Detecting and Exploiting Stability in Evolving Heterogeneous Information Spaces

George Papadakis[$,♯], George Giannakopoulos[†],
Claudia Niederée[♯], Themis Palpanas[‡], and Wolfgang Nejdl[♯]

[$] National Technical University of Athens, Greece gpapadis@mail.ntua.gr
[†] SKEL - NCSR Demokritos, Greece ggianna@iit.demokritos.gr
[♯] L3S Research Center, Germany {surname}@L3S.de
[‡] University of Trento, Italy themis@disi.unitn.eu

ABSTRACT

Individuals contribute content on the Web at an unprecedented rate, accumulating immense quantities of (semi-) structured data. Wisdom of the Crowds theory advocates that such information (or parts of it) is constantly overwritten, updated, or even deleted by other users, with the goal of rendering it more accurate, or up-to-date. This is particularly true for the collaboratively edited, semi-structured data of entity repositories, whose entity profiles are consistently kept fresh. Therefore, their core information that remain stable with the passage of time, despite being reviewed by numerous users, are particularly useful for the description of an entity.

Based on the above hypothesis, we introduce a classification scheme that predicts, on the basis of statistical and content patterns, whether an attribute (i.e., name-value pair) is going to be modified in the future. We apply our scheme on a large, real-world, versioned dataset and verify its effectiveness. Our thorough experimental study also suggests that reducing entity profiles to their stable parts conveys significant benefits to two common tasks in computer science: information retrieval and information integration.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering

General Terms

Algorithms, Experimentation, Performance

Keywords

Entity Evolution, Stability Detection, N-gram graphs

1. INTRODUCTION

Over the last few years, Web 2.0 applications have revolutionized the Web and the way people interact with it. User-generated and community-mediated knowledge collections, as well as entity repositories, such as Wikipedia[1] and Freebase[2], are among the most popular and useful applications of Web 2.0. Their reactivity, ease of use and self-organizing nature, enable and encourage users to contribute new and to modify existing content. As a result, this content comprises nowadays a constantly increasing portion of the information available on the Web.

User-generated content possesses three important properties: (a) very high level of heterogeneity, which results from its distributed, self-organized creation, (b) high growth rates in terms of volume, and (c) fast evolution, which implies reactivity as well as volatility. This fickleness, in particular, constitutes a salient aspect of user-generated data: the community of users tends to update and extend the information shared on the Web, by continuously adding new content and modifying the existing one. However, not all this information are of the same importance; some parts reflect reality, but others comprise wrong, obsolete or slightly irrelevant content. As more and more users of Web 2.0 systems examine these profiles, they gradually identify the false pieces of information and remove or replace them with more relevant ones (*Wisdom of the Crowds* [25]).

In this context, user-generated content can be roughly split into two parts: *stable* content that remains unchanged over a certain period of time, and *unstable* content, that is eventually deleted or corrected by the community. It would be beneficial to identify the former in advance (i.e., predicting stability), because it is more useful for the description of an entity and, thus, of higher value for tasks leveraging on user-generated content.

In this paper, we present an approach for predicting the stability of entity profiles (i.e., sets of name-value pairs describing entity properties) in highly heterogeneous user-ge nerated content collections, based on statistical and content patterns. In doing so, we can reduce the volume of these collections by ignoring pieces of data that are volatile. This study demonstrates that the proposed approach allows us to execute complex operations on these data collections (e.g., entity resolution) with increased efficiency, without sacrificing effectiveness.

The prediction of future behavior based on past and present dynamics (as well as usage patterns) is a common practice in various areas, such as operating system functionality, caching for Web search and recommendation systems. More

JCDL'11, June 13–17, 2011, Ottawa, Ontario, Canada.

[1]See `http://www.wikipedia.org`.
[2]See `http://www.freebase.com`.

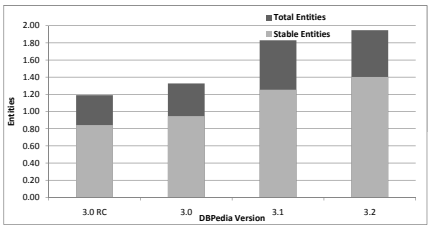

Figure 1: Evolution of DBPedia infobox entities (in millions).

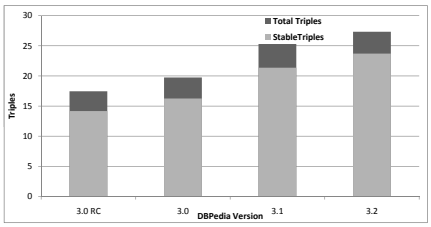

Figure 2: Evolution of DBPedia infobox triples (in millions).

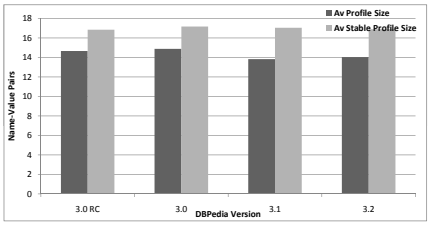

Figure 3: Evolution of DBPedia infobox entities's profile size.

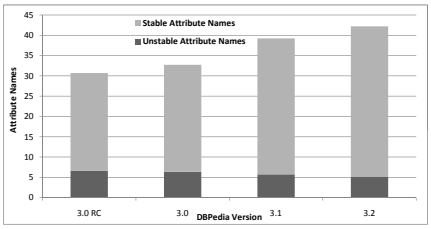

Figure 4: Distribution of attribute names into stable and unstable ones (in millions).

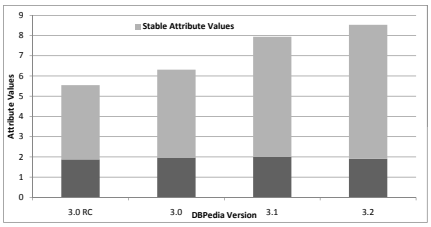

Figure 5: Distribution of attribute values into stable and unstable ones (in millions).

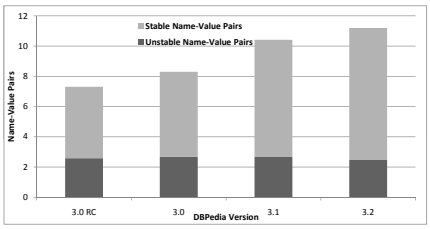

Figure 6: Distribution of name-value pairs into stable and unstable ones (in millions).

related to our work are approaches that look into content dynamics and content volatility to decide for its processing. For instance, in [22] the authors introduce a method for deciding whether data found on the Web, over a certain period of time, pertain to a particular object or not. Additionally, relevant works can be found in Web crawling [21], improving term weighting for retrieval indexing [2] and for ranking in Web search [9]. None of these works, though, focuses in the dynamics of entity profiles. In this work, we introduce methods for reducing an entity description to its stable part, and demonstrate the benefits that it conveys to common tasks: information retrieval and integration.

