Support for Decision Making: Electoral Search

Valentin Jijkoun Maarten Marx Maarten de Rijke Frank van Waveren
ISLA, University of Amsterdam
Kruislaan 403
1098 SJ Amsterdam, The Netherlands
jijkoun,marx,mdr,fwaveren@science.uva.nl

ABSTRACT

The Netherlands had parliamentary elections on 22 November 2006. We built a system which helps voters to make an informed choice among the 16 participating parties. One of the most important pieces of information in the Dutch election and subsequent coalition government formation is the party *manifesto*. This is a text document with an average length of 45 pages. Our system provides the voter with focused access to party programs: given a search term, the system displays a ranked list of relevant paragraphs from the programs. Ranking is possible by relevance or by party. By selecting only the parties of interest, the voter can make a topic-wise comparison of party view-points.

The paper describes the system in full detail, including design, requirements, technical details, and user statistics. It can be used at `www.verkiezingskijker.nl`. The system is presented as a case-study in building applications which aid decision making.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous; D.2 [**Software**]: Software Engineering

Keywords

Elections, Democracy, Information Retrieval, Language Technology

1. INTRODUCTION

In this paper we describe—in the form of a case study—a decision support system based on information retrieval techniques. It addresses the problem of providing easy access to information that, on the one hand, users are very interested in and is highly relevant to their lives, but which, on the other hand, may be almost inaccessable to a broad audience. This inaccessibility exists because of lack of focused, topic-driven access facilities, as well as the lack of instruments for comparing information from different sources.

Often such hard to reach informational niches are occupied by commerical/professional brokers, advisors or mediators. They offer help with complex decisions with long-term consequences on e.g., the welfare, wealth, or well-being of a person. Examples of such decisions include:

- choosing among several bidders on a contract;
- buying an important and expensive object for which there are many vendors, e.g., a mobile phone provider, health insurance, mortgage, marriage ceremony, choosing a school for your children, etc.

One common aspect to these cases is the necessity to make a well-informed and judicious decision, often under high time pressure. The decision consists of making a unique choice among many competitors, who often swamp the decision maker in poorly structured textual information.

The search engine described in this paper is not intended to guide the user in making decisions. It merely helps her by providing quick and easy access to those parts of the data that are relevant to the user, and in putting together different competitors along a dimension chosen by the user. Hence, our search engine can be seen as an important backend for a real-world decision support system.

But even without the front-end of the decision support system, such a search engine can be very useful. In the case study we describe in this paper, we find 16 competitors, namely, political parties, competing to attract user's votes, and each describing in their manifestos (election programs in our case) their standpoints and advantages and sometimes even drawbacks and flaws of their competitors. The number of dimensions through which a user might approach this complex information knot is almost limitless and different users might be interested in very different dimensions.

Making an informed decision in such a case is neigh impossible without additional informational support. In our case study of election programs, the best a user can do without such a support is to locate online (or even offline, e.g., printed) versions of election programs and study and/or compare them by performing string searches in the electronic documents or even by reading entire programs hoping to come across the relevant information.

Main contributions. We describe an electoral search engine aimed at helping the general public in its electoral decision making. Based on interest from real users, on user feedback and media coverage, we believe that this application of search and language technology is one of wide interest. We motivate the choices made in our design, describe the technical challenges and our solutions, and lift our findings to the general decision making problem outlined above.

The main contribution of the paper is the description of our case study. This can be viewed as a recipe describing how to use *off-the-shelf technology* to quickly build a web accessible search engine for cases which fit the given problem: make an informed choice among several competitors which

DIR 2007.
.

drown the choice-maker in textual, mostly unstructured, information.

Website. The website for the system described in this paper is available at `http://ilps.science.uva.nl/verkiezingen`.

Organization. Section 2 provides background on the Dutch electoral system, and describes the motivation for the system that we built. Section 3 describes how the perceived problem was translated into functional requirements and design of the system. Preprocessing and indexing of the data is covered in Section 4. Technical details and results are in Sections 5 and 6, respectively. Section 7 describes related work. We conclude in Section 8.

