 4=Tx+, 5=Tx-).

Predicting F1

Figure 17 shows the resulting correlation between the estimated and actual F-measures on the test data. We obtain a correlation coefficient of 0.50 and a rank correlation coefficient of 0.48. These results are a significant improvement over using precision/recall or F-measure, where we could not measure any clear correlation.

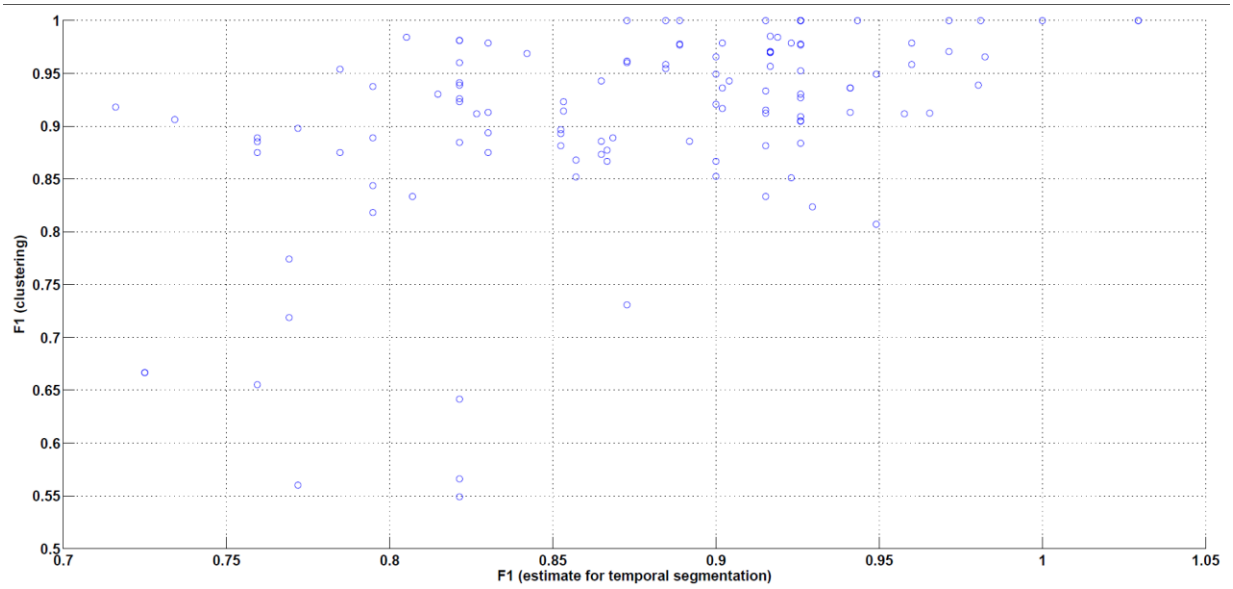

Figure 17: Scatter plot of the F-measure obtained from clustering results against the estimated F-measure from the temporal segmentation.

7.1.4 Experiment 2: simple content prior

The model does not explicitly include the dependency of the performance of the content properties of a specific video, but only adds prior probabilities for the output errors for each video. Figure 18 shows the modified Bayesian network, where the nodes with subscript 0 represent the respective error when using ground truth segmentation as input.

The input is the same data as in experiment 1, but we now compare the output against the actual ground truth.

The results are slightly worse than in experiment 1, probably due to added complexity of the model (adding five more nodes, but only providing little data, i.e. one constant value per video and node). However, the improvements over the random output shown in Table 5 are better for many of the error types than in experiment 1. In additions, this approach allows comparison against actual ground, rather than deviation from a result generated with input ground.

For individual videos no improvement could be expected, as just a constant for each type of error is used. For the set, the model is only trained from 5 sets of prior error probabilities, which does not seem to be sufficient.

Table 5: Difference of root median square error between prediction and random prediction.

	Tc+	Tc-	Ts	Tx+	Tx-
per video	0.0060	0.0053	0.0077	0.0147	0.0334
across videos	0.0002	0.0013	0.0314	0.0079	0.0737

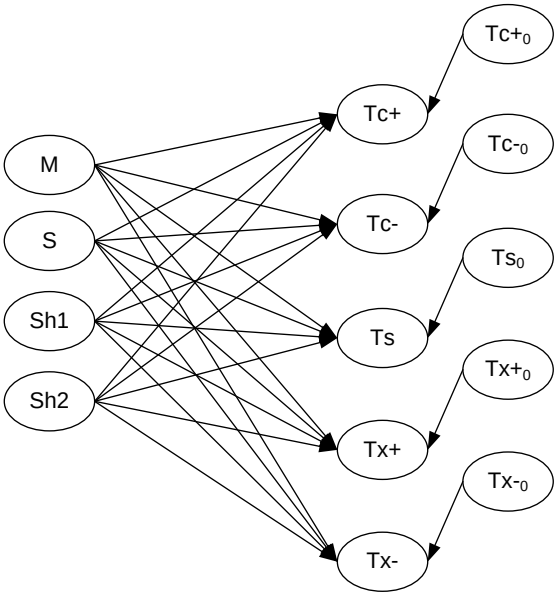

Figure 18: Modified Bayesian network including the prior output errors for a video.

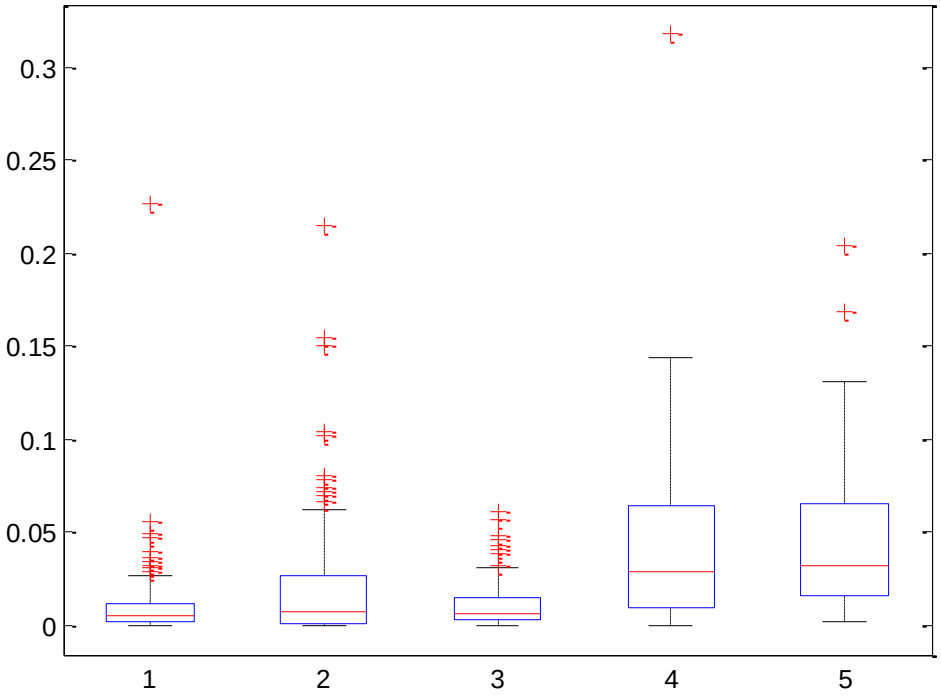

Figure 19: Prediction (against ground truth), absolute fraction of output errors (per video) for the different types of errors (1=Tc+, 2=Tc-, 3=Ts, 4=Tx+, 5=Tx-).

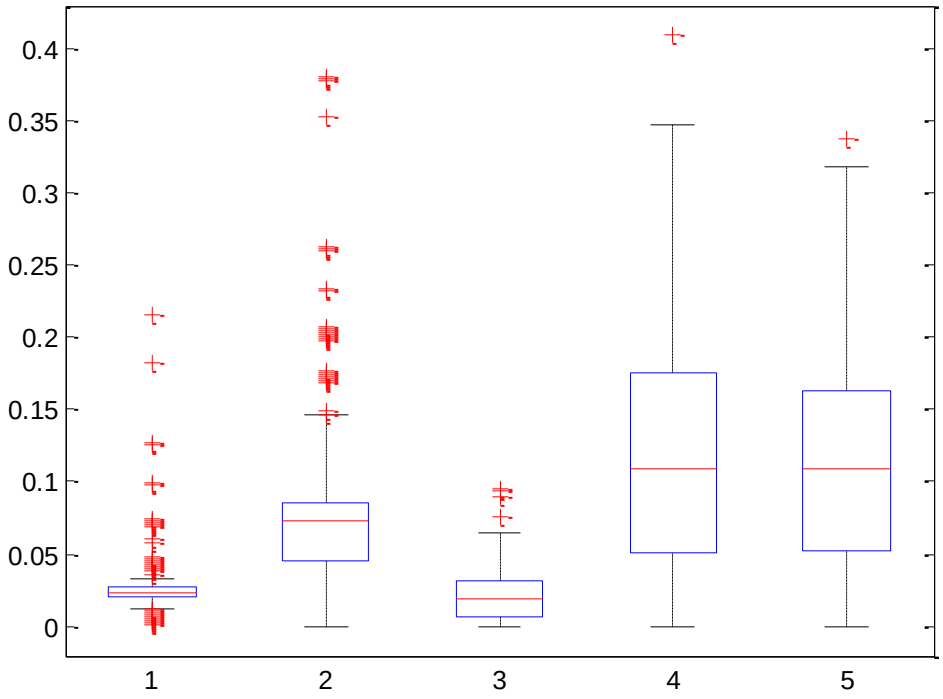

Figure 20: Prediction (against ground truth), absolute fraction of output errors (across videos) for the different types of errors (1=Tc+, 2=Tc-, 3=Ts, 4=Tx+, 5=Tx-).

7.1.5 Experiment 3: missing data

The experiment is the same as experiment 1, but we omit some of the data on input errors. We assume we have no values for merged shots in the input (as it might be easier just going from shot boundary to shot boundary and checking the correctness, rather than trying to find missed ones). In training with this data, the EM algorithm needs significantly longer to converge, but the results are only slightly worse than the ones obtained from complete data. This experiment shows that the Bayesian network is capable of dealing with missing data and still provides satisfactory results, which increases the practical value of the approach for cases where only partial ground truth can be obtained (e.g., from user feedback).

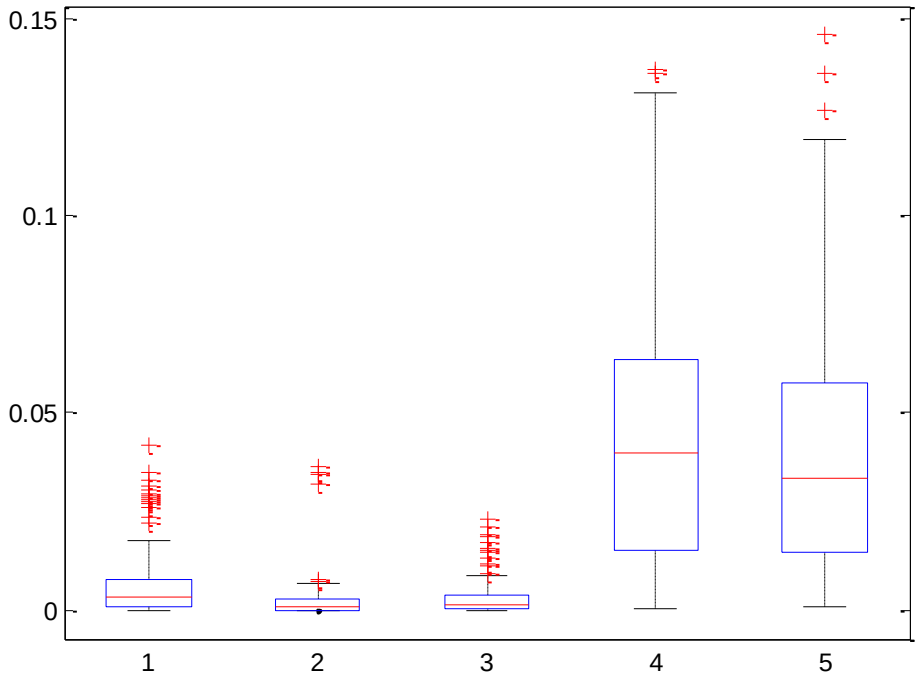

Figure 21: Prediction with missing data (against result from ground truth input), absolute fraction of output errors (per video) for the different types of errors (1=Tc+, 2=Tc-, 3=Ts, 4=Tx+, 5=Tx-).

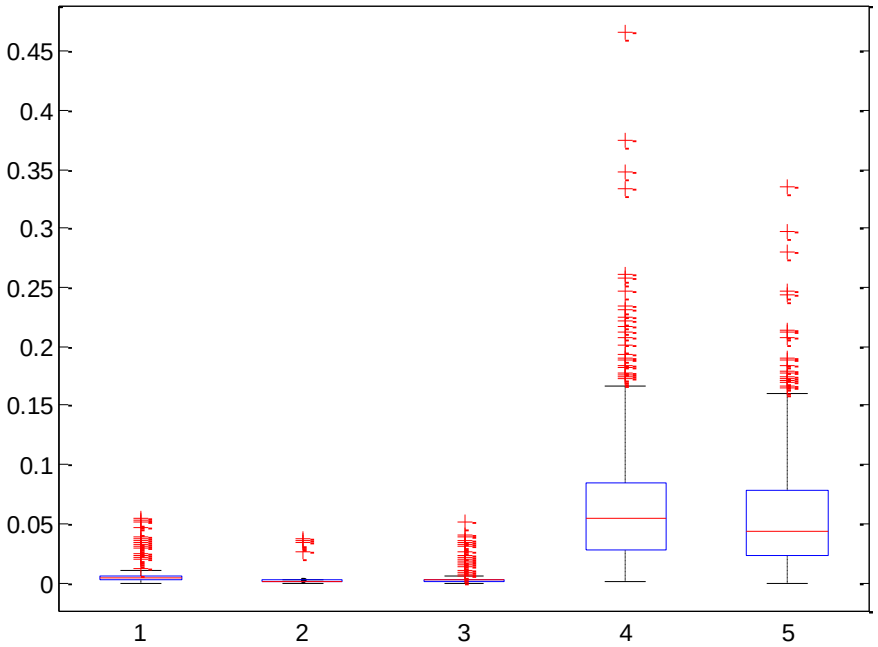

Figure 22: Prediction with missing data (against result from ground truth input), absolute fraction of output errors (across videos) for the different types of errors (1=Tc+, 2=Tc-, 3=Ts, 4=Tx+, 5=Tx-).

7.1.6 Experiment 4: precision/recall for input

In this experiment we try to verify, whether the higher granularity of the deviation against the ground truth is actually needed. We use the data from experiment 1, but we convert the input errors into precision and recall. Merged and split shots naturally translate into false negatives and false positives. Small shifts of shot boundaries are ignored, and large shifts contribute to both false positives and false negatives. We remodel the network so that it only has two nodes on the input side, representing precision and recall.

As can be expected, the results are slightly worse as when using the more detailed errors of the input ground truth.

