D4.3 Opinion Mining v2

Deliverable Co-ordinator: Diana Maynard
Deliverable Co-ordinating Institution: USFD

Other Authors: Jonathon Hare (SOTON), Gerhard Gossen (LUH), David Dupplaw (SOTON), Adam Funk (USFD), Marco Fisichella (LUH)
(The list of contributors is not exhaustive.)

This deliverable describes the approaches and results concerned with the detection and dynamics of opinions from text, images and multimedia in WP4.
ARCOMEM Consortium

This document is a part of the ARCOMEM research project funded by the ICT Programme of the Commission of the European Communities by the grant number ICT-2009-270239. The following partners are involved in the project:

<table>
<thead>
<tr>
<th>The University of Sheffield (USFD) – Coordinator Department of Computer Science Regent Court 211 Portobello Sheffield, S1 4DP United Kingdom Contact person: Hamish Cunningham, Wim Peters E-mail address: <a href="mailto:ARCOMEM-coord@lists.dcs.shef.ac.uk">ARCOMEM-coord@lists.dcs.shef.ac.uk</a></th>
<th>Leibniz Universität Hannover (LUH) Forschungszentrum L3S Appelstrasse 9a 30169 Hannover Germany Contact person: Thomas Risse E-mail address: <a href="mailto:risse@L3S.de">risse@L3S.de</a></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Yahoo Iberia SLU (YIS)</strong> Avinguda Diagonal 177, 8th floor, Barcelona, 08018, CAT, Spain Contact person: Alejandro Jaimes E-mail address: <a href="mailto:ajaimes@yahoo-inc.com">ajaimes@yahoo-inc.com</a></td>
<td><strong>Internet Memory Foundation (EA)</strong> 45 ter rue de la revolution 93100 Montreuil France Contact person: Julien Masanes E-mail address: <a href="mailto:julien@internetmemory.org">julien@internetmemory.org</a></td>
</tr>
<tr>
<td><strong>University of Southampton (SOTON)</strong> Room 4011, Building 32 Highfield campus University of Southampton SO17 1BJ Contact person: Paul Lewis E-mail address: <a href="mailto:phl@ecs.soton.ac.uk">phl@ecs.soton.ac.uk</a></td>
<td><strong>Athens Technology Center (ATC)</strong> 10, Rizariou Street 15233, Halandri Athens, Greece Contact person: Dimitris Spiliotopoulos E-mail address: <a href="mailto:d.spiliotopoulos@atc.gr">d.spiliotopoulos@atc.gr</a></td>
</tr>
<tr>
<td><strong>ATHENA Research and Innovation Center in Information Communication &amp; Knowledge Technologies (ATHENA)</strong> National Technical University of Athens School of Electrical and Computer Engineering Division of Computer Science Iroon Polytechniou 9 Athens, 15780 Greece Contact person: Nectarios Koziris E-mail address: <a href="mailto:nkoziris@imis.athena-innovation.gr">nkoziris@imis.athena-innovation.gr</a></td>
<td><strong>Télécom ParisTech (IT)</strong> 46 rue Barrault 75634 Paris Cedex 13 France Contact person: Pierre Senellart E-mail address: <a href="mailto:pierre.senellart@telecom-paristech.fr">pierre.senellart@telecom-paristech.fr</a></td>
</tr>
<tr>
<td><strong>Deutsch Welle (DW)</strong> Neue Medien / Distribution Voltastr. 6 13355 Berlin, Germany Contact person: Birgit Gray E-mail address: <a href="mailto:birgit.gray@dw-world.de">birgit.gray@dw-world.de</a></td>
<td><strong>SUDWESTRUNDFUNK (SWR)</strong> Hans-Bredow-Strasse, D-76522 Baden-Baden Germany Contact person: Robert Fischer E-mail address: <a href="mailto:robert.fischer@swr.de">robert.fischer@swr.de</a></td>
</tr>
<tr>
<td><strong>HELLENIC PARLIAMENT (HEP)</strong> Amalias 14, 10557, Athens Greece Contact person: Dimitris Koryzis E-mail address: <a href="mailto:dkoryzis@parliament.gr">dkoryzis@parliament.gr</a></td>
<td><strong>PARLAMENTSDIREKTION (AUP)</strong> Dr. Karl Renner-Ring 3 A-1017 Vienna Contact person: Guenther Schefbeck E-mail address: <a href="mailto:guenther.schefbeck@parlament.gv.at">guenther.schefbeck@parlament.gv.at</a></td>
</tr>
</tbody>
</table>
Work Package Participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to the writing of this document or its parts:

1. University of Sheffield (USFD)
2. Leibniz Universität Hannover (LUH)
3. University of Southampton (SOTON)

Change Log

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Amended by</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>01/04/12</td>
<td>Diana Maynard</td>
<td>Initial outline</td>
</tr>
<tr>
<td>0.2</td>
<td>10/06/12</td>
<td>Diana Maynard</td>
<td>Contributions added from LUH and SOTON</td>
</tr>
</tbody>
</table>

Executive Summary

This deliverable describes the approaches and results concerned with the extraction and dynamics of opinions within WP4, deploying text mining, NLP, image analysis and semantic data integration technologies. We focus on three main types of media: textual documents, images and multimedia. We discuss the main challenges arising from the ARCOMEM requirements, and show how we attempt to tackle them. For each task, we describe the methodology used, the links with related work in other WPs and projects, and the first experimental results. Finally, we give some outlook about the remaining steps to be undertaken, areas for improvement and prospects for the future.
Table of Contents

1 Introduction..................................................................................................................8

2 Opinion Identification and Classification from Text.........................................................9
   2.1 Sentiment Analysis Application for English..............................................................11
   2.2 RDF generation........................................................................................................13
   2.3 Opinion Aggregation and Summarisation.................................................................16
   2.4 Detecting opinions in comments .............................................................................17
   2.5 Sarcasm detection.....................................................................................................21
   2.6 Opinion Aggregation and Summarisation.................................................................27
   2.7 RDF generation........................................................................................................28
   2.8 Other improvements.................................................................................................29
   2.9 Machine Learning applications for Opinion Mining................................................29
   2.9.1 SentiWordNet features........................................................................................31
   2.9.2 Linguistic and Semantic Features.........................................................................31
   2.9.3 ML Applications....................................................................................................32
   2.10 Evaluation................................................................................................................32
       2.10.1 Customer reviews ..........................................................................................32
       2.10.2 US Election dataset tweets ............................................................................34
       2.10.3 General tweets .............................................................................................34
   2.11 Ongoing work..........................................................................................................35
   2.12 Discussion and Future Work....................................................................................36

3 Opinion Event Detection in the Social Web....................................................................38
   3.1 Related Work............................................................................................................38
   3.2 Opinion Event Detection System.............................................................................39
   3.3 Opinion Timeline Extraction....................................................................................40
   3.4 Opinion Event Detection Algorithm.........................................................................40
   3.5 Opinion Event Description.......................................................................................41
   3.6 Exploration and Evaluation Tool .............................................................................42
   3.7 Evaluation................................................................................................................44
   3.8 Evaluation................................................................................................................46
   3.9 Integration into the ARCOMEM system...................................................................47
   3.10 Future Work............................................................................................................48

4 Image-opinion Identification and Classification..............................................................49
   4.1 Facial Expression Recognition...................................................................................49
   4.2 Image Reuse Analysis...............................................................................................51
   4.3 Future plans and challenges......................................................................................53

5 Multimodal Opinion analysis (SOTON, LUH).................................................................54
   5.1 Attractiveness classification of images......................................................................54
       5.1.1 NicePic!............................................................................................................54
       5.1.2 System Architecture.........................................................................................55
       5.1.3 Future plans and integration with the ARCOMEM system...............................56
   5.2 Privacy classification of images................................................................................57
   5.3 Sentiment classification of images............................................................................58
   5.4 Evaluation................................................................................................................58
List of Tables
Table 1: Examples of opinionated sentences................................................................. 19
Table 2: Aggregated scores for entities in the document.............................................. 20
Table 3: Aggregated score for the document............................................................... 20
Table 4: Experiments on sarcastic corpus................................................................. 23
Table 5: Experiments on non-sarcastic corpus............................................................ 24
Table 6: Evaluation of rule-based system on customer reviews.................................. 33
Table 7: Confusion matrix for corpus 1....................................................................... 33
Table 8: Confusion matrix for corpus 2 ..................................................................... 33
Table 9: Evaluation on US elections corpus............................................................... 34
Table 10: Evaluation on General Tweets..................................................................... 35
Table 11: Detected types of opinion events............................................................... 41
Table 12: Sample events extracted using burst detection on opinion values............... 45
List of Figures
Figure 1: Sentiment analysis for Whitney Houston..........................9
Figure 2: Positive and negative tweets about Whitney Houston, as determined by Sentimen140 .11
Figure 3: Architecture of Opinion Mining Application in GATE.................................................................13
Figure 4: Annotated opinionated sentence in GATE..........................................................14
Figure 5: Article about rigged voting.........................................................................................19
Figure 6: Screenshot from the opinion mining demo......................................................................36
Figure 7: Architecture of the opinion dynamics module...........................................................39
Figure 8: Opinion extraction pipeline......................................................................................47
Figure 9: Expression recognition in a constrained office setting.............................................50
Figure 10: Examples of the CLM model fitted to images from the US Elections crawl........51
Figure 11: Visualisation of the average opinion of articles associated with duplicated images....53
Figure 12: Overview of the NicePic! system...........................................................................55
Figure 13: NicePic! classifier performance with different data modalities............................56
Figure 14: The NicePic! classification interface. .................................................................58
Figure 15: The NicePic! diversified search interface. .........................................................60
1 Introduction

The work described in this deliverable is closely related to that described in D3.3 since it forms the remaining component of the ETOEs (Entity, Topic, Opinion and Event) detection and enrichment. The opinion mining components described here are therefore all designed to conform to the approach and architecture used in WP3, and we do not reiterate the details here. Opinion mining is not included in the online phase, however, but only in the offline phase, since it cannot be carried out satisfactorily in real time. It follows the same procedure as for the other ETOEs; similarly, the representation of opinions is defined with respect to the same data model and stored in a similar way in the Arcomem database. The data model has already been detailed in D3.2 and D3.3 and is not repeated here.

The work package aims ultimately to produce a set of reusable components for opinion mining. From the textual processing side, components are implemented in an open framework which has state-of-the-art properties for rapid development and deployment of language processing (GATE\(^1\)) and which has been proven as a vehicle for service-based computing in previous projects including the LIRICS implementation of ISO TC37/SC4 lexicographical and linguistic processing standards. The components are open source under the LGPL licence. Components related to visual and multimodal opinion analysis are being implemented in the OpenIMAJ\(^2\) library, which is released as open source under the BSD license.

For Task 4.1, text-based opinion mining, we have primarily worked on three main areas:

- experimentation with Machine Learning algorithms;
- development of methods and tools for extracting and aggregating opinions from specific document parts and about specific entities;
- developed new subcomponents for sarcasm detection, hashtag splitting, and an algorithm for measuring opinion interestingness
- development of a method for finding opinion events and measuring dynamics.
- development of a demonstrator for the opinion event detection algorithm,
- evaluation of the various components

For Task 4.2, Image Opinion Identification and Classification, we have primarily worked on the development and integration of new approaches to Facial Expression recognition and Image Opinion reuse.

For Task 4.3, Multimodal Opinion Analysis, we have worked on the development of computational models of attractiveness with the NicePic! system.

For these last two tasks, we have been exploring how crowd-sourced annotations can best be generated, with the overall aim of using these annotations as the basis for training the automatic multimedia opinion and sentiment classifiers and techniques that have been described in earlier deliverables. Our evaluation of techniques for the improvement of crowdsourced data took place within the 2013 MediaEval Crowdsourcing challenge, which had entries from a number of international teams.

Tools to perform all these tasks have been integrated in the ARCOMEM pipeline and are functioning.

---

1 http://gate.ac.uk/
2 http://openimaj.org
2 Opinion Identification and Classification from Text

The aim of this task is the creation of an opinion identification module in order to detect opinionated text at predicate, sentence and paragraph level. The relevance of the identified opinion is determined based on the relevant social media context items, while extracted opinions are related to the correct part of the archived item.

In this section, we describe a set of reusable text processing components for extracting opinionated information from social media. The components are implemented in GATE. We have developed applications to extract named entities, terms and events (described in D3.2 and D3.3) and to detect opinions about them. The opinions are then aggregated over larger sections of text, to give some overall sentiment about topics and documents. We go beyond traditional opinion mining techniques in a number of ways: by focusing on specific opinion-target extraction related to key terms and events, by examining and dealing with a number of linguistic phenomena, and by aggregating the opinions in different ways for a more flexible view of the information contained in the documents.

Extracted entities and events can be used to drive the extraction of opinions from tweets. It is not enough to simply know whether a tweet is positive or negative in general, but rather, we need to know what exactly it is positive or negative about. It is thus important to relate the opinion to a target (topic); for example, a tweet may be negative overall (e.g., sadness about the death of a famous person) but positive about the actual person. For example, after Whitney Houston's death, many tweets expressed sadness at her death. However, most sentiment analysis tools (such as Sentiment140\(^3\)) interpreted these tweets as being negative about Whitney Houston because of the expression of sadness, as depicted in Figures 1 and 2.

