BlogNEER: Applying Named Entity Evolution Recognition on the Blogosphere*

Helge Holzmann[1], Nina Tahmasebi[2**], and Thomas Risse[1]

[1] L3S Research Center,
Appelstr. 9, 30167 Hannover, Germany
{holzmann,risse}@L3S.de
[2] Computer Science & Engineering Department,
Chalmers University of Technology,
412 96 Gothenburg, Sweden
ninat@chalmers.se

Abstract. The introduction of Social Media allowed more people to publish texts by removing barriers that are technical but also social such as the editorial controls that exist in traditional media. The resulting language tends to be more like spoken language because people adapt their use to the medium. Since spoken language is more dynamic, more new and short lived terms are introduced also in written format on the Web. In [1] we presented an unsupervised method for Named Entity Evolution Recognition (NEER) to find name changes in newspaper collections. In this paper we present BlogNEER, an extension to apply NEER on blog data. The language used in blogs is often closer to spoken language than to language used in traditional media. BlogNEER introduces a novel semantic filtering method that makes use of Semantic Web resources (i.e., DBpedia) to gain more information about terms. We present the approach of BlogNEER and initial results that show the potentials of the approach.

Keywords: Named Entity Evolution, Blogs, Semantic Web, DBpedia

1 Introduction

The introduction of new technology changes the way we express ourselves [2]. In Social Media, like blogs, everyone can publish content, discuss, comment, rate, and re-use content from anywhere with minimal effort. The constant availability of computers and mobile devices allows communicating with little effort, few restrictions, and increasing frequency. As there are no requirements for formal or correct language, authors can change their language use dynamically. Under these circumstances we expect people to adapt their language to the means of communication by using more creative language and unconventional spellings.

* This work is partly funded by the European Commission under ARCOMEM (ICT 270239)

** This work was done while the author was employed at L3S Research Center

Also words which might otherwise have been reserved for use only in conversations between friends can be introduced in written text.

These changes lead to a more dynamic language where new and short lived terms are introduced also in written format. Local as well as global language trends can spread via forums on the Web to a larger audience. This shortened gap between written "Web Language" and spoken language coupled with the inherent dynamics of spoken language leads to the introduction of new terms and high dynamics also in written language.

With the increasing efforts in documenting and preserving the public view on certain events and topics like the Financial Crisis or the Olympic Games, there is also an increasing need to make use of this content. To turn user generated content into valuable information requires a better "understanding" of the content. A systems that is aware of this knowledge can support information retrieval by augmenting the query term. Awareness of language evolution is in particular important for searching tasks in archives due to the different ages of the involved texts.

Language evolution is a broad area and covers many sub-classes like word sense evolution, term to term evolution, named entity evolution and spelling variations. In [1] we presented our approach for Named Entity Evolution Recognition (NEER). NEER is an unsupervised method to find name changes without using external knowledge sources. As an example consider *Pope Benedict XVI*, formerly known as *Joseph Ratzinger*. NEER can detect those changes in a high quality newspaper dataset that reports this evolution by analyzing co-occurring terms.

In this paper we present a first extension of NEER towards "Web Language" by adapting and applying the method to blog content. The language used in blogs is often closer to spoken language than to language used in traditional media [3]. BlogNEER, an extension of NEER that introduces a novel semantic filtering method, makes use of semantic resources (here exemplarily DBpedia) to gain more information about terms.

In the following section we present the related work in the field of named entity evolution. In Section 3 we give an introduction to NEER and motivate BlogNEER. Section 4 explains our novel filtering method utilizing external resources from the Semantic Web. In Section 5 we describe our experiments and show an example. Section 6 concludes the work and gives an outlook on future work.

2 Related Work

Previous work on automatic detection of language evolution has mainly focused on named entity evolution. The interest has mainly been from an information retrieval point of view as search results can be affected by named entity evolution.

