Extracting the Discussion Structure in Comments on News-Articles

Anne Schuth
Universiteit van Amsterdam
aschuth@science.uva.nl

Maarten Marx
Universiteit van Amsterdam
marx@science.uva.nl

ABSTRACT

Several on-line daily newspapers offer readers the opportunity to directly comment on articles. In the Netherlands this feature is used quite often and the quality (grammatically and content-wise) is surprisingly high. We develop techniques to collect, store, enrich and analyze these comments. After giving a high-level overview of the Dutch 'commentosphere' we zoom in on extracting the discussion structure found in flat comment threads; people not only comment on the news article, they also heavily comment on other comments, resembling discussion fora. We show how techniques from information retrieval, natural language processing and machine learning can be used to extract the 'reacts-on' relation between comments with high precision and recall.

Categories and Subject Descriptors

H.3.1 [**Information Systems**]: Information Storage and Retrieval—*Content Analysis and Indexing*

Keywords

Web mining, Web data extraction

1. INTRODUCTION

In the past few years, the World Wide Web saw some fundamental changes among which the phenomenon known as *user generated content*: users, as opposed to owners of websites, are now able to add their content. Examples are Wikipedia, discussion boards, blogs, social networking sites, customer review sites, experience or photo sharing sites, etc.

User generated content existed long before the web, in the form of a *Letter To the Editor (LTE)*, which was part of newspapers from the start. With newspapers going from printed form to the web, some of them have also adopted the web analogue of the LTE: comments to news-articles in a manner familiar from weblogs. But an LTE is very different from a comment on a webpage: an LTE is slower in appearing, it may be edited, and the newspaper decides if, when and where it is placed.

In the Netherlands we see that newspapers have different strategies to deal with this new form of LTE. Just 2 out of the 8 national daily newspapers allow comments on all their articles; one allows comments on a small collection, and one allows them only on special "discussion articles". Interestingly, the two Dutch quality newspapers which both devote a lot of printed space to opinionated articles from authors not affiliated to the newspaper do not have a comment facility at all.

Why would one study comments on newspaper articles? We see two reasons. The first is web-technical. Comments are an integral part of the blogging experience according to various user studies [5, 11]. Users would like to have support in assessing comments (search, more structured presentation) [3]. We believe this also holds for comments on news articles. The second reason is sociological. The Netherlands are a politically very polarized nation. Recently there were two political murders involving very controversial figures. There are heated debates on immigration issues, and newspaper articles on these tend to generate loads of comments and debate [12]. To understand and analyze this period, these comments (of ordinary people) may be a valuable source.

These two reasons come together in the main topic of this paper: extracting discussion structure within flat lists of comments. On all of the Dutch news-sites it is presently not possible, for someone posting a comment, to specify whether he or she is commenting on the article or on some previous comment by another author. (Neither is this possible on e.g. `blogger.com`). The effect is that discussions are often quite confusing and hard to read. It would be very helpful to bring forth the underlying structure, and to present this structure within the comment thread by, for example, hyperlinks between comments standing in the reacts-on relation.

Thus our goal is to make as much implicit information explicit, and we use all available tools and techniques to do this. Explicit information may result in better user interfaces to comments. It is also key to both qualitative and quantitative OLAP style analysis of this huge amount of data.

Research Questions. We have two main research questions. The first explores the dataspace: *What does the Dutch 'commentosphere' consist of and what does it look like?* This is an umbrella for questions like: Who is posting comments? Where do commenters come from? Do they react on each

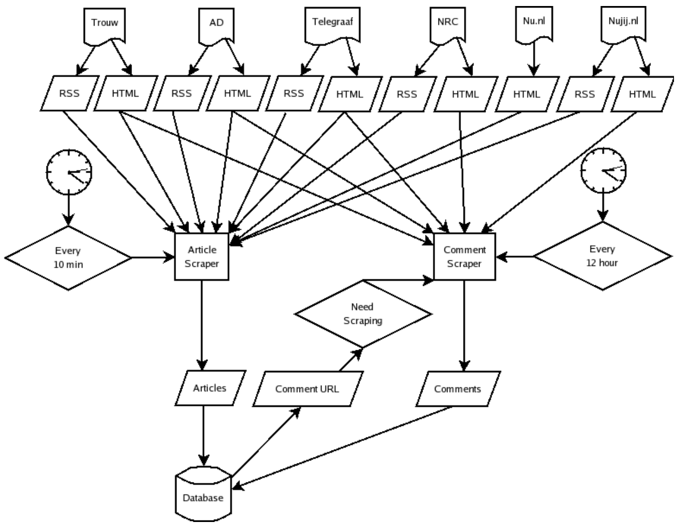

Figure 1: Flowchart of the data acquisition process.

other? In what way do they react? What does a comment look like? What kind of language is used? The second is technical. *How can the discussion structure of comment threads be made explicit?*

Organization of the paper. We first describe the data 'as-it-is' and how it is acquired (Section 2). We explore the data in Section 3, answering the first research question. In Section 4 we develop methods to extract the discussion structure of threads and evaluate these. We end with conclusions and directions for future work.