---

3 http://sentiment140.com/

---

Figure 1: Sentiment analysis for Whitney Houston
Even detecting opinions related to the right entity "Whitney Houston" in this case is not sufficient -- what we actually need is to relate the opinion to the event of her death. We therefore use the entities and events as possible targets to which the opinions are anchored. Opinions and sentiments are first gathered at the sentence and word level from text-based documents, based on the recognition of sentiment referring to the entities and events previously identified. Opinions can then be aggregated over wider elements such as whole documents or individual blog posts, and fed into the knowledge store along with the individual sentiments.

It has been argued recently [42] that traditional opinion mining techniques which focus primarily on identifying positive and negative sentences from product reviews or tweets are not sufficient for many more complex needs, particularly with respect to the analysis of social media. We believe, indeed, that this case holds exactly here, where detecting opinions in pre-defined areas of text is only partially useful, and that simply beating existing state-of-the-art evaluation scores on such tasks is not sufficient research. Instead, we return to some more fundamental research which, although it may not obtain such good scores, is far more useful in a practical sense for the tasks at hand. In the case of ARCOMEM, being able to connect opinions across entities and documents and enable the perspective of the end user to be taken into account: for example, knowing whether a document is positive or negative is not useful, but visualising the diversification of opinions about a topic is far more interesting. We investigate such issues further in Sections 2.3 and 2.4.

2.1 Sentiment Analysis Application for English

The sentiment analysis application is designed to run on text annotated with entities and events, and the relevant linguistic processing associated with these (described in D3.3) - namely sentences, tokens, POS tags, mophological analysis, Noun Phrase (NP) and Verb Phrase (VP) chunks etc. Figure 3 shows the architecture of the system in GATE. The sentiment analysis application comprises the following components:
• **Flexible Gazetteer Lookup**: this matches the lists of affect/emotion words against the text. We use a flexible gazetteer which means that the words in the list are matched according to their root form. This enables different lexicalisations, e.g. plurals, different verb forms etc. to match against each other. However, we do restrain the matching (at a later point in the grammar rules) so that a match is only valid if the same POS applies to both, i.e. a verb will not be matched with an adjective. This is because many sentiment-bearing words differ in sentiment when used as different parts of speech (compare e.g. "I like it" with "someone like me").

• **Regular Gazetteer Lookup**: this uses a regular gazetteer, and matches lists of affect/emotion words against the text only if they occur in exactly the same form as the list, i.e. different lexicalisations are not matched, because these tend to be specific terms such as swear words or phrases. For example, "bloody" is often used as a swear word, but "blood" is not, so we only want a match with a swearing sentiment when it is used as an adjective.

• **Comment Detector**: this set of JAPE grammars identifies comments found in news texts and blogs and annotates them separately from the main articles, so that sentiments can be aggregated appropriately.

• **Sentiment Grammars**: set of hand-crafted JAPE rules which annotate sentiments and link them with the relevant targets and opinion holders. Includes modules for conditional sentence detection, question detection, etc.

• **Sentiment Aggregation**: Groovy scripts which combine the scores for sentiments over sentences, paragraphs and documents and output an aggregated score for each.

• **RDF Generation**: creates the relevant RDF-XML according to the data model from the annotations

This application improves on the one reported in D4.2 in a number of ways, as will be explained in the following sections. In particular, we have incorporated improved sentiment gazetteers and scoring mechanisms for modifiers, implemented a sentiment aggregation module, and implemented a module for distinguishing comments in blogs and news texts, and for aggregating scores appropriately.
2.2 RDF generation

The GATE opinion mining application has been wrapped in the off-line module (see D3.3 for more details of this). Currently, it annotates sentiment-containing sentences in English or German with the features "score", "polarity", "entity_string" (see Figure 4 for an example of an annotated opinionated sentence). The document is also now annotated with its averaged sentiment score plus the standard deviation of the scores (see Section 2.3).
For each annotated document, we generate RDF triples according to the ARCOMEM data model, in a similar fashion to that described in D3.3 for entity and event extraction. RDF triples relating to instances of sentiment-containing sentences are generated from annotations and their features created on the documents by the GATE applications. We have developed a custom processing resource (PR) for this purpose, which is configured by a file of RDF statement templates. This PR identifies annotations whose types and features match the templates' input specifications, and fills in the statements in the corresponding template, using information from each matched annotation (annotation features in particular). The OpenRDF statements are gathered from all the documents in the corpus into one Collection<Statement>, which can be uploaded to the Sesame repository.

RDF output is generated for each instance of “Opinion” with “scorePolarity” and “scoreScale” datatype properties. An “isRealizedBy” property points to the Text and WebResource representations of the document and position where the relevant annotation (sentiment-containing sentence) was found. A “hasTarget” property points to an instance of Entity with the entity_string value as its rdf label.

A “type” property has one of the following string values:

- “individual”: a single opinion in the text;
- “commentSet”: a composite opinion from the average score of the users' comments in an article;
- “document”: a composite from the average of all the individual opinions in the document;
- “documentTarget”: a composite for each target in the document, scored from the average of all the individual opinions in the document with the same target.

The composite opinion scores will be explained in Section 2.3 and 2.4. We illustrate below some examples of RDF snippets generated for the sentence shown in Figure 4. First, we show triples directly related to the opinion in the SentenceSentiment annotation. Here, we see the string of the sentence, plus the target, polarity and score values.

For each annotated document, we generate RDF triples according to the ARCOMEM data model, in a similar fashion to that described in D3.3 for entity and event extraction. RDF triples relating to instances of sentiment-containing sentences are generated from annotations and their features created on the documents by the GATE applications. We have developed a custom processing resource (PR) for this purpose, which is configured by a file of RDF statement templates. This PR identifies annotations whose types and features match the templates' input specifications, and fills in the statements in the corresponding template, using information from each matched annotation (annotation features in particular). The OpenRDF statements are gathered from all the documents in the corpus into one Collection<Statement>, which can be uploaded to the Sesame repository.

RDF output is generated for each instance of “Opinion” with “scorePolarity” and “scoreScale” datatype properties. An “isRealizedBy” property points to the Text and WebResource representations of the document and position where the relevant annotation (sentiment-containing sentence) was found. A “hasTarget” property points to an instance of Entity with the entity_string value as its rdf label.

A “type” property has one of the following string values:

- “individual”: a single opinion in the text;
- “commentSet”: a composite opinion from the average score of the users' comments in an article;
- “document”: a composite from the average of all the individual opinions in the document;
- “documentTarget”: a composite for each target in the document, scored from the average of all the individual opinions in the document with the same target.

The composite opinion scores will be explained in Section 2.3 and 2.4. We illustrate below some examples of RDF snippets generated for the sentence shown in Figure 4. First, we show triples directly related to the opinion in the SentenceSentiment annotation. Here, we see the string of the sentence, plus the target, polarity and score values.
Next, we show triples directly connecting the opinion to the text. Here we see the string of the sentence, plus the start and end offsets of the sentence in the document.

Now we show triples directly related to the target of the Opinion (in this case, "President Obama", who has the entity type “Person”):
Finally, we show the triples connecting the Person to the Text:

(http://www.gate.ac.uk/ns/arcomem/instances#d19543b6-3cf4-49c9-becb-6c43c14f614b, http://www.gate.ac.uk/ns/ontologies/arcomem-data-model.owl#isRealizedBy, http://www.gate.ac.uk/ns/arcomem/instances#d19543b6-3cf4-49c9-becb-6c43c14f614b)

(http://www.gate.ac.uk/ns/arcomem/instances#d19543b6-3cf4-49c9-becb-6c43c14f614b, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, "President Obama")

(http://www.gate.ac.uk/ns/arcomem/instances#d19543b6-3cf4-49c9-becb-6c43c14f614b, "6677"^^<http://www.w3.org/2001/XMLSchema#integer>)

(http://www.gate.ac.uk/ns/arcomem/instances#d19543b6-3cf4-49c9-becb-6c43c14f614b, "6692"^^<http://www.w3.org/2001/XMLSchema#integer>)

(http://www.gate.ac.uk/ns/arcomem/instances#d19543b6-3cf4-49c9-becb-6c43c14f614b, http://www.gate.ac.uk/ns/ontologies/arcomem-data-model.owl#rawText, http://www.gate.ac.uk/ns/arcomem/instances#d19543b6-3cf4-49c9-becb-6c43c14f614b)

(http://www.gate.ac.uk/ns/arcomem/instances#d19543b6-3cf4-49c9-becb-6c43c14f614b, http://www.gate.ac.uk/ns/ontologies/arcomem-data-model.owl#startOffset, "6677"^^<http://www.w3.org/2001/XMLSchema#integer>)

(http://www.gate.ac.uk/ns/arcomem/instances#d19543b6-3cf4-49c9-becb-6c43c14f614b, http://www.gate.ac.uk/ns/ontologies/arcomem-data-model.owl#endOffset, "6692"^^<http://www.w3.org/2001/XMLSchema#integer>)

(http://www.gate.ac.uk/ns/arcomem/instances#d19543b6-3cf4-49c9-becb-6c43c14f614b, http://www.gate.ac.uk/ns/ontologies/arcomem-data-model.owl#contains, "President Obama")


2.3 Opinion Aggregation and Summarisation

In terms of the user requirements, it has become clear that simply finding opinions about entities and events is not particularly useful in isolation: what is needed is some kind of aggregation at a higher level, in order to find e.g. opinions about documents as a whole, or to correlate all the opinions by different people about the same topic or entity. A novel aspect to our work thus concerns the type of aggregation that can be applied to opinions to be extracted from various sources and co-referred. In classical information extraction, this can be applied to the extracted data in a straightforward way: data can be merged if there are no inconsistencies, e.g. on the properties of an entity. Opinions behave differently in social media, however: multiple opinions can be attached to an entity and need to be modelled separately, as explained below. In temporal terms, an important question is whether one should just store the mean of opinions detected within a specific interval of time (as current opinion visualisation methods do), or if more detailed approaches are preferable, such as modelling the sources and strength of conflicting opinions and how they change over time. Effectively, we advocate here a form of opinion-based summarisation, e.g. displaying positive/negative opinion timelines, coupled with opinion holders and key features. This is detailed further in Section 3.

Typically, opinion aggregation involves simply combining all the positive and negative scores about a topic, document or corpus. However, there is some debate in the literature about the relative balance between positive and negative opinions in a document set. In some cases, people are more likely to post positive than negative reviews or opinions; in other cases, the opposite is true. Moreover, in product reviews, positive and negative comments outweigh neutral ones, but this is not necessarily true of news articles, for example, or even random tweet collections. Some variations have therefore been proposed, depending on the domain. Michael Wu from Lithium performed a study of sentiment data on various social web apps about presidential candidates in March 2012. His analysis involved taking the positive sentiments minus the negative sentiments, over a 2 week period, as one might expect, but also including the neutral sentiments. Neutral sentiments were weighted at 1/10 and added to the net sentiment. The theory behind this is that a neutral sentiment expressed actually has slightly more value than no sentiment expressed at all.

---

*http://lithosphere.lithium.com/t5/Building-Community-the-Platform/Big-Data-Big-Prediction-Looking-through-the-Predictive-Window/ba-p/41068*
So for example a combination of positive and negative sentiment expressed about a single topic probably possibly swings very slightly towards the positive side and should be included.

In general, the ARCOMEM users are of the opinion that while having a single opinion about a document (i.e., whether it is overall positive or negative) is not useful unless it is related to something specific (e.g., "This document is positive about Obama"). However, a simple aggregation of document scores will not provide this information, because it is often impossible to associate a single entity as the target of the aggregated opinions. Furthermore, one cannot simply associate the overall opinion with the most frequent entity, or even an entity which is clearly the topic of the document, because not all opinions will be about this entity. On the other hand, knowing the range of diversity of opinions in the document is potentially useful, so that one can see whether the document is highly opinionated, and whether the opinions cover a wide range of topics.

One possibility for finding "interesting" documents in terms of opinions would be to find documents which contain outlying opinions about entities. This is in some sense the opposite of calculating reputation trust. There, a user is considered trustworthy if their opinions typically match those of the majority of users, since we typically trust the majority opinion about a topic (such as a book or film review). In ARCOMEM, however, we want to find things which are unusual, such as a lone voice of dissent. An approach we are currently investigating involves finding entities with outlying opinion scores in a document, as follows:

- For each entity which is the target of an opinion in the document, calculate the average opinion score for that entity across all documents in the corpus.
- Rank documents according to the number of outliers they contain (where an outlier is an entity with an opinion score widely differing from the average for that entity, according to some threshold)
- Documents with the highest number of outliers could be considered the most interesting from an opinionated point of view.

We have developed a component for aggregating the opinions contained in different sentences. Initially, we tested this on product reviews, where they may be several different opinions expressed in a product review, which can then be summarised to give an overall rating. We compared this approach with machine learning methods which produce overall ratings for each review without considering individual sentences (see e.g. [38] for an example). The evaluation is described in Section 2.10. While the product review corpus and task is slightly different from the Arcomem data and task, it gives us a good baseline to compare different approaches, and has the advantage of providing us with readily available training data and a gold standard set to evaluate against.

