Terminology Evolution Module for Web Archives in the LiWA Context*

Nina Tahmasebi
L3S Research Center
Appelstr. 9a
Hannover, Germany
tahmasebi@L3S.de

Gideon Zenz
L3S Research Center
Appelstr. 9a
Hannover, Germany
zenz@L3S.de

Tereza Iofciu
L3S Research Center
Appelstr. 4
Hannover, Germany
iofciu@L3S.de

Thomas Risse
L3S Research Center
Appelstr. 9a
Hannover, Germany
risse@L3S.de

ABSTRACT

More and more national libraries and institutes are archiving the web as a part of the cultural heritage. As with all long term archives, these archives contain text and language that evolves over time. This is particularly true for web archives as content published online is highly dynamic and changing at a fast rate. The language evolution causes gaps between the terminology used for querying and the one stored in long term archives. To ensure access and interpretability of these archives, language evolution must be found and handled in an automatic manner. In this paper we present the LiWA Terminology evolution module, TeVo which takes us one step closer to fully automatic detection of terminology evolution. TeVo consists of a pipeline for finding evolution from web archives based on the UIMA framework. The LiWA TeVo module consists of two main processing chains, the first for Warc file extraction and text processing and the second for finding terminology evolution. We also present the terminology evolution browser, the TeVo browser, which aids in exploring evolution of terms present in archives.

Categories and Subject Descriptors

H.3.6 [**Library Automation**]: Large text archives; H.3.1 [**Content Analysis and Indexing**]: Linguistic processing

General Terms

Terminology Evolution, Semantics, Information Extraction

*This work is partly funded by the European Commission under LiWA (IST 216267).

1. INTRODUCTION

Preserving knowledge for future generations is a major reason for collecting all kinds of publications, web pages, etc. in archives. However, ensuring the archival of content is just the first step toward "full" content preservation. It also has to be guaranteed that content can be found and interpreted in the long run.

Currently the semantic accessibility of web content suffers due to changes in language over time, especially when considering time frames beyond ten years. Language changes are triggered by various factors including new insights, new political and cultural trends, new legal requirements or high-impact events. For example, consider the name of the city Saint Petersburg: the Russian city was founded in 1703 as "Sankt Piter Burh" and soon after renamed to "Saint Petersburg". From 1914-1924 it was named "Petrograd" and afterwards "Leningrad", then changed back to "Saint Petersburg" in 1991. Terminology evolution is not restricted to location names but refers to all added, changed or removed senses for a term.

The work in this paper is done within the scope of the LiWA project[1]. In LiWA, short for Living Web Archives, the objective is to turn web archives from mere web page repositories into *living web archives*. LiWA aims at improving web archives by filtering out irrelevant content such as web spam [8]; and dealing with issues of temporal web archive coherence [19], as well as improving long-term usability.

Within the LiWA project an abstract model presented in [21] and [23] has been developed, that allows the representation of terminology snapshots at different moments in time, extracted from large digital corpora. In order to apply the terminology evolution detection algorithms to web archives we have implemented a terminology extraction pipeline based on the Apache UIMA[2] framework.

The remaining paper is organized as follows. We begin by giving an overview of the architecture for the TeVo module

[1]http://www.liwa-project.eu/
[2]http://incubator.apache.org/uima/

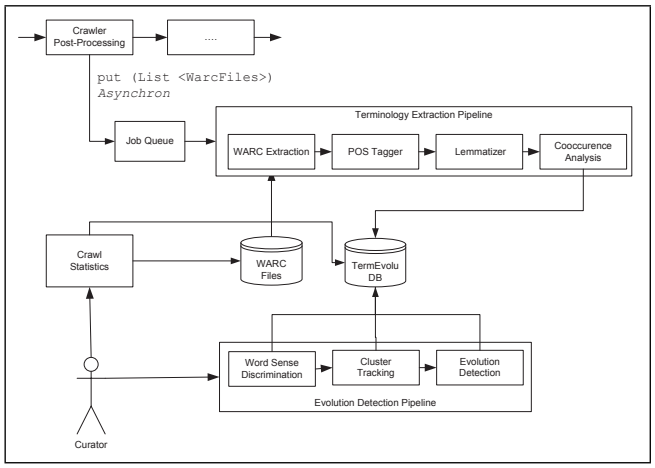

Figure 1: TeVo architecture in LiWA.

in Section 2. The first part of our module concerning terminology extraction is explained in detail in Section 3. The second part concerning word sense detection is explained in Section 4. We present our visualization tool in Section 5. Experiments conducted with the module on a excerpt of a web archive is given in Section 6. We review related work in Section 7 and conclude our paper as well as discuss future work in Section 8.