Berberich et al. [4] proposed a solution to this problem by reformulating a query into terms prevalent in the past. They measure the degree of relatedness between two terms when used at different times by comparing the contexts as

captured by co-occurrence statistics. This approach requires a recurrent computation each time a query is submitted as it requires a target time for the query reformulations which reduces efficiency and scalability. The results presented in this paper are "anecdotal" (to use the words of the authors) and thus do not provide a basis for comparison. However, because of the promising results we use the same method for defining a context.

Kaluarachchi et al. [5] propose to discover semantically identical concepts (or named entities) used at different times. They discover these changing entities using association rule mining by associating distinct entities to events. Sentences containing a subject, a verb, objects, and nouns are targeted and the verb is interpreted as an event. Two entities are considered semantically related if their associated event is the same and the event occurs multiple times in a document archive. The temporally related term of a given named entity is used for query translation (or reformulation) and results are retrieved appropriately w.r.t. specified time criteria. They present precision and recall for three queries and evaluate only indirectly on the basis of retrieved documents.

Kanhabua et al. [6] define a time-based synonym as a term semantically related to a named entity at a particular time period. They extract synonyms of named entities from link anchor texts in Wikipedia articles using the full history. The paper evaluates the precision and recall of the time-based synonyms by measuring increased precision and recall in search results rather than directly evaluating the quality of the found synonyms.

In more recent work, Mazeika et al. [7] consider semantically similar entities from different time periods. They extract named entities from the YAGO ontology and provide a visual analytics tool to analyze the evolution of named entities of the New York Times Annotated Corpus. No name changes are tracked but the tool offers a visualization of the evolution of an entity in the relation to other entities.

3 Named Entity Evolution Recognition

The NEER approach addresses the problem of automatically detecting named entity evolution. It works unsupervised and without incorporating external resources. This section gives an overview of NEER and its limitations on blog data.

3.1 Definitions

We consider a **term** w_i to be a single or multi-word lexical representation of an entity at time t_i. The **context** C_{w_i} is the set of all terms related to w_i at time t_i. Similar to Berberich et al. [4] we consider the most frequently co-occurring terms within a distance of k words as the context, however, other contexts can be used. We consider a **change period** to be a period of time in which one term evolves into another. We consider **temporal co-references** to be different lexical representations that have been used to reference the same concept or entity at the

different periods in time. **Direct temporal co-references** are temporal co-references that are variations of each other with some lexical overlap. **Indirect temporal co-references** are temporal co-references that lack lexical overlap on the token level. A **temporal co-reference class** contains all direct temporal co-references for a given named entity, denoted as $coref_r$ $\{w_1,\ w_2\ ,\ \ldots\}$. Each temporal co-reference class is represented by a class representative r which is also a member of the class. For example, Joseph Ratzinger is the representative of the co-reference class containing the terms {*Joseph Ratzinger, Cardinal Ratzinger, Cardinal Joseph Ratzinger, ...*}.

3.2 Overview of NEER

The major steps of the Named Entity Evolution Recognition (NEER) approach are depicted in Figure 1. NEER utilizes change period for finding named entity evolution. These periods are identified by detecting high frequency bursts of an entity. Those are considered to indicate a change period. Texts from the year around a burst are regarded for collecting the co-reference candidates by extracting the relevant terms. These are used to build up contexts represented as graphs. Based on the contexts four rules are being applied to find direct co-references among the extracted terms. These are merged to co-reference classes as follows:

1. *Prefix/suffix rule*: Terms with the same prefix/suffix are merged (e.g., Pope Benedict and Benedict).
2. *Sub-term rule*: Terms with all words of one term are contained in the other term are merged (e.g., Cardinal Joseph Ratzinger and Cardinal Ratzinger).
3. *Prolong rule*: Terms having an overlap are merged into a longer term (e.g., Pope John Paul and John Paul II are merged to Pope John Paul II).
4. *Soft sub-term rule*: Terms with similar frequency are merged as in rule 2, but regardless of the order of the words.