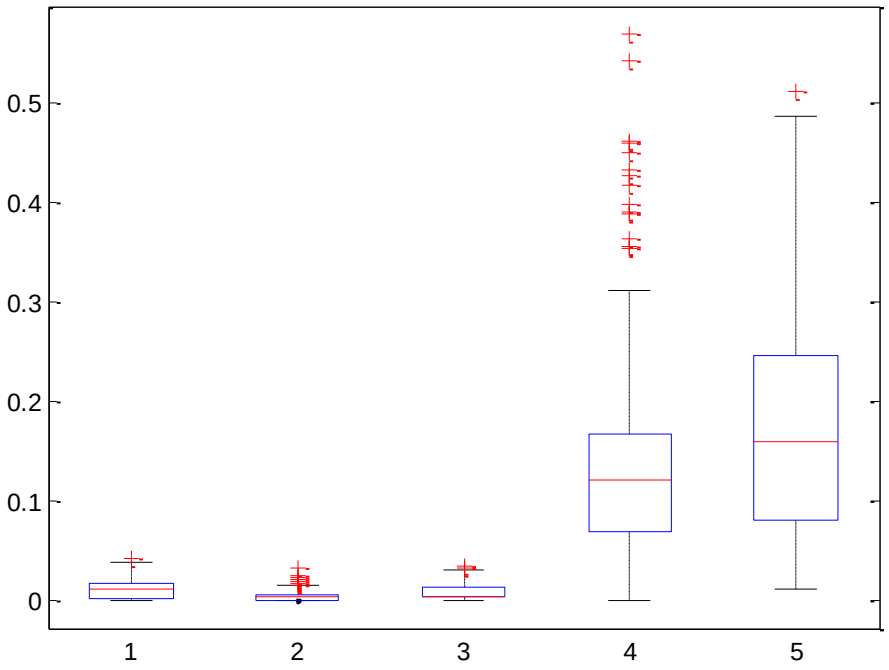

Figure 23: Prediction from precision/recall (against result from ground truth input), absolute fraction of output errors (per video) for the different types of errors (1=Tc+, 2=Tc-, 3=Ts, 4=Tx+, 5=Tx-).

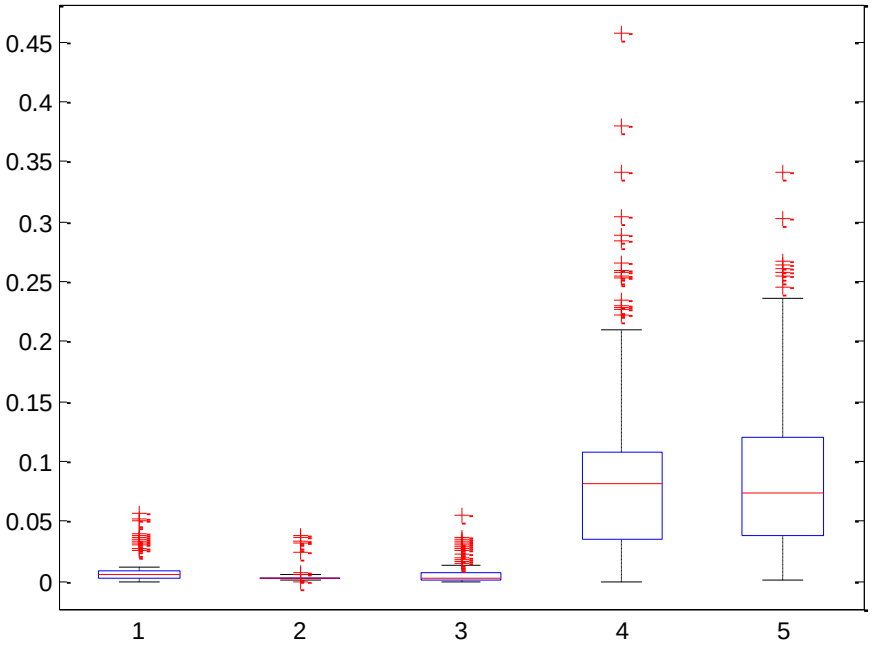

Figure 24: Prediction from precision/recall (against result from ground truth input), absolute fraction of output errors (across videos) for the different types of errors (1=Tc+, 2=Tc-, 3=Ts, 4=Tx+, 5=Tx-).

7.1.7 Experiment 5: no ground truth for intermediate results

In this experiment, we assume that we have no ground truth for the shot segmentation, but only for the final output of the process. We thus generate segmentations of varying quality levels, and try to learn the network parameters from this data.

We use the data from experiment 1. We generate shot boundary results from the ground truth, adding errors of each type. We generated 5 levels of quality, adding a normally distributed fraction of errors with means 0.05; 0.1; 0.2; 0.3 and 0.4. We then use one of these like the ground truth in experiment 1, i.e. we generate variation of it and run the clustering component.

Figure 25 and Figure 26 show the median error per and across videos. As expected, the error increases with decreasing quality of training data. The increase of the error is higher for the per video results than across videos. This seems to be due to the fact that the across video results are based on a larger set of data and are thus more robust. Despite the increase, the error rates are still in a reasonable range. For many analysis tools, error rates not higher than 20-30% can be expected from actual results. This means that the approach can be applied even if ground truth is missing for an intermediate step. The parameters of the network can also be learned from simulations based on actual results.

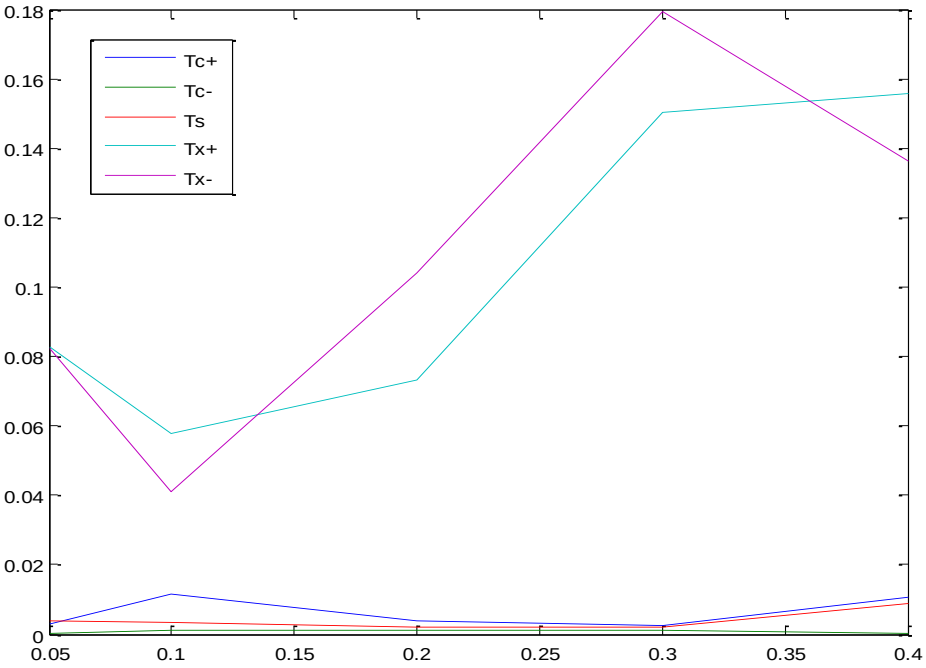

Figure 25: Median absolute error per video determined from input segmentation with different mean errors.

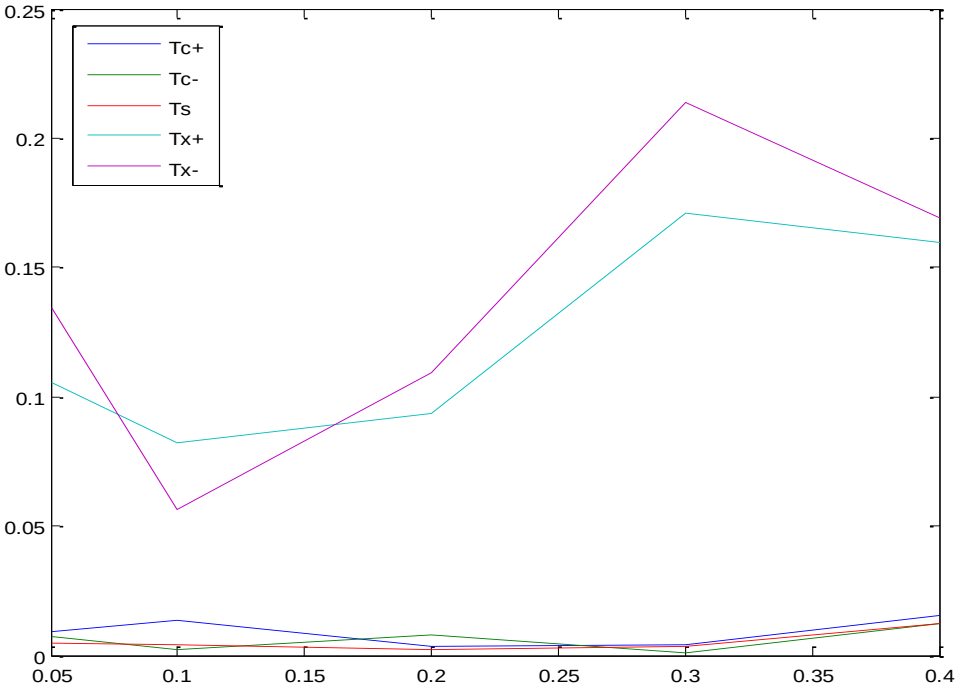

Figure 26: Median absolute error across videos determined from input segmentation with different mean errors.

7.2 Quality analysis

We assess the performance of a video breakup detection algorithm, depending on a previous shot boundary detector and content properties of the video, in particular the visual activity (including camera and object motion).

7.2.1 Experiment setup

We use the video breakup detector described in [Fassold, 2012].

The video breakup detector has the following inputs:

- The content itself
- A temporal segmentation of the video (shots)

The component has the following outputs:

- Segments with start and end times where video breakups have been detected
- A confidence value for each detection

For the input segmentation, we consider the following edit operations (corresponding to possible error types) of the input segment w.r.t. a ground truth:

- Merge of two segments (M)
- Split of two segments (S)

All input values are expressed in terms of fractions, using the number of segments as denominator.

We consider the following edit operations of the output wrt a ground truth:

- Fraction of false detections (FP)
- Fraction of missed detections (FN)

The corresponding Bayesian network is shown in Figure 6 (right).

7.2.2 Data

For assessing the video breakup detection algorithm, we use a data set collected by the vdQA project[20]. The data set consists of 84 videos (9,267 shots) which contain 89 video breakup defects.

For the activities, we only use the actual measurements from the videos. We generate a set of 1,000 segmentations for each video, where we introduce random errors. As we do not have ground truth shot boundaries, we use the results of an actual shot boundary detector as input to the generation.

The parameters of the errors are as follows (normally distributed):

- Fraction of merged segments: $N(0.1;0.05)$
- Fraction of split segments: $N(0.1;0.05)$, the relative split position is $N(0.5;0.2)$, i.e., around the center of the segment

We generate outputs of the video breakup detection algorithm, not taking shot information into account (one would usually discard detections across shot boundaries). We can apply the shot boundary information later, i.e., without rerunning the feature extraction, and generate the video breakup detection results for each of the input shot boundaries. We perform 5-fold cross validation, training on 80% of the generated shot boundaries and activities of the respective videos and testing on 20% each time.

As a baseline, we determine the mean absolute error rate for a prediction only based on the statistics of output errors of the video breakup detector. This baseline is 0.0298 for false positives, and 0.0250 for false negatives.

7.2.3 Prediction of error rates

Using shot boundaries only for the prediction, we obtain 0.0277 for false positives and 0.0117 for false negatives, using only activities 0.0220 for false positives and 0.0112 for false negatives. These results show that activity provides a better prediction, especially for false positives. The error rate for false

[20] http://vdqa.icg.tugraz.at/

positives is in general higher, mainly due to a few large outliers. The MAE for using both shot boundaries and activities for the prediction is the same as for activities only. Figure 27 shows the statistics of ground truth errors and predicted errors.

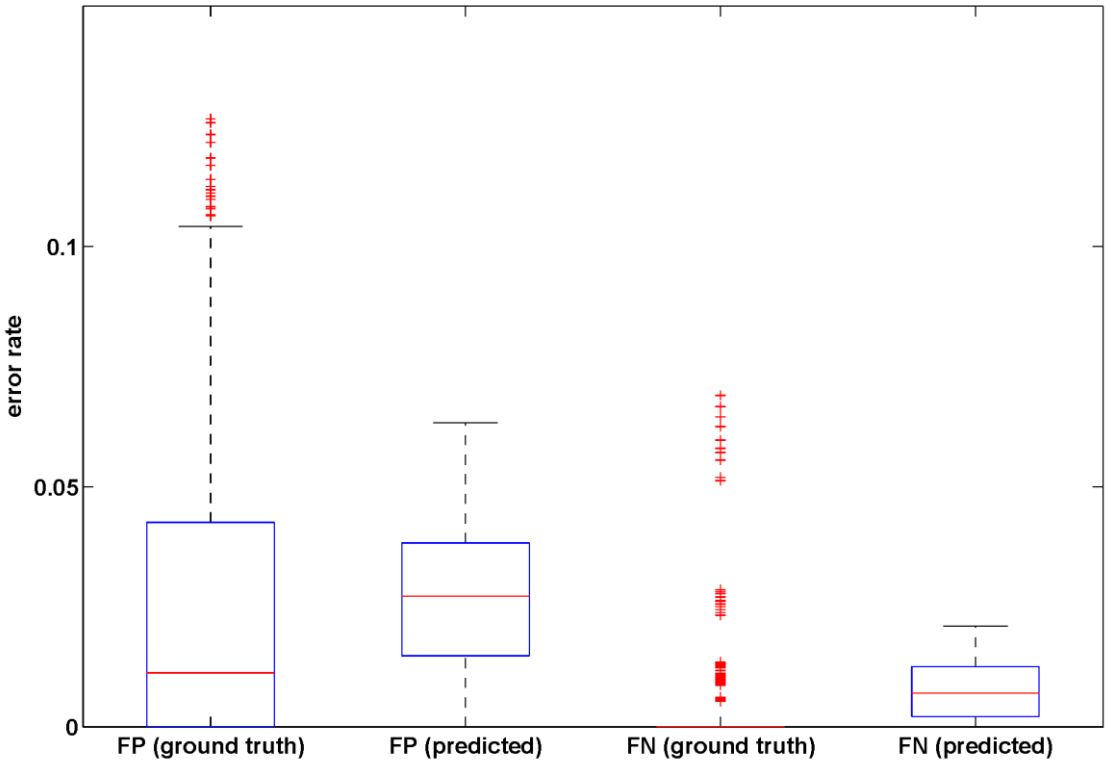

Figure 27: Ground truth and predicted error rates for video breakups, using both activities and shot boundary errors as input.

7.3 Using ASR output for named entity extraction and matching

We assess the influence of the performance of the ASR output on subsequent tasks, in particular, named entity recognition (NER) and semantic similarity matching.

There is work from the area of spoken language understanding that shows that the performance of keyword recognition or task identification from speech is not always correlated with the word error rate (WER). [Riccardi, 1998] showed that in a system for call routing, small WER improvements can result in significant overall performance improvements. [Wang, 2003] observe that there is no correlation or even negative correlation between WER and task identification in a language understanding system, when WERs are moderate. In contrast, for keyword-based topic detection, [Park, 2008] report that the keyword error rate increases more than the WER when the quality of the transcript deteriorates. [He, 2011] investigated the relation between ASR and subsequent machine translation, and reports that optimizing the ASR component of the BLUE score of the subsequent translation improves the results over optimizing the ASR component for WER.