The challenge of identifying the stable content in advance is amplified by the high level of heterogeneity of user-genera ted content. Indeed, there is little limitation in the way users can contribute content: every individual is free to add information in her own proprietary format, bringing about unprecedented levels of heterogeneity. In the case of user-generated entity repositories, this situation affects not only the schemata describing the same entity types, but also the different descriptions (profiles) of the same entity. Google Base[3], for instance, encompasses 100,000 distinct schemata corresponding to 10,000 entity types [19]. Therefore, effectively identifying stable content becomes very challenging.

User-generated content is also prone to accumulate to immense volumes, even over relatively short periods of time [19]: Wikipedia's user base is rapidly increasing, thus fueling an exponential growth in its content [3]; Blogosphere is steadily expanding by 175,00 new blogs and 1.6 million new blog posts per day [5, 24]; micro-blogging emerged only recently to become one of the most popular Web applications, with Twitter[4] being its most prominent instantiation [18]. Last but not least, video-sharing web sites, such as Youtube[5], amass content at such a fast pace that its manipulation and maintenance poses significant challenges [4]. This makes applications that want to leverage on this content very challenging. The early identification of stable content within such collections effectively reduces the amount of the useful data that needs to be considered, enabling the development of more efficient applications on top of user generated content.

To illustrate the phenomenon of stable and unstable content over a certain period of time, we briefly examine the evolution of Wikipedia entities, before sketching our approach.

Case Study: Content Evolution in DBPedia. DBPedia[6] has made available 6 versions of the English Wikipedia infoboxes, spanning 2 years, from October 2007 (version $3.0RC$) to October 2009 (version 3.4). The figures discussed below were created by comparing the content of each DBPedia version with all its subsequent versions. We consider as *stable* those items that remained unchanged for all subsequent versions. Consequently, we cannot identify the stable items of the last version, namely 3.4. We also excluded version 3.3 due to lack of sufficient history, as explained in Section 4.1. Thus, the last two versions of DBPedia were only considered in the estimation of the evolution of the earlier versions.

Figures 1 and 2 are indicative of the immense growth rate of Wikipedia content. In the two years covered by our dataset, $1,400$ entities and $27,500$ triples were added every day (on average) to its repository. Among them, $1,300$ entities and $25,000$ triples remained unmodified in the following versions.

From these numbers, we can also notice the volatility of the Wikipedia content. A considerable portion of the new entities appearing in each DBPedia version are not stable; their entire profile may have been modified, deleted as noise, or merged with some other entity (duplicate entries). In fact, the portion of stable entities ranges from 70% for version $3.0RC$ to 73% for 3.2. The same applies to the triples of DBPedia infoboxes (Figure 2): 81% of those contained

[3]See `www.google.com/base`.
[4]See `http://twitter.com`.
[5]See `http://www.youtube.com`.
[6]See `http://dbpedia.org`.

in version 3.0*RC* are stable, raising gradually to 87% for 3.2. Similarly, the portion of unstable attribute names (Figure 4), attribute values (Figure 5) as well as name-value pairs (Figure 6) declines in the later versions of DBPedia. Thus, *the longer is the period of time that user-generated content is exposed to the eyes of the community, the larger is the extend of its revision.* In this context, we can consider those pieces of information that remain stable across all subsequent versions to be more valuable and more descriptive of an entity.

Figure 3 also advocates the argument about higher usefulness of stable information items. The average profile size of stable entities (i.e., number of triples describing them) is consistently higher than that of all entities. Moreover, the former increases slightly, but steadily with the passage of time, whereas the latter fluctuates extensively. This is a strong evidence that unstable entities tend to have small profiles, thus lowering the overall average profile size; on the other hand, the description of stable ones is gradually enriched with more relevant information.

The high heterogeneity of Wikipedia entities is demonstrated in Figure 4 along with its evolution. From $30,000$ distinct attribute names in the first version, it raises to $42,000$ in version 3.2.

On the whole, Figures 1 to 6 are indicative of Wikipedia's content volatility. They strongly support the argument that only a portion of it remains stable after it has been added to the repository. Stable content, though, is more useful than unstable one, comprising the cornerstone of entity profiles. Therefore, it should suffice to exclusively consider stable content when leveraging user-generated data.

Approach. In this paper, we focus on the evolution of collaboratively edited, heterogeneous information spaces, like Wikipedia. We advocate the idea that information contained in the profile of an entity at time t_1, but is missing at time $t_2 > t_1$, tends to be less useful (it can be even incorrect, irrelevant or simply updated). Information that remains stable should be correct, thus, comprising the most valuable part of an entity profile. Therefore, reducing user-generated information spaces to their stable parts, should have a positive impact on the efficiency of applications operating on them, at a negligible cost on effectiveness.

In more detail, we present a novel approach for extracting evolution patterns from heterogeneous information spaces, on the basis of statistical and content features. Our method works on the level of an attribute-value pair, predicting whether it will be modified in the future or not. We apply our method on a voluminous, real-world, versioned data set and demonstrate the benefits of employing only their stable part in two common tasks: information retrieval and information integration.

The contributions of our paper can be summarized as follows:

- We formalize the problem of "entity evolution", and develop a novel approach to deal with it, using established machine learning algorithms in combination with natural language processing ones.
- We demonstrate how to exploit the concept of stable entity profiles in practice, by applying it to two common tasks: keyword queries over structured data and entity resolution.
- Finally, we evaluate our approach, as well as its application on the above tasks through an extensive experimental study on a large, versioned, real-world data set.

The rest of the paper is structured as follows: Section 2 formalizes the problem and the notions we are handling, whereas Section 3 introduces our approach to entity evolution prediction. We present the thorough experimental study of our approach in Section 4, and discuss the results of applying it to our use cases in Section 5. Finally, Section 6 discusses relevant works, and Section 7 concludes the paper and presents ideas for future work.

2. PROBLEM FORMULATION

The prediction of the stable parts of an information space in the context of fast evolution is an issue in most user-created data collections. However, we focus our work on *entity repositories*, i.e., information spaces that comprise statements about real-world entities, such as persons, places, and products. The reason is twofold: the availability of large data sets and the growing popularity of the entity-centric approach.

We distinguish two types of evolution in the context of entity repositories: the *macroscopic* and the *microscopic* one. The former pertains to modifications in an information space as a whole (e.g., new entities added in Wikipedia), while the latter refers to changes in individual entities (e.g., some values are updated, and/or some others are deleted from the entity description).