2. DATA DESCRIPTION AND ACQUISITION

We study comments attached to news-articles placed on the websites of four national Dutch daily newspapers,

de Telegraaf	`http://www.telegraaf.nl/`
Algemeen Dagblad	`http://www.ad.nl/`
Trouw	`http://www.trouw.nl/`
NRC	`http://weblogs.nrc.nl/weblog/discussie/`

and one web-only news-source, `www.nu.nl`.

Terminology

We use the following terminology. A *(news-)site* is a source that produces news-articles.

An *article* is a complete news-article published on a news-site. Usually the author who wrote the article is mentioned, as is its place of origin. All news-sites also state the date and time of publishing, except for Trouw, who states only the publishing dates. This will be a shortcoming as will be seen later on.

The *comments* on articles consist of at least the name of the author and the comment itself. At a few sites, an email address and the city the commenter lives in, are also required. As is the case with articles, comments have, on all sites, a date and time of publishing attached to them.

A *comment-thread*, or just *thread* is a flat list consisting of comments, in reverse chronological order. There are no explicit relations between comments other than the publish-time, as opposed to, for example, fora were usually features exist to mark a post as a reply to a specific previous post.

A *commenter* is person who comments on an article. Usually a commenter only leaves a name behind, but on some site a city and email address as well. Commenters may comment on an article more than once, and they can comment on more than one article on more than one site. Since usually only a name indicates a commenter it is difficult to discover the identity of a commenter [4].

site	article	comment
id *(int, key)*	id *(int, key)*	id *(int, key)*
name *(string)*	site_id *(int, foreign)*	article_id *(int, foreign)*
url *(string)*	source_url *(string)*	full_text *(blob)*
	comment_url *(string)*	author *(string)*
	title *(string)*	city *(string)*
	description *(blob)*	email *(string)*
	full_text *(blob)*	date_published *(time)*
	date_published *(time)*	date_scraped *(time)*
	date_scraped *(time)*	

Table 1: Structure of the Database: site, article and comment relation with their attributes.

Data Acquisition

All five sites offer an RSS-feed with headlines and links to the newest articles. Only NRC publishes the full content of posts, that can be commented on, through their feed. So for all sites but NRC the articles need to be extracted from the HTML version of the site.

Specialized wrappers are written to fit each site and make sure as little HTML as possible is attached to an article. Scraping comments is done separately from scraping the articles, the reason for this is a timing issue: we do not want to miss articles so we check the RSS often. The harvesting of comments is, as was the case with most articles, done through the HTML interface of the news sites, with specialized wrappers for each site. The complete data-flow is illustrated in Figure 1. Nu.nl forms an exception as comments on its articles are published on nujij.nl

Tricky points in the data collection phase are exactly when and how often to scrape, and character encoding issues. Maintaining an almost realtime mirror of all comment threads is nearly impossible because we have to poll all open articles for new comments. Making comments available through RSS would solve this problem. For a more in depth discussion of these issues, see the full report [9].

Scalability. Even though the collection of data poses a serious constraint on the time taken by harvesting algorithms, this time only goes up linear with the number of sources or threads considered. This also holds for the algorithm, discussed later, used for analyzing the threads.

Storage. All the collected data is simply stored in a relational database, mainly consisting of three relations: *site*, *article* and *comment* with attributes and their type as listed in Table 1.

3. DATA EXPLORATION

In order to gain some insight in the data considered and to give ground to choices made later on we provide basic data on the four important concepts in our data: *articles, comment threads, comments, and commenters.*

3.1 Articles

So far we have collected the following amount of data: (For `Nu.nl` only articles which do have comments are indexed)

site	days	articles allowing comments	
		total	avg. per day
Trouw	47	285	6
AD	35	8139	232
Telegraaf	42	7198	171
Nu.nl	85	2109	24
NRC	71	20	0
		17751	

The number of days that are included in the data set differ per site due to differences in feed lengths and site–policies. Trouw and NRC both show very low numbers of articles, compared to the other sites. This is due to the fact that both allow only a few articles to be commented on. Also, the numbers for Nu.nl are a little lower than those of AD and Telegraaf because only those articles with comments on nujij.nl are indexed.

3.2 Comment threads

A comment thread is a collection of comments all belonging to one article. The amount of threads and their average sizes per site is as follows:

site	articles with comments		number of comments	thread-size		
	total	%		avg	min	max
Trouw	80	28%	1297	16	1	88
AD	3465	42%	33383	9	1	202
Telegraaf	636	8%	45760	71	1	826
Nu.nl	2088	99%	34165	16	1	689
NRC	19	95%	2557	134	34	314
	3431	35%	117162	23		

When are comments on an article generated? And for how long commenters keep publishing their comments. Figure 2 gives an impression of the amount of comments that react on a article in the first week after an article has been published. The graph clearly shows a high peak right after the article was published, and then declining peaks, each 24 hours later. These peaks are the result of much less people commenting during the night. It should be noted that the width of a peak is partly a result of the spreading of publishing times of articles during the day.