2. BACKGROUND

Dutch parliamentary elections. The last Dutch parliamentary elections were held on November 22, 2006. Members of the Dutch parliament (150 seats) are chosen according to the principle of proportional representation. In 2006, 65.591 votes were needed for a single seat (`http://www.kiesraad.nl`). This system leads to a proliferation of political parties; in general, some twenty parties participate in the national elections (in 2006: 24 parties). Before the election, the Dutch parliament contained representatives of 10 regular parties, as well as a small number of independent members. 10 parties are represented after the elections, two of them are new. In this system, it is nearly impossible for a party to obtain an absolute majority, and the Dutch government thus always consists of a coalition of parties.

The global trend of parties moving towards the center of the political spectrum is also visible in The Netherlands. For each election, all parties create manifestos (a.k.a. party platforms) which contain their plans and promises should they be elected into government. These documents tend to be quite long (45 pages, on average) and are mostly read by professionals.

The general public has a hard time making up their minds in this complex setting, as is witnessed by the existence of multiple (we found 12) "voting advice websites." The eldest and most popular of these (`www.stemwijzer.nl`, providing over 4.5 million advices during the last parliamentary elections) measures the similarity between a voter and the parties, and returns a ranked list of parties. Similarity is measured by answering agree/disagree on 30 theses covering the "hot election topics." This advice site is set up and created by the Instituut voor Publiek en Politiek (IPP, [15]), a public-private non-profit foundation with the aim of bringing politics and the general public closer together. The Stemwijzer site does not aim to *predict* what a person will vote. On the contrary, it advices the voter by showing those parties which most agree with the voter on the "hot topics." The top ranked (most similar) party is often unexpected by the voter. This aspect generates heated debates among voters, ranging from hate-mail to the site to opinionated articles by professionals in leading national newspapers.

The problem and our proposed solution. IPP analyzed the reason for this and concluded that the general public just does not know the views of the participating political parties, but rather has stereotypical images of them. They asked our help in building a system which can easily inform voters on the viewpoints of parties. Because the manifestos are considered to present the official party-line, we decided on a system which makes these easily accessible, with the following requirements:

1. users can choose a topic from a predefined list, and possibly a number of parties; the system returns all relevant passages from the programs (of the chosen parties);
2. users can also decide on their own search terms;
3. for the predefined topics, the precision at 5 should be high;
4. because the manifestos were made available less than one month before the elections, manual tagging of passages was not possible, so the system had to be fully automatic.

The next section translates these informal desiderata into functional requirements on the system.

3. FUNCTIONAL REQUIREMENTS AND DESIGN

In this section we describe the requirements from the perspectives of the user, the developers, and the system. It ends with the architecture of the main modules.

Requirements from the user's perspective

Pinpointing to relevant passages. Voters do not read manifestos because they are too long. The system must therefore provide focused access and return only the relevant passages in response to a search query, *not* the complete document. To implement this, we divided the manifestos into paragraphs and fed these as separate documents to the search engine.

Thematic and free search. Based on previous elections and current hot topics, the contractor provided an isa-hierarchy of election topics. The hierarchy was 3 levels deep, with 10 topics at the top level, and contained 179 topics in total. For these topics, the contractor wanted high recall and precision at five. We obtained this through query expansion. Section 4.2 describes how the expansion terms were collected.

Besides this, we offered a familiar search box in which the user could provide her own search terms.

Figure 1 shows the query interface. The open menus display a path in the isa-hierarchy.

Comparison of topics between parties. Given a topic, the system should help the voter in comparing the different parties on their views on that topic. This translated into an interface in which the user can sort results by relevance and by party. The user can also restrict the search to certain parties using checkboxes. In the case of two parties the relevant results are displayed next to each other, in two columns, as illustrated in Figure 2.