Assuming an infinite set of attribute names $\mathcal{N}$, along with an infinite set of possible values $\mathcal{V}$, and an infinite set of identifiers $\mathcal{ID}$, an information space I can be formally defined as follows:

Definition 1. *An* ***information space*** $\mathbf{I}$ *is a tuple* $\langle N_I, V_I, ID_I, P_I \rangle$*, where* $N_I \subseteq \mathcal{N}$ *is the set of attribute names appearing in it,* $V_I \subseteq \mathcal{V}$ *is the set of values used in it,* $ID_I \subseteq \mathcal{ID}$ *is the set of local identifiers contained in it, and* $P_I \subseteq ID_I \times \mathcal{P}(A_I \times V_I)$ *is the set of entity profiles that it comprises. We call* $\mathbf{N_I}$ ***schema space*** *and* $\mathbf{V_I}$ ***value space of*** $\mathbf{I}$.

In the following, we focus on evolution on the microscopic (entity) level. Therefore, at the core of our approach lies the notion of an entity profile over time.

Definition 2. *An* ***entity profile*** $\mathbf{p_i}$ ***of an information space*** $\mathbf{I}$ ***at time*** $\mathbf{t_i}$ *is a tuple* $\langle id, A_p \rangle$ *that corresponds to a real-world entity, where* A_p *is a set of attributes* a_i*, and* $id \in \mathcal{ID}$ *is a local identifier for the profile. Each* ***attribute*** $a_i \in A_p$ *is a tuple* $\langle n_i, v_i \rangle$*, consisting of an optional* ***attribute name*** $n_i \in \mathcal{N}$ *and an* ***attribute value*** $v_i \in \mathcal{V} \cup \mathcal{ID}$*. We call the set* $n_i \subseteq \mathcal{N}$ ***schema space*** *and the set* $v_i \subseteq (\mathcal{V} \cup \mathcal{ID})$ ***value space of*** $\mathbf{p_i}$. ∎

As real-world entities evolve and change over time, their corresponding profiles should be modified appropriately. Thus, a single entity is described over time by a sequence of profiles, p_i, at distinct points in time t_i. This is, for example, the case with Wikipedia entities, where numerous users collaboratively update and improve the entity profiles according to real-life events. However, some core properties of an entity - such as the name of a person or her id number - tend to remain unmodified (possibly after an initial phase,

where errors are corrected). The set of these name-value pairs comprises the stable profile of an entity, p^s. More formally:

Definition 3. *A **stable entity profile** $\mathbf{p^s}$ is an entity profile that will retain its tuple $\langle id, A_p \rangle$ unmodified in all points of time t' following a point in time t: $t > t'$.* ∎

Of course, stability in infinity can be neither observed (verified) nor expected in most cases. In this work, we focus on predicting stability for relevant observation time frames and verify our prediction based on available versions of information spaces.

Wisdom of the Crowds Theory advocates the principle that noisy information are gradually removed from collaborative environments. As a result, their content tends to reflect the shared notion of reality, which is typically the true state of affairs, as long as users are allowed to contribute information without any restriction. Thus, the content of information spaces is likely to converge to a stable representation, that will remain unchanged in the future. In the case of entity profiles, this can be interpreted as the convergence of individual profiles towards their stable representation. To describe this phenomenon formally, we introduce the notion of *profile distance*:

Definition 4. ***Profile Distance** $\mathbf{d(p_i, p_k)}$ between two profiles of the same real-world entity at times t_k, t_i ($t_k > t_i$) is defined as the ratio of the different name-value pairs over the common ones.*

$$d(p_i, p_k) = \frac{A_{p_i} \ominus A_{p_k}}{A_{p_i} \cap A_{p_k}} = \frac{(A_{p_i} \bigcup A_{p_k}) \setminus (A_{p_i} \cap A_{p_k})}{A_{p_i} \cap A_{p_k}}$$

The higher the value of $d(p_i, p_k)$, the fewer the content that these two profiles have in common. ∎

Based on the notion of profile distance, we can define evolving entity descriptions in an information space.

Definition 5. *An **evolving entity description** $\mathbf{ed}$ is a sequence of entity profiles p_1, p_2, ..., p_k with $t_i > t_{i-1}$ for the same entity that converges towards the stable profile, p^s, i.e. from some point in time the distance to the stable profile reduces over time: $\forall ed \exists K_{ed} \geq 0 : (\forall t_i, t_k \geq K_{ed} : t_i \leq t_k \Rightarrow d(p_i, p^s) \geq d(p_k, p^s))$* ∎

On the basis of the above definitions, the problem we want to solve can be formalized as follows:

Problem 1. *Given two snapshots of an information space I at times t_1 and t_2 ($t_1 < t_2$), identify the part p_2^s of each entity profile $p_2 \in P_I$ that is highly likely to remain stable in the future points in time $t > t_2$.* ∎

In other words, the goal is to develop an effective method that predicts for each entity profile p_i its stable part p_i^s with the following properties:

1. p_i^s describes the same real-world entity as p_i (i.e., they share the same *id*).
2. The schema and the value space of p_i^s is a subset of the schema and value space of p_i, respectively.
3. $d(p_i^s, p^s) < d(p_i, p^s)$.
4. The name-value pairs space of p_i^s will remain unmodified in the future ($p_i^s \subseteq p_k^s \ \forall t_i < t_k$).

Apparently, the challenge here is that the stable profile is not known in advance. Instead, it has to be predicted from the evolution patterns identified over time in the given information space.

3. APPROACH

To solve the aforementioned stability prediction problem, we cast it as a binary classification problem: each information item — i.e., an attribute name or an attribute value — will be classified as either stable or unstable.

The most crucial aspect of our approach is the set of features that will form the basis for this classification. In our approach we use a combination of: **(i)** *statistical features*, which are numerical values representing measurable dimensions of the data at hand (e.g., the length of an attribute value), and **(ii)** *content-based features*, which encapsulate patterns in the content (i.e., string representation) of entity profiles.

For ensuring the applicability of the features in a wider range of settings, we restrict our approach to features that can be extracted from a single snapshot of the given information space I (i.e., features that do not involve multiple versions of I, like the longevity - active versions - of an item). There is no restriction, though, in the number of entities, to which a feature can pertain: the feature can be extracted either from one or from many profiles. The actual features we used are discussed later in this section.

Based on the selected features, we train a state-of-the art classification algorithm using a large versioned information space. We use the earliest version of I for training the classifiers. In fact, we train one classifier for attribute names and another one for attribute values, independently from each other. The reason for not considering a joint classifier for the schema and value space is twofold: first, this approach suffers from lower efficiency, due to the larger number of features and instances (compare Figures 4 and 5 with Figure 6). Second, a classifier for name-value pairs involves higher levels of noise, since the name of an attribute can be the value of another one and vice versa. In case they differ in stability, content features will be seriously distorted, affecting the judgment of the classifier [12]. We, thus, distinguish: **(i)** *Schema Space stability*, which refers exclusively to attribute names, and **(ii)** *Value Space stability*, which pertains solely to attribute values.