Ultimately, the graphs are being consolidated by means of the co-references classes. Afterwards filtering methods filter out false co-references that do not refer to the query term. For this purpose, statistical as well as machine learning (ML) based filters were introduced. A comparison of the methods revealed their strengths and weaknesses in increasing precision while keeping a high recall. The ML approach performed best with noticeable precision and recall of more than 90%. While it is possible to deliver a high accuracy with NEER + ML, training the needed ML classifier requires manual labelling.

3.3 Limitations of NEER Applied on Blog Data

Tahmasebi et al. [3] showed that language in blog texts behaves differently than traditional written language. Blog language is much more dynamic and closer to spoken language than written language in traditional media. Therefore, we treat blog texts differently than texts from newspapers.

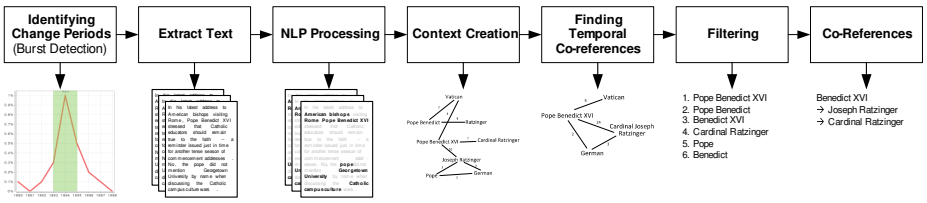

Fig. 1. Pipeline used to detect temporal co-references[1].

The machine learning filter, which delivered best results in NEER experiments, achieved a precision of more then 90% by filtering out false detected co-references. Applying this to blog data leads to a much wider contexts, containing many unrelated terms due to the large amount of relatively low quality texts. Therefore the NEER filtering methods would have a much lower effect.

NEER makes no use of external resources like DBpedia since the main development goal was to apply on historical document collections. Incorporating the Semantic Web allows us to filter out false detected names using semantic information, which is reasonable when working with data from the Web, like blogs.

4 Semantic Filtering Approach

Semantic Filtering is a novel a-posteriori filtering method for NEER incorporating the Semantic Web. With this approach we exemplarily use external data from DBpedia to augment a term with semantic information. Employing these, we are able to filter out names that do not refer to the same entity. Two terms referring to entities of different types or categories can not be evolutions of each other. A-posteriori means we apply this filter after applying NEER to our dataset given a query term and one or more change periods. At this step we have access to the NEER results which consist of a collection of indirect co-references and a co-reference class for the query term, composing the direct co-references. Using this filter, all co-references that could be identified as names for other entitites will be filtered out.

The semantic filter incorporates semantic information from DBpedia which are structured as resources. A **resource** on DBpedia is the structured representation of a Wikipedia page, which is automatically extracted as described by Bizer et al. [8]. While an ambiguous name can refer to multiple resources, every resource has its own unique name and every name only points to one resource directly. This is realized by using **disambiguation resources**. E.g., *Apple_(disambiguation)* is the disambiguation resource of the resource *Apple* (the fruit) and *Apple_Inc.* Unlike this example, disambiguation resources do not always have the "disambiguation" suffix. However, every resource has **properties**, which either point to a textual or numeric value, or to another resource.

Disambiguation resources can be identified by the existence of **disambiguation properties** that point to their corresponding unambiguous resources.

Other properties which are important for our work are the **types** of resources as well as **subjects**, which can be conceived as categories. In addition to the property relations (resource → property → value), DBpedia also provides the inverse relations (value → is-property-of → resource). These can help to detect ambiguous resources where the corresponding disambiguation resource points to the ambiguous one (e.g., *Apple (dismbiguation)* disambiguates *Apple*).