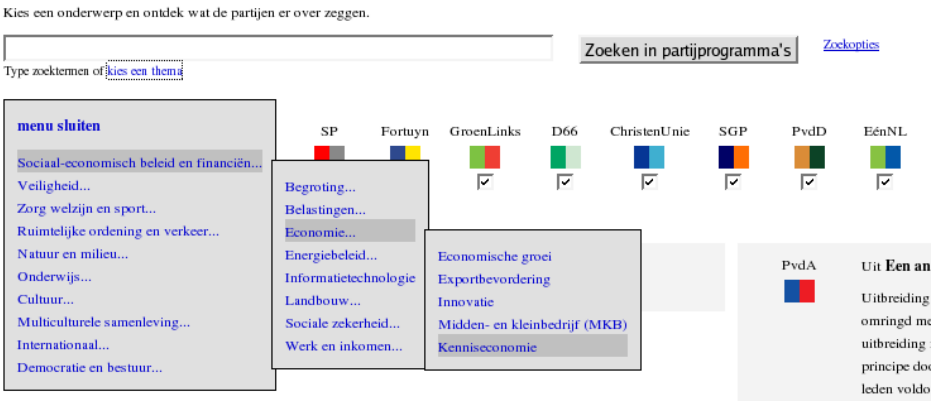

Figure 1: Query interface showing pop-ups with the thematic search terms (in the grey boxes).

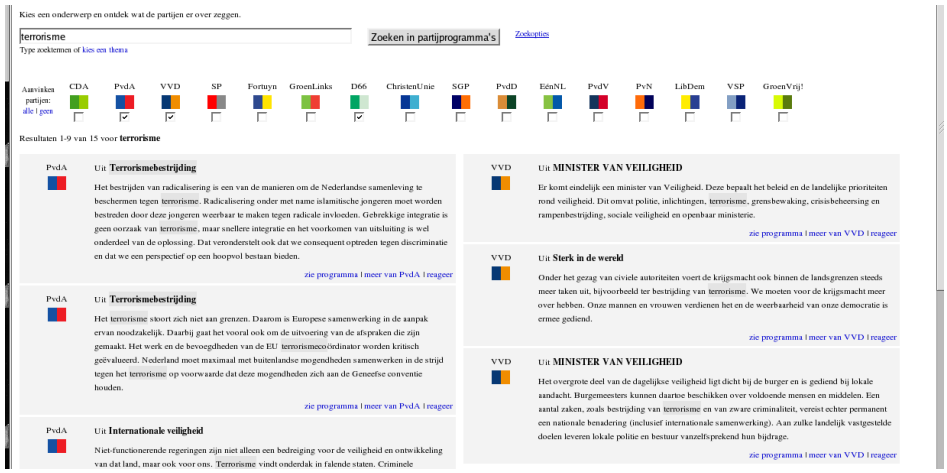

Figure 2: Comparison of two parties on the topic of terrorism.

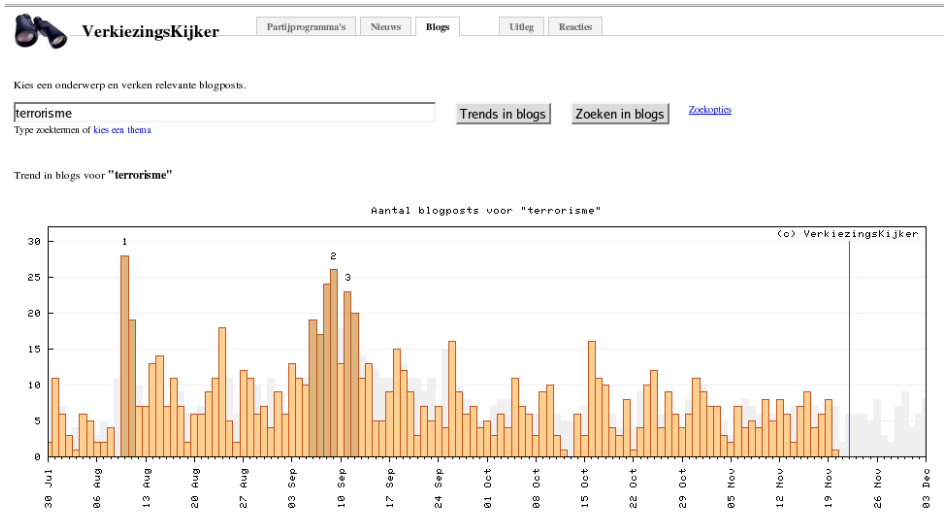

Figure 3: Trend in the volume of blog posts on the topic of terrorism.

Multiple sources of information. While the manifestos are the primary source of information, we also want to offer integrated access to other independent sources. We chose for information coming from professionals and from the general public. For this, we collected newspaper articles and weblog postings, respectively. By clicking the corresponding tab in the interface, the user can switch, keeping his search term, between these three different sources. See Figure 4. How the data from these sources was collected is described in Section 4.