### 2.4 Detecting opinions in comments

In this section, we focus on opinion extraction from news articles. Many articles in online newspapers and other text types offer some kind of opinionated comment about a topic, person or event. However, such webpages often also contain comments from readers. It is useful not only to understand the opinions in the main story, but to separate these from the opinions of the different readers. Furthermore, the opinions expressed in these comments may relate either to the main story, or to one or more of the other comments. There are many reasons why we might want to analyse not only the story itself, but also the comments: to give us a better understanding of the different sides to the main story; to give us an idea about the prevailing views of (admittedly a narrow section of) the public; to give us insight into further issues not mentioned in the story, or related stories, and so on.
Typically, opinion mining tools either look at the webpage as a whole, and ignore this distinction between story and comment, or they look at the story and disregard the comments. In the case of the former, this might provide erroneous opinion analysis over the document as a whole. For example, a story in the online press about rigged voting machines in the recent US elections (see Figure 5) had a very negative tone overall, but many of the comments were positive (often about other, related, issues), so an overall opinion analysis did not show the negativity adequately. The aggregated score for the document was +0.286, showing a small amount of positivity, but the standard deviation was 0.567, showing a lot of variance. For archiving purposes, we are particularly interested in getting a cross-section of the views of the community about certain topics and events, so this analysis of comments is vital.

Table 1 shows some examples of opinionated sentences from a document, including the entity that is considered the target of the opinion, and the scores associated with the opinion. We can see that the mentions of the entities have already been co-referenced with the longest mention in the document of that entity (e.g. “Romney” has been co-referenced with “Mitt Romney”, which is used as the label for the entity and its URI: see D3.3 for more details). Table 2 shows the aggregated opinions for two of these entities (Obama and Romney), while Table 3 shows the overall opinion score for the document.
Our overall strategy is as follows. First, we have to identify which is the main story and which is the comment section, if such exists. For this we use a number of heuristics, such as looking for some specific <div> and <span> tags in the document metadata, or a section labelled “Comments” or similar. We annotate these two sections as Text and CommentSection respectively. Next, we identify the individual comments, again using similar sets of heuristics, and use the results of our NE recognition tools to also identify the author name, date/time of posting and so on, and to associate these with the correct comment. An example rule is given below, which matches the two most common pieces of metadata (originally found in the HTML markup) denoting a comment in a news text. The original HTML markup is transformed automatically by GATE into annotation types with features and values. The rule looks for either a <div> annotation with a feature called “class” whose value is “dsq-comment-message”, or a <div> annotation with a “class” feature, whose value is “comment-body”. If either of these are matched, a new annotation is created over the same span, called “CommentMessage”, which is then used in future rules to define the span over which the content of the comment itself can be found (excluding things like the time and author of the comment, which are dealt with separately).
Next, we analyse the opinion in each comment separately (if one exists) and relate it to the correct target, determining whether it is about the main story or something else (e.g. another comment). Finally, we aggregate these opinions by author and by target (entity).

We thus output opinions in a variety of different ways: all are represented as an Opinion, with different types (as mentioned in Section 2.2).

1. a single opinionated sentence, represented with type “individual” and with an Entity or Event as the target, and which has “score”, “polarity” and “hasTarget” properties.

2. an average opinion over a whole document, represented with type “document”, and which has no Target (since it makes no sense over a whole document). It has “score”, “scoreStdDev”, and “polarity” properties.

3. an average opinion produced from all the opinions in the document which have the same target. This has type “documentTarget” and has properties “hasTarget”, “score”, “scoreStdDev” and “polarity”.

4. an average opinion produced over all the comments in the document, without regard to the targets, and which has the type “commentSet”. It has “score”, “scoreStdDev”, and “polarity” properties.

Of course, this kind of approach will not solve the problem on all kinds of documents, but it is nevertheless a useful first step which can easily be adapted to other document types which do not match the same kind of format. The formats we have examined and to which these methods have been applied do occur very frequently and are of a standard type used in news articles, forums and so on, so we can be confident that many cases are covered. A more general approach would not be feasible given the widely differing kinds of texts we deal with.

2.5 Sarcasm detection

Sarcasm is one of the most cited reasons for failure of opinion mining systems to perform well, primarily because it is so difficult for a machine to detect. The vast majority of opinion mining systems thus choose to ignore it. Given that sarcasm is prevalent in certain kinds of social media, especially tweets, it is useful to be able to incorporate at least the easily detectable uses of sarcasm into our tools, although there will always be cases which are impossible to identify reliably. In a corpus of 1 million randomly selected tweets, we found approximately 2% of them contained the hashtags #sarcasm or #irony; however, this does not reflect all sarcastic tweets by
any means, and in certain domains such as politics, we can expect the proportion of sarcastic
tweets to be higher.

Even for a human, sarcasm can be difficult to detect. One needs to have a good understanding of
the context of the situation, the culture in question, and perhaps the very specific topic or people
involved in the sarcastic statement. This kind of real-world knowledge is often impossible for a
machine. Furthermore, even correctly identifying a statement as sarcastic is often insufficient to be
able to correctly analyse it, especially in terms of sentiment, due to issues of scope.

Almost all current research on sarcasm detection has only studied the issue of classifying
sentences (usually tweets) as sarcastic or not. The French company Spotter recently hit the news
for its state of the art sarcasm detection tool, reputed to achieve an 80% success rate in finding
sarcastic utterances, as part of its reputation analysis platform. But it is not entirely clear how it
then processes them. Furthermore, sarcastic statements are particularly prone to novelty in
vocabulary use, which makes it hard to train Machine Learning algorithms to successfully spot
them when no external clues (such as hashtags) are present. We have investigated the use of
sarcasm in tweets, and in particular their effect on sentiment analysis. Our early experiments show
that correctly detecting sarcasm improves sentiment detection by nearly 50 percentage points (on
a corpus of sarcastic tweets), but that even when a tweet is correctly identified as being sarcastic,
accuracy of sentiment analysis is still far from perfect. This work is also described in Error:
Reference source not found.

There have been a number of recent works attempting to detect sarcasm in tweets and other user-
generated content. Tsur et al. Error: Reference source not found use a semi-supervised approach
to classify sentences in online product reviews into different sarcastic classes, and report an F-
measure of 82.7% on the binary sarcasm detection task (although Precision is much higher than
Recall). Liebrecht et al. Error: Reference source not found use the Balanced Winnow Algorithm to
classify Dutch tweets into sarcastic or not, with 75% accuracy, training over a set of tweets with the
#sarcasm hashtag. Reyes et al. Error: Reference source not found use a similar technique on
English tweets to detect ironic tweets, using the #irony hashtag, with 70% accuracy. Davidov et al.
Error: Reference source not found achieved 83% accuracy on sarcasm detection in tweets.
However, it appears that none of these approaches go beyond this step: even when a statement is
known to be sarcastic, one cannot necessarily predict how this will affect the sentiment expressed.

2.1.1 Sarcasm-aware opinion mining

We modified the ARCOMEM opinion mining pipeline to consider sarcasm as part of the grammar
phase. As a first step, we simply reversed the polarity of an opinion whenever a sarcastic
statement was found. We manually collected a list of sarcastic hashtags from a corpus of random
tweets, and then extended this by automatically collecting pairs of hashtags where one hashtag
contained an existing sarcasm hashtag (e.g. #sarcasm), using the GazetteerListCollector GATE
plugin\(^5\).

For example, the following tweets contain pairs of sarcastic hashtags::

*I love living with a 7 year old #NotReally #sarcasm

The best feeling in the world is definitely being ignored. I love it #keepdoingit #bestthingever
#sarcasm

We then added the other hashtag to our list of sarcasm indicators, e.g. #notreally. These indicators
were used to signify when sarcasm was present in a sentence: if one or more of these was found
within the same sentence as an opinion, or within the same tweet, the existing polarity was
reversed. Figure 1 depicts an example of two sarcastic sentences analysed by our improved
application. Here we show the results via our online demo system\(^6\).

\(^5\) http://gate.ac.uk/userguide/sec:gazetteers:listscollector
\(^6\) demos.gate.ac.uk/arcomem/opinions/
2.1.2 Experiments on sarcasm detection

In our first experiment, we collected a corpus of 134 tweets containing the hashtag #sarcasm, from a larger set acquired via GardenHose on Oct 16 2012. Error: Reference source not found and manually annotated the sentences with a sentiment (positive, negative or no sentiment). Out of 266 sentences, 68 were found to be opinionated (approximately 25%), and of these 62 were negative while 6 were positive. Of these 68 opinionated sentences, 61 were deemed to be influenced by sarcasm while 8 were not (note that it is possible for a tweet to be sarcastic, or to at least have a sarcastic hashtag, without every sentence contained in that tweet being sarcastic). Unsurprisingly, we can see that the vast majority of sarcastic tweets have negative polarity. This is due to the nature of sarcasm. Error: Reference source not found for discussion of this. However, more than 10% of sarcastic tweets involved sentiment-containing sentences not directly affected by the sarcasm, i.e. whose polarity was unaltered.

We evaluated the performance of our regular sentiment analyser (SA-Reg) and the analyser which considered sarcasm (SA-Sar); results are shown in Table 4. We measured detection of opinionated sentences, detection of opinions with the correct polarity of sentiment, and detection of correct polarity of sentiment only for those opinionated sentences correctly identified. Note that the results for opinion detection are identical for both analysers.

<table>
<thead>
<tr>
<th>Sarcastic corpus</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opinionated</td>
<td>74.58</td>
<td>63.77</td>
<td>68.75</td>
</tr>
<tr>
<td>Opinion+polarity - Regular</td>
<td>20.34</td>
<td>17.39</td>
<td>18.75</td>
</tr>
<tr>
<td>Polarity-only - Regular</td>
<td>27.27</td>
<td>27.27</td>
<td>27.27</td>
</tr>
<tr>
<td>Opinion+polarity - Sarcast</td>
<td>57.63</td>
<td>49.28</td>
<td>53.13</td>
</tr>
<tr>
<td>Polarity-only - Sarcast</td>
<td>77.02</td>
<td>77.28</td>
<td>77.28</td>
</tr>
</tbody>
</table>

Table 4: Experiments on sarcastic corpus

For comparison, we also manually annotated the same tweets without taking sarcasm into consideration, in order to have a gold standard set of tweets that matched the first corpus in terms
of distribution (since it was heavily skewed towards negative polarity). We then evaluated the performance of the same two analysers on this dataset; results are shown in Table 5.

<table>
<thead>
<tr>
<th>Regular corpus</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opinionated</td>
<td>57.89</td>
<td>58.93</td>
<td>58.41</td>
</tr>
<tr>
<td>Opinion+polarity - Regular</td>
<td>45.61</td>
<td>46.43</td>
<td>46.02</td>
</tr>
<tr>
<td>Polarity-only - Regular</td>
<td>78.79</td>
<td>78.79</td>
<td>78.79</td>
</tr>
<tr>
<td>Opinion+polarity - Sarcastic</td>
<td>22.81</td>
<td>23.21</td>
<td>23.01</td>
</tr>
<tr>
<td>Polarity-only - Sarcastic</td>
<td>39.40</td>
<td>39.39</td>
<td>39.39</td>
</tr>
</tbody>
</table>

Table 5: Experiments on non-sarcastic corpus

### 2.1.3 Discussion

Table 4 shows a reasonable performance for the sarcastic corpus on detecting whether sentences are opinionated or not, with higher Precision than Recall. The regular application performs very poorly on the polarity detection tasks, unsurprisingly. What is interesting, however, is that the sarcastic-tuned application still does not perform that highly. In other words, even when we know that a tweet is sarcastic, we do not always get the polarity right. There are several reasons for this. One is simply that we do not always correctly identify the polarity in the first place, before the sarcasm is considered. More importantly, however, is the effect of sarcasm scope.

A naive interpretation of sarcasm (as used in the first version of our sarcasm detection application) blindly assumes that sarcasm has a similar impact on polarity to negation. For example, if we have a phrase such as "this project is great!", we would reverse the polarity of the sentiment from positive to negative if we knew that it was meant sarcastically. However, the scope of sarcasm is not always easy to determine. Take the following example:

*I am not happy that I woke up at 5:15 this morning.. #greatstart #sarcasm*

If we reversed the polarity of the opinion "not happy", we would end up with a positive sentiment, but this is not correct. The sarcasm applies only to the hashtag #greatstart and not to the previous sentence. In the next example, however, the sarcasm refers to the main utterance "you are really mature" and not to the other hashtag #lying.

*You are really mature. #lying #sarcasm*

Furthermore, sarcasm does not always reverse polarity. Take the example

*It's not like I wanted to eat breakfast anyway. #sarcasm*

When uttered sarcastically, this statement indicates negative sentiment, but without the sarcasm, it doesn't particularly indicate positive sentiment. To address this problem, we have developed a number of heuristics. In general, when someone is being sarcastic, they are saying the opposite of what they mean, so as long as we know which bit of the utterance is the sarcastic bit, we can simply reverse the polarity (as in the original version of the application described above). If there is no polarity on the original statement, we can assume that the statement is negative, for example:

*It's not like I wanted to eat breakfast anyway #sarcasm*
To get the polarity scope right, we need to investigate the hashtags. We adopt the following procedure. If there is more than one hashtag, we look at any sentiment contained in those hashtags.

- A negative sentiment in the text, followed by both a positive hashtag and a sarcasm hashtag indicates (most likely) that the sarcasm has flipped the polarity of the positive hashtag to negative, and left the negative sentiment as it is. Take for example, in the following tweet:

  *I am not happy that I woke up at 5:15 this morning. #greatstart #sarcasm*

Here we have a negative sentiment "not happy", a positive hashtag "greatstart" and a sarcasm hashtag. The sarcasm refers to the "great start" and not to the "not happy", so we leave the sentiment of the tweet as it is (i.e. negative).