By mapping a query term as well as all of its co-references (direct and indirect) to DBpedia resources we can augment the terms with semantic properties. These properties can help to filter out false positive results derived by NEER as new names for the entity. It is important to mention that we only make use of descriptive properties and will not utilize already known name evolution information and co-references from DBpedia. In this paper we focus on a term's types and subjects, but also make use of redirects and disambiguations. Although, in some cases redirects represent a name change as well by redirecting an old name to its new name, we do not use this information explicitly. Hence, we treat all terms separately, even if they redirect to the same resource, like there is no redirection available (e.g., for *Czechoslovakia* and *Czech Republic* or *Slovakia*).

4.1 Disambiguation and Aggregation of Properties

To map a term to a DBpedia resource, we replace spaces with underscores and append it to the DBpedia resource URI (e.g., for "Project Natal" the resource URI becomes http://dbpedia.org/resource/Project_Natal). In case we are able to resolve a term to a resource we fetch all property relations as well as the inverse relations and save them in a lookup table. In this table, every property gets indexed twice, by the complete property URI (e.g., http://www.w3.org/1999/02/22-rdf-syntax-ns#type, short *rdf:type*) and by the name extracted from the URI (e.g., type). In the lookup table, every property for a term points to a list of values, either URIs or strings for textual/numeric values. By indexing the property names in addition to the unique identifiers we are able to retrieve a list of all types independently from the used ontology. This is important since some resource have assigned same properties from different ontologies (e.g., http://dbpedia.org/property/type in addition to *rdf:type* from the example above). By indexing these using their name (i.e., type), we unify them to the same property.

After mapping the found terms to their corresponding resources, we follow four strategies to extend and disambiguate their semantic meanings. The first strategy is to follow DBpedia redirections if present. The second strategy is to explore disambiguation resources for ambiguous terms that do not redirect to a disambiguation resource. The remaining two strategies disambiguate ambiguous terms.

Redirection Strategy Redirections are realized on DBpedia by a redirection property (i.e., http://DBpedia.org/ontology/wikiPageRedirects, short *dbpedia-*

owl:wikiPageRedirects). This is assigned to the resource that is supposed to redirect to another. We leverage this by fetching the resource the property points to (s. Figure 2). Redirects are followed recursively. During this procedure we fetch and index all new found properties and aggregate them. The rationale behind this is that, in case there is a redirection pointing to another resource, this is supposed to give a better entity description. Therefore, it represents the same entity and its properties belong to the entity as well.

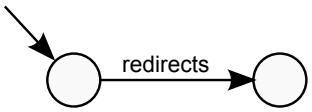

Fig. 2. Follow redirections.

Ambiguation Strategy If a resource has an ambiguous meaning, it mostly points to a disambiguation resource using the *dbpedia-owl:wikiPageRedirects* property. In this case, we apply the first redirection strategy. However, there are ambiguous resources that do not redirect. For instance, the resource *Apple* (i.e., http://dbpedia.org/resource/Apple) represents the fruit, even though *Apple* is an ambiguous term. The disambiguation resource for *Apple* is *Apple_(disambiguation)*, but there is no redirection between these two. Therefore, *Apple_(disambiguation)* uses the *dbpedia-owl:wikiPageDisambiguates* property to point to its non-ambiguous resources, like *Apple* (the fruit).

To discover ambiguous terms, we analyze all inverse disambiguation relations of a resources and follow backwards if there is a relation originating in a resource with the exact same name as the original term, but with the suffix "(disambiguation)" appended (s. Figure 3). Unlike for the redirection, we do not collect all properties. Instead, we only keep the properties of the disambiguation resource, because the original term might not the one we are interested in (e.g., Apple fruit).

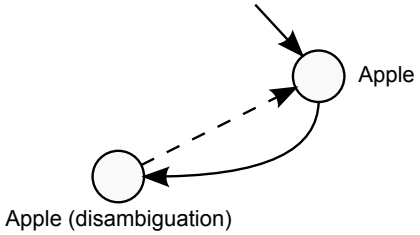

Fig. 3. Redirect to disambiguation resource, in case it exist with the same name.