Figure 4: Tabs for the three sources of information: *partijprogramma's* (manifestos), *nieuws* (news), and *blogs* (indeed: blogs).

Events and trends. To guide the user to interesting events related to a topic, we provided a trend button. Trends of a topic are shown measured in the volume of news and blog postings per day, with peaks indicated in the graph. Similar displays are used by other blog-tracking sites, e.g., [2, 4]. To provide meaning to the trend plots, we offer automatically generated explanations by finding the corresponding events which probably lead to the peak; see Figure 3. The details of the peak explanation method are provided in Section 5.

Requirements from the developer's perspective

Collecting domain knowledge. Due to time constraints this had to be done with very little effort on the part of the domain expert. Section 4.2 describes how the expansion terms for the predefined topics were collected.

Data preprocessing. Again due to time constraints (the data is basically available at the exact moment the system should be working), this had to be simple but robust and effective. Details are given in the next section.

Requirements from the system's perspective

- Use only open source, off-the-shelf technology: Lucene for the search engine; MySQL for the database; PHP, Javascript, CSS for the website; Perl and shell scripting for data processing scripts.
- Provide a simple API to the search engine, because it has to be hooked up to several websites: implemented using the GET method in forms, so all query variables are visible in the URL.
- The system has to run in real time with many users.

Architectural design. The design of the main modules of the system and their interaction is given in Figure 5.

4. DATA PREPROCESSING AND INDEXING

We describe how we processed the three data streams — manifestos, news articles and blogs— and how we obtained query expansion terms for the predefined topics from the domain expert.

4.1 Manifestos

We could collect 16 official and final manifestos. When printed, these had 670 pages in total. Although most of them were discussed at the assembly of the members of the parties, just two of them were available in web-readable HTML format. Five were only available in Word format, the rest in PDF. Hyperlinks were almost absent; these texts were edited as to be read from start to end.

Based on the physical layout, the manifestos were divided into paragraphs, yielding a corpus of 4618 documents. Table 1 contains the breakdown of the number of paragraphs per party.

Party	Number of paragraphs
Christenunie	749
Partij van de Arbeid	581
SP	481
SGP	444
Partij voor Nederland	381
D66	373
Partij voor de Dieren	316
CDA	291
Groen Links	232
LibDem	172
Een NL	147
VVD	144
Fortuyn	128
Partij voor de Vrijheid	92
VSP	67
Groen Vrij!	20

Table 1: Number of paragraphs in manifestos per party.

In order to put the paragraphs into context and to boost recall, we recursively added chapter, section, subsection and paragraph headers to each individual paragraph. This was all stored in XML format, which made it easy to produce uniformly looking HTML versions of all manifestos.

The manifestos were processed by a generic Perl script which could be fine-tuned for each party. On average half an hour of manual cleanup was needed for each party program.

4.2 Obtaining domain knowledge

For the predefined topics, our goal was to be able to retrieve paragraphs relevant to the topic with good precision at high rank (in top 5). To achieve this in a simple way we expanded the topic title with additional query terms which

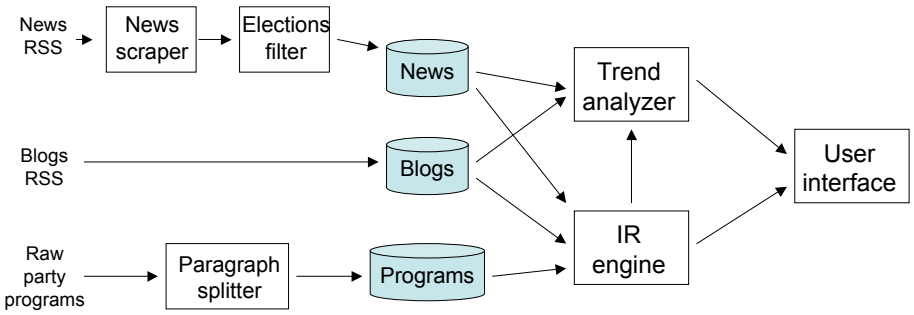

Figure 5: Architecture of the system.

we obtained indirectly from the domain expert as follows. We used the search engine to find candidate paragraphs for each topic simply using the title of the topic as search query, and asked the expert to provide relevance feedback: mark the returned paragraphs as relevant or not relevant for the topic. We asked the expert to find at least 5 relevant paragraphs for each of the 179 topics. For most topics, 5 relevant paragraphs were found in the top 20 hits, making the task relatively easy. In total, the domain expert needed just over two days for the task. (We note that during the task some topics were dropped because they were not present in the manifestos at all).