- A positive sentiment in the text, followed by two sarcasm hashtags, or by a negative hashtag and a sarcasm hashtag, indicates (most likely) that both hashtags are flipping the polarity of the positive sentiment in the tweet, with the result that the tweet becomes negative overall. In the following example:

  *You are really mature. #lying #sarcasm*

the positive sentiment "really mature" is flipped to negative by the hashtags "lying" and "sarcasm".

- When in any doubt, it is usually safe to assume as a default that a sarcastic statement carries negative sentiment

If there are no hashtags that indicate sarcasm, there are other clues we can look for. One indicator might be word combinations with opposite polarity, e.g. a typically negative word such as "rain" or "delay" together with a positive sentiment word such as "brilliant". This is particularly likely if the sentiment word is a strong one ("rules" for sarcasm dictate that one should normally use a highly emphatic positive word such as "excellent" rather than just a mildly positive one such as "good" if one wants to use it sarcastically). Modifying the sentiment word with a swear word also increases the likelihood of sarcasm. Another possibility is to include world knowledge, especially in combination with the above. For example, knowing that people typically do not enjoy going to the dentist, or that people typically like weekends better than weekdays, could help indicate that a positive sentiment about such things is likely to be sarcastic. However, these kind of issues are incredibly hard to solve, and we should assume that there are many instances of sarcasm that a machine is highly unlikely to ever identify.

2.1.2 Collecting and tokenising hashtags

Much useful sentiment information is contained within hashtags, but this is problematic to identify because hashtags typically contain multiple words within a single token, e.g. #notreally. We therefore developed an algorithm to extract the individual components from the hashtag. First, if a hashtag is camelcased, we use the capitalisation information to create separate tokens, by modifying the GATE tokeniser to split tokens on capital letters, e.g. notReally is separated into "not" and "Really". Second, if the hashtag is all lowercase or all uppercase, we try to form a token match against the Linux dictionary which we have converted into a GATE gazetteer. We also try to match against our existing gazetteers of entity types such as Locations, Organisations etc (but these only include some well-known types - see Section 2.1.3 for discussion), and against the same dictionary of common slang words used in the TwitIE normalisation tool (see D3.3 for more information about this). After some initial experimentation, we manually edited the list to remove most single-character "words" such as "h" (but leaving "real" words such as "i", "a" etc.) and a few other entries that we considered non-words. Working from left to right, we use a Viterbi-like algorithm to look for the best possible match that combines a set of known words from the lookups, and completes the tokenisation to the end of the hashtag. If a combination of matches can be
found without a break, the individual components are converted to tokens and the original single
Token annotation covering the whole span of the hashtag is removed. In our example, #notreally
would be correctly identified. However, some hashtags are ambiguous: for example, "#greatstart"
gets split wrongly into the two tokens “greats” + “tart” rather than “great” + “start”. These problems
are hard to deal with; in some cases, we could make use of contextual information to assist. We
could also consider the POS tags, for example adjective+noun combinations are more probable
than verb+noun combinations. Figure 2 shows an example of retokenised hashtags in GATE: we
can see, for example, that #conflictpalmoil has been tokenised as “conflict” + “palm” + “oil”.

In addition to using the sentiment information from these hashtags, we also collect new hashtags
that typically indicate sarcasm, since often more than one sarcastic hashtag is used. For this, we
used the GATE gazetteer list collector to collect pairs of hashtags where one was known to be
sarcastic, and examined the second hashtag manually. From this we were able to identify a further
set of sarcasm-indicating hashtags, such as #thanksdude, #yay etc. Further investigation needs to
be performed on these to check how frequently they actually indicate sarcasm when used on their
own.

![Figure 2: Screenshot of retokenised hashtags](image)

### 2.1.3 Evaluating hashtag tokenisation

We conducted an experiment to measure the accuracy of hashtag tokenisation, using a corpus of
1000 tweets randomly selected from the US elections crawl. 944 hashtags were detected in this
corpus, of which 408 were identified as multiword hashtags (we included combinations of letters
and numbers as multiword, but not abbreviations). 281 were camelcased and/or combinations of
letters and numbers, 27 were foreign words, and the remaining 100 had no obvious token-distinguishing features. Since the camelcased words and letter/number combinations were all correctly identified, and the foreign words would normally have been excluded by the language identification module, it is the remaining 100 words that we are most interested in. We compared a gold standard set of tokenised hashtags annotated manually, with a system-annotated set, achieving 95.2% Precision and 93.3% Recall.

For a fairly simple solution, these initial results are pleasing. Error analysis showed that most of the errors came from an overgeneration of a word, for example #freespeech was analysed as frees + pee. This kind of error could be rectified by not allowing successful decomposition if the end of the hashtag is not reached. Another error is due to the fact that the Twitter normaliser is only run over Tokens and not hashtags. Running the hashtag decomposition and converting the result to regular Tokens, or running a special version of the normaliser over the hashtags, would solve this. These issues are work in progress and/or left for future work. A final error is due to the presence of unknown named entities (people, locations and organisations) forming part of the hashtag. While some of these are recognised by our gazetteer lookup (especially locations), many of them are unknown. The named entity recognition component in GATE cannot identify these until they have been correctly tokenised, so we have a circular problem. In collaboration with the SemanticNews project\(^7\), we have been working on a solution for disambiguating user names in Twitter by means of DBpedia; we are planning to adapt this technique to hashtags also so that such entities can be recognised.

2.6 Opinion Aggregation and Summarisation

As mentioned in D4.3, feedback from the ARCOMEM users revealed that producing a single opinion about a document (i.e. whether it is overall positive or negative) is not useful unless it is related to something specific (e.g. “This document is positive about Obama”). However, a simple aggregation of document scores will not provide this information, because it is often impossible to associate a single entity as the target of the aggregated opinions. Furthermore, one cannot simply associate the overall opinion with the most frequent entity, or even an entity which is clearly the topic of the document, because not all opinions will be about this entity. On the other hand, the users agreed that knowing the range of diversity of opinions in the document is potentially useful, so that one can see whether the document is highly opinionated, and whether the opinions cover a wide range of topics.

2.3.1 Calculating interestingness

We have therefore developed an algorithm for identifying which documents are likely to be interesting to the users, based on the extent to which they contain outlying opinions about the entities within them. Essentially, this is calculated as follows:

- For each entity which is the target of an opinion in the corpus, calculate the mean and standard deviation of the opinion scores for that entity (across all documents in the corpus). Opinions that do not have a specific target are included in the global (corpus-wide) statistics, but not in any target’s statistics.
- Calculate the target-interestingness of each targeted opinion based on how far out of a range around that target’s mean its opinion score is (opinion scores within the range have an interest score of 0). Calculate the global interestingness of each opinion the same way, but using the global statistics (mean and standard deviation).

\(^7\) http://semanticnews.org.uk
Calculate the interest score of each document as the mean of the global interest scores of the opinions it contains. Outliers therefore make a document more interesting, and opinions close to the average make it less interesting.

The approach is realised in the following way. After annotating and scoring all the opinions in a corpus as usual, we then:

1. Scan through the whole corpus and build up a data structure that contains for each opinion annotation:
   - a document ID
   - an annotation ID
   - the target entity or event
   - the score.

2. Calculate the mean and standard deviation ($\sigma$) of opinions for each target.

3. Scan the corpus again and for each opinion, calculate the “target interestingness” of its score with respect to all the opinions in the corpus and with respect to all the opinions about the same target, and add these as features of the opinion annotation. For a given set of opinions (all the opinions in the corpus, or all the opinions about a particular target), we calculate from the scores $x$ the mean and standard deviation, and using a boringness coefficient $b$ to mark the “middle of the road opinions” (scores in the range $\bar{x} \pm b\sigma$) as uninteresting, we calculate the interestingness of each opinion based on the distance between its score and the edges of the boring range. We calculate the “global interestingness” of each opinion the same way, but using the global mean and standard deviation (rather than any target’s statistics).

$$I(x_i) = x_i - (\bar{x} + b\sigma) \text{ if } x_i > \bar{x} + b\sigma;$$
$$I(x_i) = (\bar{x} - b\sigma) - x_i \text{ if } x_i < \bar{x} - b\sigma;$$
$$I(x_i) = 0 \text{ if } (\bar{x} - b\sigma) \leq x_i \leq (\bar{x} + b\sigma)$$

4. Add to each document an annotation with a feature containing the mean interestingness score of the opinions in it.

5. The annotations result in RDF triples. To cover interestingness, we have added new properties to the ontology: WebResource interest decimal, Opinion interest decimal, and Opinion targetInterest decimal. Examples are given in the RDF snippets in the next section.

### 2.7 RDF generation

The coverage of RDF output has been expanded to include opinions about events and entities, using the arco:Opinion arco:hasTarget arco:InformationObject property, and to use the arco:type string property’s value to distinguish a single opinion in a text (“individual”), an aggregate score over a document (“document”), an aggregate score for a particular target over a document (“documentTarget”), and an aggregate score over the “comments” section of a document (“comment”). Opinions are also scored for how interesting they are (as explained in Section 2.3 above) and the resulting scores are expressed with the arco:interest and arco:targetInterest numeric properties (domain arco:Opinion). Each document is scored using the mean of the
interest values of the opinions it contains, and expressed with the arco:interest numeric property (domain arco:WebResource).

The RDF output for instances of arco:Event now uses the arco:hasParticipant property to identify instances of Location, Organization, and Person involved in the event, whenever these are identified by the event detection analysis.

The following snippet (with long prefixes and UUIDs trimmed for readability) shows examples of these properties.

(inst:Opinion_eb0f665547e6, arco:hasTarget, inst:America)
(inst:Opinion_eb0f665547e6, arco:interest, "0.1661257918788578"^^http://www.w3.org/2001/XMLSchema#decimal)
(inst:Opinion_eb0f665547e6, arco:scorePolarity, "negative")
(inst:Opinion_eb0f665547e6, arco:scoreScale, 
"-0.5"^^http://www.w3.org/2001/XMLSchema#decimal)
(inst:Opinion_eb0f665547e6, arco:targetInterest, "0.0"^^http://www.w3.org/2001/XMLSchema#decimal)
(inst:Opinion_eb0f665547e6, arco:type, "individual")
(inst:Opinion_eb0f665547e6, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, arco:Opinion)
(inst:Event_c8bbc47b6063, arco:hasParticipant, inst:President_Obama)
(inst:Event_c8bbc47b6063, arco:hasType, "win (unspecified)")
(inst:Event_c8bbc47b6063, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, arco:Event)
(inst:Event_c8bbc47b6063, http://www.w3.org/2000/01/rdf-schema#label, "Obama Win on PBS")

2.8 Other improvements

We also performed some minor improvements to the general opinion mining detection algorithms. First, we modified the scores on the opinionated words to take into account things like capitalisation. Sentiment words (such as “good”, “bad” etc.) have their score doubled if they are entirely capitalised, or if an adverb preceding them is entirely capitalised. So for example, if “this is good” has a score of +0.5, “this is GOOD” will get a score of +1.0. We also plan to do the same for words which have extra repeated letters, e.g. “gooood", and to investigate other similar cases where the score might be intensified.
2.9 Machine Learning applications for Opinion Mining

Statistical machine learning approaches to information extraction include the use of Hidden Markov Models (HMM), Support Vector Machines (SVM), and Conditional random Fields (CRF). With HMMs [22], the information extraction task is cast as a tagging problem where, given a sequence of input words, the system has to produce a sequence of tags; the words are observations and the tags are hidden states in the HMM. CRFs [21] are state-of-the-art techniques for IE and tend to do better than other classification methods, but are less suitable for opinion mining. SVMs are very competitive supervised models for information extraction [17], which treat the task as a binary classification problem (or set of intersecting binary problems; each label gives rise to a binary classification problem) by seeking to optimise a hyperplane in the vector space of instances that maximally separates positive from negative instances. SVMs have been used in a variety of NLP problems which are instances of multi-class classification problems (for more than two classes; in NER, for example, there are a considerable number of names to be recognised such as location names, organisation names, personal names) and perform well in this field [24][25]. We have initially adopted the SVM learning paradigm, not only because it has been used with success in different tasks in NLP, but it has been shown particularly suitable for text categorisation [18]. In previous classification experiments, we have tried other machine learning algorithms such as Decision Trees, Naive Bayes Classification, and Nearest Neighbour from the Weka toolkit [48], but support vector machines gave us best overall classification accuracy.

Almost all these statistical approaches adopt the same steps: first they transform the problem into a multi-class classification task; they then convert the multi-class problem into several binary classification problems using a one-vs-all or one-vs-another approach (for example); then an SVM classifier is trained for each binary classification task; finally, the classifiers’ results are combined to obtain the solution to the original NLP problem. In our methodology, each information extraction learning problem is transformed into a classification problem. Each learning instance is transformed into a vector representation in a high dimensional feature space (we use lexical, syntactic, and semantic features). The SVM learns a hyperplane that separates positive from negative instances with the maximal distance to all training examples. In this work we use SVM with uneven margins (SVMUM) as proposed by [26], because the data is generally not evenly balanced.

When applying SVMUM to a problem, we need to identify the value for the uneven margins. If the problem has just few positive training examples and many negative ones, then a margin smaller than 1 could be used. The margin parameter can be empirically determined by cross-validation on training data. A reasonable estimation of the margin parameter can help achieve better performance than using a standard SVM. Some problems or data sets may not be sensitive to changes in the margin, thus a standard SVM can be applied. A second parameter which has to be carefully selected in the SVM algorithms is the probability threshold (between 0 and 1) to be used to accept or reject a particular classification. In order to estimate these parameters empirically, we use a set of n documents from the corpus and carry out an experiment for each possible combination of probability and margin using values between 0.10 and 1.00 with steps of 0.10. For each pair of values, n iterations are executed where document i is removed from the corpus, the n-1 documents remaining documents are used for training the SVM with the given parameters, and the i document is used to test the algorithm. At each iteration, Precision, Recall, and F-measure are computed. The probability and margin are chosen as the ones maximising the F-measure. For our current work, we use the same parameters as for the original experiments detailed in [38], but also test a few different variants.