2. ARCHITECTURE

The LiWA TeVo Module is split into terminology extraction and tracing of terminology evolution. It is a post-processing module and can be triggered once a crawl or a partial crawl is finished. As input the module takes WARC or ARC files created, e.g., by Heritrix.

The terminology extraction pipeline is implemented using UIMA, as presented in Figure 1. Apache UIMA (originally developed by IBM) is a software framework for unstructured information management applications. The UIMA framework is very scalable and can analyze large amounts of unstructured information. Furthermore its modular design allows for easy extension and adoption for the TeVo module. The data exchange between pipeline components is done via the Common Analysis System (CAS) as described in [9].

The evolution detection pipeline is manually triggered by the archive curator based on crawl statistics gathered during terminology extraction. When enough data is extracted or the desired time frame is reached, the curator can start the evolution detection pipeline.

3. TERMINOLOGY EXTRACTION

For extracting terminology from web archives we have build a pipeline, as can be seen in the Terminology Extraction UIMA Pipeline in Figure 1, with the following UIMA components:

- *WARC Extraction*: Archive Collection Reader, using BoilerPipe [10] for extracting text from web documents.

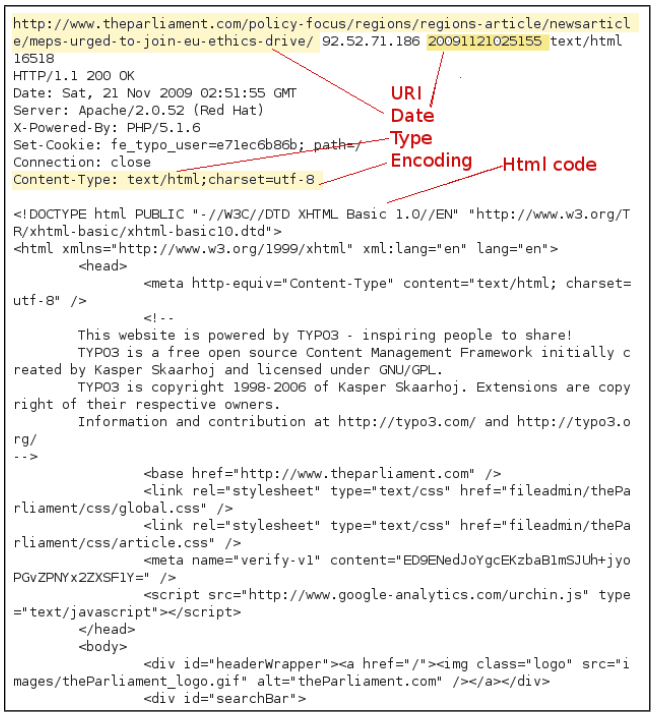

Figure 2: Example of an ARC file and the extracted information.

- *POS Tagger* and *Lemmatizer*: Natural Language Processing using DKPro [13] UIMA components.
- *Cooccurrence Analysis*: AnnotationsToDB, writing the terminology and document metadata to a MySQL database, TeVo DB.

When the processing finishes and the extracted terminology is indexed, terminology co-occurrence graphs can be created for different time intervals. The extraction pipeline can be called several times before the curator initiates the evolution detection pipeline.

3.1 Collection Reader

The first task in our system is to extract and annotate the text from web archives, which have the ARC or WARC format[3]. In order to iterate through the web pages crawled and stored in an archive we integrated the archive reading tools from Heretrix Java Api[4] from Internet Archive.

For each archive file we retrieve the URL, the crawl date, the content type and the encoding from the archive header as shown in Figure 2.

For the archive files with content type *text/html* we retrieve the html content and then, based on the encoding, we extract the textual content using the BoilerPipe Java Api [10]. One of the major issues when dealing with web data is extracting the text from the html document. By using simple

[3]http://www.digitalpreservation.gov/formats/fdd/fdd000236.shtml
[4]http://crawler.archive.org/

HTML stripping tools, it is inevitable to extract also unwanted text, like table headers and advertisement tags. By using the boilerpipe library algorithms we better extract the main textual content of a web page. Extracting content is very fast (milliseconds) and no global or site-level information is required.

For each text file from the archive we create a document with the URI annotated as document id and text as ArticleText. As additional document metadata we annotate the crawl date and the crawl title as given by the archive name.

3.2 NLP Annotator

In the next step we annotate the article text using the components of the DKPro (Darmstadt Knowledge Repository)[5], a collection of UIMA-based components for NLP tasks. We process the text of each article using two annotators from DKPro: the *BreakIteratorSegmenter* for sentence splitting and tokenization, and the *TreeTaggerPosLemmaTT4J* annotator for part-of-speech tagging and lemmatization. The *POSTagger* and *Lemmatizer* from DKPro are wrappers for TreeTagger [17].