7.3.1 Experiment 1: Named Entity Recognition

Experiment setup

We use the speech recognition service developed by FBK and the named entity recognition service developed by KUL (see D2.3).

The named entity recogniser uses the ASR output as its input and outputs a list of named entities, as well as their type (person, organisation, location).

For the ASR output, we consider the following edit operations (corresponding to possible error types) of the input segment w.r.t. a ground truth:

- Inserted words (I_{ASR})
- Deleted words (D_{ASR})

- Substituted words (S_{ASR})

All input values are expressed in terms of fractions, using the total number of words as denominator. The sum of the error fractions corresponds to the word error rate (WER).

We consider the following edit operations of the output wrt a ground truth:

- Fraction of false named entity detections (FP)
- Fraction of missed named entity detections (FN)

The corresponding Bayesian network is shown in Figure 28 (left).

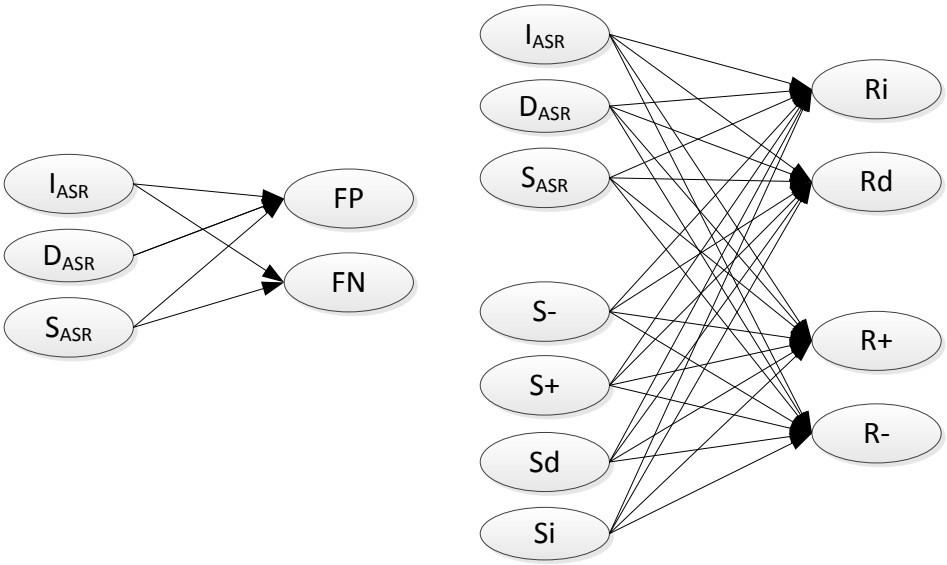

Figure 28: Bayesian network modelling the use of ASR outputs for named entity recognition (left) and in the MediaEval Search & Hyperlinking task (right).

Data

We use 9 videos from the TOSCA-MP data set. These videos are news broadcasts from DW-TV in English with a total duration of about 25mins each. For these files, a manually created speech transcript has been created by FBK/CELCT, and NER ground truth has been created by KUL and JRS.

The NER service has been run on three sets of inputs:

- The manually generated ground truth
- Actual output of the ASR service
- A simulated ASR output, with errors added as follows: The target word error rate was chosen to be the same as the ASR output, with 60% of the errors being substitutions, and 20% each deletions and insertions. The errors were uniformly distributed over the text and independent of word categories. It was assumed that the substituted and inserted words contain the same fraction of named entities as the ground truth of the respective video.

Results

We assessed the fractions of matching, inserted and deleted words depending on the level and distributions of errors in the input text. As expected, performance is best for the ground truth, setting a maximum performance level for the NER algorithm under test. The actual ASR output and the simulated ASR output have the same word error rate, but the actual ASR output results in a higher error rate of the named entities. The results (shown in Figure 29) indicate that the ASR algorithm is more likely to have errors on named entities than on other words, which can be expected due to the number of names and places appearing in news that have not been encountered before. In some cases, also the inconsistent pronunciation of foreign names contributes to this issue.

In addition to the numbers for matches, insertion and deletions shown in Figure 29, we also assessed partial matches of named entities (e.g., only first or last name of a person). The rates for partial matches are 0.038 for the ground truth, 0.066 for the actual ASR result and 0.096 for the simulated one. This is the only case where the ranking of the results for the three input types is different, i.e. the actual ASR

outperforming the generated one. The ASR algorithm seems to more likely succeed or fail on the entire named entity, while the simulated result treats each word of multi-word named entities independently. We additionally checked for matching entities, but with a wrong entity class assigned. However, this has not been observed in the outputs for this data set.

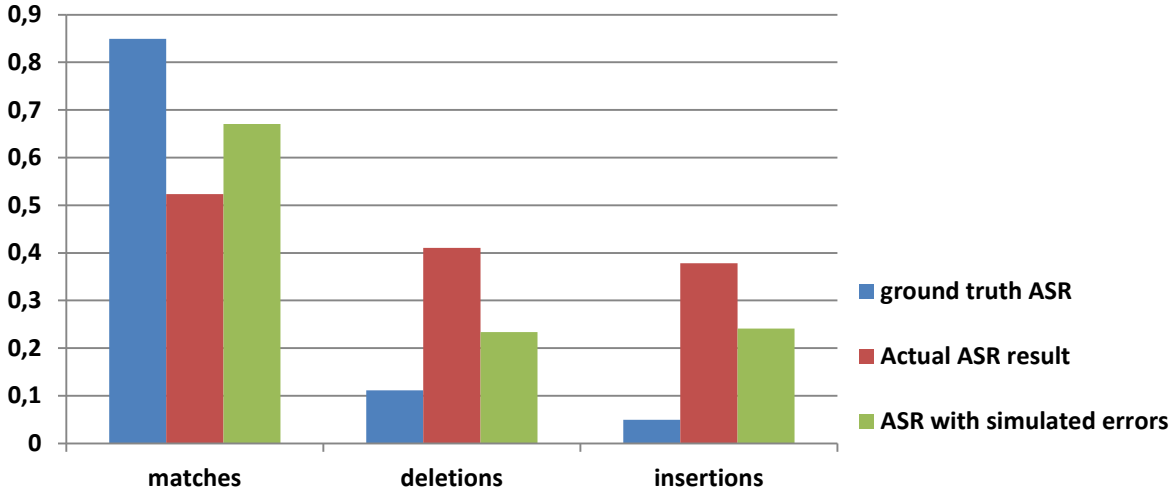

Figure 29: Matches, deletions and insertions of named entities.

7.3.2 Experiment 2: MediaEval Search & Hyperlinking Task

Experiment setup

We have used the linking step of the MediaEval 2013 Search & Hyperlinking (S&H) task to assess the dependency of the text-based linking on the ASR performance. For short summary of the S&H task, see section 7.5. The linking step determines video segments from the collection related to an anchor segment based on their similarity. We use only text-based similarity between the segments, taking the context segment of the anchor into account. The method is described in [Lokaj, 2013].

The textual linking component depends on the ASR output for the segments to be matched and the search result for which the anchors are chosen.

For the ASR output, we consider the following edit operations (corresponding to possible error types) of the input segment w.r.t. a ground truth:

- Inserted words (I_{ASR})
- Deleted words (D_{ASR})
- Substituted words (S_{ASR})

All input values are expressed in terms of fractions, using the total number of words as denominator. The sum of the error fractions corresponds to the word error rate (WER).

In this experiment we do not model the dependency from the performance of the search system, but assume the input anchor as fixed. We describe experiments considering this modification in Section 7.5.

We consider the following edit operations of the output w.r.t. a ground truth:

- Fraction of segments added (Ri)
- Fraction of segments deleted (Rd)
- Fraction of segments ranked higher (R+)
- Fraction of segments ranked lower (R-)

The corresponding Bayesian network is shown in Figure 28 (right).

Data

We use the manual subtitles provided by BBC as the starting point of the experiment. We generate versions of the subtitles by randomly introducing errors with a certain target WER, assuming that 20% of

the errors will be insertions, 20% deletions and 60% substitutions. We then run the linking approach described in [Lokaj, 2013], using textual similarity only.

Results

Figure 30 shows the results of the experiment, reporting mean average precision (MAP) and precision for the top 5 and 10 ranks. The performance decreases only moderately with increasing WER, esp. the precision at the top ranks stays at the same level up to a WER of 0.4. At a WER of 0.7, the MAP is half of the value for the unmodified input. This is in line with the official results, which show that there is no significant performance difference between the subtitles and ASR outputs (however, the WERs of these ASR outputs are not known).

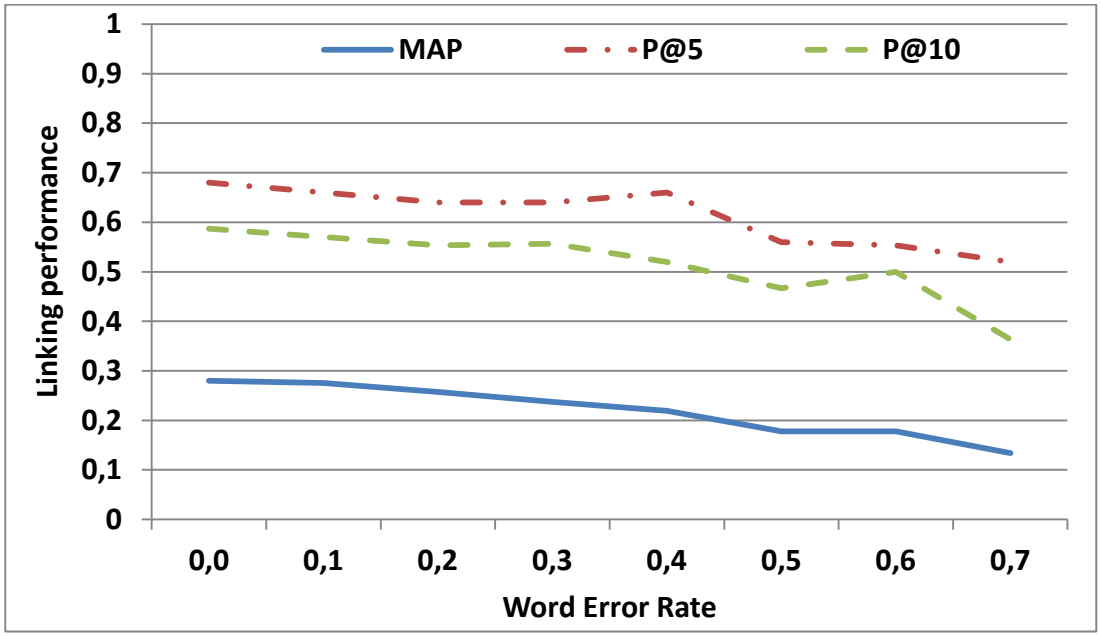

Figure 30: Dependency of linking performance on word error rate.

7.4 Search benchmark

This following experiment is based on the Search and Hyperlinking Task at MediaEval 2013 [Eskevich, 2013]. MediaEval is a benchmarking initiative for multimedia evaluation that brings together researchers who participated in benchmark tasks to report on their findings, discuss their approaches and learn from each other. MediaEval offers different inspiring new scenarios of user interaction to deal with the increasing amount of digital multimedia content available. In this case, the underlying use scenario is a popular information seeking strategy that is executed millions of times every day: users search for relevant information using a query and then follow hyperlinks to further explore the collection. For the textual web, this use scenario is already supported by search engines and embedded hyperlinks in web pages. Furthermore, there is research in searching for multimedia content and in, mainly manually, linking such content. However, there is virtually no research in supporting searching and hyperlinking videos. The Search and Hyperlinking Task models a specific instance of this use scenario to encourage researchers to develop technology that supports home users in their daily routine.

About the dataset

The dataset for the task was formed by textual resources and visual cues.

On the one hand, a collection of 1,260 hours of video was provided by the BBC. The average length of a video was roughly 30 minutes and most videos were in the English language. The collection was used both for training and testing of systems. The BBC also provided human generated textual metadata and manual transcripts for each video. Participants were also provided with the output of two automatic speech recognition (ASR) systems:

- All audio files were transcribed by LIMSI-CNRS/Vocapia using the VoxSigma vrbs_trans system (version eng-usa 4.0) [Gauvain, 2002]. The models used by the system have been updated with partial support from the Quaero program [Gauvain, 2010].
- The LIUM system [Rousseau, 2011] is based on the CMU Sphinx project, and was developed to participate in the evaluation campaign of the International Workshop on Spoken Language Translation 2011. LIUM generated an English transcript for each audio.

On the other hand, participants are provided with shot boundaries, one extracted key frame per shot, as well as the outputs of concept detectors and face detectors for these key frames. For each video, shot boundaries were determined and a single key frame per shot was extracted by a system kindly provided by Technicolor [Massoudi, 2006]. The extracted frame was the most stable I-frame within its shot. In total, the system extracted approximately 1,200,000 shots/key frames. Concept detection scores for a list of concepts were provided. These concepts were selected by extracting keywords from metadata and spoken content. We used the on-the-fly video detector Visor, which was kindly provided by the Computer Vision Group of University of Oxford [Chatfield, 2011]. To make the confidence scores comparable over multiple detectors, organizers used them as variables in a logistic regression framework, which ensures the scores lie in the range [0:1]. They set the logistic regression parameters to the expected value of the parameters from over 374 detectors on the Internet archive collection used in TRECVid 2011. INRIA [Cinbis, 2011] kindly provided possible bounding boxes in key frames with a confidence score that the bounding box contains a face. Additionally, the tool also contained for each bounding box, the N most similar faces (bounding boxes) in the dataset.

About ground truth and evaluation

For the definition of realistic queries, organizers conducted a study with 30 users between the ages of 18 and 30. By browsing the collection, the users selected items, a segment of a video with a start and an end time, that were interesting to them. The users were then instructed to consider these items as a known-item which they have to rediscover. Organizers asked the users to formulate text and visual queries that they would use in a search engine to carry out their re-finding. The study resulted in 50 known-items and corresponding multimodal queries.

Organizers used the following three metrics in order to evaluate the submissions of the workshop participants:

- Mean reciprocal rank (MRR), which assesses the ranking of the relevant units.
- Mean generalized average precision (mGAP), which rewards techniques that not only find the relevant items earlier in the ranked output list, but also are closer to the ideal point to begin playback (the \jump-in" point) of the relevant content.
- Mean average segment precision (MASP), which takes into account the ranking of the results and the length of both relevant and irrelevant segments that need to be listened to before reaching the relevant content.

About the approach

The TOSCA-MP system was not originally oriented to retrieve segments or scenes but media assets exploiting general textual information (e.g. description), time-based textual information (e.g. ASR or subtitles), manual or automatic semantic annotations, and other low-level features extracted by external analysis services (e.g. concept detection). This basically means that, for any query, the scoring function return a list of media assets based on all the metadata available for the media assets. The change in the overall behaviour has implications in how the whole system ingests and indexes media assets. Thus, the first step is to produce the segmentation of the media assets. There are several approaches [Eskevich, 2013] for that:

- Sliding window: the segmentation is produced into short segments of length approximately the same as that of an average relevant segment.
- Speech segment: the segmentation is produced into consecutive silence bounded by utterances from the same speaker.
- Based on Lexical cohesion: the segmentation is based on the lexical cohesion. For example, the algorithm TextTiling computes the cosine similarity between adjacent blocks of sentences with fixed size.