Perceptron is a simple, fast and effective learning algorithm, which has successfully been applied to named entity recognition [8]. The system uses a two-layer structure of classifiers to handle the imbalanced data. The first layer classifies each word as entity or non-entity. The second layer classifies the named entities identified by the first layer in the respective entity classes. Li et al. [23] proposed another variant of Perceptron, the Perceptron algorithm with uneven margins (PAUM), designed especially for imbalanced data. In [24], Li et al. explored the application of PAUM to IE,
and found that the PAUM system performed slightly worse than the SVMUM system. On the other hand, training time of PAUM is only 1% of that for the SVM and the PAUM implementation is much simpler than that of SVM. Therefore, when simplicity and speed are required, PAUM presents a good alternative.

In our experiments (described in Section 2.10), we are dealing with the problem of classification of opinionated texts (product reviews) as positive/negative or on a more fine-grained classification scale (e.g., very bad to excellent). Because the product reviews give us access to considerable free annotated training data, we solve the classification problem using a supervised ML framework to recognise positive and negative opinions and to filter out opinionated vs non-opinionated sentences. In our learning framework each text represents a learning or testing instance.

Each learning instance is represented as a vector of feature-values, in our case features are created from linguistic annotations produced by different linguistic processors. The features to be used are selected according to a hypothesis one may have about what may influence recognition of a class. In the case of sentence or text classification, the features are either lexical (morphological information), syntactic (relying on POS information), semantic (relying on a sentiment dictionary), or discursive (textual).

### 2.9.1 SentiWordNet features

In order to identify the positivity or negativity of a given word in text, one first needs to perform general word sense disambiguation, i.e., when observing a word such as “good” in text, and assuming it is an adjective, one would have to decide for one of its 21 senses (as specified in WordNet). However, we do not apply any word sense disambiguation procedure: instead, for each entry in SentiWordNet (each word#sense) we compute the number of times the entry is more positive than negative (positive > negative), the number of times is more negative than positive (positive < negative) and the total number of entries word#sense in SentiWordNet. This enables us to consider the overall positivity or negativity a particular word has in the lexical resource. We are interested in words that are generally “positive”, generally “negative” or generally “neutral” (not much variation between positive and negative). For example a word such as “good” has many more entries where the positive score is greater than the negativity score, while a word such as “unhelpful” has more negative occurrences than positive. We use this aggregated score in our experiments on opinion identification. A language resource has been implemented in GATE to access the SentiWordNet resource and an algorithm to compute the “general” sentiment of a word has been implemented.

### 2.9.2 Linguistic and Semantic Features

Here we describe the features we use to represent instances. For each token in the document, the following features are used in our experiments:

- **string**: the original, unmodified text of the token;
- **root**: the lemmatised, lower-case form of the token (for example, run is the root feature for run, runs, ran, and Running);
- **category**: the part-of-speech (POS) tag, a symbol that represents a grammatical category such as determiner, present-tense verb, past-tense verb, singular noun, etc.);
- **orth**: a code representing the token’s combination of upper- and lower-case letters (if it has been classified as a word);
- **countP**: the word’s positivity score (based on our description);
• countN: the word’s negativity score;
• countF: the total number of entries for the word in SentiWordNet.

We additionally use the following syntactic features (syn_features):
• ADJ: the lemmatized form of an adjective;
• ADV: the lemmatized form of an adverb;
• ADJ_ADJ: a bigram of adjectives’ lemmas;
• ADV_ADV: a bigram of adverbs’ lemmas;
• ADV_ADJ: a bigram of adjective’s lemma and adverb’s lemma.

For each sentence in the document the following features are used (sentence_features):
• countP: (at sentence level) the number of positive words in the sentence (words which have been observed with a positive polarity more times than with a negative polarity);
• countN: (at sentence level) the number of negative words in the sentence (words which have been observed with a negative polarity more times than with a positive polarity);
• senti: a value ‘pos’ or ‘neg’ or ‘neutral’ according to the distribution of sentiP and and sentiN in the sentence.

For each target text fragment in the document, the following features are used (text_features):
• count_pos: the number of sentences with senti value ‘pos’;
• count_neg: the number of sentence with senti value ‘neg’;
• count_neutral: the number of sentences with senti value ‘neutral’.

2.9.3 ML Applications

We developed some variations of the machine learning application used in [38] for analysing product reviews, incorporating some of the information we use in our ARCOMEM rule-based system for opinion mining. For example, we experiment with features such as POS tags, NP and VP chunks, terms and named entity information and so on.

2.10 Evaluation

2.10.1 Customer reviews

We experimented with the set of product reviews used in this work, using a small corpus of reviews from a consumer forum (http://www.price-grabber.co.uk). The corpus consists of HTML pages, each containing a number of separate comments product or company reviews. Each review
consisted of a paragraph or two of natural-language text entered by one of the forum’s users and the same user’s rating of the company from one to five stars. Each rating was represented by an <img> tag pointing to a GIF image of a row of one to five adjacent stars, with an alt attribute of 1 Star Review, 2 Star Review, etc. For the experiment, we first converted the gold standard annotations from a star rating to a positive, negative and neutral rating. 1 or 2 stars was designated negative, 3 as neutral, and 4 or 5 as positive. We then ran our conditional application (described in Section 2.1) over only the sentences that formed part of the product review, using the <span> annotation from the original HTML document markup as the covering annotation set. Because the spans did not always correspond to exact sentences, we added a special processing resource (a groovy script to constrain the sentences appropriately) to get the right sentence spans. The application annotated the sentences individually, such that every sentence deemed to have a sentiment was annotated with positive or negative and a score indicating the strength of the sentiment. All sentiment-containing sentences within the span of a single review were then collected and their scores averaged (total score divided by the number of sentiment containing sentences) to produce a single sentiment score for the review. The averaging was done using another groovy PR written for this purpose. Any reviews which did not contain any sentiment containing sentences were deemed to be neutral and given a score of 0. We used the Corpus QA tool in GATE to compare the system output with the gold standard, as defined by the star ratings.

We also experimented with two corpora: (1) the test set of documents used in previous experiments, containing 198 reviews, and (2) a larger set of documents from the same corpus, consisting of 552 reviews.

<table>
<thead>
<tr>
<th></th>
<th>Documents</th>
<th>Reviews</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corpus 1</td>
<td>16</td>
<td>198</td>
<td>66.7</td>
</tr>
<tr>
<td>Corpus 2</td>
<td>40</td>
<td>552</td>
<td>63.59</td>
</tr>
</tbody>
</table>

Table 6: Evaluation of rule-based system on customer reviews

<table>
<thead>
<tr>
<th>Response Key</th>
<th>negative</th>
<th>neutral</th>
<th>positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>0</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>neutral</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>positive</td>
<td>6</td>
<td>46</td>
<td>130</td>
</tr>
</tbody>
</table>

Table 7: Confusion matrix for corpus 1

<table>
<thead>
<tr>
<th>Response Key</th>
<th>negative</th>
<th>neutral</th>
<th>positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>7</td>
<td>31</td>
<td>4</td>
</tr>
<tr>
<td>neutral</td>
<td>0</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>positive</td>
<td>16</td>
<td>138</td>
<td>334</td>
</tr>
</tbody>
</table>

Table 8: Confusion matrix for corpus 2
Table 6 shows the results of two experiments we performed on this data, while Tables 7 and 8 show the relevant confusion matrices. Several things are immediately evident. First, the system finds erroneously many neutral sentences, i.e. there are many opinionated sentences for which no sentiment is found. This is largely because our application is designed to maximise precision over recall, and to only find opinionated sentences in cases where it is more certain. Second, the application is much better at finding positives than negatives. This is most likely because the negative recognition component still needs more work: in particular, negation is not always explicit and the negative marker can sometimes be located outside the clause, which means that we do not recognise it as being relevant. Third, the scores are a little lower than might be expected. There are a number of reasons for this. First, there are many reviews with no text, or with no obvious sentiment-containing text, but which still have a star rating that is positive or negative. Second, some comments do not correspond to the star rating. For example, someone might give a 4 star rating but a quite negative set of comments overall, or a one-star rating but some positive comments. These are known problems with using a star rating as a gold standard, although it is standard practice due to the ready-made nature of the gold standard corpus. Furthermore, ML systems are typically trained on star ratings, which biases them to this kind of behaviour, whereas our rule-based system is not trained on such data. Finally, we should point out that this work was done before subsequent improvements were made to the system.

2.10.2 US Election dataset tweets

We performed another experiment on the US election dataset tweets from the ARCOMEM crawl. For this, we randomly extracted 70 tweets from the crawl and manually annotated them with opinion (opinionated or not), polarity (positive or negative) and sarcasm (present or absent). Results are shown in Table 9. As previously, we have deliberately favoured Precision over Recall, in order to minimise wrongly opinionated results being generated, which would negatively affect the end system. Note that the sarcasm detection was perfect, largely because there were no sarcastic tweets in our corpus. For a better evaluation of sarcasm detection, see Table 1.

<table>
<thead>
<tr>
<th>Regular corpus</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opinionated</td>
<td>87.50</td>
<td>63.64</td>
<td>73.68</td>
</tr>
<tr>
<td>Opinion+polarity</td>
<td>75.00</td>
<td>54.55</td>
<td>63.16</td>
</tr>
<tr>
<td>Polarity-only</td>
<td>85.71</td>
<td>85.72</td>
<td>85.72</td>
</tr>
<tr>
<td>Sarcasm detection</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 9: Evaluation on US elections corpus

2.10.3 General tweets

We performed a third experiment on a set of general tweets collected and manually annotated with named entities as part of the TrendMiner project. The corpus is described in Section 3.1 of D3.3.1. We selected from this annotated a corpus a random sample of 400 tweets and manually annotated
them with opinion, polarity and sarcasm information, and evaluated our system against this set. We used two annotators for this and measured the inter-annotator agreement at 87.5% observed agreement for polarity detection and 91.07% for sarcasm detection. Results are shown in Table 8: we can see quite a difference here compared with the US Election tweets. Interestingly, we scored much higher on recall but lower on precision. It is not yet clear why this should be the case: the system is slightly improved since the earlier versions used for the first experiment, since we have extended the gazetteers. However, we would not expect this to have such a detrimental effect on Precision, so more work needs to be performed to determine the reasons for this. The detection of opinion polarity was actually lower than in the US elections corpus, but given the amount of sarcasm in the tweets, and the vague and often highly ambiguous nature of the content, this is not entirely unexpected. For example, one tweet read "Complete Tosca on the Tube." It is not clear whether this was a pun on the word "tosser", and was a sarcastic and negative statement about an annoying person, or if it was talking about the opera Tosca and some kind of performance (perhaps they were listening to it on their iPod). The sarcasm detection performed very well at 91% Precision and Recall. Out of 400 tweets, there were 91 sarcastic sentences, which is quite a high proportion, and many of these were not indicated by any kind of sarcasm marker.

<table>
<thead>
<tr>
<th>Twitter corpus</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opinionated</td>
<td>65.69</td>
<td>77.31</td>
<td>71.02</td>
</tr>
<tr>
<td>Opinion+polarity</td>
<td>52.61</td>
<td>61.92</td>
<td>56.89</td>
</tr>
<tr>
<td>Polarity-only</td>
<td>80.08</td>
<td>80.03</td>
<td>80.05</td>
</tr>
<tr>
<td>Sarcasm detection</td>
<td>91.03</td>
<td>91.04</td>
<td>91.03</td>
</tr>
</tbody>
</table>

Table 10: Evaluation on General Tweets

2.11 Ongoing work

There are a number of issues which still need to be handled: some of these are outside the scope of this project, however, in terms of the accuracy of the sentiment analysis application, there are many unresolved problems, some of which are not tackled successfully by any state-of-the-art system, such as sarcasm detection, opinion ambiguity, and many issues related to linguistic analysis of social media. We are working on some linguistic issues which can more easily be solved within the scope of the project, however, such as incremental improvement of the tweet handling application, better detection of negatives and their scope, better target detection and so on. We are also experimenting further with machine learning techniques and relevant features, and of adapting these tools to different domains and to German.

Another important issue involves finding clusterings of the opinions expressed in social media, according to influential groups, demographics and geographical and social cliques. Consequently, the social, graph-based nature of the interactions requires new methods for opinion aggregation and reputation finding. In co-operation with work in other WPs, especially WP2, we have begun to investigate such topics, and will continue to do so in the final months of the project and beyond.
Opinion Mining (Version 1)

Please type or paste some text (note that long texts may take a little while to analyse, and that the input will be treated as UTF-8):

We shouldn’t have been voting for Obama anyway. The machine knew that. Now THAT is what I call artificial intelligence.