Direct Disambiguation Strategy If a disambiguation resource has been identified we need to decide for one of the suggested resources as a representation

for the entity name under consideration. In case one of the candidates proposed by DBpedia is also a direct co-reference of the term we take this one as shown in the example in Figure 4. The term we try to resolve in the example is *Pope Benedict*. The corresponding disambiguation resource proposes all popes with name Benedict up to XVI. Since *Pope Benedict XVI* is a direct co-reference in the co-reference class of *Pope Benedict* derived by NEER we follow this resource as described for our redirection strategy and aggregate its properties with the properties that have been fetched so far.

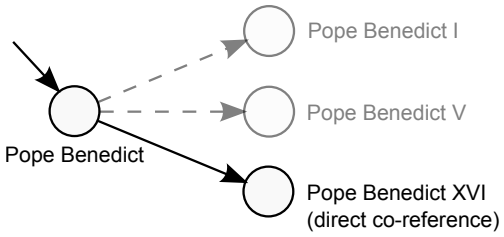

Fig. 4. Disambiguate entity by following resource with the same name as a direct co-reference.

Indirect Disambiguation Strategy For the disambiguation of terms for which we do not have a direct co-reference as disambiguation candidate, we make use of indirect co-references derived by NEER for that term. Using these indirect co-references ind_1, ind_2, ... we form a term vector. Additionally, a term vector is formed for each disambiguation candidate based on the property values of the corresponding resource. These vectors consist of the frequencies of every indirect co-reference occurring in the property values: ($freq(ind_1)$, $freq(ind2)$, ...). Similar to [9] we calculate the cosine similarity between two vectors to measure which resource fits the term in our context best. That resource will be selected as the semantic representation for the ambiguous term. This procedure is illustrated in Figure 5.

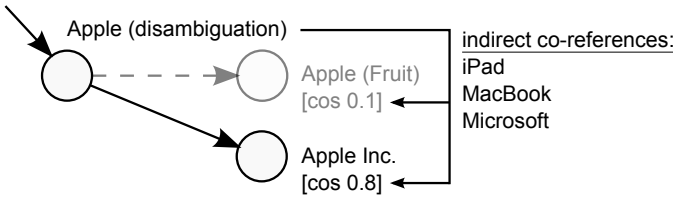

Fig. 5. Disambiguate entity by following resource that is most similar to the indirect co-references.

4.2 Filtering

After the disambiguation and aggregation of properties from DBpedia we proceed with the filtering. We consider the properties *type* and *subject* despite their ontology or namespace (i.e., URI), as described in Section 4.1. We treat DBpedia under the open world assumption. That means the fact a resource does not have a certain property does not mean that the corresponding entity does not have the property either. The resource has perhaps just not been annotated with the property. However, if a resource has a certain property, we consider this to be complete. For instance, if a resource is annotated with types, we assume these are all types it has and there is no type missing.

Similarity Filtering The first filter we apply to the result set of co-references derived by NEER compares the similarity of the query term with its co-reference candidates based on the their types and subjects from DBpedia. We compare the set of types and subjects of the query term with sets of each co-reference, direct and indirect. This only works if the query term or its corresponding DBpedia resource respectively has been annotated with types or subjects at all. Otherwise, this filtering method is not applicable. The same holds for the co-references. It would be wrong to consider two term referring to different entities just because one of them has not been annotated with types or subjects while the other one has (open world assumption, s. above). In this case we treat them as correct co-references for the query term and keep them in our result set. In case the query term's resource and the resource of the co-reference under consideration have both been annotated with types or subjects we require them to have at least one type and/or subject in common. To check this requirement, we compute the intersections of their type sets as well as their subject sets. In case one of the set intersections is empty, we consider the two terms as different and filter out those co-references. Otherwise, we keep them in our result set and pass them to the type filter.