In our case, the system is based on the use of textual resources - manual curated subtitles and transcripts (LIMSI) – with a sliding window with a fixed size (30, 90). The system also indexed the

metadata of the media assets but we realised that only using metadata showed very low results and the tests did not show a clear pattern on how better use metadata plus other information and we omitted it in the process. Although we were interested in including visual descriptions into the process, we did not converge towards a relevant implementation in time. Thus, for each of the 3.076 videos provided by the challenge, the system creates segments with the corresponding subtitles and transcripts for the different sliding windows.

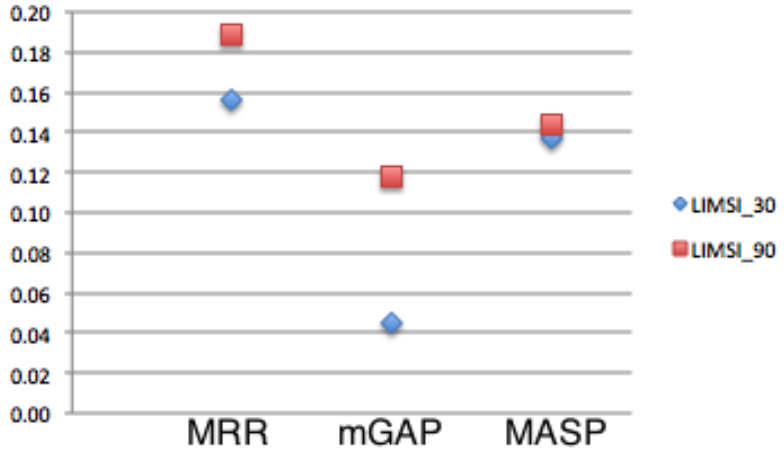

Figure 31 - Results for MRR, mGAP, and MASP

Figure 31 shows MRR, mGAP, and MASP scores for all the runs with window sizes of 30 and 90 seconds. As would be expected, smaller window size decreases the scores, however the trend of the method effectiveness remains the same except for the MASP score (this is because the sliding windows are not overlapping).

The current system is based on textual resources and the effect of scene decomposition and visual features on the results should be investigated in details. On the one hand, scene decomposition has been reported useful for a better definition of video segments. On the other hand, visual descriptions of video content can potentially help for searching as it seems that other approaches using the visual concepts in the query slightly increases the results for all measures.

7.5 Search Results and Linking

This experiment is based on the Search and Hyperlinking Task at MediaEval 2013 [Eskevich, 2013], which uses the following scenario: A user is searching for a segment of video that they know to be contained in a video collection (“known item”). If the user finds the segment, she may wish to find additional content related to some aspect of this segment. The MediaEval linking subtask is either based only on this relevant item (“anchor”) returned by the search system, or one a larger contextual segment around the search result (e.g., a news story, a scene).

Experiment setup

In this experiment, we perform the linking step of the task, using the method described in [Lokaj, 2013].

In the benchmark data set, the input to the linking task is well-defined by a relevant anchor segment and its contextual segment. In a real-world setting, we cannot expect that the correct anchor is always ranked near the top of the results, or that we have a precise definition of the contextual segment. We thus investigate the performance of our linking method on actual search results of different systems and with a coarse estimate of the context. These experiments answer the question whether having video segments that are not correct, but only close-by hits are good enough as input for the subsequent linking step. If this was the case, adding the linking step in the search process would increase the practical value of search systems, even if the performance of the initial search result is limited.

For the search result output, we consider the following edit operations (corresponding to possible error types) of the input segment w.r.t. a ground truth:

- Result deleted (Sd)
- Result inserted (Si)
- Result ranked up (S+)

- Result ranked down (S-)

As described below, only the top 10 results of the search output will be considered.

In this experiment we do not model the dependency from the performance of the ASR output, but assume the input anchor as fixed. We describe experiments considering this modification in 7.3.2.

We consider the following edit operations of the output w.r.t. a ground truth:

- Fraction of segments added (Ri)
- Fraction of segments deleted (Rd)
- Fraction of segments ranked higher (R+)
- Fraction of segments ranked lower (R-)

The corresponding Bayesian network is shown in Figure 28 (right).

Data

The MediaEval dataset is a collection of 1,260 hours of video provided by the BBC. The average length of a video is roughly 30 minutes and most videos are in English. The BBC kindly provided human generated textual metadata and manual subtitles for each video. Participants are also provided with the output of two automatic speech recognition (ASR) systems, one by LIMSI-CNRS/Vocapia using the VoxSigma vrbs trans system [Gauvain, 2002], and the other using the LIUM system [Rousseau, 2011] based on the CMU Sphinx project.

We use the ground truth of the official MediaEval evaluation created from crowd sourcing. Like with all pooled evaluations, other runs may contain relevant segments not in the set of segments originally judged. We use the evaluation measures described in [Aly, 2013], as well as the evaluation script referenced there. As an evaluation measure we report mean average precision (MAP) of a ranked list of linked segments, calculated using segment overlap as described in [Aly, 2013]. We determined results for the other two methods for MAP calculation described in that paper (binned, tolerance to irrelevance), and the results are similar (though those MAP scores are generally lower). We follow the same rules as in the official evaluation, i.e., we discard result segments coming from the same programme as is the anchor.

Experiment 1: Automatically Determined Context

Based on the finding from the MediaEval benchmark, that the use of contextual information around the anchor segment provides significant improvements of the results, we investigate how important it is to know the precise boundaries of these context segments. We apply the proposed linking method to the manually defined context segments from the ground truth and to automatically determined context segments that are created by adding k minutes before and after the anchor segment, with k={1,3,5}. These values have been chosen to cover different lengths of context segments.

The results obtained from automatically determined context in terms of MAP and precision at top ranks of the linked segments are slightly lower but comparable to those obtained from the manually defined context segments from the ground truth. Figure 32 provides an overview of runs with different context lengths and textual resources.

Short context segments (±1 minute) have the best scores for top 5 runs, although the performance is slightly lower than for the manually defined ones. In terms of MAP medium length context segments (±3 minutes) score best. These results show that the additional information from the story around the anchor segment is useful, even if it may be incomplete, or parts from adjacent stories are included in the context.

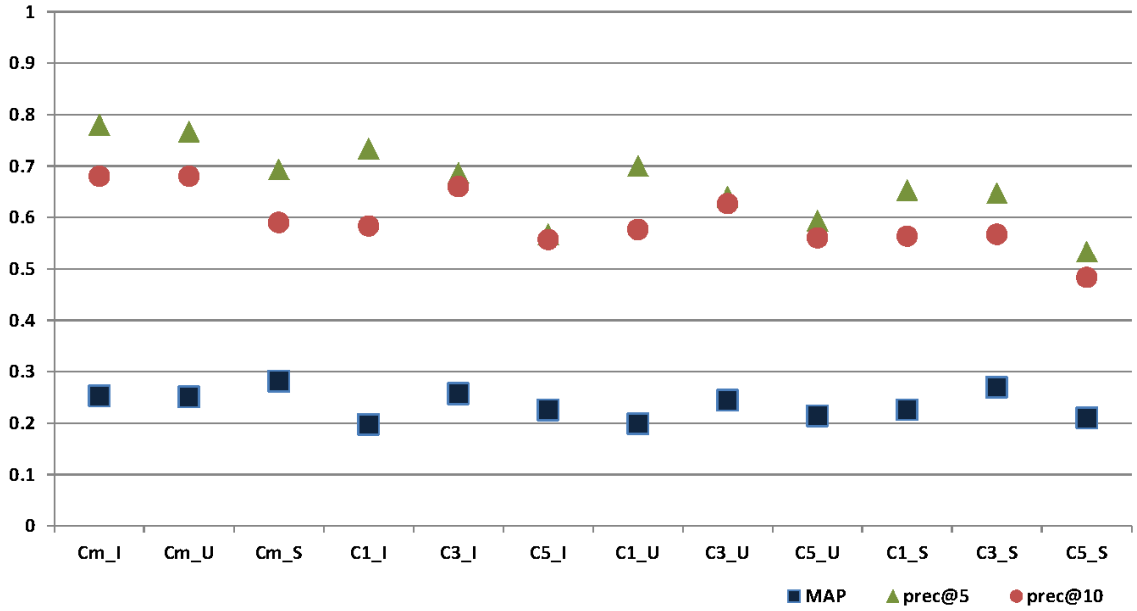

Figure 32: MAP, precision at 5 and 10 over all anchors, using manually defined context (Cm), or automatically defined context segments of certain lengths (C1, C3, C5) and different text resources (I, U, S).

Experiment 2: Starting from an Actual Search Result (Top-ranked)

The aim of the linking step is to obtain a set of content items that fulfil the user's information need. In a real-world setting, linking would have to start from a video segment in the search result. We can assume that the top-ranked search result, even if it is not among the correct target items of the search, is somewhat relevant to the query. We thus use the top-ranked search result as input to linking. We run this experiment on the search submissions from different participants of the MediaEval 2013 Search & Hyperlinking task in order to answer the following questions:

- Is the performance of the linking results comparable to the performance when starting from the true anchor item?
- Is adding context also useful in this case?
- Is there a correlation between the performance of linking from the top-ranked search result and the overall search performance?
- What is the performance, if the linking result is assessed as the search result? Does it correlate with the performance of the initial search result?

We perform linking both using the actual video segment from the search result as anchor segment, as well as using the context around the search result. Due to the lack of information about the actual stories/scenes, we define the context as the anchor segment ±3 minutes. For the experiment, we select a subset of search submission that reached different performance levels. As several variants of the same method (e.g., using different text resources) often yield similar scores there is no added value in using all the runs.

Assuming no knowledge (and user feedback) about the relevance of the video segments in the search result, we start the linking process from the top-ranked item in a search result. Figure 33 shows the MAP and precision at rank 5 and 10 obtained from a set of actual search runs submitted to the MediaEval 2013 Search & Hyperlinking task, using the proposed linking method. The horizontal lines indicate the best results obtained from starting with the known item as input for the linking step. We can see that the results when starting from the top-ranked item are clearly below this upper limit. We also see that the performance decrease affects the top items of the linking result more, while MAP is still comparable, at least when starting from some of the search runs.

An analysis of the results reveals that the main reason for the lower performance is that the context around the anchor segment does not improve the results, if we are not starting from the correct item. While we observed significant improvement when using context when starting from the correct item, this disappears when starting from the top-ranked item of the search results: the linking MAP is 0.164 with the anchor only, and 0.160 with the context (this is very similar to the MAP of anchor only when starting from the ground truth segment). We can see that for the runs that only use the anchor the performance

is still comparable to that obtained when starting from the correct item. Thus, the performance difference we observe when starting linking from another item than the correct search result is due to the fact that we do not gain improvements from using contextual information. In contrast, while the anchor segment (i.e., the search result item) might still be somewhat relevant, the context might be completely different, thus decreasing the relevance of the linked items.

We also investigate the correlation between the correctness of a search result (using mean reciprocal rank MRR to assess the search performance as described in [Aly, 2013], with window size 10) and the relevance of the items obtained from linking the top-ranked item of the result list. As can be assumed, there is a positive correlation, although not very strong (see Figure 34). This means that linking from the top-ranked item of a search result list that has the relevant items ranked higher, will result in better linking performance. Note that we only use the first item from the search result for linking. Given the rather low MRR scores of the search results, the top-ranked one is likely not a relevant one for most of the queries. However, the correlation we observe indicates that search systems that rank the relevant items higher will have top-ranked items that are more similar to the relevant items, thus resulting in higher linking performance.

Finally, we are interested to know whether linking can help to find the relevant items, that might be further down the result list in the initial search result. Linking from the top-ranked item is used here as a refinement step of the search result. We treat the linking result as an updated search result and compare the MRR obtained. Figure 35 shows the correlation between the MRR from the linking result and that of the initial search result (correlation coefficient 0.44, *p*-value 0.0003). One interesting observation is that the MRR from the linking results (mean 0.072, median 0.064) are overall slightly worse, but still comparable to those from the initial search results (mean 0.078, median 0.074). For several runs, an improvement is achieved, i.e. the linking result has relevant items ranked higher than initial search results. It can be assumed that this effect is much stronger, if we do not automatically start from the top-ranked segments, but let the user choose the start segment for linking from the top-ranked items in the search results.

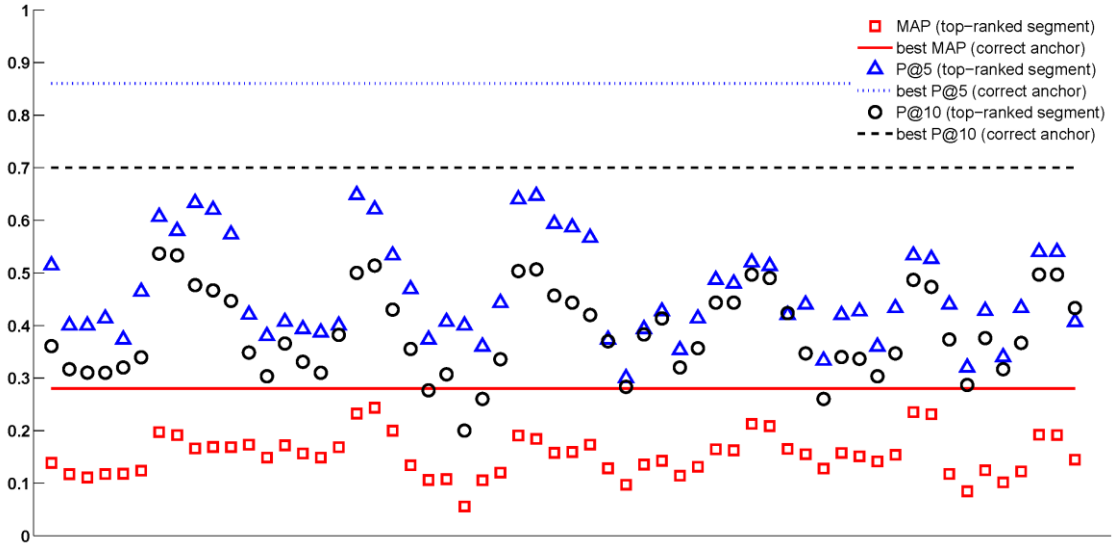

Figure 33: Mean of MAP, precision at 5 and 10 of the linking results generated from the top-ranked video segments in actual search runs submitted to the MediaEval 2013 task. The lines indicate the best MAP/precision obtained using the same method when starting from the correct result item of the search.