For each topic, using the paragraphs manually annotated as relevant, we collected the 15 most overused terms as characterizing the topic. Over-usage of terms was determined using the log-likelihood statistical test [6]. Specifically, for each topic we compared two text corpora: (1) consisting of paragraphs marked as relevant, and (2) consisting of all paragraphs in the collection. Let O_i be the observed frequency of a term in corpus i ($i = 1, 2$), N_i the size of corpus i, and $E_i = (N_i \cdot \sum_i O_i)/\sum_i N_i$ its expected frequency in corpus i (where i takes values 1 and 2 for the corpus of relevant paragraphs and the entire corpus, respectively). Then, the log-likelihood value for a term is calculated according to this formula:

$$-2 \ln \lambda = 2 \sum_i O_i \ln \left(\frac{O_i}{E_i} \right).$$

We subsequently performed a manual check of the terms identified by this method and removed those that would likely lead to topic drift. This occurred mostly with very general topics at the top of the IS-A-hierarchy, e.g., "Europe" or "Rules".

This semi-automatic method for extracting characteristic terms showed good results. The terms found were often unexpected but judged correct by the domain expert ("If you had asked me directly, I would have never thought of that term for this topic".) Table 2 lists the overused words on the topic "Traffic jams".

This method can be improved by considering multi-word compounds like *public transport* as phrases. The terms removed during the manual check were often part of compounds which were excellent as phrases, but would lead to topic drift if taken as single words (For Dutch readers: "lik op stuk" was a nice example showing up in the topic "criminality"). Unlike in English, noun compounds in Dutch are often written as one word (see the examples in the table) and therefore did not cause many problems at this stage.

Dutch	**English translation**
vervoer	transportation
kilometerheffing	"kilometer charge"
mobiliteit	mobility
openbaar	public
werkverkeer	"work related traffic"
forensenforfait	commuter allowance
ondertunneling	tunnelling
overkappingen	coverings
heffing	charge
auto	car
wegennet	road net
snelwegen	highways
motorrijtuigenbelasting	car tax.

Table 2: Semi-automatically extracted characteristic terms for the topic "Traffic jams".

The resulting per-topic lists of characteristic terms were used straightforwardly in the search interface: when a user deployed topic-based search, rather than keyword-based, the list of terms for the chosen topic was used as a query to the paragraph retrieval engine.

4.3 News

Besides manifestos, our system had news articles and blogs as information sources. Unlike manifestos, news items have to be collected, cleaned and converted to a simple format (header, content, source, date, time, etc.) in real time. The requirements for the news collection and extraction module were as follows:

1. Cleanup of news articles should be fast and easy; there was no time to develop separate scrapers for different news sites;

2. Only news items somehow relevant to the elections (or, more generally, politics) should be considered by the search engine, as opposed, for example, to sport or entertainment news;

3. Many important news events should be represented in the corpus and covered from different sides of the political spectrum.

Because our main interest was events related to the national elections, we chose nation-wide daily newspapers. The whole political spectrum is present in this population, and fortunately, all selected newspapers make content easily available with RSS-feeds. Table 3 lists the newspapers included in our corpus, along with their political orientations and the average number of political news articles per day.

Daily newspaper	#	Political orientation
Algemeen Dagblad	9.3	Centre
Nederlands Dagblad	5.3	Christian Democrate
NRC Handelsblad	5.5	Centre
Het Parool	2.6	Left-wing
De Spits	2.8	Right-wing
De Telegraaf	12.6	Right-wing
Trouw	13.1	Christian Democrate
Volkskrant	11.6	Left-wing

Table 3: Newspapers included in the search engine. The middle column contains the avarage number of election-related articles per day.

The preprocessing of news data consisted of 4 stages:

1. getting RSS feeds and fetch full articles (HTML);
2. extracting the text content of the articles from HTML;
3. filtering news items related to the elections; and
4. storing and indexing the data.