To classify $\mathbf{P_I}$ (i.e., the set of entities contained in I), we iterate over its elements examining each profile p separately. The attribute name, n_i, and attribute value, v_i, of each attribute $a_i \in A_p$ are tested against the corresponding classifiers. If at least one of them matches the instability patterns encapsulated in the respective classifier, the whole name-value pair is considered unstable. Otherwise, it is classified as stable. Thus, the set of attributes consisting of both a stable name and a stable value, compose the stable profile of the entity.

In the following, we analyze the features selected for each of the aforementioned levels of stability. Before elaborating on these features, though, we provide some background knowledge on the n-gram graphs, necessary for the discussion that follows.

N-Gram Graphs Background. At the core of our

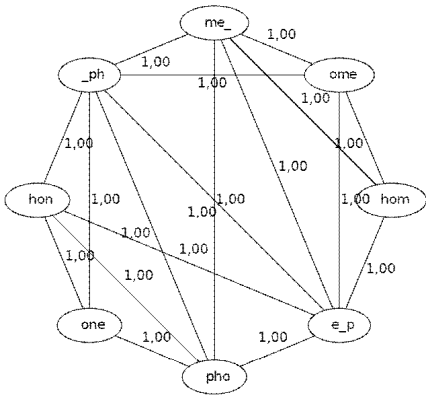

Figure 7: Sample tri-gram graph representing the string "home_phone". Every node of the graph tri-gram corresponds to a tri-gram of the string, and the edges connect tri-grams, whose distance is less than three letters, regardless of their relative position (i.e., they are undirected edges).

content-based features lies the *n-gram graphs* framework [11]. We chose this method over other string similarity techniques for a number of reasons: apart from representation of instances, it also provides a comprehensive representation of sets of instances (classes) under a unique perspective (the graph), which is the required functionality in our case. Further, this representation allows for subword matching and graded matching between strings, with bounded values of similarity that facilitates the functionality of the classifiers.

In essence, this framework allows to represent a text as a graph (see Figure 7 for an example), keeping more information than a bag-of-words and supporting expressive representation of sets of text using a single graph [12]. Assuming that each attribute name constitutes a small text T, we can combine all stable names of the training set into a common graph and all its unstable names into another graph. The same applies to attribute values. The corresponding graphs capture patterns common in the content of stable and unstable items, such as recurring and neighboring character sequences, special characters, and digits. Thus, they can be employed to measure the similarity of an information item (of the testing set) with the stable and the unstable class. The similarity is calculated between the corresponding graph representation G^i of the information item and a class representative graph G^j. The following three kinds of similarity between the n-gram graphs are used in the scope of this work:

Containment Similarity (CS), which expresses the proportion of edges of a graph G^i that are shared with a second graph G^j. Assuming that G is an n-gram graph, e is an n-gram graph edge and that for function $\mu(e, G)$ it stands that $\mu(e, G) = 1$, if and only if $e \in G$, and 0 otherwise, then:

$$\mathrm{CS}(G^i, G^j) = \frac{\sum_{e \in G^i} \mu(e, G^j)}{\min(|G^i|, |G^j|)}, \quad (1)$$

where $|G|$ denotes the size (i.e., the number of edges) of a graph G.

Size Similarity (SS), which denotes the ratio of sizes of two graphs:

$$\mathrm{SS}(G^i, G^j) = \frac{\min(|G^i|, |G^j|)}{\max(|G^i|, |G^j|)}. \quad (2)$$

Value Similarity (VS), which indicates how many of the edges contained in graph G^i are contained in graph G^j, considering also the weights of the matching edges. In this measure, each matching edge e having weight w_e^i in graph G^i contributes $\frac{\mathrm{VR}(e)}{\max(|G^i|, |G^j|)}$ to the sum, while not-matching edges do not contribute (i.e., for an edge $e \notin G^i$ we define $w_e^i = 0$). The *ValueRatio (VR)* scaling factor is defined as:

$$\mathrm{VR}(e) = \frac{\min(w_e^i, w_e^j)}{\max(w_e^i, w_e^j)}. \quad (3)$$

The equation indicates that the *ValueRatio* takes values in the interval $[0, 1]$, and that it is symmetric. Thus, the full equation for *VS* is:

$$\mathrm{VS}(G^i, G^j) = \frac{\sum_{e \in G^i} \mathrm{VR}(e)}{\max(|G^i|, |G^j|)}. \quad (4)$$

VS converges to 1 for graphs that share both the edges and their weights, with a value of VS = 1 indicating perfect match between the compared graphs.

A derived, important measure is the **Normalized Value Similarity (NVS)**, which is computed as:

$$\mathrm{NVS}(G^i, G^j) = \frac{\mathrm{VS}(G^i, G^j)}{\mathrm{SS}(G^i, G^j)}. \quad (5)$$

The NVS is a measure of similarity where the ratio of sizes of the two compared graphs does not play a role.

An n-gram graph is characterized by three parameters: (a) the minimum n-gram rank $L_{\min}$, (b) the maximum n-gram rank L_{MAX}, and (c) the maximum neighborhood distance D_{win} [11]. In the following, we exclusively consider tri-gram graphs, i.e., $L_{\min} = L_{\mathrm{MAX}} = 3$ with a neighborhood distance of $D_{\mathrm{win}} = 3$, which were experimentally shown to provide a good trade-off between effectiveness and efficiency for the English language [11].

The next two subsections discuss our prediction methods in more detail. The features we present were selected among several candidates based on a preliminary Information Gain analysis on the versioned data set of DBPedia Infoboxes. In the future, we plan to investigate whether this set of features exhibits similar performance in other information spaces, as well.

3.1 Schema Space Classification

To categorize attributes names into stable and unstable ones, we conducted a preliminary statistical analysis. The following *statistical features* were identified as most promising (the average values accompanying each feature for both categories pertain to DBPedia, version 3.0*RC*):

Entity Frequency (EF_A) represents the total number of entities that contain the given attribute name in their profile. The higher this number, the more stable the attribute name is expected to be; names existing in few entities are more likely to be replaced by other ones. Indeed, stable names are associated with 494.17 entities, on average, whereas unstable ones with just 14.74 entities.

Statement Frequency (SF_A) denotes the total number of name-value pairs that involve the given attribute name.

It differs from EF_A in that it considers the cases where the same attribute name can be assigned to multiple values within a single entity profile. Similar to EF_A, though, higher values are expected to correspond to more stable attribute names. Unstable names have a mean SF_A of 18.06, while for stable ones the average is 718.79.

Name Length (NL) expresses the number of characters comprising the given attribute name. The longer a name is, the more likely it is to be replaced by a synonymous, but more concise one (i.e., the less stable the attribute is). For this reason, stable names consist of 14.22, while unstable ones of 35.96 characters, on average.

Value Distinctiveness (VD) represents the total number of *distinct* values associated with the given attribute in I. The larger and more diverse the set of values of an attribute name, the more stable it is. For example, in case an excessive number of DBPedia entities share the same value for a specific attribute name, this name-value pair is apparently stemming from a template; thus, it is by no means representative of these entities and very likely to be removed in the future. Stable attribute names receive a mean of 300.29 distinct values, whereas unstable ones are associated with just 10.47.