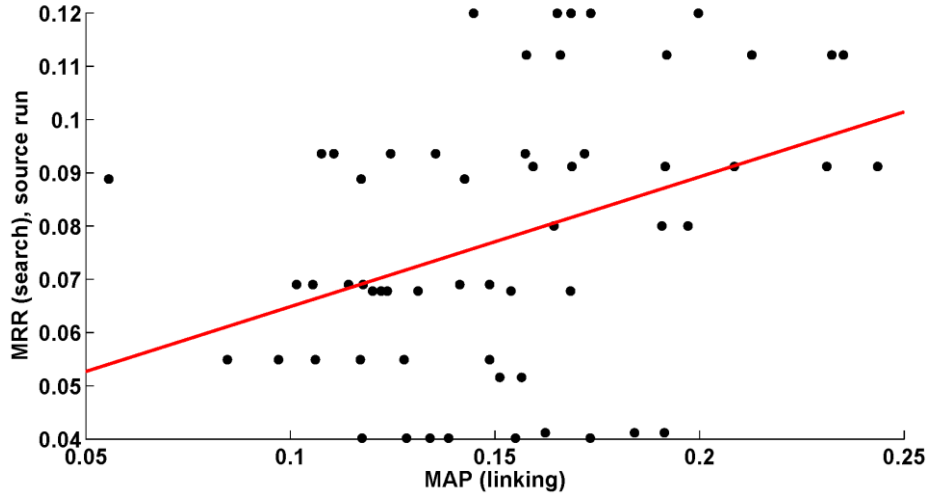

Figure 34: Correlation between MRR of search runs and MAP of the resulting linking result, using the top-ranked item of a search run as input (correlation coefficient 0.30, *p*-value 0.0178).

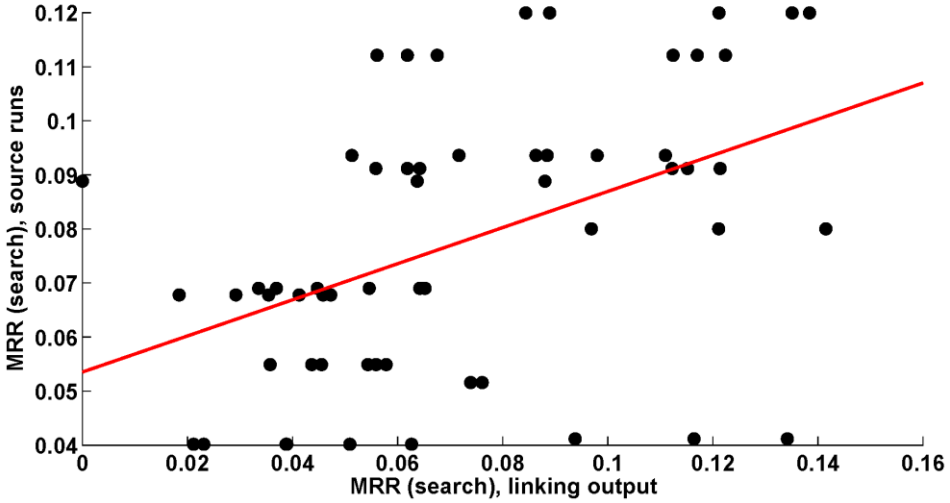

Figure 35: Correlation between MRR of search runs and MRR of the linking result, when treating the list of linked items as an updated search result (correlation coefficient 0.44, *p*-value 0.0003).

Experiment 3: Starting from an Actual Search Result (Related)

The setup of the experiment is the same as when starting from an actual top-ranked search results, but we investigate whether these results change, if we use the first item among the top 10, that is related to the query. We thus simulate a user interaction of choosing the first related segment from the search results. In the experiment, we define a segment as related to the query, if it occurs in the ground truth of the corresponding linking task.

As in the previous experiment, we start from an actual search result, but we choose the first segment among the top 10, which is in the ground truth of the linking result. This is not the known-item result of the search task, but a segment that is related to the query. In this way we simulate the interaction of a user choosing the first segment in the search result that is somewhat related to the query.

As can be expected, the results improve significantly, and are comparable to those obtained starting from the correct anchor segment. Figure 36 shows the results of this experiment. While the differences between using anchor only or also context are not as large as when starting from the correct segment, there is still a clear improvement from using context. The mean of MAP over the anchor-only runs is 0.245, while that of the runs using context is 0.337.

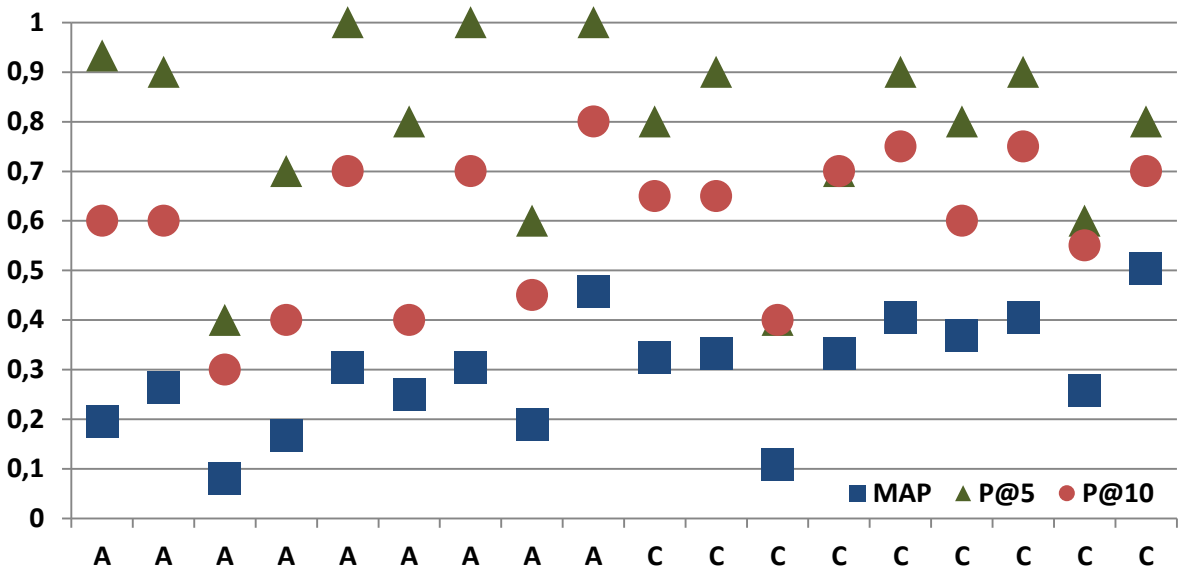

Figure 36: MAP, precision at 5 and 10 of the linking result when using relevant search result among the top 10 as input.

7.6 Benchmarking of visual clustering

7.6.1 Experiment setup

For the evaluation of the visual clustering method, a ground truth set was created.

Users on the web-service 'Amazon Mechanical Turk' annotated a set of 500 key frames (50 key frames from 10 video's, some of them Royal wedding related) by comparing each pair of two images in this set (500*499/2 = 124,750 comparisons in total), and assessing whether these images are 'Very similar', 'Somewhat similar' or 'Not similar'. Some control questions were added to increase the reliability of people's answers.

Also for these 500 images, descriptor values were calculated for the CSD, EHD and CEDD descriptors. All 124,750 distances between all image pairs were calculated using the distance measure corresponding to the descriptor type.

Comparing the calculated distances for each of the descriptors to the ground truth data set allows to evaluate the performance of each of the descriptor types, although a mathematical comparison of descriptor types to each other gets difficult since for each type, the calculated distances lay in a different range with different thresholds or no thresholds at all.

7.6.2 Results

The results are shown as box plots in Figure 37 Figure 37: Box plots of the calculated distances of the image pairs that were user-rated as 'Not similar', 'Somewhat similar' or 'Very similar', for the descriptor types CSD, EHD and CEDD. Box plot boundaries are in Table 6.

the data for these box plots is in Table 6. For completeness, we also show the mean and standard deviation for all the data and the sample size in Table 7.

Table 6: Maximum, Q3, Median, Q1 and Minimum values of the calculated distances of the image pairs that were user-rated as 'Not similar', 'Somewhat similar' or 'Very similar', for the descriptor types CSD, EHD and CEDD.

		Not similar	Somewhat similar	Very similar
CSD	Max	7937	6543	6381
	Q3	3861	3336	3132
	Median	3224	2703	2439,5
	Q1	2654	2173	1868,75
	Min	424	407	132
EHD	Max	240	196	170
	Q3	103	96	90
	Median	85	78	73
	Q1	68	62	57
	Min	5	17	2
CEDD	Max	1,000	1,000	0,975
	Q3	0,799	0,628	0,568
	Median	0,661	0,504	0,430
	Q1	0,524	0,394	0,308
	Min	0,042	0,052	0,007

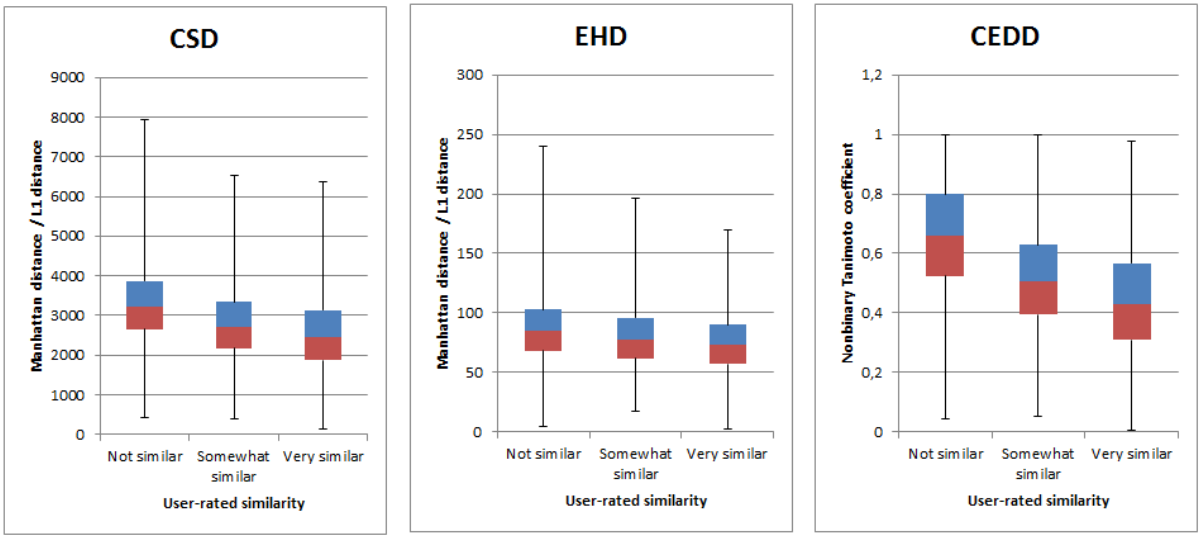

Figure 37: Box plots of the calculated distances of the image pairs that were user-rated as 'Not similar', 'Somewhat similar' or 'Very similar', for the descriptor types CSD, EHD and CEDD. Box plot boundaries are in Table 6.

Table 7: Mean and standard deviation of the calculated distances of the image pairs that were user-rated as 'Not similar', 'Somewhat similar' or 'Very similar', for the descriptor types CSD, EHD and CEDD. Sample size indicates how many of the 124,750 image comparisons fall into each user-rated category.

		Not similar	Somewhat similar	Very similar
	Sample size	124750		
		116376	7132	1242
CSD	Mean	3300,40	2793,02	2494,97
	Standard dev.	900,82	868,83	984,15
EHD	Mean	86,996	80,008	73,609
	Standard dev.	26,84	24,40	24,37
CEDD	Mean	0,660	0,513	0,435
	Standard dev.	0,184	0,171	0,184

In the box plots in Figure 37, we notice how for CEDD, there is almost no overlap between the boxplot 'Not similar' and 'Very similar', which means there is a significant difference in calculated distances between these two categories. For CSD, there is more overlap between these boxes, and for EHD even more, although for all descriptor types a positive correlation is present between the user-rated values and the calculated values (e.g. mean, median).

These results could prove how CEDD outperforms CSD and EHD in image similarity matching. This was also found in [Chatzichristofis, 2008] using the ANMRR (Averaged Normalized Modified Retrieval Rank) as an objective measure. The results also learn us that image similarity based on colours is good (CSD) but profits from using edges (EHD) since image similarity based on both colours and edges (CEDD) is even better. EHD may not be the best descriptor for image similarity, but the reason for its lower performance may also be that users understand similarity as equal colours rather than equal edges, which is reflected in their ratings (the formulation of the question was: *Please rate each of the 11 pairs of images: do they look visually similar? You should consider similarity as visual similarity of the images i.e. colour, appearance, intensity, brightness etc., not semantic or conceptual similarity e.g. if both images contain similar objects (humans, cars, animals etc.)*).

8 Measuring performance by cost

8.1 Introduction

The description of task goals received from the responses to the survey (see D4.1) is rather qualitative and informal. While it is mostly stated which type of information is expected as a result of the task, there is no indication of information quality that could be used in an automatic process. For example, when is the annotation of relevant persons in news material good enough? If the person that is subject of the news item is identified? Which of the detected persons are the key persons? Is it sufficient to have the annotation on a temporal granularity of 30 seconds?

Clearly, the answers to these questions (and consequently how the ground truth needs to be defined) depend on the task context and need a human validation in the loop to specify what that means for a specific content item/set.

Most existing evaluation methods assess the difference to some ground truth. For many annotation tasks it will neither be necessary nor feasible to have a “perfect” result (i.e. no errors such as omissions at all), but a result that makes the annotation usable for a certain purpose. Some evaluation methods go into that direction by assigning costs to certain error types. However, this cannot be done without considering the context represented by the process in which a specific tool has to be integrated.

We can use costs as a measure for the performance of a tool in the process. One aspect are actual costs for a human operator performing validation and correction of automatic results, the other aspects are costs which are penalties for error that remain in the results. For the first, the effort needed to make the result acceptable in a specific task context allows turning performance of a diverse range of tools into working time (and thus cost) as performance measure. The number and types of errors corrected provide feedback on the types of error that are critical in a specific task context and more fine grained costs for different categories of errors can be derived. The task model can be used to trace back the component from which the errors originate and thus provide evaluation information for the component. Of course the effort for correcting certain types of errors is not a constant, but also depends on the efficiency of the user interface available for validation and correction. Thus, providing better tools for the manual intervention also influences the equation of how individual components’ performance impacts the overall annotation effort.

Given a manually defined ground truth, the assessment step can be defined using a kind of metadata edit distance (cf. Section 4.3). The costs for the different edit operations will differ on the type of error, where it occurs, whether it is masked by other components, etc.

8.2 Cost simulation model

Real media production processes are a complex combination of human factors and system operations, and as such quite distant from the aseptic laboratory settings in which automatic information extraction tools are developed. Furthermore, workflows are the result of established practices that involve not only practical technical constraints but also personnel-related issues like shifts, contractual regulations and professional roles. As a result, costs connected with workflows cannot be estimated taking into account individual operations, but considering the whole process. As a consequence, expected workflow optimizations introduced by the introduction of automated tools cannot easily be assessed by just considering their individual performance (e.g., in terms of precision and recall).

The proposed approach is then to simulate the entire process under consideration introducing two distinct modalities for the specific function being optimized, fully manual operations and computer-assisted operations, and then perform an analysis on the minimum performance needed by the automatic tools to improve the overall cost figures of the process.

We can model a workflow as a sequence of orchestrated actions, to each of which a cost and an execution time can be associated. Both costs and execution times can be either fixed or depending on some functions that model the intrinsic operations taking place when the action occurs. Execution times, in particular can be associated to personnel costs and equipment costs, depending on variable cost models or distributed overhead models.