Simon Macgillivray

Figure 6: Screenshot from the opinion mining demo

A simple stand-alone web-based demo of the English opinion mining application has been implemented and is now available. This demonstrates the application described in Section 2.1 for extracting and classifying opinionated sentences from text and analysing the opinions in them. Positive and negative opinionated sentences are extracted from the text along with a 4-category rating, and if appropriate, the target of the opinion (highlighted in red). The rating ranges from very negative (score of less than -0.5), negative (-0.5 to 0), positive (0 to +0.5), and very positive (more than +0.5). Figure 6 shows a screenshot. The current version works only on English text; future versions will allow the possibility of German text and/or a combination of the two within a document. We also plan to make available versions which demonstrate the opinion aggregation work described in Sections 2.3 and 2.4, as well as a version which works specifically on tweets. These demos are intended to be used for training and demonstration purposes, rather than as a replacement for the visualisations in SARA.

2.12 Discussion and Future Work

Opinion mining from text, and particularly from social media which is difficult to analyse, is still very much in its infancy in terms of research, while very much a hot topic. This means that our tools are far from perfect, although they exhibit considerable advances over the state-of-the-art in certain aspects, and there remain therefore a number of issues which have not been handled or where further research would be fruitful. These form part of our ongoing work, which will extend not only until the very end of this project, but beyond its life and into future projects. Such improvements

http://demos.gate.ac.uk/arcomem/opinions/
include use of more detailed discourse analysis in order to provide better mechanisms for scope handling -- not only of things like negation and sarcasm, but also of the opinions themselves. Recall is also a key factor: in this work, we have strongly favoured precision over recall because we believe it to be more important for the users to get good quality information. The exploration of converting the tools to German has also not been explored to its full potential, and it is clear that the German opinion mining tools we have developed in the project are very much proof of concept only: we again made a deliberate choice to focus primarily on the quality of the English tools. Further experimentation with machine learning tools could also be beneficial: while we performed initial experiments with such methods, we chose to retain the rule-based systems in the final applications since they achieved better results. Finally, more research is needed on the integration of information from other sources such as reputation and trust of authors, which could be used to highlight the most important opinions.
3 Opinion Event Detection in the Social Web

Timelines of opinions of social network users are becoming more and more frequent in the reporting of current events, as they show how public sentiment evolves. However, interpreting a simple timeline requires extensive background knowledge in order to understand the rise and fall of the aggregated sentiments. The same also holds in web archives, where users may want to look at historical events where they have no contextual knowledge at all.

In this section, we describe a method to detect interesting periods (“events”) in such a timeline based on sudden large changes (“bursts”) of aggregated sentiments. Furthermore, we describe these events using the entities involved in them. The result will be an annotated timeline that allows users to quickly understand and explore the web archive, even if they have little or no background knowledge on its content.

Our main contributions on this topic are:

- We propose a simple unsupervised event detection strategy based on detected opinions.
- We describe and implement an event labelling method based on entities and relations described in the documents. We improve, adapt, and formalize the problem of dynamic relationships and event extraction, earlier introduced on query logs in [Error: Reference source not found].

3.1 Related Work

Information retrieval algorithms and applications have been successfully proposed in the last twenty years. Surprisingly little work has been conducted on Temporal Retrieval on large temporal collections. In this area, [2] and [5] present approaches for extracting temporal information in order to cluster search results. Lately, recent works concentrated on annotated timelines to retrieve temporal information.

Google’s news timeline9 is an experimental feature that allows a user to explore news by time. Extensions to document operations such as comparing the temporal similarity of two documents in the context of news articles are presented by Makkonen and Ahonen-Myka [Error: Reference source not found]. An interesting approach that combines event detection and tracking with timelines as a browsing interface is presented by Swan and Allan [44]. Time information is also used in temporal mining of blogs to extract useful information [Error: Reference source not found].

The problem of event detection has been examined using news articles as part of a broader initiative named topic detection and tracking [1]. The holy grail in this body of work has been to automatically acquire a landscape view of a document collection, which answers in a compact manner the questions of: “What happened?” and “What is new?”. The event detection task can be divided into two categories: retrospective event detection (RED) and new event detection (NED), also called “First Story Detection”, in either on-line or off-line mode [49]. Retrospective event detection refers to the detection of previously unidentified events from an accumulated historical collection. More recent works such as [Error: Reference source not found] and [Error: Reference source not found] apply RED using two kinds of information contained within articles: contents, as previously, and timestamps. Authors of both papers prove the importance and usefulness of them in detecting events, going beyond the focus of previous works on finding only better utilizations of contents. In conclusion, the authors propose and improve a probabilistic model for RED, in which both content and temporal information are used. New event detection refers to the discovery of the onset of new events, either from live feeds in real-time (online model) or under a closed-world assumption. In

9 http://www.newstimeline.googlelabs.com
particular, the new event detection task is defined to be the task of identifying new events in a set of stories. Each story is processed in sequence, and a decision is made whether or not a new event is discussed. A decision is made after each story is processed [7] and [53]. In this work, we choose to apply a retrospective event detection algorithm using data historical collection in the social network, in order to build a timeline of opinions on the extracted events.

### 3.2 Opinion Event Detection System

#### Definitions

An event is a tuple \( (i, c, d) \), where \( i \) is a time interval, \( c \) is a change indicator and \( d \) is an event description. The event description \( d \) can be for example a weighted vector of prominent keywords or a textual summary. The change indicator \( c \) gives type and direction of change (e.g. average sentiment more positive, average subjectivity lower).

A proto-event \( (i, c, D) \) consists of the time interval \( i \), the change indicator \( c \), and the set of evidence documents \( D = \{d_1, d_2, ..., d_k\} \) used to find the proto-event.

#### Architecture

The system consists of three components, depicted in Figure 7. The first component is the opinion timeline extraction which takes as input the raw, time-stamped documents corresponding to an entity of interest, and outputs an opinion timeline. The second component is the event detection component, which takes as input the opinion timeline and outputs a list of proto-events. The last component is the event labeling component, which transforms the proto-events into events with a full description, using the evidence documents of the proto-events as source material.

The opinion timeline extraction runs an opinion detection algorithm on the source documents and extracts all opinions about the target entity. They are grouped into buckets \( w_j \), where each bucket contains all opinions in a certain time period. For each bucket we can compute a number of different values:

- the sum and average of the sentiments;
- the sum and average of the subjectivity as the sum resp. average of the absolute value of the sentiments;
- the number of positive and negative opinions;
- the total number of opinions.

Each of those values is computed for each bucket, giving us a number of different time series. Using several time series instead of only using the average sentiment allows us to detect and distinguish different kinds of opinion change, e.g. an increase/decrease in average sentiment vs increase/decrease in polarization.

The event detection component takes the opinion timeline and runs a burst detection algorithm against each contained time series to find opinion change events. As the time series are highly cor-
related, we will often have parallel bursts in more than one time series. We merge these bursts if
they have an overlap larger than \(\omega\%\). Each of those bursts corresponds to a proto-event, where
the timespan \(ii\) is equal to the timespan of the burst, in the case of merged bursts to the union of
the original bursts’ timespan. The change indicator of the proto-event is the name of the time series
where the burst occurs and the sign of the burst (values in the burst are larger/smaller than outside). We assign the documents published in \(ii\) containing the target entity to the proto-event as ev-
dence documents.

Finally, the event labelling component takes the proto-events and detects a useful label for them
based on the evidence documents. We create the label from the \(k\) terms of the evidence
documents that have the highest tf-idf value. However, based on the use case, many other
possible labelling algorithms such as multi-document summaries are possible.

### 3.3 Opinion Timeline Extraction

The timeline extraction module uses the output of the opinion mining module described before in
Section 2.1. Starting from a given set of entities, e.g. those specified in the crawl specification, we
gather all opinions where that entity is the opinion target. We also find the date of each opinion.

For the date, we assume that the publication date of the containing document is equal to the date
when the opinion was expressed. This assumption does not hold e.g. when we have reports in
news articles about opinions expressed by third parties at an earlier time. However, even in this
case the time difference is rather small, and in the main focus area of our research, namely social
media, the date when an opinion is expressed and the publication date coincide (as do the author
of the document and the opinion holder). Therefore this approximation should be enough and the
additional complexity in runtime and algorithms that the detection of such reports would require,
could improve the quality of our results only by very little.

In addition to opinions where the opinion mining module has detected an explicit target, we also
assign the most frequent entity of a document as the target of the contained opinions without a de-
tected target. This greedy algorithm tries to improve the recall of opinions with a given target at the
cost of some precision. The results may not be good enough to show to the end user, but in the
event period detection stage the additional instances smooth out the data and reduce the number
of noisy events detected.

We sort the opinions for a target entity by the date when they were expressed and partition them
into weekly bins. For each bin we compute the average sentiment, the number of positive and neg-
ative opinions, as well as the sum of positive and negative opinions as the number of subjective
opinion mentions.

### 3.4 Opinion Event Detection Algorithm

An opinion event is for us a period of time where there is a sudden and significant change in either
the number or the value of opinions expressed about an entity. We can therefore use burst detec-
tion algorithms to find the time span of an event. A burst detection algorithm takes as input a list of
values \([v_1; v_2; \ldots ; v_n]\), where each value \(v_i\) represents a time period \(i\) of fixed size and outputs the
ranges of indices that differ significantly from their context.

For our experiments we used a variant of Kleinberg’s algorithm [19]. We chose this algorithm be-
cause it is less sensitive to noise in the input value and therefore less prone to split continuous
events into multiple parts than other, threshold-based, methods. Kleinberg’s algorithm models a
stream as being produced by a probability distribution that has multiple states. Each state corresponds to a specific output frequency. The states are arranged as a sequence such that low output frequencies correspond to small state indices and high output frequencies to high ones. Each state transition is associated with a cost. The burst detector finds an optimal state sequence that balances the difference of the value \( v_i \) and the expected value of the state at index \( i \) against the cost of state transition. Bursts correspond to the indices where the optimal state sequence has a state other than the base state.

We adapted Kleinberg’s algorithm in two ways. Firstly, the algorithm is designed only for frequency values. In our implementation, we introduce states that are shifts by a multiple of the standard deviation of the input data. This allows us to find bursts in any real-valued time series. Secondly, the algorithm can only detect positive bursts, as it only models states where the value is above average. We introduce additional burst states that correspond to values below average and are thus also able to find negative bursts or slumps, i.e. periods where the value is significantly below average. The latter modification is especially useful for real-valued time series such as the series of the average sentiment.

To detect events, we run the modified burst detection algorithm on each of the time series described above and assign event types based on the bursting time series and direction of burst (positive/negative), as depicted in Table 11. As the different time series are often correlated, we typically find similar bursts on more than one time series. We merge the resulting event when they overlap for at least \( \omega \% \) of their durations by setting the time span of the final event as the union of the individual event’s time spans. The final event also has multiple event types. When \( \omega \) is too small, different events often blend into each other through repeated merging of intermediary events, therefore we chose \( \omega = 25 \% \) in our experiments.

<table>
<thead>
<tr>
<th>Time series</th>
<th>positive</th>
<th>negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>average sentiment</td>
<td>sentiment+</td>
<td>sentiment-</td>
</tr>
<tr>
<td>avg. absolute sentiment</td>
<td>subjectivity+</td>
<td>subjectivity-</td>
</tr>
<tr>
<td># positive opinions</td>
<td>positive+</td>
<td>positive-</td>
</tr>
<tr>
<td># negative opinions</td>
<td>negative+</td>
<td>negative-</td>
</tr>
<tr>
<td># opinions</td>
<td>frequency+</td>
<td>frequency-</td>
</tr>
</tbody>
</table>

Table 11: Detected types of opinion events

3.5 Opinion Event Description

The final component provides the descriptions of the detected events. Here we tested two different approaches: The first approach is to label each event with some descriptive keywords; the second is to find and connect the most prominent entities involved in the event.

For the first approach we use the documents that contain opinions about the target entity and are in the time span of the event. We rank the terms in these documents by their tf-idf value and choose the \( k \) highest values. The number of keywords \( k \) can be easily changed to accommodate different use cases.

The second approach is based on the Dynamic Relationship Discovery algorithm [Error: Reference source not found]. We start by finding potentially relevant named entities. Das Sarma et al. use a pre-defined set of potential entities. This requires domain knowledge about the relevant entities and is thus not applicable in our scenario. Instead we use entities that also occur in the evidence documents. However, this includes many spurious entities (e.g. from unrelated texts or site boiler-
plate such as author names). As the processing cost of the following steps increases quadratically in the number of entities, we remove spurious entities using a threshold of at least 5 occurrences.

The algorithm creates a relation between entities if they occur together in the same time span. As our candidate related entities are by definition in the same weekly time span, we can look at periods smaller than a week. This also allows us to better separate different strands of an event that occur in close proximity. We create daily frequency counts for the target as well as the candidate entities, run burst detection on the resulting time series and link entities that have bursts at the same time. From the resulting set of candidate relations, we only keep those that co-occur significantly more often in the co-burst period.

Starting from the target entity, we extract temporally connected components of the relation graph, to detect distinct sub-events which consist of groups of related entities (e.g. “Obama”, “Romney”, “Presidential Debate”). These connected components, which consist of a set of entities and the relationships between them, are added to the detected events as the event description. The user interface can show the entities and relationships to the user as a network graph, so that they can gain an immediate understanding of the major actors in an event and how they are connected. In addition, events that occur at a similar time and share entities can be linked to each other, allowing an easy exploration of related events.