Also note that peaks for the NRC site are much lower than the others, but they keep recurring for a much longer time; on the NRC site, commenters keep commenting on an article as much as a week after it was published. This as opposed to the curve for the Telegraaf that shows almost no new comments after as little as 36 hours after the article was published. This is not the effect of closing the comment thread, it stays open.

3.3 Comments

We now look at some of the properties of comments itself. A tokenizer, as described in [10], was used to split up comments in words and sentences. Figure 3 shows the amounts of comments per article, and displays a similar power law as found for comments on blogs in [6]. We also compare the comment length and the number of comments of our corpus to the figures reported for comments on blogs in [6]:

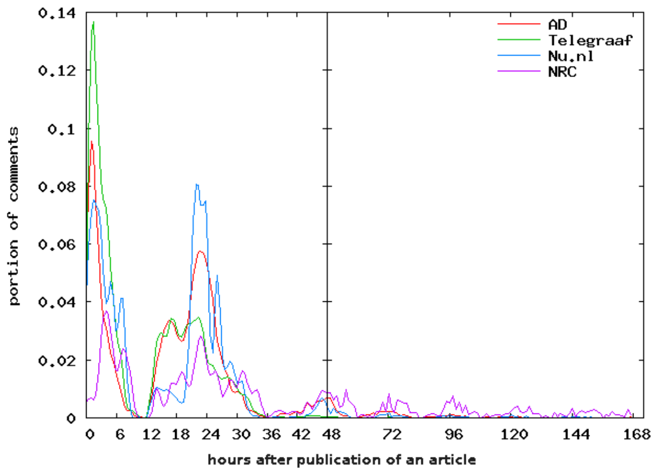

Figure 2: The average timing of publishing comments after an article is published, for a week. Note that the data for Trouw is left out because no publishing time is provided. Also note the differences in scaling on the x-axis, used to give a more detailed view of the first 48 hours.

	Comments on blogs	Comments on news-articles
Comment length (words)		
Mean	63	46
StdDev	93	48
Median	31	38
Comments per post/news-article, excluding uncommented posts/news-articles		
Mean	6.3	18.2
StdDev	20.5	41.8
Median	2	5

Figure 4 plots the number of sentences used per comment against the portion of comments that have this number of sentences. There seem to be two types of curves here, the very steep and high ones around three sentences per post, for Telegraaf, AD and Nu.nl, and the much less steep ones having a peak around 6 sentences per post, for Trouw and NRC.

Another interesting property of comments is the length of sentences used in posts. This says something about the *style* used. Figure 5 plots the average sentence length per comment against the portion of comments that have this property. As with the number of sentences, Trouw and NRC jump out again. The average sentence length in comments on articles for those two sites is around 13 words per sentence, while the other three sites have comments with an average sentence length of around 9 words.

Both the number of sentences and the average length of these sentence is significantly higher for comments on the NRC and Trouw site than they are on the other three sites. This means that comments on those two site are much more substantial.

3.4 Commenters

Where a lot of research on blogs is ultimately directed at getting a picture of the blogger [6], our interest in comments also derives from an interest in the persons behind the comments: the commenters. Of them, in most cases only a (nick?)name is known. But on some sites also the city they live in and their email address is given. As the names commenters use, are chosen by themselves, it is hard, if not

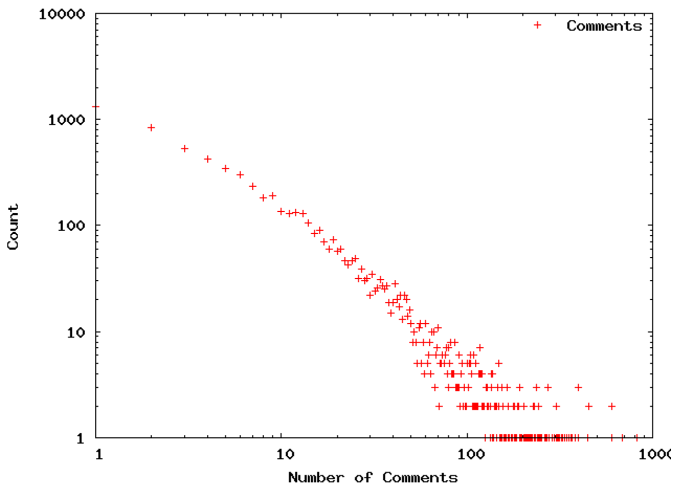

Figure 3: Distribution of the amount of comments per news-article (with at least one comment).

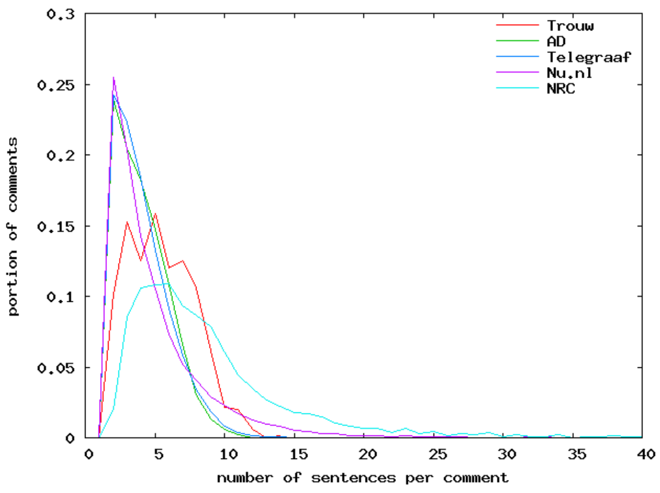

Figure 4: A histogram of the number of sentences per comment. Each point on a line indicates the portion of comments that has the number of sentences measured on the x-axis.