A workflow *W* can be composed of *M* actions orchestrated in ways determined by logical operators that constrain how actions have to follow each other or if one or more actions can run in parallel. In turn, any

action in the workflow, to be accomplished, may need the execution of a set of functions on its input data.

Thus, if we want now to analyse what is the impact of substituting a manual implementation of a certain subset of all the functions in a workflow with a corresponding automatic implementation we have to address, in general terms, a condition which can be expressed as:

$$\sum_{i=0}^{N}\left(C_i^A + C_i^{chk}\right) < \sum_{i=0}^{N} C_i^M \quad (1)$$

where *N* is the number of functions in the workflow in which some automatic tool is introduced, C_i^A is the cost connected with the individual execution of the automatic tool implementing function *i*, C_i^{chk} is the cost connected with the manual check of the automatic tool's results, and C_i^M is the cost connected with the fully manual implementation of function *i*. Costs can have a manifold nature and depend both on personnel costs and on systemic costs, and these may vary from function to function in the workflow. We can assume that fixed costs related to e.g. software licensing and maintenance are absorbed in each term C_i^A.

This means that analytical estimation of Eq. 1 in real cases can be very complex and expensive. Thus, to practically evaluate such trade-off conditions we consider the whole workflow as the function *W* to be evaluated and estimate total costs connected with each of the two workflow versions (automatic ($C_W{}^A$) + check ($C_W{}^{chk}$) and fully manual ($C_W{}^M$)):

$$C_W^A + C_W^{chk} < C_W^M \quad (2)$$

Developing Eq. 2 distinguishing systemic and personnel costs we obtain:

$$C_{Ws}^A + C_{Wp}^{chk} + C_{Ws}^{chk} < C_{Wp}^M + C_{Ws}^M \quad (3)$$

where we assume, for the sake of simplicity, that $C_{Wp}^A = 0$, i.e. that costs of personnel for the execution of the automatic tool are negligible.

It has to be noticed that all variables in Equations 1-3 have to be considered as random variables since the actions of the underlying workflow can be orchestrated in such a way that the execution of each of them is subject to some probabilistic occurrence of some conditions. This for example applies in those cases in which a parallel execution of a set of actions at the model level (e.g., expressed in BPMN) actually results in the instantiation of only a subset of the possible parallel actions, depending on some statistical properties of the input data at each running cycle of the workflow.

For these reasons, and since a general approach at analytical estimation is not possible, a possible way to study the impact of automatic tools on workflows passes through differential simulation, i.e. the comparison of two simulation models in which the first is the starting condition (i.e., a fully manual workflow) and the second is differing by the change of one or more functions with automatic counterparts.

To obtain meaningfully comparable results, however, an additional condition in the simulation has to be implemented, which is directly related to Eq. 3, namely that results of functions implemented by automatic tools have to be corrected till complying to what a manual function would produce. This part of the simulation is finally depending on the intrinsic performance (in terms of some standard metric as precision, recall, F-measure) of the tool implementing the function. If the tool has perfect performance then the terms C_i^{chk} of Eq. 3 have all value zero. As long as the performance decreases, accordingly those terms differ from zero in a monotonic way.

8.3 Cost simulation for segment-based annotations

We consider the common case of segment-based annotation (e.g., classification), which depends on the quality of an earlier segmentation and the component producing the annotation for the segment. We can express the probability of having wrong label classification for a segment of media, $Pr(\epsilon_o)$ as depending on the probability of correct (resp. wrong) segmentation $Pr(Cs)$ (resp. $Pr(Ws)$) and in turn decomposed in terms of the individual labels generated by the classifier:

$$Pr(\epsilon_o) = Pr(\epsilon_o|Cs)Pr(Cs) + Pr(\epsilon_o|Ws)Pr(Ws) = \\ Pr(Cs)\sum_{i=1}^{N_l} Pr(\epsilon_{o,l_i}|Cs)Pr(l_i) + \\ +Pr(Ws)\sum_{i=1}^{N_l} Pr(\epsilon_{o,l_i}|Ws)Pr(l_i) \quad (4)$$

where N_l is the number of labels which the classifier is able to assign and l_i denotes the $i-th$ label. $Pr(\epsilon_{o,l_i}|Cs)$ is the probability of wrong classification for label l_i when segmentation is correct while

$Pr(\epsilon_{o,l_i}|Ws)$ is the probability of wrong classification when the segmentation is wrong. We assume the latter to be equal to:

$$Pr(\epsilon_{o,l_i}|Ws) = 1 - (1 - Pr(\epsilon_{o,l_i}|Cs))\alpha_i(\Delta_s, \Delta_e) \quad (5)$$

where Δ_s is the segmentation mismatch (positive or negative) at the start of the segment, and Δ_e at the end, both expressed as fractions of the true segment. The function α_i accounts for the sensitivity of the classifier to the wrong segmentation for label i. Therefore it holds that $0 \leq \alpha_i(\Delta_s, \Delta_e) \leq 1$, whereas values near to 1 indicate more robustness of the classifier against segment mismatches than values near to 0. To identify the functional form and the fine-grained behaviour of such functions the approach is to use a numerical fitting technique on a small set of data produced by the classifier under varying segmentation error conditions.

After few algebraic substitutions using the fact that

$$Pr(Ws) = (1 - Pr(Cs)) \quad (6)$$

we have:

$$Pr(\epsilon_o) = Pr(C_s)\sum_{i=1}^{N_l} P\,r(l_i)[Pr(\epsilon_{o,l_i}|C_s)(1 - \alpha_i(\Delta_s, \Delta_e)) - (1 - \alpha_i(\Delta_s, \Delta_e))] \\ + \\ + \sum_{i=1}^{N_l} Pr(l_i)[(1 - \alpha_i(\Delta_s, \Delta_e)) + \alpha_i(\Delta_s, \Delta_e)Pr(\epsilon_{o,l_i}|C_s)] \quad (7)$$

which expresses the global probability of wrong classification under wrong segmentation conditions depending on the classification accuracy at each label level $Pr(\epsilon_{o,l_i}|Cs)$, the observed label probability of each label $Pr(l_i)$ and the global segmentation accuracy $Pr(Cs)$.

As a further step, we can decompose $Pr(\epsilon_{o,l_i}|Cs)$ taking into account the different classification performances a classifier may have for the different labels as follows: $Pr(\epsilon_o, l_i|Cs) = \sum_j Pr(\epsilon_{o,j}, l_i|Cs, l = j)Pr(l = j), j = 1..N_l$ with *l=j* being the ground truth label.

Now we want to calculate the probability of wrong classification of the ground truth segments $S = \{s_1, \dots, s_N\}$, based on the probability of wrong classification of the segments detected automatically $\Sigma = \{\sigma_1, \dots \sigma_M\}$.

In general, for a given true segment s_k we can have the following conditions:

- the true segment is perfectly corresponding to one detected segment
- the true segment corresponds to more than one consecutive detected segments (oversegmentation)
- the true segment corresponds to one larger segment which merges more than one true with other segments (undersegmentation)

Under these conditions, for each true segment s_k we can always associate a set of detected segments whose combination covers s_k and minimising start and end disalignments. Let's denote this set to be $\Sigma_k = \{\sigma_{k,1} \dots \sigma_{k,N_k}\}$.

Assuming a majority voting mechanism to select the most likely label among the ones associated to each of the elements of Σ_k, we can express the probability of wrong classification of s_k as:

$$Pr(\epsilon_{s_k}) = \sum_{i=\lceil N_k/2 \rceil}^{N_k} \frac{1}{\binom{N_k}{i}} \sum_{n=1}^{\binom{N_k}{i}} \prod_{j \in M(i,n)} Pr(\epsilon_{o,j}) \prod_{l \in m(i,n)} (1 - Pr(\epsilon_{o,l})) \quad (8)$$

where $M(i,n)$ is the $n-th$ possible subset of Σ_k of cardinality i and $m(i,n) = \Sigma_k \setminus M(i,n)$.

Now let's relate this model with costs. We can assume that costs related with the model are connected with the two main operations that are needed to transform the information coming from the automated segmentation and subsequent classification into the ground truth information by some human corrector.

This is composed by two elements: a) costs related to correcting segmentation; b) costs related to correct classification. The first kind of costs can be modelled, for each ground truth segment $s_k \in S$, as depending on the gap between Σ_k and s_k. A difference holds for cases of merged segments and of split segments. The first kind of error being in general heavier than the second since correcting false detections is in generally lighter than missed detections. Furthermore, we can assume that correcting split segments has a dependency only on the cardinality of Σ_k since the corrector should be able to decide about a false segmentation point just watching material before and after its boundary.

Thus, the average cost for correcting the automated segmentation Σ to be compliant with the ground truth S has two components Γ_m and Γ_s respectively modelling costs related to correcting merged and split segments.

$$\Gamma_m = \sum_{i=1}^{N_m} L(\Sigma_i) C_m \quad (9)$$

$$\Gamma_s = \sum_{i=1}^{N_s} \| \Sigma_i \| C_s \quad (10)$$

where N_m is the number of merged ground truth segments in S, N_s is the number of split segments in S, C_m is the unit cost (per unit of time) for correcting merged segments, C_s is the unit cost (per unit of split boundary) for correcting splits. Finally $L(\Sigma_i)$ is the length of segment coverage Σ_i of s_i.

Notice that both quantities expressed in Equations 9 and 10 yield 0 when detected segmentation is compliant to ground truth since both N_m and N_s would be 0 in that case.

The second kind of costs can be considered depending linearly on the length of the true segment s_k, since a corrector must analyse the whole segment to be really sure about the assigned ground truth classification.

Thus, costs related to correcting classification of a certain ground truth segment s_i, $\Gamma_{c,i}$, depends linearly on the length of the ground truth segment:

$$\Gamma_{c,i} = L(s_i) C_c \quad (11)$$

where $L(s_i)$ is the length of segment s_i and C_c is the unit cost (per unit of time) for correcting classification.

Since we want to evaluate an expected cost in probabilistic terms, we have to connect this to Equation 5, so that we have that the expected classification correction cost $\overline{\Gamma_c}$ is:

$$\overline{\Gamma_c} = \sum_{i=1}^{\|S\|} Pr(\epsilon_{s_i}) L(s_i) C_c \quad (12)$$

Similar considerations can be made to obtain expected segmentation correction costs starting from the probability of wrong segmentation $Pr(W_s)$, and considering the independent effect of merging and splitting.

We can finally reach the following constraint, which is compliant with the general model of automated vs. manual annotation introduced in Section 8.2:

$$\overline{\Gamma_s} + \overline{\Gamma_m} + \overline{\Gamma_c} < M_s + M_c \quad (13)$$

where M_c and M_s are the costs related with the manual operations respectively related to segmentation and classification.

8.4 Costs for correcting segment-based annotations

For an analysis chain that performs temporal segmentation and classifies segments, we need to consider four types of edits:

- delete annotation (D)
- add annotation (A)
- change segment boundaries (Sh)
- substitute content of annotation (S)

For tasks with a defined set of annotation values (e.g., classification), D, A, and Sh can be treated separately for each class. S may link several classes, as it consists of D for class 1 and A for class 2.

The costs for each of these edits are structured differently. There is the cost for inspecting the video segment for each reported segment, i.e. true and false positives. Depending on the frequency of annotations in the video, it may be more efficient to go through reported events by confidence (if their total duration is clearly smaller than the duration of the video) or linearly through the video (if the duration of annotations is close to the duration of the video). In addition to the inspection time, the cost for D is the interaction time needed for deleting an annotation, for Sh the time for adjusting the segment boundaries and for S the time for selecting the new content/modifying the content.

The situation is different for A. As soon as only a single item is to be added, the time for looking through the whole video has to be added once. Note that this does not necessarily mean linear time of the video, depending on the available tools and type of annotation to be made this step may use fast forward or navigation using key frames. In addition, A requires the time for creating the annotation, i.e. selecting/entering the content; this will depend on the number of classes/values to select from, or if free text needs to be entered, and possibly time for setting the segment boundaries of the segment to be labelled.

For comparing with manual annotation, the frequency and distribution of segments are deciding factors, i.e., several annotations in parallel will require to spend more time with a segment, while only one annotation at a time will allow to progress closer to real-time. The costs for a manual annotation can be determined by requiring edits of type A for every segment in the ground truth.

8.5 Example 1: Person identification

We have selected a task model for “identification of persons in news material” and performed a simulation analysis using the general cost model introduced in Section 8.2 and under the following conditions. (1) We assume that the function related to manual person identification could be substituted by a combination of automatic analyses: identification by face, by speaker’s voice, or by open caption/graphics. (2) We implement the whole workflow in a simulation tool in the two versions (manual and automatic + check). (3) We run the simulation of the manual process. (4) We run a series of simulations of the automatic process, under a range of values for the F-measure of the automatic tools. (5) We compare results and find a trade-off condition. As shown in Figure 38, the proposed approach can be used to relate the performance of an automatic content analysis tools to the cost saving in a specific process. Individual cost items (in Euros) have been estimated based on the knowledge of typical labour anf system-related costs observable in media production enterprises. Based on measured data from actual process, this methodology can help to assess whether the performance of a specific automatic content analysis tool is good enough for a specific practical application.

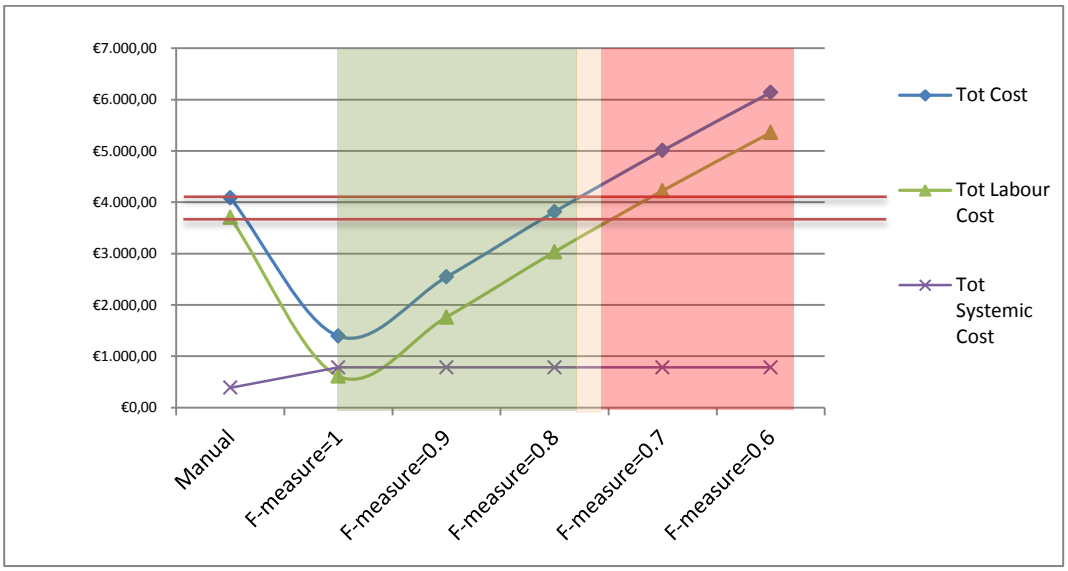

Figure 38. Simulation results for person identification.