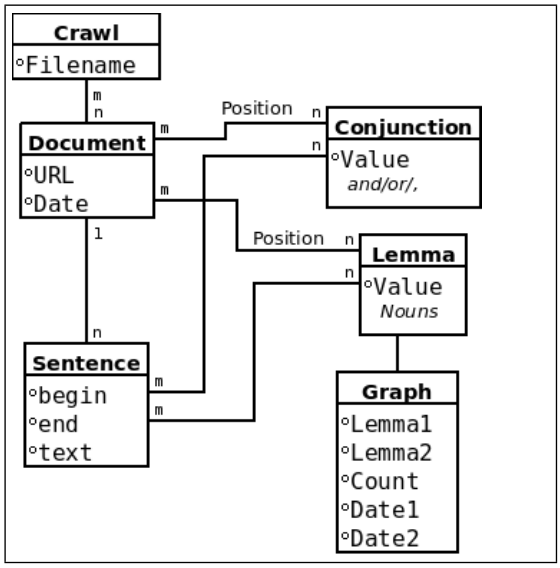

Figure 3: TeVo DB Schema

3.3 Terminology Indexing

In order to make the terminology extracted from the archives available for further analysis, we implemented a database index for the found documents and terms. For detecting terminology evolution, the curvature clustering algorithm presented in [7] works on lemmas of nouns co-occurring within *and* and *or* clauses, or in lists separated by commas.

In the *AnnotationsToDB* annotator we insert metadata about each analyzed document, i.e. its URL, the crawling date and the crawl name, into the database (see the diagram in Figure 3). Additionally we insert all the annotated sentences with the information regarding sentence position.

As a start, we are interested in the evolution of nouns. Therefore we only insert lemmas of nouns occurring in documents into the database, along with their position information, beginning and end offset. All remaining annotated tokens, other than nouns and conjunctions are omitted. These would make the preprocessing expensive both in terms of storage as well as running time. When building the co-occurrence graphs we consider nouns appearing in the same list, thus we have to tackle the problem of articles and cardinals appearing before a noun. For example, in the sentence "It involves forgiveness and a readiness to accept ...", the nouns *forgiveness* and *readiness* appear in an "and" clause. Inserting the original offsets of the two nouns would hinder us from retrieving the relations between them using a simple *select* based on position. To overcome this limitation for all nouns preceded by articles (a, the, ...) or cardinals (one, 20, ...) we use the beginning offset of the preceding token as beginning offset.

One issue that frequently appears when dealing with archives generated by regular crawls is duplicate data. Pages that are not changed between two crawls will be duplicated in the archive. At terminology level, duplicate data may negatively bias the terminology analysis [23]. To overcome this issue, when a document is found with a URL which already exists in the index, we check if the two documents have the same set of lemmas. We consider two documents to be the same, or have non-significant changes if the overlap between the two lemma sets is higher than a threshold. The threshold can be configured, based on initial experiments we used a threshold of 80%.

Web pages can be static or dynamically generated. While the static pages have a unique fixed URL, the dynamic ones are being generated based on the parameters in the POST or GET parameters from the HTML request. Thus, within the same crawl there are possibly different pages with the same URL. For this situation, we first check that the pages have distinct lemma sets as in the previous scenario. If we can conclude that the pages indeed have different content, we append a random number to the URL of the second page before inserting it as the document identifier in the database.

Finally the co-occurrence graph is stored in the TeVo DB and can be fetched by the evolution detection module.

4. TERMINOLOGY EVOLUTION

After finishing the terminology indexing step, we can start extracting word senses and tracking evolution. In the first step we fetch a co-occurrence graph from the TeVo DB. We keep all co-occurrences and consider the graph as an unweighted graph. We then extract word senses by clustering the co-occurrence graph. The clustering algorithm is based on clustering coefficient of each term in the graph, i.e., the interconnectedness of the neighbors of a term. Terms that have a highly interconnected neighborhood are likely to present stable terms, while terms with a sparsely connected neighborhood are likely to be ambiguous [7]. By removing the terms with low clustering coefficient, the graph falls apart into coherent subgraphs (clusters) which we interpret as word senses. Previous works [6, 7, 14] have used a clustering coefficient of 0.5 which provides stable word senses. In

[5]http://www.ukp.tu-darmstadt.de/research/projects/dkpro/

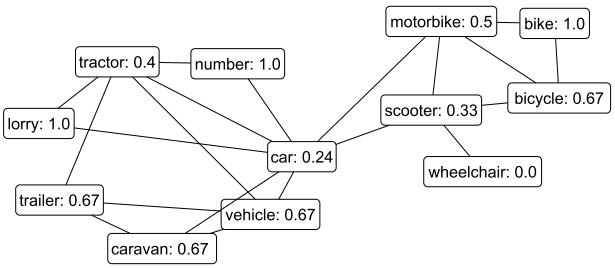

Figure 4: Subgraph from .gov.uk archive from 2006, each node contains also its clustering coefficient from the local graph. The graph shows two clusters which both correspond to means of transportation.

our experiments we use a clustering coefficient of 0.3 which has shown to provide a larger number of word senses as well as a higher probability of evolution. It should be noted that the extracted word senses represent the data available in the collection from which the co-occurrence analysis has been gathered.