impossible, to distinguish between two commenters that call themselves, for example, *John*. A manual inspection shows that adding a city often does not help: e.g., there are many "Jan, Amsterdam" pairs in our corpus. Other, more exotic names, seem to provide more evidence that the name is indeed referring to just one person but a 100% quarantee is never given. The cases were one can be sure, are those were an email address is provided, since these systems all work with authentication emails.

Even though the issues above are serious, it does give insight in the data when comments are grouped by email addresses and commenter names. The graph in Figure 6 does this, and plots each commenter against the number of comments she posted. The graph shows the expected power law, with 50% of the comments being posted by less than 5% of the commenters. Of course, we should discount for the very common names, of which many appear in the top 100 of most posting commenters. How this should be done exactly is not clear, but even when these names are filtered out, it is evident that a large portion of the comments is produced by a relatively small number of people.

As some sites allow for entering a city-name, it is interesting to see what the geographical distribution of commenters is. Using the geocoder from Google Maps,[1] all coordinates for cities given by commenters were requested. Those cities which coordinates in between (50°N 3°W) and (54°N 8°W), roughly the coordinates of a bounding box of the Netherlands, were selected and plotted in Figure 7. As can be seen, this plot resulted in quite a detailed map of the Netherlands with a high density of commenters in the so-called *Randstad*, which is the most highly populated area.

4. THREAD STRUCTURE

Commenters not only comment on news articles, they also react on other comments, *using the news article commenting facility*. We are interested in this "reacts-on" relation between comments in one thread. However, as was mentioned earlier, this relation is not explicit in the data; commenters do not have the option to comment on a specific comment, as is the case on most fora on the web. Consequently, for each news-article, its associated comment thread is just a flat list of comments. The only explicit relation between them is the later-then relation in publishing time.

We can model the reacts-on relation (a pair $\langle x, y\rangle$, where x reacts on y) between an article and a set of comments as a *rooted directed acyclic graph* (DAG). The nodes in this graph are the article and the comments and the arcs denote the *reacts-on* relation. The root is the article. The graph is directed and acyclic because one can only react on something which happened earlier: reacts-on is a sub-relation of the later-than relation. The structure is not a tree since it is possible for one comment to refer to more than one comment (and also to the article).

Commenters have no formal way of indicating that they refer to a comment, not to the news-article. They use a variety of techniques to indicate the relation, ranging from citing an authors' name, quoting part of a comment, to being completely absent (most probably, referring to the last posted comment). Here is a sample of observations:

1. Referring happens by citing an authors' name ('Eens met "bringer of torture"')
2. One comment can refer to more than one author ('Bravo *Djieff* en *Frabkster*')
3. The '@' character may be used to indicate a reference. (*@ouwerocker*)
4. A citation usually seems to refer to the latest post by the cited author.
5. A referring statement might be misspelled (*Frabkster* instead of *Frankster*).

Using the commenter's name is a frequently used technique but not the only one. In this paper, we will only attempt to extract reacts-on relations that involve citing an author's name. We do, however, allow for misspellings in the referring statement and multiple relations from and to a comment.

[1]see: `http://www.google.com/apis/maps/documentation/index.html#Geocoding_HTTP_Request`

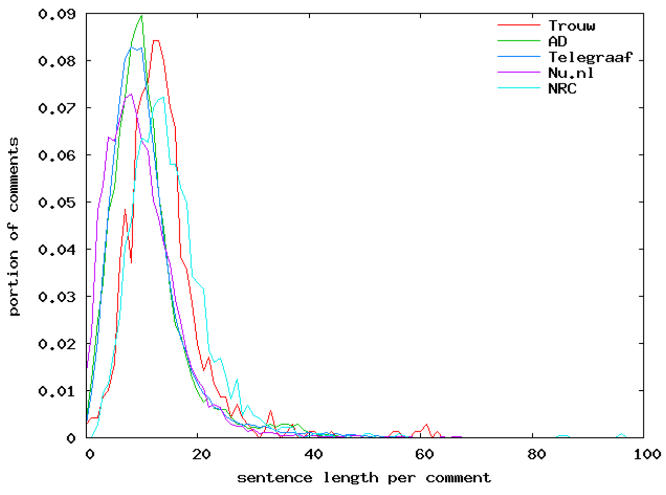

Figure 5: A histogram of the average length of sentences, in words, per comment. The sentence length is averaged per comment and the size of the portion of comments having the same average sentence length is plotted against the sentence length.