Figure 38 reports the variation of the total cost of the process w.r.t. the F-measure of the three detectors (face-based, speaker-based and OCR-based, we have considered that all of them share the same performance). It can be observed that if F-measure is above 0.8 both total costs and labour costs of the automatic version of the workflow are less than the manual counterpart (the first column of the graph). Conversely, if F-measure drops towards 0.7 both costs are higher than the manual ones. In the middle, there is a region of F-measure values for which at least the labour costs are lower than in the manual case. These are of course influenced by the manual check costs.

8.6 Example 2: Video breakup detection

We apply the cost simulation to the video breakup detection task described in Section 7.2 using the analytical cost model introduced in Section 8.3. Based on the error probabilities of the shot boundary detection (playing the role of segmentation) and video breakup detector (playing the role of the classifier), the cost simulator assumes a workflow with a manual verification step after this automatic part of the process. In the verification step, a user corrects segmentation errors (splits, merges) as well as classification errors (setting the correct label for a segment). Missed video breakups are modelled as merged segments and false positive detections as split segments. In case of a mismatch of the segmentation, the cost simulation tool uses majority voting for the classification, thus both false positives and false negatives will result in classification errors. The case of a classification error despite correct segmentation is quite rare for short defects like video breakups. Some correction costs depend on duration of the involved segments (e.g., reviewing a merged segment to split it correctly) while others do not (e.g., deleting a split). We use the same data set as in the previous section, and the error probabilities obtained from actual runs of the algorithm. The unit costs were set as follows (with a time unit of 0.1 seconds for duration dependent costs): correct split 10, correct merge 100 and correct classification label 10. Merges require reviewing the entire segment, and are thus most costly. We used the following expression for the function $\alpha_i(\Delta_s, \Delta_e)$ introduced in Equation 8:

$$\forall i \; \alpha_i(\Delta_s, \Delta_e) = \frac{2}{1 + e^{A(|\Delta_s + \Delta_e|)}}$$

where A is a parameter calculated to fit the average behaviour of the classifier under wrong segmentation conditions.

Figure 39 shows the result of 100 runs of the cost simulation, varying both segmentation and classification performance. The costs are shown depending on the classification accuracy obtained downstream the segmentation correction, so to take into account the effect of a varying segmentation performance on the various kinds of costs. This variation may for example be related to the application of the same segmentation algorithm to a different content genre than the one used to train the segmenter or to the application of a different configuration of the same segmenter, or to the usage of a range of different segmenters with varying performance. This has been simulated by modelling a

segmenter performance in terms of probability of over- and under-segmenation and of the corresponding average number of respectively split and merged segments.

We see that the segmentation correction costs are dominant, and only get down to the same range as classification costs towards 0.94 accuracy. This threshold may be for example useful to filter out segmenters or segmenter configurations among the one simulated.

Of course, the obtainable cost figure may depend on the organisation and the staff at hand. A parallel simulation yielded to the result that using the same cost assumptions as for the corrections, viewing the entire content and making annotations for the video breakups would result in costs in the range of 2.9e7 units, meaning that in the studied case the automatic tool remain quite profitable even under stressful segmentation error conditions.

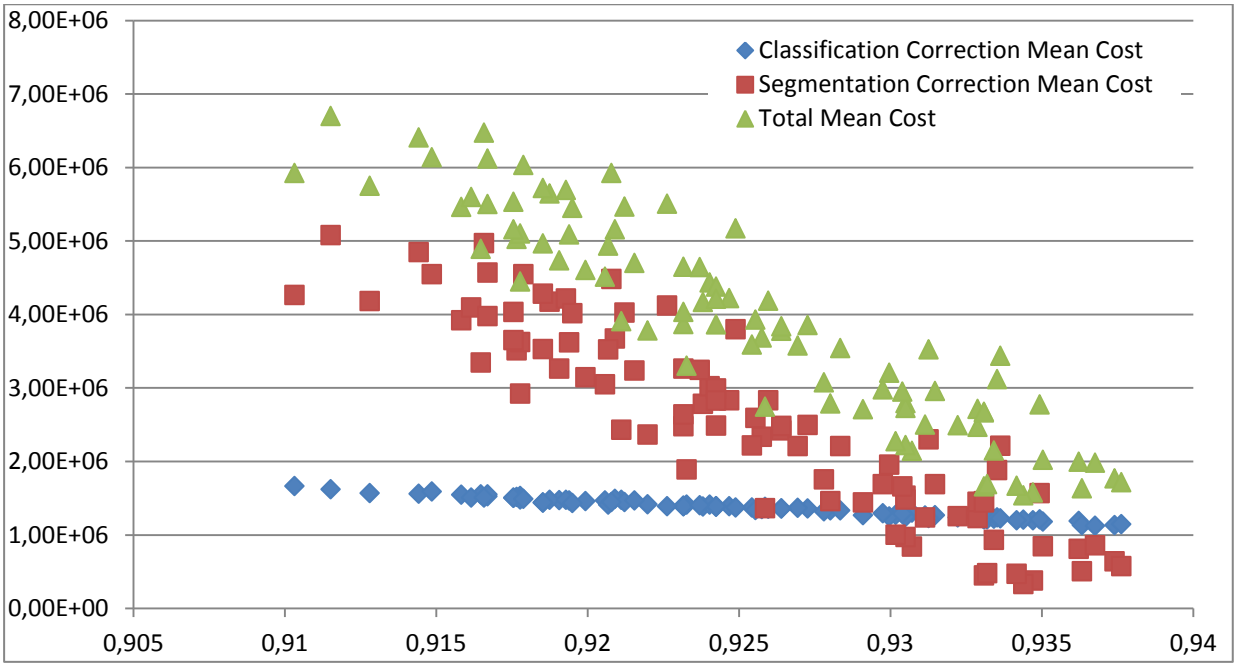

Figure 39: Simulated costs depending on classification accuracy.

8.7 Conclusions

This part of the work has demonstrated that performance of individual tools can be difficultly put into direct relationship with advantages in terms of costs implied by the overall workflows in which they are included. Specifically, the intrinsic imperfect performance of these tools introduces the need to have manual check points in the automated workflows, which are normally not needed in fully manual workflows. Simulating these conditions in a differential way allows however to develop methodologies to assess what thresholds in tools' performance have to be requested in order to improve cost figures.

The drawback of such an approach resides in the fact that it is normally quite complex to model real workflows in practice, specifically to derive coherent figures in terms of execution times and costs and in terms of a complete breakdown analysis of actions into individual functions. The recommendation is to develop this kind of approaches as part of a general methodology that all industrial processes should follow (e.g., as that defined in processes and methods engineering), so that simulation parameters can be estimated along with production process design and aligned and optimised during operations.

9 Integration

This section discusses the integration of the proposed benchmarking and simulation methods into systems, in particular, a service oriented architecture such as the one developed in TOSCA-MP.

9.1 Benchmarking workflow

The benchmarking workflow consists of the following steps:

- Generate a set of jobs for a content basket, possibly with varying parameters
- Run these jobs and generating results
- Determine the differences between results and a reference (e.g., ground truth), resulting in a set of edits for each pair of result and reference (cf. Section 4.4)
- Perform one or more of the following
  - Determine performance measures from the edits
  - Train a model of the error dependencies and run error prediction (cf. Section 5) or cost simulation (cf. Section 8).
- Based on the results, select parameters and create/update a workflow configuration with these parameters

Figure 40 provides an overview of this workflow. The steps in the dashed box are implemented in the benchmarking service itself. Preparing jobs and displaying the benchmarking results is implemented in the control & configuration UI. Job execution is done by the MPMF. Details on the implementation of the service and the control & configuration UI can be found in D6.6.

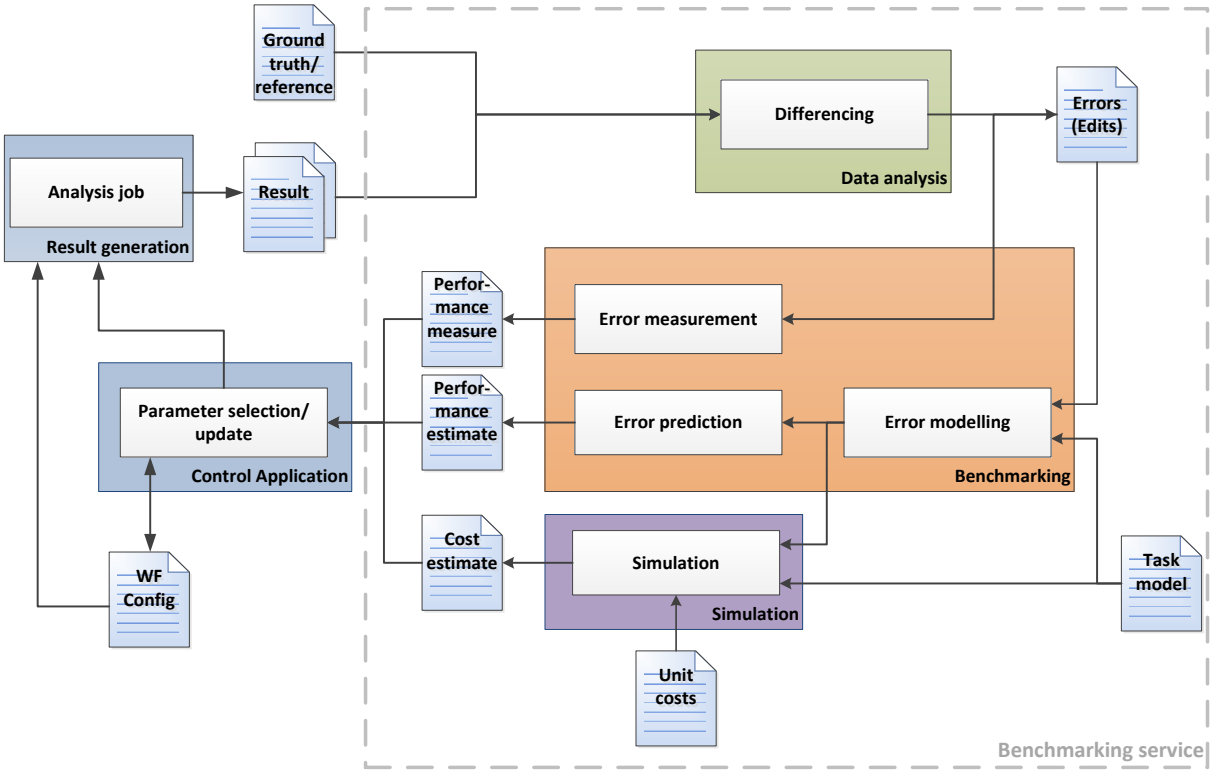

Figure 40: Integrated of benchmarking workflow.

9.2 Configuration

9.2.1 Bayesian network

As described in Section 5, the structural information for the Bayesian network is available in the task model. This step of the process could thus be automated (e.g., by generating code needed to construct the Bayesian network using a toolkit of choice). However, the following information is not included in the current implementation of the task model:

- The types of data being exchanged between two tasks
- The edits to this data that are of interest

These two annotations are added to the task model, CTT provides support for this.

In addition, there is other information which is typically not available when starting with the model from scratch, but needs analysis of actual data.

- Statistical information about the error data in order to choose the appropriate model for the nodes
- Statistics about the relation between input errors, configuration and content properties and output errors, in order to decide about the need for hidden layers in the network design.

The latter is especially tricky, as it may require starting with an initial model, and then refining it. In the experiments performed, hidden layers turned out not to provide any advantage.

9.2.2 Simulation model

The model used for the simulation is represented in BPMN. As described in D4.3, the task model represented in CTT XML can be automatically transferred to BPMN using an XSL style sheet. Thus, this process does in theory not involve any manual interaction. However, in practice, incompatibilities between different tools implementing the BPMN standard have been encountered.

9.3 Gathering data

Apart from the task model, it is of course necessary to gather as much data as possible in order to assess components and run simulations. While ground truth can be generated offline in limited amounts, much larger amounts of data can be gathered from a running system. This data includes in particular

- Collecting all intermediate data of all automatic processing in the system,
- Logging all user changes in verification tools, including time stamps,
- Logging all user annotations, including time stamps, and
- Logging all user interactions in search tools, including time stamps.

In the first three cases, one result of the action is a new or updated metadata document. Storing this document in the Distributed Repository Framework (DRF), which supports versioning, allows tracing back the history of the metadata document and extracting the modification done in every step.

In addition, for assessing the effort needed for certain manual interventions and for cost simulations, user actions and their time stamps need to be accessed. However, this is not done on the fly, but the unit costs are configured based on previous measurements.

9.4 Benchmarking service functionalities

In order to run benchmarking and simulation as part of the workflow (assuming the networks/models need have been created), we implemented the following functionalities in the service:

- Extract the set of edits to the difference of two metadata documents (using the approach described in Section 4.4.2)
- Modelling error dependencies, i.e., training the Bayesian network
- Running predictions and/or cost simulations
- Derive cumulative measures and statistics

A use case for a benchmarking service, including the functionalities described above as well the option to update configurations, has been proposed the EBU/AMWA FIMS RfT phase 2. The text of this proposal can be found in Annex A.

10 Conclusion

In this document we have described task-based approaches to benchmarking and simulation, and we have provided experimental results for both content analysis and search/result visualisation.

Several of the experiments confirm the expectation, that independent metrics for content analysis components are not always suitable in order to assess the performance of the overall analysis or search process. Thus, these tools need to be analysed under consideration of the task context. We have proposed a model in order to perform prediction of performance based on propagation of errors throughout a processing chain.

In order to assess the practical value of a specific content analysis tool, it is important to represent an assessment in terms of cost saving in a workflow. The cost modelling and simulation tools allow assigning costs to manual verification and correction of imperfect automatic analysis results, and thus to compare total workflow costs of tools with different performance levels as well as with manual annotation.

11 References

[Aly, 2013] R. Aly, M. Eskevich, R. Ordelman, and G. J. Jones, Adapting binary information retrieval evaluation metrics for segment-based retrieval tasks, Technical Report 1312.1913, ArXiv e-prints, 2013.

[Bai, 2005] C. G. Bai, Q. P. Hu, M. Xie, and S. H. Ng. “Software failure prediction based on a Markov Bayesian network model,” J. Syst. Softw. (74)3: 275-282, 2005.

[Bailer, 2009] W. Bailer, F. Lee and G. Thallinger, “A Distance Measure for Repeated Takes of One Scene,” The Visual Computer, vol. 25, no. 1, pp. 53-68, Jan. 2009.

[Bailer, 2009b] Werner Bailer and Herwig Rehatschek, “Comparing Fact Finding Tasks and User Survey for Evaluating a Video Browsing Tool,” in Proceedings of ACM Multimedia, Beijing, CN, Oct. 2009.

[Belkin, 1995] Belkin, N. J., Cool, C., Stein, A. & Thiel, U. (1995). “Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems,” Expert Systems With Applications, 9(3), 379-395.

[Belkin, 2000] Nicolas J. Belkin, “Helping people find what they don't know,” Commun. ACM, 43, 8 (August 2000), 58-61.