Figure 4 shows a small part of the graph created for the last quarter of 2006 from .gov.uk crawls. The term *car* has a low clustering coefficient (0.24) since it has many neighbors which are not connected to each other. When *car* is removed, the graph falls apart into two smaller subgraphs. In order to make these clusters capture also ambiguous words, the terms which have been removed will be added in the clusters where they have neighbors. The result is (*bicycle, bike, car, motorbike, scooter, wheelchair*) and (*car, vehicle, caravan, trailer, tractor, lorry, number*). Each term in one cluster is closer in meaning to the other words in the cluster than words from the other cluster. Both clusters correspond to means of transportation but differ in that the first cluster corresponds to smaller, mostly two wheel vehicles, while the second cluster corresponds to larger, four or more wheeled vehicles.

Once we have created clusters, i.e., word senses, for each period in time, we can compare these clusters to see if there has been any evolution. Consider the cluster $C1$ =(*bicycle, bike, car, motorbike, scooter, wheelchair*) from 2006. In 2007 we find the exact same cluster, most likely indicating that the web pages from 2006 stayed unchanged in 2007. In 2008, we find a cluster $C2$ =(*motorcycle, moped, scooter, car, motorbike*). Because of the high overlap between the clusters, e.g., *car, motorbike, scooter*, we can draw the conclusion that they are highly related. Still we see some shift. In $C1$, the words *bicycle, bike* and *wheelchair* are representatives of unmotorized means of transportation, while in $C2$, only motorized means of transportations are present. Because of the short time span we cannot draw any conclusions of real terminology evolution, however we can say that there has been a shift in usage in the archive. In 2003-2005 we cannot find a cluster related to *motorbike*. This is likely a consequence of data selection and comes from using a random sample of each crawl.

Current LiWA tracking technology use Jaccard similarity to compare clusters. This means that for two clusters, we look at the fraction between the overlapping terms and the total number of distinct terms contained in both clusters. The similarity scores for two clusters lie between 0 and 1. Similarity of 1 indicates that two clusters are exactly the same and a similarity of 0 indicates that two clusters have no terms in common. For $C1$ and $C2$ the Jaccard similarity is $\frac{3}{8}$. We consider two clusters which have a similarity higher than α to represent the same word sense. In the above example $C1$ would keep its meaning even if one word was removed from one archive to another, e.g., $C1' =$ (*bicycle, bike, car, motorbike, scooter*) and $C1$ can be considered to represent the same sense. When two clusters have a similarity below β we consider the clusters to have no relation. Clusters with similarity above β but below α are candidates for evolution.

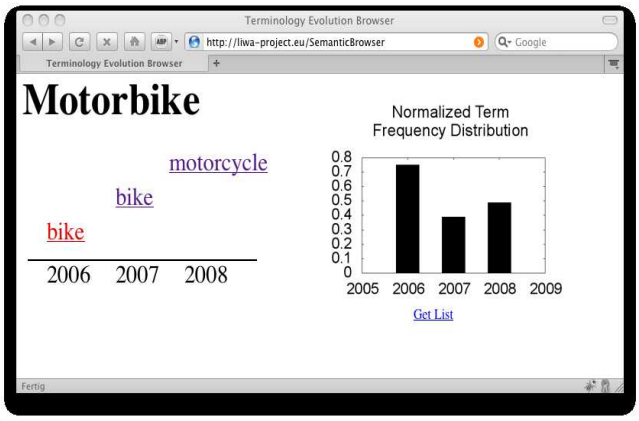

Figure 5: User-interface showing clusters for term *Motorbike*

year	cluster members
1867	yard, terrace, flight
1892	hurdle race, flight, year, steeplechase
1927	flight, england, london, ontariolondon
1938	length, flight, spin, pace
1957	flight, speed, direction spin, pace
1973	flight, riding, sailing, vino, free skiing
1980	flight, visa, free board, week, pocket money, home
1984	flight, swimming pool, transfer, accommodation

Table 1: Selected clusters and cluster members for the term 'flight' from The Times Archive.