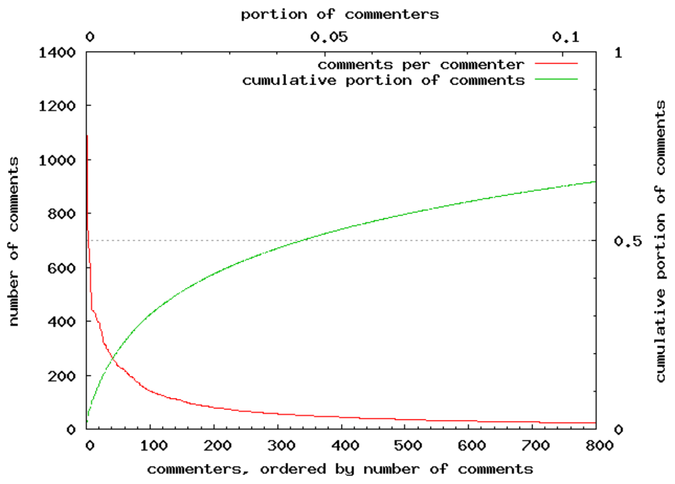

Figure 6: Commenters' names are grouped by email address and name and are then plotted against the number of comments they posted. The top and bottom axis correspond, the left and right axis do not. The left axis belongs to the red (falling) curve, the right to the (rising) green curve. Less than 5% of the commenters post 50% of the comments.

We can define this reacts-on relation declaratively as follows:

comment x reacts-on comment y if
x is published later then y, and
(a minor spelling variation of) the
commenter's name of y appears in x.

The difficult part, as witnessed by the given examples, is the operationalization of the second clause: when do we say that a name *appears* in a comment? We will give three different implementations and show that a combination of them yields the best results.

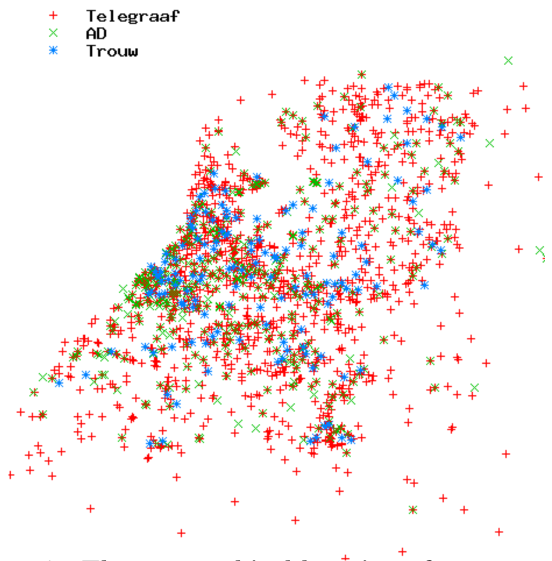

Figure 7: The geographical location of commenters, between (50°N 3°W) and (54°N 8°W), plotted for the sites that support this. Trouw data is plotted on top of AD data, which is plotted on top of Telegraaf data since the Telegraaf, followed by AD, had most data available. The contours of the Netherlands and the most densely populated areas in the Netherlands can easily be recognized.

4.1 Experimental setup

We created a hand-annotated dataset consisting of 4 comment-threads, one for each news-site. The threads were picked semi-randomly, that is, we ensured that they were rather long and had relations occurring in them. We could not find any thread for the news-site Trouw which had reactions on comments[2]. The dataset consists of 478 comments in which 193 reacts-on relations between comments were identified by hand. Table 2 gives a breakdown per site.

news-site	comments	relations
AD	34	9
Teleg.	56	8
Nu.nl	313	163
NRC	75	13
Total	478	193

Table 2: Breakdown of the hand annotated dataset per news-site.

The task is the following: given a comment-thread, we need to identify pairs (x,y) of messages in the thread such that they agree with was annotated by hand. Now we describe our baseline algorithm for identifying the reacts-on relation between comments. It implements the clue *commenters name appears in comment* by a case-insensitive string match between each preceding commenter and the full text of the current comment. A problem with this method is that it will find names included in normal words as well (e.g., *Boy* in *loverboy*). On the other hand, a more sophisticated method that would look at word boundaries, would have difficulties with names spanning more than one word (i.e.: *bringer of torture*). The baseline method is illustrated

[2]This is the only news-site which *in advance* inspects and possibly edits *all* comments left behind on their website.

in Algorithm 1.

We evaluate the number of reaction pairs that we do find,

Algorithm 1 Baseline

```
initialize the set of Commenters as $\emptyset$
order Thread ascending by time
for each comment $\in$ Thread do
  for each commenter $\in$ Commenters do
    if commenter_of(comment) $\neq$ commenter then
      if in_string(comment, commenter) then
        assert reacts_on(last_post_of(commenter), comment)
      end if
    end if
  end for
  add commenter_of(comment) to Commenters
end for
```

and the number found pairs that are indeed reaction pairs. For this, we use the standard information retrieval measures recall, precision and the F_1 measure to evaluate our implementations [1]. The results of the baseline method on the hand annotated dataset are in Table 3. It is often, as it

	recall	precision	F_1
AD	0.8888	0.2580	0.4000
Teleg.	0.3750	0.1363	0.2000
Nu.nl	0.4723	0.5789	0.5202
NRC	0.6923	0.1764	0.2812
Average	0.6071	0.2874	0.3901

Table 3: Recall, precision and the F_1 measure for the baseline method.

is here, easy to get a high recall, at the cost of a low precision, by simply returning all possible relations, and vice verse. Our objective, therefore, is to maximize both *recall* and *precision* at the same time.