### 3.6 Exploration and Evaluation Tool

![Figure 3: List of detected events for the topic “abramoff bush”](http://okkam.l3s.uni-hannover.de:8090/)

To show the results of the algorithm, we developed a web application\(^\text{10}\) that allows users to explore a document collection using the detected events. The document collection used was the TREC-BLOG dataset, as described in Section 3.2.1. On the first page of the application (see Figure 3), we show a list of all detected events, including the corresponding time period and the extracted labels. From here, the user can click through to a detailed view of an event (Figure 4), which shows all available metadata about the event, the label and also a list of the documents belonging to the events. The screenshot depicts an extracted event for the entity “ariel sharon” as well as some timelines and related documents. Elements of the interface include: (1) frequency of

\(^{10}\) Available at http://okkam.l3s.uni-hannover.de:8090/
topic mentions, (2) average opinion, (3) event metadata, (4) titles and links to documents from which the event was extracted, and (5) event label (keywords).

In addition, the system allows a keyword search in the dataset, showing the distribution of result documents over time in a graph (Figure Error: Reference source not found). Together these functions allow an easy and efficient exploration of the document collection.

3.7 Evaluation

3.7.1 Evaluation method

We ran our algorithm using the dataset described above and extracted the labeled events. Two raters annotated at least 64 randomly chosen events as to whether they were correct events, and provided a rating on a scale from 1–5 (1 worst, 5 best) for the labels of correct events. In total we got 134 ratings for 72 distinct events.
3.7.2 Results

Opinion Event Detection

The results for the event detection show that it is hard even for human annotators to detect when an opinion event happens, as we only got a moderate inter-annotator agreement (calculated as Pearson’s correlation) of 0.52.

For 44 of the 72 events both evaluators agreed that the event was correct, resulting in a precision of 61%. If we count an event as correct if at least one of the annotators marks it as correct, which is reasonable given the low inter-annotator agreement, the precision increases to 72%.

In this work, we did not specifically check the recall of the event detection algorithm, as we had no ground truth to compare with. Creating such a baseline would require an explicit definition of when a real-world event involves an opinion change. This is however an inherently subjective question. Therefore we concentrated in our work on correctly identifying the most prominent events. In order to find the largest number of possible events, end user applications should allow an adaptation of the event detection parameter to more sensitive levels.

Opinion Event Description

Finding good event descriptions is necessary to make the extracted events useful as an exploration tool for a web collection. In this work, we used a tag cloud of relevant keywords, which provides a simple and well understood method to provide a brief overview of the content.

In our experiment, two annotators gave a rating of 1–5 (1 worst, 5 best) to each event determined as relevant earlier. The inter-annotator agreement, calculated as Spearman’s rank correlation $\rho$, was moderate with 0.39. This again shows that this problem has a high amount of subjectivity and is therefore hard to solve algorithmically.

The average rating for the labels was 3.25. An analysis of the labels revealed that many contained non-content keywords from web page boilerplate text. When several documents containing such keywords are included in an event, these non-descriptive terms appear to be more relevant than other keywords.

Error: Reference source not found shows some examples of extracted event descriptions. Figure 4 above presents a screenshot of the application used during the evaluation, which shows the event metadata, the frequency and average Opinion time series used to detect bursts and the source documents.
### Table 12: Sample events extracted using burst detection on opinion values

<table>
<thead>
<tr>
<th>Topic</th>
<th>time span</th>
<th># docs</th>
<th>description</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ariel sharon</td>
<td>2005-11-29–2005-12-06</td>
<td>44</td>
<td>isra, israel, likud, palestinian, minist, prime, jewish, arab, polit, parti</td>
<td>opinion+</td>
</tr>
<tr>
<td>brokeback mountain</td>
<td>2005-12-02–2006-02-17</td>
<td>4085</td>
<td>movi, gai, lm, cowboi, heath, ledger, oscar, gyllenha, award, stori</td>
<td>negative+</td>
</tr>
<tr>
<td>bruce bartlett</td>
<td>2006-02-03–2006-02-17</td>
<td>23</td>
<td>impostor, bush, reagan, republican, bankrupt, betrai, legaci, georg, critic, conserv</td>
<td>subjectivity+</td>
</tr>
<tr>
<td>muhammad cartoon</td>
<td>2005-12-03–2006-02-25</td>
<td>772</td>
<td>muslim, danish, ciliegi, scandal, controversi, islam, arab, newspap, cameroonian, hama</td>
<td>opinion+, frequency+</td>
</tr>
</tbody>
</table>

#### 3.8 Evaluation

We evaluate our approach on the data from the TREC-BLOG challenges 2006 [16] and 2008 [17]. We removed all the known spam blogs from the TREC-BLOG 2006 dataset and extracted the textual content using the GOOSE system\(^{11}\), removing superfluous content like navigation and sidebars. This gives us a dataset of 26.7M documents spanning the time between 2004 and 2008. All documents without a publication date or with a date outside this time range are ignored.

The target entities are derived from the 50 topics of the BLOG 2006 opinion retrieval challenge. The topic names correspond to one or more entity names. For Person entities, we also added additional queries, so that mentions of the form “firstname lastname” as well as only “lastname” are found, if the last name is unambiguous enough (for example, we add “cheney” but not “bush”). We use all documents that contain all of these names as the source documents for finding the opinion events for this topic.

The evaluation of this component is currently still ongoing. We evaluate using two different criteria: (1) the precision of the detected events, and (2) the quality of the provided labels. The precision is checked against manually created event timelines for at least some of the evaluation entities. We are working on criteria to include events into the set of opinion events. To do so, we compare the results of our component to that of a standard event detection algorithm. The quality of the provided labels will be assessed using human annotators, who will be asked to rate each label according to how well it describes the event. They will also get a list of the evidence documents for this event to allow them to perform this judgment. We will perform this evaluation using both kinds of labels.

\(^{11}\) https://github.com/GravityLabs/goose
3.9 Integration into the ARCOMEM system

The opinion dynamics component will be integrated into the final ARCOMEM system in the next few months. During the development we used simplified implementations of some sub-modules as stand-ins until the final modules from WPs 3 and 4 were completed. In the following, we will describe the new sub-modules.

We developed a naïve opinion classification module (depicted in Figure 9) that allows us to extract opinions quickly from a large number of documents, albeit at much lower levels of accuracy than that of a linguistically motivated model such as the ARCOMEM one described in Section 2. Our current module uses polarity scores from the AFINN [18] sentiment dictionary and some simple negation rules based on a chunking of the input text and a word list of negating terms (“negators”). We calculate a sentiment for each sentence containing at least one sentiment indicating word as the average of the final polarity scores. All sentences’ sentiments are aggregated into a document opinion using the average of the sentences’ sentiment scores.
The experiments described above were conducted using a stand-in component for the opinion detection, as the opinion detection modules were not finished when we started the implementation and evaluation of the opinion dynamics module. Since then, we have implemented an integration of the opinion detection module such that it can be used to find opinions about the target entities. The higher accuracy of the opinion detection module will improve the relevance and accuracy of the detected opinion events.

3.10 Future Work

In the future we are going to explore new methods to provide an event description that help users better understand the data. We will look at different types of users (e.g. expert users looking for specific types of information vs. users trying to get an overview of events in a certain topic area and time period) and develop description methods for those scenarios, such as textual snippets that cover the entities relevant to an event and their relations.
4 Image-opinion Identification and Classification

The opinion analysis of visual data has a number of applications within the ARCOMEM scenarios. Specific examples include sentiment-based search (e.g. “find me positive media about X”), and the determination of the key influential media objects. The general aim of this task is to investigate novel techniques that allow exploration and classification of the ways in which images are used to form opinions. It considers two aspects: the diversity in image content of images used to represent the same opinion, and the diversity in opinions of a given image as it is re-used throughout an archive.

Images are often used to illustrate the opinions expressed by the text of a particular article. By themselves, images also have the ability to convey and elicit opinions, emotions and sentiments. In order to investigate how images are used in the opinion formation process, we have been developing tools that (a) allow the reuse of images within an archive to be explored with respect to diverse time and opinion axes; and (b), allow in-depth analysis of specific elements (in particular, the presence and expression of human faces) within an image to be used to quantify opinion and sentiment.

4.1 Facial Expression Recognition

The analysis of the facial expressions of people can tell us a lot about the sentiment associated with an image or the opinions associated with the image in the broader context of the document within which the image is embedded. Following on from D4.2, we have continued to explore and develop techniques that can potentially help us move towards accurate facial expression recognition. From the beginning, however, it must be stressed that these techniques are still relatively immature, and applying them effectively to real-world images is a massive challenge.

In D4.2, we described our initial experiments with a set of parametric modelling techniques called Active Shape Models [9] and Active Appearance Models [10]. We have continued to explore this with a hybridisation of these two approaches called a Constrained Local Model (CLM) [41].

Our OpenIMAJ CLM implementation fits a statistical 3D face model to the detected face. The 3D face model can be used to locate facial keypoints within the image and also to determine the pose of the face. As with the ASM and AAM, the CLM model is a form of parameterised statistical shape model called a "point distribution model"; this means that the 3D model has an associated set of parameters which control elements of its shape (i.e. there are parameters that determine whether the mouth is open or closed, or how big the nose is). When we fit the model to a detected face in an image or video, we get back a numerical vector of parameters that describe the shape of the face.
Using the shape parameter vector, under constrained conditions in our office, we have had some success in training classifiers that can detect certain facial action units [14]. Action units basically describe certain muscle movements within a human face, and correspond to things like raising the eyebrows and opening the mouth. Facial action units are attractive because they provide a direct way to classify emotions; for example an expression of happiness in a face is represented by a combination of AU6 and AU12. An example of the detection of action units in a laboratory setting is shown in Figure 9. The left image is the raw video with the classifications overlaid; the second from the left shows the shape model; the green bars illustrate the model parameters; the right image is the aligned face mask used for face recognition and gender classification.

Unfortunately, training a system to detect the full set of action units required for the different emotional states is difficult due to the lack of publicly available data. A second problem directly relates to the facial models themselves, in that it is quite difficult to build a shape model (ASM, AAM or CLM) that will accurately fit all faces, which is essential for the accurate measurement of the shape parameters needed for expression classification. A third and final problem is that accurate detection of a face is required to initialise the fitting of the CLM model; whilst face detection techniques are quite mature, they can still have major problems working in real-world images where the faces are not exactly frontal to the camera, or there are shadows or contrast issues. Figure 10 shows some examples of our CLM model applied to examples from the ARCOMEM US Elections Crawl. Notice in particular how poorly the model fits to Michelle Obama’s face (and causes the misclassification of gender as a side effect).

In terms of the ARCOMEM platform, the CLM facial analysis model is already integrated, and is capable of producing some annotations related to facial expression. However as previously described, the performance is generally quite low on the kinds of images found in web-crawls, so
this functionality should be regarded as rather experimental. That being said, this is an important research area that we are continuing to explore and push. In addition to shape-based expression classification, we are also exploring some simpler models based on the classification of pixel patches around facial keypoints; in the shorter term these may turn out to be more robust on real-world imagery.

Figure 10: Examples of the CLM model fitted to images from the US Elections crawl.

4.2 Image Reuse Analysis

Investigating the way images are reused across opinionated documents on the web allows us to estimate, through association, the opinion of images for which the sentiment is unknown and, subsequently and conversely, estimate the sentiment of articles which are ambiguous.
To estimate the opinion of an image, the sentiment of which is unknown, we can aggregate the opinion from the set of all articles which contain this image. If there is an overall consensus in the opinion of articles where the image is used, in general we can assume the image supports this opinion and therefore inherits its opinion.

When an article has conflicting opinions, or an opinion is not evident, then the image may be able to provide clues as to the article’s sentiment: if it contains an image which has been reused many times in articles that have particular opinions, the ambiguous article can be associated with that opinion through the association with the image. Because the image matching is purely visual, this technique will work across language barriers, such that articles in a language that cannot be analysed could still have sentiment scores associated with them.

The pipeline for is as follows:

**Textual Opinion Analysis** - The text associated with articles needs to be analysed to determine whether sentiment for entities in the article can be extracted.

**Image Duplicate Detection** - The links between articles utilising the same image needs to be made.

**Analysis of the Opinions** - The sentiment from the articles associated with duplicate images needs to be analysed to find trends or clusters.

For the ARCOMEM system, the first part of the pipeline is achieved using the GATE offline analysis tool. This generates sentiment polarity and scores for text extracts in web resources. The web resources may have images associated with them. As all this data is stored into the ARCOMEM knowledge base, it is possible to query for those images. The following query finds images belonging to web-resources that also have opinionated text:

```sql
where {
}
```

Image duplicate detection is described in D2.3 and is based on finding connected components in a graph constructed from collisions in a hash table of 128-bit sketches of SIFT features (locality sensitive hashing). The images from the opinionated articles can all be fed into the image duplication algorithm to find matching images.

Once we have a set of duplicate images and the articles with which they are associated, it is possible to perform analysis or create a visualisation to investigate the way the image is reused. We have created a visualisation (both in Java and as a web application) that allows the display and investigation of a set of objects that are ordered on some diversity axis. In our case, the objects are the images and the diversity axis the opinion of the article. Figure 11 shows an example of the web application displaying the duplicate images arranged by an opinion score.
4.3 Future plans and challenges

So far we have only considered the overall opinion score of an article. But this may, in some cases, be ambivalent across the entities in the article. When opinions are associated with specific entities in an article, the matching of images is still important to find links between articles, but it is also important to make the association with the entities in the image. For example, article $a$ uses image $I$ and may be negative towards entity $e$ and neutral about entity $f$, but article $b$ which also uses image $I$ may be negative towards entity $f$ and neutral about entity $e$. In this case the link should not be made, as despite having similar opinion scores and using the same image, they are not of the same cluster. The context of the opinion should impact the match. The problem is that identifying entities in visual terms in an image is considerably more difficult than in text. Although we are working towards this in our work in WP3, we do not think that it is sufficiently mature and accurate to be used in this way yet. Indeed, datasets that allow the testing of this type of occurrence would be very expensive in terms of time and effort to construct.
5 Multimodal Opinion analysis (SOTON, LUH)

Opinion and sentiment are rather complex notions that can be very difficult to predict purely from visual data alone. A more fruitful approach is to consider the image (or other media modality) in the context in which it appears, whether that be an image on Flickr or video on YouTube surrounded by tags and comments provided by humans; or an image in a news item surrounded by the text of the article to which it relates. State-of-the-art research on the sentiment analysis of images (see e.g. [50], [47], [Error: Reference source not found], [43]) has already begun to explore how the analysis of textual content and the analysis of visual content can complement each other.