Because available web archives span a relatively short period of time, true terminology evolution becomes difficult to find. Therefore, as an example of cluster evolution, we show clusters from The Times Archive [11]. The archive spans from $1785 - 1985$ and is split into yearly sub-collections and processed according to [22]. In Table 1 we see clusters for the term *flight*. Among the displayed clusters it is clear that the senses for flight are several and mostly grouped together. Between 1867-1894 there are 5 clusters (only two of them displayed here) that all refer to *hurdle races*. Between the years 1938 - 1957 the clusters are referring to *cricket*, the terms in the clusters are referring to the ball. Starting from 1973 the clusters correspond to the modern sense of flight as a means of travel, especially for holidays. The introduction of among others *pocket money, visa, accommodation*, differ-

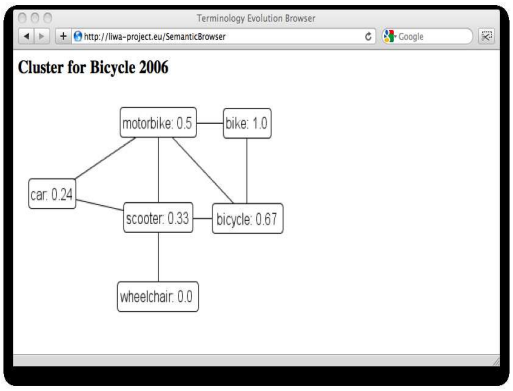

Figure 6: Graphical representation of 2006's *Motorbike*-Cluster

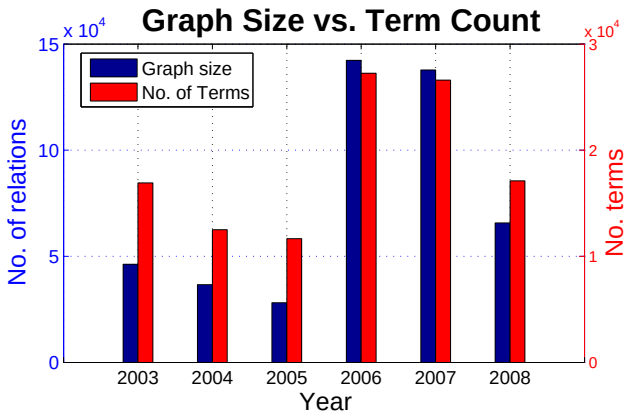

Figure 7: Number of relations shown for samples from 2003-2008. We also see the amount of unique terms from these relations.

entiates the latter clusters from the earlier. Also the cluster in 1927 refers to a flight but not necessarily in a holiday sense.

5. VISUALIZATION OF EVOLUTION

In order to make the results of the terminology evolution process end-user accessible, we devised a web-based user interface which allows for exploring the evolution of a given term. As running example we will use the term *motorbike* present in clusters from the 2006-2008 .gov.uk crawls (see Section 6). After the user specifies the term of interest, here *motorbike*, we show all clusters representing this term over time by displaying the term with the highest clustering coefficient for each cluster over a time line (right side, Figure 5). Furthermore, we give the term frequency distribution of the term over time (left side, Figure 5).

By assessing the term frequency distribution, and possibly combined with a changing cluster representative as seen on the right side, the user can infer if a significant change of the word usage happened at a given point in time.

To get a deeper understanding of the context of a given year, the user can click on a cluster representative. As shown in Figure 6, all cluster members are displayed along with their connection.

The TeVo visualization browser enables the user to get a quick look at what happens to a term over time. First of all the raw (or normalized) term frequencies over time can give an indication of an event, or evolution, for a term. If the term 'motorbike' spikes in frequency in one year it is worth the effort to investigate further into that term. In addition to the term frequencies, the clusters help in getting term context. Assume that in 2007, a music group named *motorbike* is started. Then an additional cluster would appear, containing terms such as *music, concerts, CD, release* etc. From this, the user would get an indication of an added (or inversely) removed word sense. In addition, if there has been more subtle evolution within a cluster, the user can click on that cluster and see the other cluster members. In Figure 6, based on the connections between the cluster members, i.e., only to one other term in the cluster, the user can deduce that the term *wheelchair* is less relevant to the cluster for *motorbike*. The TeVo visualization browser saves the user time in finding and reading web pages from the archive for the term 'motorbike' from different periods in time to get this overview.

6. EXPERIMENTS

Our experiments are conducted on sample archives from .gov.uk crawls available at European Archives[6]. Archives from December each year are chosen and processed. The results are varying sized samples for which we will present details. Firstly it is clear that the amount of relations extracted from the yearly samples vary heavily because of the type of archive. As web archives can contain multimedia files, images, videos etc, it is difficult to predict the amount of text from such a sample. This limits the control over the amount of text extracted and indexed. Furthermore, even if we can control the amount of text that is processed, if the crawl is too wide, the extracted relations become sparse and the amount of useful information in each co-occurrence graph varies heavily.