As mentioned above, we use the n-gram graph framework for modeling and capturing the patterns in the string representation of the attribute names classes. Considering the similarity measures CS and NVS, which were introduced above, we additionally introduce the following four *content-based* features for schema classification: (a) *CS* with the *stable graph*, (b) *NVS* with the *stable graph*, (c) *CS* with the *unstable graph*, and (d) *NVS* with the *unstable graph*. They are calculated by comparing the n-gram graph representation of an attribute name of the testing set with the class-representative graphs (derived from the instances of the training set).

In total, we, rely on a limited number of features, thus allowing for high efficiency; both the training and the application of the classifiers are practical and scalable to large datasets.

3.2 Value Space Classification

In line with schema space classification, our preliminary statistical analysis indicated the following features as more suitable for predicting the stability of values in the given information space I (the mean values are again derived from DBPedia, version $3.0RC$):

Entity Frequency (EF_V) denotes the number of distinct entities that encompass the given value in their profiles. Frequently used values are expected to be more stable. In fact, stable values are associated with 3.57 entities, while unstable ones only with 1.25.

Statement Frequency (SF_V) expresses the number of name-value pairs that contain the given value. The higher this number, the more stable the value is. Stable values are found in 4.09 statements, on average, while unstable ones in 1.29.

Value Length (VL) corresponds to the number of characters in the string representation of the value. The longer a value, the more prone to change it is. Indeed, stable values comprise a mean of 27.89 characters, much less than the unstable ones (42.37 characters).

Attribute Distinctiveness (AD) is equal to the number of distinct attribute names associated with the given value. The higher this frequency, the more stable the value is (1.44 on average for stable values and 1.05 for unstable ones).

For creating adequate content-based features, we use again the n-gram graphs, in this case to model and capture the patterns in the string representation of the attribute value classes. Similar to schema space classification, four content-based features are created for value space classification: (a) CS with stable graph, (b) NVS with stable graph, (c) CS with unstable graph, and (d) NVS with unstable graph. In total, we consider again a limited set of eight features that makes our approach applicable to the magnitude of millions of distinct values.

4. CLASSIFICATION EVALUATION

4.1 Evaluation Methodology

In the context of our experimental study, we employed an established, versioned data set that is freely available on the Web: the *DBPedia Infobox Dataset*. Every version of it comprises all attribute-value pairs of all infoboxes and templates within all Wikipedia articles, recorded exactly as they appear in it (i.e., without any pre-processing) at specific points in time. In total, we used six snapshots, spanning a time period of two years (from October, 2007 to October, 2009).

To derive the *ground-truth* of each version (i.e., the labels of the attribute names and values, indicating them as stable or unstable), we compared each version with all its subsequent versions. An item (i.e., attribute name or attribute value) is considered *unstable* if it is missing from at least one of the following versions. Items that are persistent in all versions are marked as *stable*. Apparently, this procedure does not apply in the case of the last two snapshots (versions 3.3 and 3.4), since there are not enough versions following them chronologically[7]. Both of them are, thus, considered solely in the process of estimating the evolution of the other versions.

It should be noted at this point that the degree of evolution varies substantially between the separate versions of DBPedia. In fact, the earlier a version is, the higher its observable degree of evolution, and vice versa. The reason is that it is compared against more subsequent versions, which increases the likelihood of change in its content. As a result, the *class imbalance problem*, inherent in all versions, becomes more intense for later versions; although all versions contain a majority of stable items, the skew increases in proportion to the recency of the data (as shown in Figures 1 to 6).

To overcome this problem and its impact on measuring the actual performance of our classification scheme, we conducted our tests on samples with equal distribution of classes (i.e., 50% stable and 50% unstable items). To this end, we extracted from each version 10 random, possibly overlapping samples for each category. In the case of attribute names the sample size was set to $1,600$ instances, while for attribute values it was set to $600,000$ instances. The cardinality of the

[7]Theoretically, we can identify the stable items of DBPedia 3.3 by comparing it to DBPedia 3.4. In practice, though, we cannot derive safe conclusions by considering solely one subsequent version. In fact, the unstable items of the first four DBPedia versions were found to have an average "lifetime" (i.e., number of versions in which they are present) well over 2. This applies both to attribute names and attribute values.

	Accuracy (%)		
	version 3.0	**version 3.1**	**version 3.2**
Naive Bayes	54.51 ± 1.24	54.25 ± 1.28	53.78 ± 1.01
Bayes Net	79.90 ± 1.44	78.93 ± 1.29	78.74 ± 1.40
NB Tree	84.16 ± 3.32	79.75 ± 2.66	78.87 ± 2.56
J48	82.54 ± 2.37	76.85 ± 2.44	75.78 ± 2.64
JRip	82.84 ± 1.65	77.61 ± 1.24	77.07 ± 1.12
Random Forest	82.32 ± 2.04	75.58 ± 2.83	74.43 ± 3.19
SVM	81.59 ± 0.92	68.23 ± 1.14	65.95 ± 1.11

Table 1: Comparison of the performance of seven classification algorithms on samples of the schema space (i.e., attribute names) of three versions of the DBPedia Infobox Dataset.

	Accuracy (%)		
	version 3.0	**version 3.1**	**version 3.2**
Naive Bayes	52.80 ± 0.37	52.63 ± 0.32	52.48 ± 0.29
Bayes Net	75.25 ± 0.08	73.88 ± 0.06	73.61 ± 0.05
NB Tree	81.39 ± 1.47	79.07 ± 0.99	78.63 ± 0.95
J48	82.02 ± 0.32	79.30 ± 0.28	78.84 ± 0.26
JRip	80.70 ± 2.67	78.10 ± 1.87	77.80 ± 1.39
Random Forest	77.89 ± 0.24	75.91 ± 0.15	75.59 ± 0.14
SVM	80.25 ± 0.84	76.35 ± 0.77	75.99 ± 0.74

Table 2: Comparison of the performance of seven classification algorithms on samples of the value space (i.e., attribute values) of three versions of the DBPedia Infobox Dataset.

samples was chosen on the basis of the size of the smaller class (unstable items) across all versions: larger cardinalities would result in samples of unstable items that share the majority of their elements, thus distorting the experimental outcomes. Smaller cardinalities, on the other hand, cannot guarantee representative results (i.e., they involve a high variation in the performance of the samples).

The *training set* for the classification algorithms encompassed the 10 samples of the first snapshot, namely DBPedia 3.0 RC. The resulting 10 classifiers were evaluated against the 10 random samples of the remaining versions (i.e., the *testing sets* were the samples of the versions 3.0 to 3.2). In total, we conducted 100 tests per version and classifier and considered the average accuracy as the metric for estimating the performance of each algorithm. More formally, accuracy α is defined as follows: $\alpha = \frac{TP}{TP+FP}$, where TP (i.e., true positive) stands for the items that are marked as stable in the ground-truth and are identified as such by the classifier, and FP (i.e., false positive) denotes the unstable items that were falsely identified as stable by the classifier. The outcomes of attribute names and values classification are presented in the following subsections, 4.2 to 4.4.