We briefly describe each stage.

Stage 1. Fetching the RSS feeds and the articles in HTML was done by hourly polling the RSS-feeds of the selected news sources. For each article, we store URL, publication date and time, source, headline and the HTML content.

Stage 2. Scraping news articles is notoriously difficult [5, 3] and due to the high time pressure we had to go for a generic solution.

One common way of extracting data from web pages such as news sites is to write a set of rules (usually in the form of regular expressions) for each site which is scraped. This gives accurate results, but is not robust, does not scale and requires regular maintenance.

It turns out there is a class of web pages which have the content all lumped together with only a few types of HTML tags in the text: newspaper websites are a prime example of this. The news content is presented as a single block of text with menus, search boxes, links to other sections of the paper, etc. placed around it. This visual layout translates back to an HTML document structure with the content in one block.

The HTML tags that appear in the `BODY` section of an HTML document can be either *inline* or *block-level* [16]. This distinction determines the behaviour in several respects of the tag, but most important for the task at hand is the content-model and formatting behaviour: Block-level tags cause a line-break, whereas inline tags do not (this behaviour can be changed using CSS, but generally is not). Examples of block-level tags are `BLOCKQUOTE`, `PRE`, and `H1`, whereas inline tags are those like `EM`, `FONT`, and `I`. As we are looking for large single pieces of text, encountering a block-level tag usually means we have skipped to a next piece of text.

With this model for the structure of our data, we can construct an algorithm to extract plain text from HTML. We need to look for long blocks of text outside of tags that is uninterrupted by block-level tags. This algorithm has one tunable parameter: The minimal length of a segment of text for it to be accepted, measured in numbers of words/tokens. We optimized this parameter based on a representative sample of 25 HTML pages, and found an optimal threshold value of 20 words. The errors with this threshold were negligible: 3 to 4 superfluous or missing word per article (we do not consider headings and captions to be part of the article). Note that this only concerns the words which are indexed; the user is transferred to the original page, so will not notice these errors.

This simple technique met the first requirement beyond our expectation. A complete description plus code is available [14].

Stage 3. Filtering for election related news. Although we mainly used focused RSS feeds (internal affairs, political news, etc) only 20% of the crawled articles were relevant to the elections. In order to meet the second requirement, we filtered the stream of articles using a Naive Bayes classifier, an algorithm that has gained popularity recently for filtering spam from email[7].

The classifier was trained with a few hundred articles which were manually categorized (this large number of articles was necessary as only 20% of the news articles pertain to the election). To evaluate the operation of the classifier we picked 100 news articles at random from our corpus and ran the classifier on them while manually verifying the results The results are in Table 4.

	correctly predicted	incorrectly predicted
predicted as election-related	13	1
predicted as not election-related	81	5

Table 4: Evaluation of the classifier on 100 articles

Though a fair number of wrong predictions were made, the disparity between the numbers of election-related and other articles means that this filter reduces a stream that contains only 20% election-related articles to one that consists of over 90% of them. In the process some election-related articles are lost, but given the overlap in coverage between news sources this is acceptable for our purposes.

It should be noted that most of the errors occur at the edge of the "election-related" conceptual area. For instance, the false positive is an article about the court case of a man who made threats of violence against several prominent politicians, of whom both the names and parties are given.

Stage 4. Indexing and storing.. The text of the news articles after filtering is indexed for retrieval and stored in a database, together with the meta-data from the RSS feed (headline, URL, publication date and time, newspaper section, etc.).

4.4 Blogs

We obtained our blog data from ILSE, one of the largest Dutch weblog hosts (`http://www.web-log.nl/`). At the moment of writing the corpus contains 43984 weblogs with an average of 4179 blog postings per day. Within the measured time-period there were 7768 active bloggers (having at least one post a week). Unlike for news articles, we do not perform election-related filtering on blog posts. Because we obtained clean data from the blog host, no additional cleanup or any other preprocessing was needed.

Indexing and storing. Similar to news items, blog posts are indexed for retrieval and stored in a database, along with meta-data (blogger, URL, publication date and time, etc.).