Type Filtering Other than the similarity filter, the type filter considers hierarchies of types in addition to the types a resource is directly annotated with. For instance, both *Pope Benedict XVI* and *Barack Obama* are *persons* (resources of type *dbpedia-owl:Person*). Therefore, the similarity filter would not have filtered out one of them as co-reference of the other. However, *Pope Benedict XVI* is of type *dbpedia-owl:Cleric* while *Barack Obama* is annotated with *dbpedia-owl:OfficeHolder*. Both types are sub-types of *Person*. Thus, the two terms refer to different kinds of persons on DBpedia and do most likely not correspond to the same entity.

To achieve this filtering we need to analyze the sub-class relations of all types assigned to a resource. Each type on DBpedia is represented as an URI that points to a resource of that type. To obtain the hierarchy of a type, we leverage the rdfs:subClassOf property (i.e., http://www.w3.org/2000/01/rdf-schema #subClassOf) of the resource. This points to its super-type and allows us to

perform this procedure recursively until there is no rdfs:subClassOf property available or no resource corresponding to a type's URI exists.

After we have fetched the hierarchies for all types top-down, starting by a type and fetching the super-types, we analyze them bottom-up. For all types that the query term and its potential co-reference have in common we compare all of their sub-types. For instance, for *Pope Benedict XVI* and *Barack Obama*, having type *Person* in common, we compare their sub-types of type *Person*: *Cleric* and *OfficeHolder*. As these are different we consider the two terms not to be the same or referring to the same entity respectively and do not keep the co-reference candidate in our result set. In case they are equivalent we proceed with the next sub-type. This will be done recursively as long as both terms have sub-types in common or until they are not annotated with further sub-types.

The open world assumption holds again if the terms under consideration have a type in common, only one of them has been annotated with a further sub-type though. As we cannot tell whether the sub-type is missing on the other DBpedia resource or the entity is actually not an instance of that type, we do not filter out that co-reference and keep the it in the final result set.

5 Experiments

For our experiments we created a Ruby implementation of NEER and added the introduced extensions for BlogNEER. For the entity extraction we used a Ruby implementation of the Lingua English Tagger by Coburn [10].

For the evaluation we created two datasets. The techblog dataset consists of five popular tech blogs covering five years from 2008 to 2013, fetched from Google Reader: TechCrunch, Gizmodo, SlashGear, Ubergizmo and GottaBeMobile. For the general blog dataset we fetched the top 100 blogs from nine different categories (sports, autos, science, business, politics, entertainment, technology, living, green), based on the ranking of Technorati [11], also from Google Reader. In addition, we used the Blogs08 TREC dataset, described by Ounis et al. [12].

Prior to creating contexts with NEER we applied a frequency filtering to avoid feeding NEER with too many noisy terms. Those terms often do not have a corresponding DBpedia resource and thus they cannot be filtered out by using the semantic filter with similarity or type filtering and remain as noise in the end result. Applying the frequency filter lead to much better results by keeping the contexts smaller.

To demonstrate the results of BlogNEER we use the term "Kinect" as an example. "Kinect" is the name of a gaming accessory from Microsoft. During its development it was known under the name "Project Natal" until the announcement of Kinect in June 2010. We used that month as the change period and applied the frequency as well as the semantic filter to our results. The following set of terms is a result containing both, direct and indirect co-reference without semantic filtering:

Apple, Engadget, GameStop, Project Natal, Kotaku, Nintendo, Redmond, USA Today, Microsoft Kinect, Microsoft

After applying the semantic filter (s. Section 4.2) we get an improved result set:

Project Natal, Microsoft Kinect

Due to the preliminary stage of our research, we are unable to compare precision and recall. However, in recent experiments we already reached a recall similar to the recall we achieved with our baseline, NEER on the New York Times dataset [1]. Even though the precision was still lower due to noise, the semantic filter helped with filtering out false positives as shown in the example above. In case the noise consists of misspelled, informal or rarely used terms, which are not known in DBpedia, we are not able to filter them out using semantic filtering. In future work we will tackle this problem by using advanced frequency filtering methods.