Regarding the classification algorithms we employed, we tested several of them, in order to have a better understanding of the problem we are tackling. In more detail, we considered: (i) Naive Bayes, (ii) Bayes Net, (iii) the optimized version of the propositional rule learner RIPPER, (iv) SVM, and three tree-based classifiers, namely (v) NB Tree, (vi) J48, and (vii) Random Forests (for a comprehensive overview of these methods, see [28]). Naive Bayes is the simplest of them, serving as an estimate of the difficulty of the problem: low performance advocates the need for more effective and elaborate (thus, less efficient) methods, whereas high performance suggests that simple methods should suffice. The rest are state-of-the-art methods, which are expected to exhibit comparable performances.

Note that we conducted our experiments employing three sets of features: pure statistical features, pure content-based features and the combination of them, as presented in Section 3. Due to lack of space and for the sake of readability, we exclusively analyze the results of their combination. Considering one set of features independently of the other leads to substantially lower performance across all versions (accuracy is lower by approximately 10% for both cases and for both tasks). This implies that statistical and content-based features constitute complementary sources of evidence, which are indispensable for the optimal classification scheme. These results are in accordance with previous related research [12].

Our experiments were fully implemented in Java 1.6 and performed on a server with Intel Xeon 3.0GHz. For the implementations of the classification algorithms, we used the open source library Weka[8], version 3.6, documented in [14]. The functionality of the n-gram graphs was provided by the open source library JInsect[9].

4.2 Schema Space Classification

The performance of the seven algorithms on the classification of attribute names is summarized in Table 1. The poor performance of Naive Bayes is easily noticeable, since it lies close to the performance of the random classifier ($\alpha = 50\%$) across all versions. This is indicative of the difficulty of the binary classification problem at hand. The rest of the algorithms achieve much higher effectiveness, with small variations between most of them.

It is interesting to notice that NBTree achieves the highest performance across all snapshots. However, its performance is coupled with one of the highest standard deviations in each case, thus suggesting that it significantly fluctuates around the mean value. JRip, on the other hand, has similarly high performance (especially in the case of DBPedia 3.0), while its standard deviation is among the lowest. Consequently, we consider it to be the optimal choice for attribute names classification.

Quite interesting is also the monotonic decrease in the performance of all classification algorithms for later versions: the more recent a versions is, the lower the performance of a classifier. In other words, the longer the lifetime of an attribute name, the more accurate the prediction. This can be explained by the fact that recent versions may contain actually unstable items that have been labeled as stable in the ground-truth, due to the lack of more recent data snapshots.

4.3 Value Space Classification

The outcomes of attribute value classification are depicted in Table 2. Similar to the schema space evolution, the performance of Naive Bayes exceeds that of the random classifier to a minor extent. The rest of the algorithms exhibit high effectiveness, with their average accuracy lying around 80%. There is a clear winner, though, that not only achieves the highest accuracy, but also has the (second) lowest standard deviation across all versions: J48. Last but not least, we can identify the pattern of monotonic decrease once again in the performance of all algorithms for later versions of DBPedia.

[8]See `http://www.cs.waikato.ac.nz/ml/weka`.
[9]See `http://sourceforge.net/projects/jinsect`.

	version 3.0	version 3.1	version 3.2
Total Accuracy	94.22%	89.55%	89.08%
Stable Class	80.54%	85.40%	87.95%
False Negatives	1.65%	4.46%	5.47%
False Positives	4.13%	5.99%	5.45%

Table 3: Performance of JRip on the schema space (i.e., attribute names) classification of three versions of the DBPedia Infoboxes Dataset.

	version 3.0	version 3.1	version 3.2
Total Accuracy	87.11%	86.04%	85.94%
Stable Class	68.91%	74.81%	77.63%
False Negatives	3.67%	6.22%	7.34%
False Positives	9.22%	7.74%	6.72%

Table 4: Performance of J48 on the value space (i.e., attribute values) classification of three versions of the DBPedia Infoboxes Dataset.

4.4 Classification on the whole dataset

Based on our experiments, the optimal classification scheme consists of JRip for attribute names classification and J48 for the attribute values one. To estimate their performance on all instances of each version, we trained them on the whole training set (i.e., all attribute names and all attribute values) and applied them on the entire testing sets. In each case, the *baseline* performance is equal to the percentage of the stable class[10].

Table 3 presents the performance of JRip on the tasks of schema space classification. We can see that it exceeds that of the baseline labeled as *Stable Class* for all versions. However, their difference drops from 14% in version 3.0 to 5% and 1.2% for versions 3.1 and 3.2, respectively. This reduction can be partially attributed to the increase in the skew of the testing set. Most crucial, though, is the increase of the portion of *False Negatives* FN[11] for later versions, while FP remain almost stable. This means, that the more recent the data are, the higher the number of stable names that are classified as unstable, probably due to the shortcoming of the ground-truth (as explained in subsection 4.1).

Table 4 analyses the performance of J48 on the task of value space classification. Similar to the schema space one, we can identify a strong pattern in the evolution of effectiveness across all versions: initially, J48 performs substantially better than the baseline, but for later versions, the skew and the percentage of FN increases.

On the whole, the classification of instances into stable and unstable has been shown to be feasible and effective, surpassing significantly the baseline performance. This allows us to go one step further and exploit the stability labeling to improve the performance of data-volume dependent tasks.

5. IMPACT STUDY

In this section, we demonstrate the benefits of employing exclusively the stable parts of entity profiles in two common tasks of computer science: keyword queries over structured data, and entity resolution among large-scale, heterogeneous information spaces. In both cases, we consider an *attribute-agnostic* approach, that disregards attribute names and relies exclusively on attribute values[12].

[10]In the context of skewed data, a biased classifier, that assigns each instance to the largest class of the training set, outperforms the random, binary classifier.

[11]False Negative are the stable items that are classified as unstable.

[12]The attribute-agnostic approaches we adopt do not turn schema space classification superfluous. The reason is that an unstable attribute name renders the whole attribute-value pair unstable, and, thus, unnecessary. Schema space classification is, thus, useful in discarding attribute-value pairs.

We compare our optimal classification algorithm (i.e., JRip classifier for attribute names coupled with J48 for attribute values, labeled *Classification Scheme* from now on) with the following methods: (i) the **Full Version**, which constitutes our baseline, representing the original performance of a task, when considering the entire data set, (ii) the **Stable Version**, which represents the performance of the ideal classification method[13], setting the upper limit for improvement over Full Version, and (iii) the **Random Classifier**, which exhibits the performance of a plain, biased classifier that assigns all instances to the largest class.