As reported in D4.2, the collaborative image sentiment analysis experiments between L3S and SOTON that started in LivingKnowledge have continued during ARCOMEM. Specifically within ARCOMEM, we have been investigating two aspects related to opinion formation: privacy and attractiveness, and we are now just coming full circle and beginning to revisit our older work on sentiment classification by applying new features. In all cases we have made use of large sets of images and associated metadata gathered from Flickr in order to have datasets which we can perform experiments on. We have also made use of crowdsourcing to annotate these datasets and verify our results.

5.1 Attractiveness classification of images

When considered within the context of the article or page in which it appears, we hypothesise that the attractiveness of a photograph can be a strong indicator of the opinion and sentiment expressed by the article. Currently, we are only beginning to scratch the surface of this area, but we have been investigating building computational models of attractiveness that take into account both visual features as well as surrounding contextual tags. The initial outcome of this work is a system called NicePic!, which is described below. A short demonstration paper of the NicePic! application has been submitted to ACM Multimedia 2013.

5.1.1 NicePic!

The rapid increase in size of online communities and the availability of large amounts of shared visual data make discovering relevant content a difficult task. For instance, thousands of new photos are uploaded to Flickr every minute, making effective automatic content filtering techniques a necessity.

Flickr photos are accompanied by a variety of metadata such as tags, number of views, user comments, upload date, etc. The Flickr search interface exploits the explicit and implicit ratings in the metadata to infer rankings. For instance, the number of views is an indicator for the popularity of a photo. Adding a photo to one’s favourite list is probably the most direct positive indicator of relevance assignment in Flickr, and is an explicit expression of interest in the photo.

However, for recently uploaded photos community feedback in any form might not yet be available. Furthermore, many photos are just sparsely annotated which might prevent text-based search and mining methods from retrieving this potentially attractive content.

To address these problems, we have developed the NicePic! application, based on a web service for automatically classifying and ranking photos according to their attractiveness. We exploit the vast amount of social feedback available in Flickr, to obtain a training set of photos considered as more or less attractive by the community. This allows us to build classification and regression models for aesthetic inference based on multi-modal visual and textual features, and to apply them to identify new attractive content. In a wider system context, such techniques can be useful to...
enhance ranking functions for photo search [40], and, more generally, to complement mining and retrieval methods based on text, other metadata and social dimensions.

The NicePic! application allows users to explore the live Flickr photo stream in an attractiveness-centric manner. NicePic! shows the top photos submitted to Flickr during the last hour, day and week according to our aesthetic inference model, enabling users to rapidly see the best content uploaded to Flickr in the recent past. In addition, users can introduce query terms in the search box to get results relevant to their information needs. These results are organised into two columns, separating the most attractive from the least attractive photographs.

5.1.2 System Architecture

This section describes the main components of the system. The architecture is illustrated in Figure 12. Firstly, we build a training set of images with and without favourite assignments. In the second step we extract visual, and if available, textual features from the images. We then train an SVM classifier which is used by our system for identifying potentially attractive visual content.

![Figure 12: Overview of the NicePic! system](image)

A brief overview of the system is given below; a fully detailed description of the underlying approach can be found in [39], albeit with a slightly different selection and combination of features.

Data

We randomly selected time periods of 20 minutes from a time span of 5 years 2005-2010. From each of the periods we selected at most 5 pictures from Flickr with the highest number of favourite assignments as positive examples, as well as the same number of photos without favourite assignments as negative examples. We stopped after obtaining a set of 200,000 photos from each class.

Features

Even though aesthetic and artistic quality cannot be quantitatively computed, it has been shown that certain visual features of images have significant correlation with them. For instance, appealing images tend to have higher colourfulness, increased contrast and sharpness [39]; we apply image analysis to extract these features. Bag-of-words textual features extracted from the
title and tags can also provide information about the image quality, as illustrated by the classifier performance shown below.

**Classification**

Using the dataset described above, we trained Support Vector Machine classifiers to distinguish between the classes using the visual features, textual features and combinations of both feature types. The quality measures were the precision-recall curves as well as the break-even points for these curves.

Initial results are summarised in Figure 13, where the break-even points for each curve are shown by the black dots. The combination of textual and visual features have shown the best applicability resulting in BEP 0.84. Classification with only visual features also produces promising results (BEP 0.67), and can be useful if no or insufficient textual annotations are present.

![Figure 13: NicePic! classifier performance with different data modalities](image)

### 5.1.3 Future plans and integration with the ARCOMEM system

The image features used in the NicePic! system are all part of OpenIMAJ, and also have implementations that can be used in a Map-Reduce system (which is how the features were extracted for the training set). Integration with the ARCOMEM system is relatively trivial, although not yet complete; in particular, we need to investigate how we can make use of any text-based features within the system.

The current iteration of NicePic! is only using a rather basic set of global image features. We have already started experimenting with more advanced visual-word features (quantised SIFT and variants) and also started looking at a number of attractiveness-specific features that detect elements often used by photographers in choosing/framing their pictures (for example, Bokeh, photographic composition, etc.).
5.2 Privacy classification of images

The privacy classification experiments described in D4.2 were extended and published as a full paper in SIGIR 2012 [52]. In addition, we extended the work into an online demo/prototype called PicAlert! that has two features: firstly, it is able to predict the privacy of photos uploaded by users, and secondly, it has a search interface that is able to provide a privacy-diversified search of Flickr. Figure 14 shows a screenshot of the classification interface. Here, an image of two friends has been uploaded to the system, and classified as being potentially private with a high confidence. Figure 15 shows a screenshot of the diversified search interface. Here we have searched for images tagged "Christiano Ronaldo" and classified them as either public or private. We can see that the system has been able to mark the beach pictures as private and the football images as public. A paper describing PicAlert! also appeared in CIKM 2012 [51].

![Figure 14: The NicePic! classification interface.](image)

Why this decision? The top features used for classification are:

Mouse over the features in the tables for more details

<table>
<thead>
<tr>
<th>SIFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog-sift-flm12k-md1N_1353</td>
</tr>
<tr>
<td>dog-sift-flm12k-md1N_6571</td>
</tr>
<tr>
<td>dog-sift-flm12k-md1N_2016</td>
</tr>
<tr>
<td>dog-sift-flm12k-md1N_11243</td>
</tr>
</tbody>
</table>

5.3 Sentiment classification of images

Following on from our previous work [43], we are currently working on repeating the original experiments using an improved set of image features, including Dense SIFT, HOG [11], PHOG [6] and PHOW [6] features. In addition, we would like to investigate a more granular classes of sentiment (rather than just the positive-negative polarity predictions in the original work).

5.4 Evaluation

Previously our work in this area has been focused on learning opinion-bearing features from images. We have looked at domain-specific features (such as facial expressions) which are slightly more mature in the literature, and we have also looked at non-domain-specific features such as using colour and texture to learn sentiment. Typically, all our techniques require a large amount of manually labelled training data; for most experiments this is data provided through implicit crowdsourcing (i.e. the labels that users of sites like Flickr attach to their images). It is clear from our investigations that although the correlations in some of these experiments have shown
potential, it would be also be interesting to investigate (i) how difficult labelling tasks are in a real-world scenario for humans, (ii) whether we can leverage the innate ability of human labelling through crowd-sourcing and (iii) whether we can improve on the current state-of-the-art methods for utilising crowd-sourced labels.

To this end, we participated in the crowdsourcing task at MediaEval 2013, a multimedia challenge-based competition, which was able to provide us with ground-truth data to test our algorithms for improving crowd-sourced labels and learning models for image labelling. In the following section, we describe the work we undertook as part of this challenge.

5.4.1 Crowd-Sourcing Image Labels

Crowdsourcing is increasingly becoming a popular way of extracting information. One problem with crowdsourcing is that the workers can have a number of traits that affect the quality of the work they are performing. One standard way of dealing with the problem of noisy data is to ask multiple workers to perform the same task and then combine the labels of the workers in order to obtain a final estimate. Perhaps the most intuitive way of combining labels of multiple workers is through majority voting, although other possibilities exist.

The aim of the 2013 MediaEval Crowdsourcing task was to explore techniques in which better estimates of the true labels can be created. The ground-truth datasets for the task contain Creative Commons images as well as metadata associated to those images, and annotations that have been generated by two group of annotators (AMT workers and trusted annotators). One dataset contains 4835 images, while the second contains over 30,000. Both are related to the fashion industry and have been labelled with up to 262 fashion categories that were gathered from Wikipedia. In this case, the images were category labels of items of clothing rather than sentiment-based; however, a rather more vague question “is this fashion?” was also asked. Such broad questions that, in certain cases, will undoubtedly have unclear responses, mirror the kinds of responses we find in sentiment-based labelling, so we hope that our work to aid the accuracy of these specific cases could help in similar scenarios.

Our run submissions for the task explored a number of techniques: probabilistic models of workers (i.e. estimating which workers are bad, and discounting their votes), additional crowdsourcing of images without a clear majority vote, and joint probabilistic models that take into account both the crowdsourced votes as well as extracted features. Full details can be found in Error: Reference source not found.

Overall, we had the best performing runs in the MediaEval competition, although the results were interesting; in all cases (across all the participants), it was found that the generally accepted models that tried to estimate the performance of each crowd worker did not perform particularly well (i.e. were slightly worse than majority voting). The only really big improvements came from the addition of extra crowdsourced data. Currently, we are collaborating with all the task participants on performing more experiments which will be presented in a future publication.

5.5 Outlook and future work

Mining opinions and sentiment from multimedia objects remains an unsolved problem in general, although a number of successful techniques have been created that work in limited domains. This situation will inevitably improve in the future as the technology becomes more mature. In terms of general multimedia opinion mining, one problem that has not seen any attention is that of intent. Questions like “what is the intent of the placement of this image in this article?” are massively
important to determining the opinion associated with the image in the context of the article. This opens up a whole new area of research for the future that requires the collaborative efforts of NLP research, multimedia analysis and domain expertise (the article might appear in a left-wing, right-wing or satirical context; this might not however be obvious from analysing the document itself).

In terms of specific future work, we are currently collaboratively working with the other teams that took part in the MediaEval crowdsourcing task to investigate cost-optimal approaches of getting labels for images from crowd workers. This is being prepared with the aim of putting together a conference submission early next year.

As we have seen, the biggest challenge in image-opinion analysis stems from the need to have very large amounts of high quality ground-truth and training data. As we have already demonstrated, it is possible for us to gather some forms of data from online repositories such as Flickr; however, this is usually only good for coarse-grained notions of opinion. Another issue is that the textual data associated with Flickr images is often very limited, especially when compared to news articles with large bodies of text and imagery, for example. An open challenge as we lead up to the end of ARCOMEM and beyond is to investigate if text-based opinion analysis tools can be better leveraged to help us build richer multimedia datasets.

Figure 15: The NicePic! diversified search interface.
6 Summary

This deliverable has described the continued development of tools for opinion mining from text and multimedia in the ARCOMEM system, building on the initial work described in D4.2. The main highlights of this work are as follows:

- Improvement of the original pipeline for opinion mining from text, with new linguistic subcomponents, including sarcasm detection, hashtag decomposition, and opinion aggregation (interestingness);
- Summarisation and aggregation of opinions over different elements (entities, paragraphs, documents);
- Improved handling of opinion mining from Twitter and other forms of degraded text;
- Development of a machine learning application for handling opinion mining from text, and evaluation on product reviews;
- Development of methods for finding opinion events and measuring their dynamics;
- Implementation and evaluation of the event dynamics model;
- Development of a Constrained Local Model approach to Facial Expression recognition and implementation in OpenIMAJ;
- Integration of a pipeline for image reuse analysis;
- Development of the NicePic! system, building computational models of attractiveness;
- Continuation of evaluation of the image-opinion analysis with expanded features.

Research on these topics is still ongoing and is expected to extend beyond the life of this project, since these topics are cutting-edge and still very new. It should be noted that not all of the research carried out in this WP has led to fully-developed and completed tools, and some components have not made it into the final integrated system, due to their exploratory nature. To our knowledge, however, this is the first work that attempts to combine such kinds of textual and multimedia analysis tools in an integrated system, and while there remain many unsolved issues, we have paved the way for some interesting continuations to this work in future projects.
7 References


40 J. San Pedro, T. Yeh, and N. Oliver. Leveraging user comments for aesthetic aware image search reranking. In WWW 2012.

41 J. Saragih, S. Lucey, and J. Cohn, Face Alignment through Subspace Constrained Mean-Shifts, IEEE International Conference on Computer Vision (ICCV), 2009

42 A. Sayeed. An opinion about opinions about opinions: subjectivity and the aggregate reader. Proc. of NAACL 2013, Atlanta, USA.