Our results also indicated how differently NEER behaves on blog data. Although both datasets consist of blogs, we observed much less noise with the general blog dataset specialized in certain categories than in arbitrary, unspecialized and partly private blogs from the TREC blogs. Our experiments, even if not yet final, already indicate the impact of frequency and semantic filtering. We were already able to reduce the noise and achieve a constantly high recall.

6 Conclusions and Future Work

For applying the NEER method on the Blogosphere we proposed BlogNEER, an extension to the original approach. BlogNEER uses a novel a-posteriori filtering method incorporating the Semantic Web. The semantic filter applied to the results of NEER increased the precision by making use of data from DBpedia. Using properties like types and subjects (i.e., categories) we are able to keep apart terms that refer to different entities. Therefore, we can filter out names that refer to another entity than a query term and thus, can not be an new name.

We presented a first evaluation and a simple example showed the potential of BlogNEER. However, to further reduce the noise we will need to filter terms a-priori before they are processed by BlogNEER. We are also planning on incorporating additional web resources in BlogNEER as well as making use of other web specific feature, for instance tags.

References

[1] Nina Tahmasebi, Gerhard Gossen, Nattiya Kanhabua, Helge Holzmann, and Thomas Risse. Neer: An unsupervised method for named entity evo-

lution recognition. In *Proceedings of the 24th International Conference on Computational Linguistics (Coling 2012)*, Mumbai, India, December 2012.

[2] Y.H. Segerstad. *Use and adaptation of written language to the conditions of computer-mediated communication*. PhD thesis, G?teborg University, 2002.

[3] Nina Tahmasebi, Gerhard Gossen, and Thomas Risse. Which words do you remember? temporal properties of language use in digital archives. In *Theory and Practice of Digital Libraries*, volume 7489, pages 32–37. Springer, 2012.

[4] Klaus Berberich, Srikanta J. Bedathur, Mauro Sozio, and Gerhard Weikum. Bridging the terminology gap in web archive search. In *WebDB*, 2009.

[5] Amal Chaminda Kaluarachchi, Aparna S. Varde, Srikanta J. Bedathur, Gerhard Weikum, Jing Peng, and Anna Feldman. Incorporating terminology evolution for query translation in text retrieval with association rules. In *CIKM*, pages 1789–1792. ACM, 2010.

[6] Nattiya Kanhabua and Kjetil Nørvåg. Exploiting time-based synonyms in searching document archives. In *Proceedings of the 10th annual joint conference on Digital libraries*, JCDL '10, pages 79–88, New York, NY, USA, 2010. ACM.

[7] Arturas Mazeika, Tomasz Tylenda, and Gerhard Weikum. Entity timelines: visual analytics and named entity evolution. In *CIKM*, pages 2585–2588, 2011. ISBN 978-1-4503-0717-8. doi: 10.1145/2063576.2064026. URL `http://doi.acm.org/10.1145/2063576.2064026`.

[8] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *J. Web Sem.*, 7(3):154–165, 2009.

[9] A. Garcá-Silva, M. Szomszor, H. Alani, and O. Corcho. Preliminary results in tag disambiguation using dbpedia. In *Knowledge Capture (K-Cap 2009)-Workshop on Collective Knowledge Capturing and Representation-CKCaR*, 2009.

[10] Aaron Coburn. Lingua::EN::Tagger - search.cpan.org. (accessed October 27, 2009), 2008. URL `http://search.cpan.org/perldoc?Lingua::EN::Tagger`.

[11] Technorati Inc. accessed June 05, 2013, 2013. URL `http://www.technorati.com`.

[12] Iadh Ounis, Craig Macdonald, and Ian Soboroff. Overview of the trec-2008 blog track. In *In Proceedings of TREC-2008*, 2009.