5. TECHNICAL DETAILS

The system allows users to search any of the three resources (election programs, news and blogs) by either typing keywords or selecting topics from a menu. In addition, the system provides the trend functionality for news and blogs: visualization of the volume of blog posts or news items relevant to a topic or a search query, peak detection and explanation.

5.1 Searching in programs, news and blogs

For program search, the system responds with a list of paragraphs extracted from manifestos, ordered either by party or by the relevance to the topic or query. The user may optionally select which parties to include in the search results. For news and blogs, search results can be ordered by relevance or by publication time. The system is implemented using Lucene [12] for retrieval in programs, news and blogs, and MySQL database for data storage.

5.2 Trends

The trend functionality guides the user to interesting events related to a topic. The system shows trends of a topic or a keyword, measured as the volume of news and blog postings per day, with peaks indentified and highlighted on the plot (See Figure 3). Peaks are detected by comparing actual amount of information items on a specific day to the expected amount estimated from earlier observations (previous days and previous weeks).

We go one step further and provide *explanations* for the identified peaks. For each peak, the system produces a list of overused words characterising the peak in the context of the user's search query. The overused words are extracted as described in Section 4.2, using the log-likelihood statistics for comparing a set of relevant items in the peak period to the entire set of relevant items.

As a different type of peak explanation, the system also finds and displays news items and blog posts likely to be related to the "subject" of the peak. These news and blog items are identified by querying the news and blogs indices, respectively, with the list of the peak's overused words as the query. Our peak identification method is based on the method described in [1].

Query	English	#
kinderbijslag	child allowance	5314
minister-president gekozen	elected prime-minister	5252
kinderopvang	kindergarten	4464
Turkije	Turkey	3969
ontslagrecht	law governing dismissal	3284
bijstand	social security	3123
meningsuiting	freedom of speech	3069
dieren	animals	2754
nationaliteit	nationality	2664

Table 5: The frequencies of the 10 most popular keyword search queries.

6. RESULTS

An early prototype of the system went online on October 23, 2006. In this section we present some statistics for the period of five weeks between October 23, 2006 and November 30, 2006.

- 109,954: the number of unique IP hosts accessing the system; 20,624 unique hosts (19% of the total number) accessed the system on the day of the elections (November 22);
- 76,360: the number of unique IP hosts that used the search facilities of the system;
- 148,026: the total number of searches made in the system, in particular:
  - 117,132: the number of searches by keywords;
  - 28,025: the number of searches by topic;
  - 2,788: the number of trend requests by keywords;
  - 81: the number of trend requests by topic;
- 6,014: the number of distinct keyword queries;
- 175: the number of distinct topic queries (out of 179 available topics).

The distribution of the actual frequencies of search queries follow a power law and, moreover, 40 most frequent keyword queries (1% of all distinct queries) account for 80% of all keyword searches in the system. It appears that these topic areas are extremely important in the context of these particular elections. Table 5 lists frequencies of the most popular keyword queries for parties' programs.

7. RELATED WORK

Related work comes in several flavors: *multiple-perspective search*, *site scraping and wrapper development*, *morphological normalization for Dutch*, and *log-likelihood-based methods for trend analysis*.

As increasing amounts of complex information are becoming available on the web, there is active research into search engine result presentations that support interaction, exploration, and assimilation of such information. E.g., Hsu

et al. [10] report on determining and displaying semantic and spatio-temporal correlations between web pages. And commerical search engines decorate the familiar ranked result list with ads, products, questons-and-answers, etc, thus combining informational with navigational and commercial perspectives pertinent to a user's query. Multiple perspective search is more widespread than one may realize: essentially, the tabs present in the interface of many commerical search engines each provide a different perspective on the user's query—where the difference lies with the data source, the medium or the type of response (e.g., documents vs. answers). Finally, multiple-perspective search is related to faceted retrieval, where the aim is to identify a broad spectrum of subtopics of a given topic, and to organize the search results accordingly; recent applications of the idea can, for instance, be found in media analysis [11].

Site scraping and wrapper development and induction are active areas of research, cf. e.g., [5, 3]. Our proposal of scraping blocks consisting of 20 or more words is much simpler than those reported in the literature, and gave good results on Dutch newspaper articles. For a discussion of different ways of morphological normalization of Dutch language text, see [9]. Our peak identification method is similar to the method implemented in MoodViews [1, 13].