For each task, we consider two kinds of metrics: those measuring *efficiency* and those estimating *effectiveness*. They are specialized to each application, and explained in the following subsections. The only metric that is common to both use cases is the *Index Size* (**IS**); that is, the space occupied on the disk by the corresponding inverted indices[14]. The lower its value, the higher the efficiency of the method.

5.1 Keyword Queries over Structured Data

In the context of this use case, we employed the dataset used in [8], comprising 4,000 keyword queries extracted from a query log of the MSN search engine. Each query was matched to an entity of DBPedia, version 3.0. As mentioned above, only the attribute values of the entity profiles in version 3.0 were indexed. The searching method we employed was the default ranking mechanism of the Lucene library (i.e., TF/IDF [20]). To evaluate the performance of this task, we measure the following metrics (in addition to IS):

(i) *Success at 1* (**S@1**) and *at 10* (**S@10**). They represent the percentage of queries that received the correct response at the first and the tenth place of the returned ranking list, respectively. The higher their value, the more *effective* the method is.

(ii) *Average Ranking Position* (**ARP**). It stands for the place a sought entity is found on average in the ranking list produced by the search system. Thus, ARP provides an estimation of the overall performance of the system, considering the ranking position of the correct result over all queries. The lower its value, the higher the *effectiveness* of the system.

(iii) *Average Response Time* (**ART**). It amounts to the time (in *miliseconds*) that intervenes between posing a query and receiving the relevant entities. The lower its value, the more *efficient* is the system is. To acquire a more reliable estimation of this measure (i.e., independent of the external factors, like the use of disk at the query time), we repeated

[13]As the ideal classification method we consider an oracle that, given a name-value pair, decides whether it is stable or not with an accuracy of 100%.

[14]For this functionality we used the Lucene library, version 2.9. See `http://lucene.apache.org`.

	Full Version	Stable Version	Classification Scheme	Random Classification
S@1 (%)	29.30	33.20 (+13.31%)	32.18 (+9.83%)	21.19 (−27.68%)
S@10 (%)	80.90	85.42 (+5.59%)	83.68 (+3.45%)	70.24 (−13.18%)
ARP	7.83	4.46 (−43.04%)	5.02 (−35.89%)	10.74 (+37.16%)
ART (ms)	3.89	3.35 (−13.88%)	3.49 (−10.28%)	3.23 (−16.97%)
IS (MB)	622	531 (−14.63%)	548 (−11.90%)	412 (−33.76%)

Table 5: Performance comparison on keyword queries over the DBPedia Infoboxes dataset, version 3.0. In parenthesis, we present the relative improvement each method conveys in comparison with the Full Version. For S@1 and S@10 the higher a value is, the better. For ARP, ART and IS, the lower a value is, the better.

	Full Version	Stable Version	Classification Scheme	Random Classification
Duplicates	1,044,099	1,005,481	1,003,390	868,987
Comparisons ($\times 10^8$)	2.31	0.62	1.21	2.04
PC (%)	93.30	89.85	89.66	77.66
RR (%)	-	73.38	47.62	11.69
IS (GB)	2.2	1.9 (−13.63%)	1.8 (−18.18%)	1.3 (−40.90%)

Table 6: Performance comparison in the task of entity resolution between the versions 3.0 and 3.3 of the DBPedia Infoboxes dataset.

the corresponding measurements 100 times and took their average value.

The performance of all the methods is summarized in Table 5. We can notice that Stable Version outperforms Full Version in all metrics; it improves the time and the space required by 14% and by 15%, respectively, while enhancing significantly its effectiveness (especially ARP). The same applies to our Classification Scheme, though to a lesser extent (10% improvement in time and 12% in space). This means that its performance is very close to the ideal one, although there is still some potential for improvement. Random Classification apparently deletes important parts of entity profiles, impairing its effectiveness to a large extent. Thus, we can safely deduce that stable entity profiles enhance not only the efficiency, but also the effectiveness of this application.

5.2 Entity Resolution

For this use case, we applied our classification scheme on the method presented in [23]; it constitutes a blocking technique that is able to efficiently resolve two voluminous, heterogeneous, individually clean but overlapping data sets by sacrificing effectiveness to a limited extend.

In essence, it completely disregards the attribute-names of entity profiles, and relies exclusively on their attribute values, tokenizing them on their special characters (this applies even to URLs). Blocks are then defined on the equivalence of tokens: each block corresponds to a single token, and each entity is associated with multiple blocks. To process the resulting, overlapping blocks efficiently (i.e., with as few comparisons as possible), it orders blocks according to their *utility*; that is, the estimated trade-off between the gain of detecting new duplicates in them against the cost of the comparisons. Additional strategies, like duplicate propagation and block pruning, are also devised in order to further reduce the number of required comparisons.

Similar to the experimental study of [23], we apply the attribute-agnostic blocking method on the two DBPedia versions with the least overlap in their content: versions 3.0 and 3.3. In fact, they share only 46.26% of their content and 47.36% of their entities. The *ground-truth* (i.e., the set of duplicate entities that have to be detected) consists of those entities that have exactly the same URL in both versions of DBPedia. This procedure yielded 1,119,133 matching entities, in total.

To measure the performance of our method, we consider the following, established metrics for blocking techniques [23]:

(i) *Pair Completeness* (**PC**) expresses the ratio between the matches identified by our method and the matches existing in the data set. It is computed as follows: $PC = dm/gm$, where dm denotes the number of detected matches, and gm represents the number of ground-truth matches. PC is comparable to the Recall metric of Information Retrieval and takes values in the interval $[0, 1]$. Higher values indicate higher *effectiveness* of the blocking method.

(ii) *Reduction Ratio* (**RR**) expresses the reduction in the number of pair-wise comparisons required by our method with respect to the baseline one. It is defined as follows: $RR = 1 - mc/bc$, where mc stands for the number of comparisons entailed by the considered method, and bc expresses the number of comparisons required by the baseline method. In our case, we employ the original methods as our baseline. RR takes values in the interval $[0, 1]$ (for $mc \leq bc$), with higher values denoting higher efficiency.

The performance achieved by each method is presented in Table 6. Similar to the previous application, Stable Version significantly surpasses Full Version in efficiency, both with respect to the required comparisons (by a whole order of magnitude), and the required space (by 13%). In this case, though, its effectiveness is a bit lower, as it detects slightly fewer duplicates. Our Classification Scheme performs similarly, requiring almost the half of the initial comparisons, while occupying 18% less disk space. Effectiveness is also lower, but very close to that of the Stable Version. Random Classification, on the other hand, suffers from poor PC and RR. On the whole, we can see that the stable parts of entity profiles significantly improve the efficiency of this application, sacrificing its effectiveness to a minor and, thus, acceptable extent.