[Chatfield, 2011] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, “The devil is in the details: an evaluation of recent feature encoding method,” In British Machine Vision Conference (BMVC 2011), Dundee, United Kingdom, 2011

[Chatzichristofis, 2008] S. Chatzichristofis, Y. Boutalis: "CEDD: Color and Edge Directivity Descriptor. A Compact Descriptor for Image Indexing and Retrieval", Computer Vision Systems, Lecture Notes in Computer Science Volume 5008, 2008, pp 312-322.

[Chen, 2006] Harr Chen and David R. Karger, “Less is more: probabilistic models for retrieving fewer relevant documents,” In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '06).

[Cinbis, 2011] R. G. Cinbis, J. Verbeek, and C. Schmid, “Unsupervised Metric Learning for Face Identication in TV Video,” In Proceedings of ICCV 2011, Barcelona, Spain, 2011.

[Clarke, 2008] Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon, “Novelty and diversity in information retrieval evaluation,” In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '08).

[DeVries, 2004] De Vries, Arjen P., Gabriella Kazai, and Mounia Lalmas, "Tolerance to irrelevance: A user-effort oriented evaluation of retrieval systems without predefined retrieval unit," RIAO 2004 conference proceedings. 2004.

[Eskevich, 2013] M. Eskevich, G. J. Jones, S. Chen, R. Aly, and R. Ordelman, “The Search and Hyperlinking Task at MediaEval 2013,” In MediaEval Workshop, Barcelona, Spain, October 18-19 2013.

[Fassold, 2012] Hannes Fassold, Stefanie Wechtitsch, Albert Hofmann, Werner Bailer, Peter Schallauer, Roberto Borgotallo, Alberto Messina, Mohan Liu, Patrick Ndjiki-Nya and Peter Altendorf, “Automated Visual Quality Analysis for Media Production,” in IEEE International Symposium on Multimedia, Irvine, CA, USA, Dec. 2012, pp. 385-394.

[Gauvain, 2002] J.-L. Gauvain, L. Lamel, and G. Adda, “The LIMSI Broadcast News transcription system,” Speech Communication, 37(1-2):89-108, 2002.

[Gauvain, 2010] J.-L. Gauvain, “The Quaero Program: Multilingual and Multimedia Technologies,” International Workshop on Spoken Language Translation (IWSLT 2010), 2012

[Gunawardana, 2009] Asela Gunawardana and Guy Shani, “A Survey of Accuracy Evaluation Metrics of Recommendation Tasks,” J. Mach. Learn. Res. 10 (December 2009), 2935-2962.

[He, 2011] X. He, L. Deng, and A. Acero, “Why Word Error Rate is not a Good Metric for Speech Recognizer Training for the Speech Translation Task?”, in Proc. ICASSP, IEEE, May 2011.

[Kasturi, 2009] Kasturi, R.; Goldgof, D.; Soundararajan, P.; Manohar, V.; Garofolo, J.; Bowers, R.; Boonstra, M.; Korzhova, V.; Jing Zhang, "Framework for Performance Evaluation of Face, Text, and Vehicle Detection and Tracking in Video: Data, Metrics, and Protocol," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.31, no.2, pp.319,336, Feb. 2009

[Li, 2010] Yingbo Li and Bernard Mérialdo, “VERT: automatic evaluation of video summaries,” In Proceedings of ACM Multimedia, 2010.

[Lokaj, 2013] Michal Lokaj, Harald Stiegler and Werner Bailer, “TOSCA-MP at Search and Hyperlinking of Television Content Task,” in Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop, Barcelona, ES, Oct. 2013.

[Massoudi, 2006] A. Massoudi, F. Lefebvre, C. Demarty, L. Oisel, and B. Chupeau, “A video fingerprint based on visual digest and local fingerprints,” In International Conference on Image Processing (ICIP 2006), 2006

[Mouat, 2002] A. Mouat, Delta Update Language, 2002. http://sourceforge.net/projects/diffxml/files/documentation/1.0/dul.ps

[Murphy, 2001] K. Murphy, “The Bayes net toolbox for Matlab,” Comput. Sci. Stat. vol. 33. 331-350, 2001. https://code.google.com/p/bnt/

[Oohori, 2007] Takahumi Oohori, Hidenori Naganuma, and Kazuhisa Watanabe, “A New Backpropagation Learning Algorithm for Layered Neural Networks with Nondifferentiable Units,” Neural Comput. (19)5:1422-1435, 2007.

[Over, 2008] Paul Over, Alan F. Smeaton, and George Awad, “The TRECVID 2008 BBC rushes summarization evaluation,” In Proceedings of the 2nd ACM TRECVid Video Summarization Workshop (TVS '08), 2008.

[Papineni, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, “BLEU: a method for automatic evaluation of machine translation,” In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, 2002.

[Park, 2008] Y. Park, S. Patwardhan, K. Visweswariah and S. C. Gates, “An Empirical Analysis of Word Error Rate and Keyword Error Rate,” In Proceedings of the International Conference on Spoken Language Processing, Brisbane, Australia, 2008.

[Piazolla, 2010] Pietro Piazzolla, Marco Gribaudo, Roberto Borgotallo, and Alberto Messina, “Performance evaluation of media segmentation heuristics using non-Markovian multi-class arrival processes”. In Proceedings of the 17th international conference on Analytical and stochastic modeling techniques and applications (ASMTA'10), 2010, pp. 218-232.

[Radlinski, 2009] Filip Radlinski, Paul N. Bennett, Ben Carterette, and Thorsten Joachims, “ Redundancy, diversity and interdependent document relevance,” SIGIR Forum 43, 2 (December 2009), 46-52.

[Riccardi, 1998] G. Riccardi, A. L. and Gorin, ”Stochastic language models for speech recognition and understanding,“ Proc. ICSLP, Sydney, Nov. 1998.

[Robie, 2011] J. Robie, D. Chamberlin, M. Dyck, D. Florescu, J. Melton, J. Siméon (eds.), XQuery Update Facility 1.0, W3C Recommendation 17 March 2011. http://www.w3.org/TR/xquery-update-10/

[Rousseau, 2011] A. Rousseau, F. Bougares, P. Deléglise, H. Schwenk, and Y. Estév, “LIUM's systems for the IWSLT 2011 Speech Translation Tasks,” In Proceedings of IWSLT 2011, San Francisco, USA, 2011.

[Sakai, 2007] Tetsuya Sakai, “On the reliability of information retrieval metrics based on graded relevance,” Inf. Process. Manage. 43, 2 (March 2007), 531-548.

[Sidiropoulos, 2012] Sidiropoulos, P., Mezaris, V., Kompatsiaris, I. and Kittler, J., “Differential Edit Distance: A Metric for Scene Segmentation Evaluation”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, nr. 6, pp. 904-914, 2012.

[Thomas, 2006] Paul Thomas and David Hawking,” Evaluation by comparing result sets in context,” In Proceedings of the 15th ACM international conference on Information and knowledge management (CIKM '06).

[Valdes, 2012] Victor Valdes and Jose M. Martinez, “Automatic evaluation of video summaries,” ACM Trans. Multimedia Comput. Commun. Appl. 8(33), August 2012.

[Vlachos, 2002] Vlachos, M., Kollios, G., Gunopoulos, D., “Discovering similar multidimensional trajectories,” Proceedings of the 18th International Conference on Data Engineering, pp. 673–684. IEEE Comput. Soc., San Jose (2002)

[Wang, 2003] Y.-Y. Wang, A. Acero, and C. Chelba, “Is word error rate a good indicator for spoken language understanding accuracy,” in Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), 2003, pp. 577–582.

[Wilson, 2009] Wilson, Max L., m.c. schraefel and Ryen W. White, "Evaluating advanced search interfaces using established information-seeking models," Journal of the American Society for Information Science and Technology 60.7 (2009): 1407-1422.

[Zhai, 2003] Cheng Xiang Zhai, William W. Cohen, and John Lafferty, “Beyond independent relevance: methods and evaluation metrics for subtopic retrieval,” In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '03).

12 Glossary

Terms used within the TOSCA-MP project, sorted alphabetically.

see D1.2 project handbook, version 3 or later

BN	Bayesian Network
BPMN	Business Process Model and Notation
CTT	ConcurTaskTree
CTTE	ConcurTaskTree Environment
EM	Expectation-Maximisation algorithm
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

Partner Acronyms

DTO	Technicolor, DE
EBU	European Broadcasting Union, CH
FBK	Fondazione Bruno Kessler, FBK
HHI	Heinrich Hertz Institut, Fraunhofer Gesellschaft zur Förderung der Angewandten Forschung e.V., DE
IRT	Institut für Rundfunktechnik GmbH, DE
K.U.Leuven	Katholieke Universiteit Leuven, BE
JRS	JOANNEUM RESEARCH Forschungsgesellschaft mbH, AT
PLY	Playence S.L., ES
RAI	Radiotelevisione Italiana S.p.a., IT
VRT	De Vlaamse Radio en Televisieomroeporganisatie NV, BE

13 Annex A: Submission on a benchmarking use case to EBU/AMWA FIMS RfT phase 2

This annex contains the text of a use case for a benchmarking service submitted to the EBU/AMWA FIMS phase RfT (Jan. 2012).

Summary:	*Services for automated metadata extraction (e.g., automatic speech recognition), linking of metadata (e.g., associating named entities with authority files and between media items) and advanced search services (e.g., using semantic technologies) are becoming increasingly important in media production processes. However, assessing their performance for supporting a particular task in the process is important for leveraging the potential of these technologies. This use case complements the AIEMPro[2] and search use cases by describing benchmarking services that can be used to assess and optimise these services and their configuration.*
Description:	*The performance of automatic metadata extraction and advanced search services may strongly vary depending on content properties, such as genre, resolution, spoken language etc. Thus benchmarking (i.e., checking the quality of metadata/results generated against a validated reference) such a service for a new type of content in order to (i) assess whether it is applicable, (ii) select an appropriate configuration and (iii) build a tailored process for a certain type of content is relevant for using such services in a practical workflow.* *A practical workflow might include a range of metadata extraction and search services from different vendors, and even more than one service instance for one type of operation, as they might be complementary in terms of different content types. Thus a standardised, service oriented approach to benchmarking enables efficient and integrated evaluation of metadata extraction and search services.* *A benchmarking process for a metadata extraction or search service (called "service to be evaluated" in the following) produces evaluation results and possibly updated parameters based on metadata generated by this service and reference metadata ("ground truth"). In more detail, the process can be described as follows:* - *A set of metadata documents or search results for a defined set of media items, generated by the service to be evaluated, needs to be checked. In addition, the configuration used to obtain the result (parameters, queries, etc.) is available.* - *For (a subset of) these media items, ground truth, i.e., manually annotated/validated metadata, relevant search results, etc., are available.* - *Additionally, data for weighting the importance of specific aspects of the result or types of errors might be given (e.g., penalty for missing a named entity vs. missing a verb in the ASR transcript, cost of reporting an irrelevant vs. missing a relevant result in copy detection or search).* - *Based on these inputs, the benchmarking service produces evaluation results in terms of one or more metrics implemented by the benchmarking service, represented as metadata documents. Possibly, the service also outputs visualisations of these results.* - *Optionally, the benchmarking service might output a modified/optimised set of parameters for the service being evaluated. This typically requires either a dependency between the benchmarking service and the service being evaluated or a standardised set of parameters for a certain type of metadata extraction/search task.*

	- *If the service being evaluated performs analysis on collection level, the same considerations about collections as described in the AIMEPro[2] use case apply.*
Initiating Actor:	*Archive Administrators, Automation*
Supporting Actors:	*Application Monitoring Staff, Software Specialists, R&D Staff*
Inputs:	*Content flows* - *one or more XML documents describing results of automatic metadata extraction, linking or search services for a set of media items, conforming to relevant metadata standards (e.g, ISO/IEC MPEG-7 Audiovisual Description Profile, EBU Core)* - *one or more XML documents carrying ground truth data, conforming to relevant metadata standards (e.g, ISO/IEC MPEG-7 Audiovisual Description Profile, EBU Core)* - *one or more XML documents carrying configuration information and parameters of these services* - *zero or more XML documents containing weighting information for assessing the results* - *optionally a reference to the media files on which the results are based*
	Information flows - *a set of or a reference to a collection of automatically generated metadata documents or result sets* - *a set of or a reference to a collection of ground truth documents* - *a set of configuration parameters of the services being evaluated* - *a set of parameters for the benchmarking service* - *optionally, a reference to a set of multimedia content items stored in a media file already ingested in the system or to be ingested contextually*
	Control flows - *Synchronous status monitoring at task instance level, including exceptions, management of retries, fallouts, termination of tasks and task model instances* - *Dynamic prioritisation of task instances* - *Basic access and browsing of produced metadata (e.g., HTTP access to XML metadata documents being produced, access to plots of result statistics)*
Outputs:	*Content Outputs* - *A set of (XML) documents containing the results of the benchmarking task in terms of appropriate metrics* - *A set of textual files containing detailed logs of each individual task execution*

	- *Optionally, a set of modified/optimised configuration parameters for the service being evaluated*
	Information Outputs - *A (reference to) a set of benchmarking results* - *A (reference to) a (portion of) logging information where relevant* - *Optionally, a (reference to a) set of configuration parameters*
Pre-conditions:	*An automated metadata extraction, linking and search service has been performed on a set of media items in the system and the results are available in the system as XML documents.* *Manually generated or automatically generated and manually validated ground truth is available on part of the content metadata extraction has been performed for.*
Post-conditions:	*Absence of severe or unmanaged exceptions, actual execution of tasks, presence of output metadata instances, syntactic and semantic conformance of the metadata instances to the intended schemas.*
Non-functional requirements:	*Execution and memory/storage resource allocation should be pre-estimated and communicated in some form to the caller in advance. Estimated completion time should be provided. Progress reporting and error notification should be supported by the service.*
Default flow:	

1. *The process initiator selects the set of automated processing results, the set of ground truth documents and configuration parameters for processing by a benchmarking service*
2. *The process initiator asks for the execution of a task instance t on the collection C or on the individual item I*
3. *The system estimates needed resources, and completion time for the execution of t and communicates these data to the process initiator.*
4. *If the initiator agrees on the provided estimation, the task instance is internally scheduled for activation*
5. *t gets activated, and the process initiator is notified of the activation*
6. *The initiator can optionally decide to postpone the actual start of execution or to manage prioritisation of the pool of activated task instances*
7. *t gets executed, and the process initiator is notified of the start of execution*
8. *The initiator can monitor the execution of t and browse the available benchmarking results as soon as they are completed*
9. *when the execution is complete, the system notifies the initiator and communicates coordinates of the results. These may include information for downloading files from network repositories or data from databases.*
10. *the initiator accesses the results, optionally verifies and updates proposed configuration parameters and feeds back the updates to the related automated metadata extraction/search component(s)*

Exception Handling:
- *if resource estimation fails system-level predefined boundaries (e.g., maximum execution time, maximum allocated resources per task model instance), the initiator must be notified immediately* - *if the initiator rejects the resource and completion time estimation, there should be a negotiation phase* - *if part of the internal transfers for ensuring execution-near-data fail, the system should re-estimate execution times and notify these back to the initiator.* - *If any task-level unmanaged exception occurs during the execution of the task instance, the initiator must be immediately notified. The initiator can decide to retry the task, or to terminate its execution* - *If the results are not accessible, the initiator must be able to notify the event to the system*