4.2 Mining thread structure

We will describe three methods for mining the reacts-on relation. The methods are complementary to a certain extent, so it makes sense to combine them. We find an optimal combination using machine learning techniques, which indeed has the highest recall and precision. On a high level, the three methods are the following:

method B: Word boundaries Like the baseline, but do not match substrings, but sets of tokens.

method C: POS-tagging plus loose match Do a part-of-speech tagging of all comments. A name and a word in a comment match if the word is tagged as a proper noun and is similar to the name. This method takes care of spelling variations.

method D: @-trigger plus loose match A name and an n-gram match if the n-gram is preceded by the @-symbol and the name and n-gram are similar.

The results of these three methods compared to the baseline are given in Table 4. We first briefly discuss the three methods and then look at the combination.

	recall		precision		F_1	
		δ		δ		δ
A	0.6071	-	0.2874	-	0.3901	-
B	**0.6638**	0.0567	0.8307	0.5432	**0.7379**	0.3477
C	0.4978	-0.1093	0.9335	0.6460	0.6493	0.2591
D	0.3883	-0.2188	**0.9489**	0.6615	0.5511	0.1609

Table 4: Precision, recall and F-measure compared to the baseline. Figures are averaged over all sites. A indicates the baseline, B through D match the methods listed above.

Word boundaries.

In this method we tokenize both the commenter's name and the comment, and check whether the first list of tokens occurs as a sublist of the second list of tokens. Both precision and recall show an improvement over the baseline. The improvement in precision was to be expected, and was caused by the fact that the baseline identified many false positives due to falsely identifying references within words. The slight improvement in recall might come as a surprise. This is due the normalization of white-space conducted by the tokenizer. In the original data white-spaces in the author's name do not always match the white-spaces in the candidates causing the baseline to miss these reacts-on relations.

POS tagging and loose match.

This method tries to account for the fact that commenters' names are not always spelled in the same way as they appear: in our corpus we encountered spelling mistakes, change of capitalization, and even abbreviations of names. So, instead of a strict string similarity, we opt for a loose match. In order to keep a high precision we want to limit the words that we try to match to those being tagged as proper names or proper nouns by a part-of-speech tagger.

To match two strings more loosely, the Ratcliff/Obershelp [8] similarity measure is used. This measure is based on the longest common substring that two strings share. It finds the longest substring and then recursively finds the longest substrings on either side of it. When the sum of the lengths of all matching substrings is denoted as $match(w_1, w_2)$ and the sum of the lengths of both strings as $length(w_1)+length(w_2)$ the measure is:

$$similarity(w_1, w_2) = \frac{2 \cdot match(w_1, w_2)}{length(w_1) + length(w_2)}$$

For example, the strings *willem-jan* and *willem* (both are dutch first names) have the following similarity measure:

$$similarity(\text{jan-willem}, \text{willem}) = \frac{2 \cdot 6}{10 + 6} = 0.75$$

The measure can be used as a threshold when comparing two strings. A threshold of 0 would match all strings to each other. A threshold of 1 would only match identical strings.

Setting the right threshold is not trivial, so we experimented with different values, see Figure 9. The results reported in Table 4 are based on a threshold of 0.85, which has the maximal F_1 value. Precision increases over both the baseline and the word-boundary method, but the recall goes down. The drop in recall is mainly due to multi-token names which the POS tagger does not recognize.

@-Trigger and loose-matching.