In Figure 7 we see the amount of relations as well as how many unique terms were present in the graphs. In 2003 each term has an average of 2.7 relations in the resulting graph. In 2006 or 2007, each term has an average of roughly 5.2 relations. A major factor for the observed variation is the diversity of the data in the crawl. The more diverse data that is chosen for processing, the fewer are the amount of relations per term. This varying behavior is also shown for the number of clusters extracted from the co-occurrence graphs, Figure 8. In 2003 there is one cluster for every 36th relation while in 2006 and 2007 there is one cluster for every 700th relation. In 2008 we have one cluster for every 470th relation. This is most likely a result of the sparseness in the crawls. Too many topics result in a graph with many relations that are not creating triangles, which results in many terms with a low clustering coefficient. These terms are removed in the clustering and do not contribute to creating clusters.

[6]http://www.europarchive.org

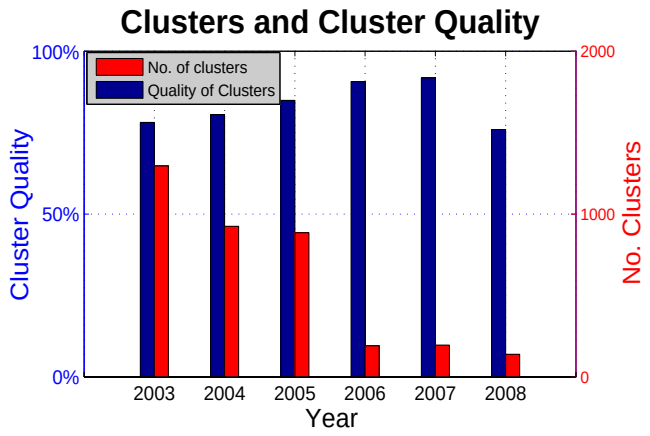

Figure 8: Number of clusters shown for samples from 2003-2008. We also see the quality of these clusters.

We measure the quality of the clusters by measuring the correspondence of clusters to WordNet synsets [15], i.e., the amount of clusters which correspond to a word sense. For this reason we evaluate all clusters with more than one term from WordNet. An average of 68% of the clusters are used for evaluation. Figure 8 shows the number of clusters per year as well as the quality of the clusters. In 2003 where we have the highest amount of clusters (shown in red), we have a fairly low quality of the clusters. Only 3 out of 4 clusters correspond to a word sense. In 2006 and 2007 on the other hand, we have fewer clusters with a high quality where 9 out of 10 clusters correspond to a word sense. In 2008 the quality is again lowered. We see that the results of the word sense discrimination algorithm is very irregular and highly dependent on the underlying archive. If the documents in the archive contain high quality, descriptive text, then the clusters have a high quality. If on the other hand, the documents in the archive contain a high amount of spam and advertisements, incorrectly written English etc., then good quality clusters are more rare.

The remaining clusters, i.e., clusters that are not representing word senses or not considered in the evaluation, are not necessarily semantically unrelated clusters. In many cases they are just not corresponding to word senses. As an example, in 2003 we have many clusters which contain people names and names of documents, forms etc. (*t.ereau, m-b.delisle, n.kopp, a.dorandeu, f.chretien, h.adle-biassette, f.gray*) are all authors of a paper about Creutzfeldt-Jakob disease and (*sa105, sa104f, sa107, sa103, sa106, sa104, sa1-03l*) are all tax return forms.

6.1 Performance Analysis for Extraction

The run-time performance of the annotators, measured over a 93.5 MB compressed web archive, is shown in Table 2. The uncompressed archive size is 194.6 MB and contains 9743 documents, out of which 7654 are of *text/html* content type and in English. All the annotators required on average less than a second per document. Most time is used by the AnnotationsToDB annotator which transfers the data to the database. As we do not need the produced CAS files, we do not need to write the CAS files on disk, thus the time needed by the CAS Writer can be deducted from the processing time.

%	Time(ms)	s/doc	Component
1.09%	59905	0.00783	ARC Reader
0.36%	19577	0.00256	BreakIteratorSegmenter
1.93%	106055	0.01386	TreeTaggerPosLemma
95.03%	5218886	0.68185	AnnotationsToDB
1.56%	85732	0.01120	CAS Writer
100%	5508361	0.71967	Entire Pipeline

Table 2: Annotator Processing Time for 7654 documents

Given the amount of data a typical crawl consists of, the performance of the annotator still leaves room for improvement. Completing the terminology extraction process for the gov.uk crawl used in our experiments took 14 days. For 2006-2008 the size of the crawls are presented in Table 3.