8. CONCLUSION

We have described an information retrieval system designed to help users in making complex decisions in the face of large amounts of poorly structured textual information. We demonstrated an application of IR techniques to a real-world information access problem: facilitating electoral search. Our system provides an easy and straightforward access to election programs of Dutch policital parties, enabling a user to identify and compare the position of parties on subjects relevant and important for her, and thus to make an informed choice. The framework described in the paper is applicable to likewise situations.

Future work. Since the time of writing we reused the framework for the Dutch provincial elections of March 7, 2007. This turned out harder than the national elections. We now had 12 (provinces) times 10–15 manifestos, and had to make a specialized topic-list for each province. We worked together with `www.kieskompas.nl` to obtain the topics: 36 for each province. This was often difficult, resulting in rather poor precision and recall on the topic queries.

Besides this system for the general public, we created a similar search engine for the developers of `www.kieskompas.nl`. They needed good and fast access to the collection to 1) create topics and theses and 2) score each party on each thesis (in total $12 \times 36 \times (> 10)$ scores for each coder). Manifestos only become available at the last moment and they are manually collected ("begging for them by phone"). They had to be indexed instantaneously. For that, we replaced the paragraph splitter by a splitter based on a fixed 400 character length, creating overlapping text tiles as in [8]. Even though the returned paragraphs were just arbitrarily cut text-snippets the focused retrieval of the search system turned out indispensable for the creators of Kieskompas.

Actually building a system which is supposed to be off-the-shelf technology still brings up new research questions. We list a few: stemming can and must be improved (*ouders* and *ouderen* both stem to *ouder*, leading to paragraphs on *child-allowance* when searching on *elderly people*); compound-splitting must be improved (*hypotheekrenteaftrek* was a hot election topic, but hardly occurred like that in the manifestos); semi-automatically finding good query expansion terms for the fixed-topic queries remains difficult: too much manual intervention was needed for a scalable system.

9. ACKNOWLEDGMENTS

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 354-20-005, 600.065.120, 612-13- 001, 612.000.106, 612.066.302, 612.069.006, 640.001.501, 640.002.501, and by the E.U. IST programme of the 6th FP for RTD under project MultiMATCH contract IST-033104.

10. REFERENCES

[1] K. Balog, G. Mishne, and M. de Rijke. Why Are They Excited? Identifying and explaining spikes in blog mood levels. In *Proceedings EACL 2006*, April 2006.

[2] BlogPulse Trends. was intelliseek, now nielsen buzzmetrics. `http://www.blogpulse.com/trends.html`.

[3] V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: Towards automatic data extraction from large web sites. In *Proceedings VLDB 2001*, pages 109–118, 2001.

[4] Database Group, University of Toronto. BlogScope. `http://www.blogscope.net/`.

[5] D. de Castro Reis, P. B. Golgher, A. S. da Silva, and A. H. F. Laender. Automatic web news extraction using tree edit distance. In *Proceedings WWW 2004*, pages 502–511, 2004.

[6] T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[7] P. Graham. A plan for spam. "`http://www.paulgraham.com/spam.html`", 2002.

[8] M. Hearst. TextTiling: segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1):33–64, 1997.

[9] V. Hollink, J. Kamps, C. Monz, and M. de Rijke. Monolingual document retrieval for European languages. *Information Retrieval*, 7:33–52, 2004.

[10] Y. W. Hsu, N. Moon, and R. Singh. Designing interaction paradigms for web-information search and retrieval. In *Proc. ICWI*, 2006.

[11] B. Liu, M. Hu, and J. Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings WWW '05*, pages 342–351, 2005.

[12] Lucene. The Lucene search engine. `http://lucene.apache.org/`.

[13] MoodViews. Tools for blog mood analysis, 2006. URL: `http://moodviews.com`.

[14] F. van Waveren. Extracting and classifying election-related news items from the world wide web. Master's thesis, University of Amsterdam, 2006. `http://www.var.cx/ac/election2006/verk.pdf`.

[15] I. voor Publiek en Politiek (IPP). `http://www.publiek-politiek.nl/english`.

[16] W3C. Html 4.01 specification. "`http://www.w3.org/TR/html4/`", 1999. Edited by Dave Raggett, Arnaud Le Hors, Ian Jacobs.