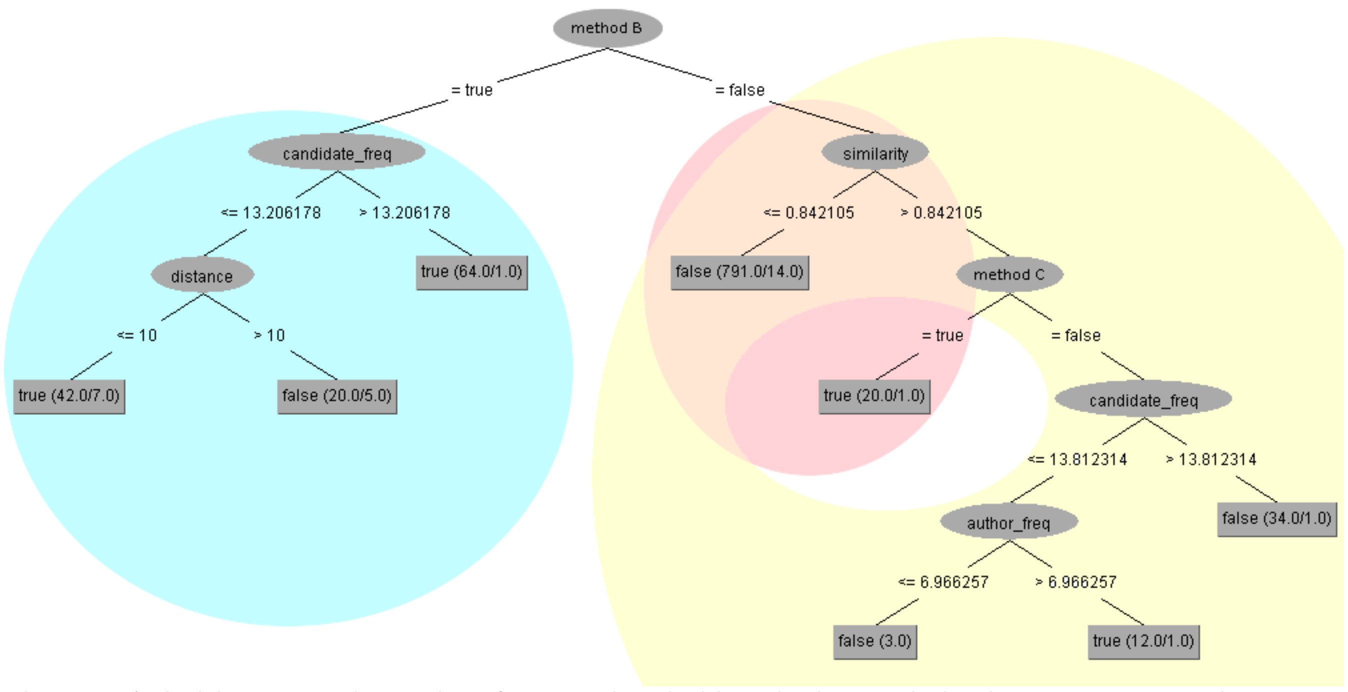

Figure 8: A decision tree, using various features that decides whether a relation between two posts is present. (method B is the *word boundary*, method C the *POS-tagging* method). Also, the areas where different methods are used are depicted by colored circles. The *word boundary* method is blue, the *POS-tagging* method is red and the *@-trigger* method is yellow.

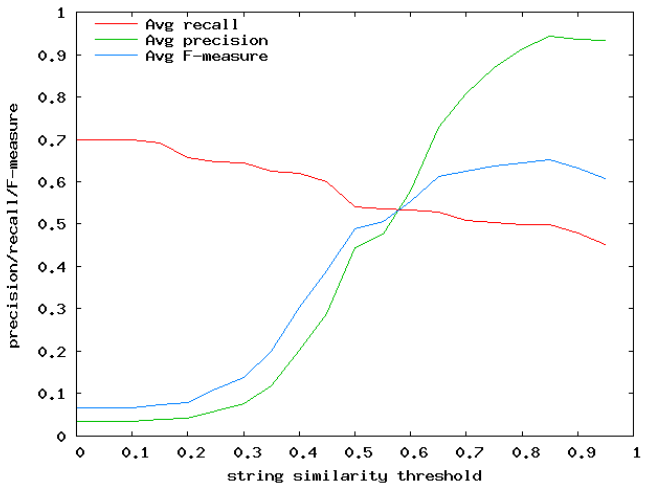

Figure 9: Precision, recall and F-measure curves for different string similarity thresholds, using the POS-tagging method.In this plot the measures for all news-sites are averaged.

Based on evidence found during the manual investigation, it makes sense to exploit the @-character as a trigger for finding references. The idea is that a word, or words, following an @-character are very likely to refer to a previous author. Therefore, these words are handled as candidates, similar to the POS-tagging case. After the candidates have been identified, they need to be matched to commenters. Again we use loose matching. The recall of this method is not very high. This is caused by the fact that only a few references are preceded by an @-character. As expected, precision is very high.

4.3 Combining the methods

The three methods all have a fairly low recall of 0.39 to 0.66, but a high precision of 0.83 to 0.95. Knowing that they are all based on different ways of finding candidates and deciding on candidates, it is likely that the sets of correctly identified relations they return do not completely overlap. If this were true, combining the three methods would lead to an improvement of the recall while a high precision could be maintained.

One way of combining the methods, which is similar to the methodology used in [6], is to let each of the three methods generate their candidates for the annotated threads, and train a classifier on this data, which then can be used to classify candidates for all threads. The supervised machine learning algorithm, that creates the classifier, needs a set of features of (commenter's name, candidate) pairs to base its classifier on. We use the following features:

- **Method.** A binary attribute for each of the three methods, set to true if the method puts the pair in the reacts-on relation.
- **Similarity** between candidate and author's name, measured using the Ratcliff/Obershelp similarity. Candidates found using method B are given a string similarity of 1.
- Temporal **distance** between two posts, measured in the number of comments that separate the comment in which the candidate appears and the comment it might be referring to.
- The **news-site** that is the source of the thread, is added as a nominal attribute.

	recall		precision		F_1	
		δ		δ		δ
A	0.6071	-	0.2874	-	0.3901	-
B	0.6638	0.0567	0.8307	0.5432	0.7379	0.3477
C	0.4978	-0.1093	0.9335	0.6460	0.6493	0.2591
D	0.3883	-0.2188	0.9489	0.6615	0.5511	0.1609
E	**0.7183**	0.1112	**0.9489**	0.6615	**0.8177**	0.4275

Table 5: Precision, recall and F-measure for all methods (B through D), compared to the baseline (A), and the combination (E). These figures are all averaged over all sites.