One option to speed up the processing is to only index the lemmas that are needed in the next step, i.e., lemmas with a preceding or succeeding conjunction. While this restriction saves processing time, it makes the data unusable for other co-occurrence filters, like sentence level or window level co-occurrence.

Year	Number of archives	Average size	Total size
2006	231	93.07 MB	21.5 GB
2007	3087	93.75 MB	289.4 GB
2008	1250	95.36 MB	119.2 GB

Table 3: .gov.uk crawls

We create co-occurrence graphs by selecting lemmas that appear to the left and right of found conjunctions. From a 3 GB snippet of the .gov.uk crawl of 2006, 59.844 co-occurrences were extracted, out of which most of the co-occurrences had a frequency of 1. This indicates that in order to produce meaningful co-occurrence graphs that do not experience the same variance as seen in Figure 7 and 8, terabytes of data is needed.

7. RELATED WORK

For finding word senses in an automatic way, i.e., word sense discrimination, several methods based on co-occurrence analysis and clustering have been proposed like [4, 16, 18]. Taking semantic structures into account improves the discrimination quality. In Dorow et al. [6, 7] it is shown that co-occurrences of nouns in lists contain valuable information about the meaning of words. A graph is constructed where the nodes are nouns and noun phrases. There exists an edge between two nodes if the corresponding nouns are separated by “and”, “or” or commas in the collection. The graph is clustered based on the clustering coeffient of a node and the resulting clusters contain semantically related terms representing word senses. Another approach of word sense discovery is focused on pattern discovery, such as the one presented in [4]. In [15] a clustering algorithm called Clustering by Committee is presented. This clustering produces

clusters with words that can be considered synonymous. An evaluation method is also proposed, where the discovered word senses can be assessed using WordNet [12].

The output from word sense discrimination is normally a set of terms to describe senses found in the collection. This grouping of terms is derived from clustering and we refer to such an automatically found sense as a cluster. Clustering techniques can be divided into hard and soft clustering algorithms. In hard clustering, an element can only appear in one cluster, while soft clustering allows each element to appear in several clusters. Due to the polysemous property of words, soft clustering is most appropriate for word sense discrimination.

Temporal aspects in information retrieval come in different flavors, such as dealing with temporal information within documents, or with temporally versioned documents, or dealing with temporal evolution of terminologies extracted from documents. According to our analysis not much work has been done on the problem of terminology evolution. Abecker et al. [1] show how medical vocabulary evolved in the MEDLINE system. McCray investigates the evolution of the MESH ontology [2]. In the latter study, psychiatric and psychological terms are manually analyzed and their evolution is studied over 45 years. Terminology evolution can also be observed in other domains. For example, in computer science the Faceted DBLP[7] allows analysis of the evolution of given keywords at different times based on the Semantic GrowBag approach [5]. However, all these approaches assess the evolution manually. Furthermore, the results cannot directly be used by information retrieval systems.

Automatic detection of cluster evolution can aid in automatically detecting terminology evolution. This has been a well studied field in the recent years. One such approach for modeling and tracking cluster transitions is presented in a framework called Monic [20]. In this framework internal as well as external cluster transitions are monitored. The disadvantages of the method are that the algorithm assumes a hard clustering and that each cluster is considered as a set of elements without respect to the links between the elements of the cluster. In a network of lexical co-occurrences, the links can be valuable since the connections between terms give useful information to the sense being presented. In [14], a way to detect evolution is presented which also considers the edge structure among cluster members.

To our knowledge only one previous work has been published in the area of terminology evolution [3]. Using language from the past, the aim here is to find good query reformulations of concurrent language. A term from a query can be reformulated with a similar term if the terms in the resulting query are also coherent and popular. Terms are considered similar if they co-occur with similar terms from their respective collections. Our approach advances on this by using word senses to find similar terms rather than pure co-occurrence information. Furthermore our approach gives more advanced knowledge on the evolution such as time information on the valid reformulations.

[7]http://dblp.l3s.de/

8. CONCLUSIONS AND FUTURE WORK

In this paper we presented the LiWA Terminology evolution module, TeVo, which takes us one step closer to fully automatic evolution detection given a long term archive. TeVo can be integrated effectively as a Heritrix post-processing module. We focused on extracting terminology needed for noun evolution detection and on overcoming the challenges appearing from dealing with web archives. We also introduce a tool which helps visualizing knowledge from, and discovering properties of a given archive.