6. RELATED WORK

Web Content Dynamics. The most similar work to ours is [22]; it explores the evolution of entities with the aim of improving object identification. To this end, it introduces a method for estimating whether observed data were generated from the same entity (due to updates in its profile) or they stem from different entities. The vast majority of the remaining works on Web Dynamics focuses on the evolution of Web Pages. [21], for example, investigates information longevity as a means of specifying the optimal

recrawl scheduling policy for each page. It estimates the actual freshness of a page, by identifying its *ephemeral* and *persistent* content fragments. Similarly, [1] detects the stable and the dynamic part of Web Pages through the *staying power* of their terms. A hypothesis similar to ours is introduced in [2]: relevant terms for a web document appear in most of its revisions and are rarely deleted. Based on this, the authors try to identify the optimal term weighting model, taking into account the history of documents in the context of social, versioned content.

Keyword Queries over Structured Data. To facilitate keyword search over structured data, recent works like [6, 26] have proposed the disambiguation of queries before the retrieval of relevant entities: each keyword query is translated into a structured query, which tries to express as accurately as possible the user's intention. To this end, the likelihood of different possible interpretations has to be assessed. Statistics pertaining to keywords, database and query logs, such as TF/IDF scores and keyword frequencies, are typically employed in this effort. Exhaustive tuning of a large number of such parameters is typically required in order to come up with the optimal estimation function [8]. As a result, we opted for an attribute-agnostic approach that disregards the attribute names, considering exclusively the values of entity profiles. Though trivial and potentially of lower performance, this approach suits better our goal: estimating the influence of stable and unstable content over this task.

Entity Resolution. *Entity resolution* (ER)—also known as *record linkage* or *duplicate detection*— aims at identifying those entity profiles or records that refer to the same real-world entity. A comprehensive overview of the existing ER work can be found in [10, 17]. The prevalent method of making ER more efficient and scalable is data blocking: data are organized into blocks of matching candidates and detailed comparisons are only performed within blocks. Blocking techniques include the Sorted Neighborhood approach [15], the StringMap method [16], q-grams-based blocking [13], Suffix Arrays approach [7] and iterative blocking [27]. However, these approaches are not applicable in the context of heterogeneous information spaces, since they require a predefined, restricted schema. Therefore, we evaluated our stability prediction approach on the attribute-agnostic blocking approach proposed in [23].

7. CONCLUSIONS

In this paper, we presented an approach for predicting stable parts of content in user-maintained information spaces, and entity repositories, in particular. The central idea is that considering exclusively the stable content of such large, heterogeneous information spaces is sufficient for many tasks. Based on statistical and content-based features, we trained an algorithm that effectively predicts the stable parts of entity profiles. We verified the effectiveness of our approach using a large, versioned data set (several versions of DBPedia). In addition, the approach was evaluated in two use cases that can profit from the prediction of and restriction to the stable part of an information space: keyword search and entity resolution. In both applications, efficiency is significantly enhanced, while effectiveness is more or less stable. In the future we plan to gain further insights into the topic by investigating whether a classifier trained on a user-generated repository (DBPedia in our case) is applicable to other repositories.

Acknowledgments. This work is partly funded by the FP7 EU Projects Glocal (Contract No. 248984) and LivingKnowledge (Contract No. 231126).

References

[1] E. Adar, J. Teevan, S. T. Dumais, and J. L. Elsas. The web changes everything: understanding the dynamics of web content. In *WSDM*, pages 282–291, 2009.
[2] A. Aji, Y. Wang, E. Agichtein, and E. Gabrilovich. Using the past to score the present: Extending term weighting models through revision history analysis. In *CIKM*, 2010.
[3] R. Almeida, B. Mozafari, and J. Cho. On the evolution of wikipedia. In *Int. Conf. on Weblogs and Social Media*. Citeseer, 2007.
[4] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *WWW*, pages 895–904, 2008.
[5] N. Bansal and N. Koudas. Searching the blogosphere. In *WebDB*, 2007.
[6] E. Chu, A. Baid, X. Chai, A. Doan, and J. F. Naughton. Combining keyword search and forms for ad hoc querying of databases. In *SIGMOD Conference*, pages 349–360, 2009.
[7] T. de Vries, H. Ke, S. Chawla, and P. Christen. Robust record linkage blocking using suffix arrays. In *CIKM*, pages 305–314, 2009.
[8] E. Demidova, X. Zhou, I. Oelze, and W. Nejdl. Evaluating evidences for keyword query disambiguation in entity centric database search. In *DEXA (2)*, pages 240–247, 2010.
[9] A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz. Towards recency ranking in web search. In *WSDM*, pages 11–20, 2010.
[10] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 2007.
[11] G. Giannakopoulos, V. Karkaletsis, G. A. Vouros, and P. Stamatopoulos. Summarization system evaluation revisited: N-gram graphs. *TSLP*, 5(3), 2008.
[12] G. Giannakopoulos and T. Palpanas. Content and type as orthogonal modeling features: a study on user interest awareness in entity subscription services. *International Journal of Advances on Networks and Services*, 3(2), 2010.
[13] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB*, pages 491–500, 2001.
[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
[15] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *SIGMOD Conference*, pages 127–138, 1995.
[16] L. Jin, C. Li, and S. Mehrotra. Efficient record linkage in large data sets. In *DASFAA*, 2003.
[17] N. Koudas, S. Sarawagi, and D. Srivastava. Record linkage: similarity measures and algorithms. In *SIGMOD*, 2006.
[18] H. Kwak, C. Lee, H. Park, and S. B. Moon. What is twitter, a social network or a news media? In *WWW*, pages 591–600, 2010.
[19] J. Madhavan, S. Cohen, X. L. Dong, A. Y. Halevy, S. R. Jeffery, D. Ko, and C. Yu. Web-scale data integration: You can afford to pay as you go. In *CIDR*, pages 342–350, 2007.
[20] M. McCandless, E. Hatcher, and O. Gospodneti. Lucene in action. *Greenwich, CT: Manning*, 2009.
[21] C. Olston and S. Pandey. Recrawl scheduling based on information longevity. In *WWW*, pages 437–446, 2008.
[22] S. Oyama, K. Shirasuna, and K. Tanaka. Identification of time-varying objects on the web. In *JCDL*, pages 285–294, 2008.
[23] G. Papadakis, E. Ioannou, C. Niederée, and P. Fankhauser. Efficient entity resolution for large heterogeneous information spaces. In *WSDM'11 (to appear)*, 2011.
[24] M. Platakis, D. Kotsakos, and D. Gunopulos. Searching for events in the blogosphere. In *WWW*, pages 1225–1226, 2009.
[25] J. Surowiecki. *The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business, economies, societies, and nations*. Random House, Inc., 2004.
[26] S. Tata and G. M. Lohman. Sqak: doing more with keywords. In *SIGMOD Conference*, pages 889–902, 2008.
[27] S. E. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina. Entity resolution with iterative blocking. In *SIGMOD Conference*, pages 219–232, 2009.
[28] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann Pub, 2005.