- The word-frequency of the author's name (**author_freq**) is looked up in a table containing word frequencies[3]. Of this, we take the $-log$, to account for precision limitations in the software used.

- The word-frequency of the candidate (**candidate_freq**) is measured equal to the feature above.

Training a classifier[4] As a wide range of supervised machine learning algorithms is available, choosing which one to use is a matter of requirements and sensible experimenting. A desired requirement in our setting is that we want the output to be easily readable and self explanatory. Two types of algorithms that have this property are tree-learners and rule-learners. The J4.8 algorithm [13], an implementation of C4.5 [7], with its default settings, gave the best results. Experimenting with other standard methods, like Bayesian networks and neural network methods, showed that very similar results[5] are obtained, so there seemed to be no reason not to use J4.8 as the classifier.

Since J4.8 is a tree-learner, it produces a decision tree. The resulting tree is depicted in Figure 8. It is interesting to see how each method contributes to the final classifier. In Figure 8, the area in which each method is used is given a colored circle. The *word boundary* method, method B, is active in the blue circle and identifies the majority, 73%, of the reacts-on relations, based mainly on the word frequency of the candidate and the distance of the reacts-on relation spans. Second comes the *POS-tagging* method with 14% of the instances by simply canceling out very low similarities. Last comes the *@-trigger* method that bases classification mostly on candidate and author word frequency.

Evaluation. Table 5 shows all methods compared to each other. It can be seen that the highest recall is measured when the methods are combined. This is what we expected. It is more surprising that we did not loose any precision by the combination, so the combination also obtains the largest F-value. Thus the combination of the three methods works the best.

5. CONCLUSIONS AND FUTURE WORK

We showed that it is quite easy to mine the Dutch 'commentosphere', to store the comments in a relational database, to enrich them and to provide aggregation data. The quality of the comments (as measured in the number of sentences and their length per comment) is remarkably high, so it seems a valuable source of information. Even though there are no discussion-linking facilities, people use the commenting facility on news-sites to react on each others comments. We showed that we can automatically extract this reacts-on relation with high recall and precision. Now, what can one do with this enriched data?

Social scientists study online debates on the web, but as far as we know, only restricted to fora and discussion sites [12], in which the reacts-on relation is explicit. Adding the discussions found in comments at news-sites may be valuable and yield other data. But then we need to be able to connect discussions held at various times and possibly on various sites to the same topic or event, thereby aggregating all discussion data on a single topic. Grouping articles this way, and measuring liveliness in the attached discussion threads would give an indication of the 'hot events', as with comments on blogs in [6]. The peak explanation technique used in `www.moodviews.com` seems well suited for this [2].

There are different types of reacts-on relations: agreement, disagreement, asking and giving advice, etc. Knowing the type of reaction could be valuable for further analysis and could also improve the 'liveliness' measure, mentioned above.

The reacts-on relation, as it is now, could readily be used by the news-sites themselves to improve their interface. It would be useful to add hyperlinks to both posts reacting on a comment and the post the comment is reacting on. A simple demo application doing just this was implemented (see `http://www.science.uva.nl/~aschuth/comments`). It showed helpful in following discussions and clicking back and forth feels quite natural. Is should however be noted that focusing on high precision in this sort of applications is crucial. It is confusing if a word gets misidentified as a link.

[3]Based on a large Dutch newspaper corpus from 2002 enriched with data extracted from all comments considered, to account for changes in language.

[4]Please refer to [9] for a detailed discussion of this subject.

[5]Only when tweaking the settings of the Multilayer Perceptron marginal improvement was found.

6. REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley, 1999.

[2] K. Balog, G. Mishne, and M. de Rijke. Why are they excited? identifying and explaining spikes in blog mood levels. In *Proceedings 11th Meeting of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, April 2006.

[3] A. de Moor and L. Efimova. An argumentation analysis of weblog conversations. In *The 9th International Working Conference on the Language-Action Perspective on Communication Modelling (LAP 2004)*, 2004.

[4] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *Proc. SIGMOD*, pages 85–96, 2005.

[5] M. Gumbrecht. Blogs as protected space. In *WWW 2004 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*, 2004.

[6] G. Mishne. *Applied Text Analytics for Blogs.* PhD thesis, University of Amsterdam, 2007.

[7] J. Quinlan. *C4. 5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[8] J. Ratcliff and D. Metzener. Pattern matching: The Gestalt approach. *Dr. Dobb's Journal*, page 46, 1988.

[9] A. Schuth. Applied text analytics for comments of news articles, 2007.

[10] E. Tjong Kim Sang. Generating subtitles from linguistically annotated text. Atranos report WP4-12, University of Antwerp, 2003.

[11] E. Trevino. Blogger motivations: Power, pull, and positive feedback. In *Internet Research 6.0*, 2005.

[12] T. Witschge. *(In)difference Online.* PhD thesis, ASCoR, Universiteit van Amsterdam, 2007.

[13] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.