We conducted experiments on selections of a large real-world dataset, which helped us to gain first insights into applying our terminology evolution algorithms to web archives. As our experiments show, the variance regarding size and quality of the co-occurrence graphs was significant. In order to overcome these issues we need to improve on data selection or significantly increase the amount of data used. It is also necessary to investigate further how to properly handle duplicate data and to find the consequences of keeping or removing such data.

As future work, we intend to optimize the indexing component to make it more applicable to web-scale data. We also want to utilize the *TreeTaggerChunker* from DKPro to annotate n-grams for detecting noun phrases. Additionally, instead of only taking cluster evolution into account, we want to specifically find term evolution. This will allow us to determine e.g. that *automobile* evolved into *car*.

Furthermore, we intend to use the TeVo browser to display parts of the co-occurrence graph for each term. This will help users to gain even more detailed insights about a term for a time period. The modules from the TeVo architecture as well as the TeVo browser will be released as open source modules before the end of the LiWA project[8].

9. ACKNOWLEDGEMENTS

We would like to thank Times Newspapers Limited for providing the archive of The Times for our research.

10. REFERENCES

[1] A. Abecker and L. Stojanovic. Ontology evolution: Medline case study. In *Proceedings of Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety*, pages 1291–1308, 2005.

[2] Alexa McCray. Taxonomic change as a reflection of progress in a scientific discipline, www.l3s.de/web/upload/talk/mccray-talk.pdf.

[3] K. Berberich, S. Bedathur, M. Sozio, and G. Wiekum. Bridging the terminology gap in web archive search. In *WebDB*, 2009.

[4] D. Davidov and A. Rappoport. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 297–304, Sydney, Australia, 2006.

[5] J. Diederich and W. T. Balke. The semantic growbag algorithm: Automatically deriving categorization

[8]http://code.google.com/p/liwa-technologies/

systems. In *ECDL*, volume 4675 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2007.

[6] B. Dorow. *A Graph Model for Words and their Meanings*. PhD thesis, University of Stuttgart, 2007.

[7] B. Dorow, J. pierre Eckmann, and D. Sergi. Using curvature and markov clustering in graphs for lexical acquisition and word sense discrimination. In *In Workshop MEANING-2005*, 2004.

[8] M. Erdélyi, A. A. Benczúr, J. Masanés, and D. Siklósi. Web spam filtering in internet archives. In *AIRWeb*, pages 17–20, 2009.

[9] T. Götz and O. Suhre. Design and implementation of the uima common analysis system. *IBM Syst. J.*, 43(3):476–489, 2004.

[10] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 441–450, New York, NY, USA, 2010. ACM.

[11] T. N. Ltd. The Times Archive. http://archive.timesonline.co.uk/tol/archive/.

[12] G. A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41, 1995.

[13] C. Müller, T. Zesch, M.-C. Müller, D. Bernhard, K. Ignatova, I. Gurevych, and M. Mühlhäuser. Flexible uima components for information retrieval research. In *Proceedings of the LREC 2008 Workshop 'Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP'*, pages 24–27, Marrakech, Morocco, May 2008.

[14] G. Palla, A.-L. Barabasi, and T. Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, April 2007.

[15] P. Pantel and D. Lin. Discovering word senses from text. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619, Edmonton, Alberta, Canada, 2002. ACM.

[16] T. Pedersen and R. Bruce. Distinguishing word senses in untagged text. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 197–207, Providence, RI, 1997. Comment: 11 pages, latex, uses aclap.sty.

[17] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK, 1994. http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html.

[18] H. Schütze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

[19] M. Spaniol, A. Mazeika, D. Denev, and G. Weikum. âĂIJcatch me if you canâĂİ: Visual analysis of coherence defects in web archiving. In *In Proceedings of 9th International Web Archiving Workshop in conjunction with ECDL 2009*, Corfu, Greece, 2009.

[20] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult. Monic: modeling and monitoring cluster transitions. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 706–711, New York, NY, USA, 2006. ACM.

[21] N. Tahmasebi, T. Iofciu, T. Risse, C. Niederée, and W. Siberski. Terminology evolution in web archiving: Open issues. In *8th International Web Archiving Workshop, Aarhus, Denmark, 18th & 19th Sep. 2008*, 2008. http://iwaw.net/08/IWAW2008-Tahmasebi.pdf.

[22] N. Tahmasebi, K. Niklas, T. Theuerkauf, and T. Risse. Using word sense discrimination on historic document collections. In *JCDL '10: Proceedings of the 10th ACM/IEEE-CS joint conference on Digital libraries*, Gold Coast, Australia, 2010. ACM.

[23] N. Tahmasebi, S. Ramesh, and T. Risse. First results on detecting term evolutions. In *In Proceedings of 9th International Web Archiving Workshop in conjunction with ECDL 2009*, Corfu, Greece, 2009.