Biperpedia: An Ontology for Search Applications

Rahul Gupta[†] Alon Halevy[†] Xuezhi Wang[§][*] Steven Euijong Whang[†] Fei Wu[†]

[†]Google Research
{grahul, halevy, swhang, wufei}@google.com

[§]Carnegie Mellon University
xuezhiw@cs.cmu.edu

ABSTRACT

Search engines make significant efforts to recognize queries that can be answered by structured data and invest heavily in creating and maintaining high-precision databases. While these databases have a relatively wide coverage of entities, the number of attributes they model (e.g., GDP, CAPITAL, ANTHEM) is relatively small. Extending the number of attributes known to the search engine can enable it to more precisely answer queries from the long and heavy tail, extract a broader range of facts from the Web, and recover the semantics of tables on the Web.

We describe Biperpedia, an ontology with 1.6M (class, attribute) pairs and 67K distinct attribute names. Biperpedia extracts attributes from the query stream, and then uses the best extractions to seed attribute extraction from text. For every attribute Biperpedia saves a set of synonyms and text patterns in which it appears, thereby enabling it to recognize the attribute in more contexts. In addition to a detailed analysis of the quality of Biperpedia, we show that it can increase the number of Web tables whose semantics we can recover by more than a factor of 4 compared with Freebase.

1. INTRODUCTION

For the first time in the history of the Web, structured data is a first-class citizen among search results. The main search engines make significant efforts to recognize when a user's query can be answered using structured data. In parallel, they are investing significant resources in building a curated database of facts extending knowledge bases like Freebase [3]. These databases contain triples such as (*France*, CAPITAL, *Paris*) that model a broad range of topics of interest. Search results from structured data are displayed prominently on the top of or to the right of normal search results.

While the databases serving structured data in search results have a relatively wide coverage of entities (e.g., Sweden, Gerhard Weikum, Ghostbusters), the number of attributes they model (e.g., CAPITAL, PUBLICATIONS, RELEASE DATE) is relatively small. For example, for the class COUNTRIES, Freebase has less than 200 attributes, whereas the number of attributes of interest is in the thousands.

[*]Work done while on leave from Carnegie Mellon University at Google Research.

Attribute: CAPITAL (***Class***: COUNTRIES, ***Type***: atomic-textual)
Primarily attached to: LOCATIONS
Synonyms: CAPITALS
Misspells: CAPITAI, CAPITA, CAPTAL, CAPTEL, CAPIT, ...
Sub relations: CITY CAPITAL, FORMER CAPITAL, FASHION CAPITAL, ...
Provenance:

Source	***InstanceCount***	***QueryCount***	***Entities with this attribute***
Query Stream	317	11M	vietnam, turkey, romania, ...
Text	441	3M	afghanistan, iraq, pakistan, ...
Freebase	n/a	n/a	n/a

Text forms: "Hanoi is the **capital** of **Vietnam**.", "Beijing, the **capital** of **China**, is ..."
Query forms: "**capital Brazil**", "What is the **capital** of **Australia**?"

Figure 1: The elements of a Biperpedia attribute

Extending the number of attributes the search engine knows about is important for several reasons. First, additional attributes will enable the search engine to more precisely answer queries from the long and heavy tail (e.g., *brazil coffee production*). Second, additional attributes enable extracting facts from Web text using open information extraction techniques [9]. Finally, a broad repository of attributes can also enable us to recover the semantics of tables on the Web [24], because it is easier to recognize attribute names in the column headers and in the surrounding text.

We describe Biperpedia, an ontology of binary attributes that contains up to two orders of magnitude more attributes than Freebase. An attribute in Biperpedia (see Figure 1) is a relationship between a pair of entities (e.g., CAPITAL of countries), between an entity and a value (e.g., COFFEE PRODUCTION), or between an entity and a narrative (e.g., CULTURE). Biperpedia is concerned with attributes at the schema level. Extracting actual values for these attributes is a subject of a future effort. Biperpedia is a best-effort ontology in the sense that not all the attributes it contains are meaningful. However, we show that it has high precision (e.g., 0.91 for the top 100 attributes and 0.52 for the top 5000 attributes).

In addition to its scope, a second distinguishing characteristic of Biperpedia is that its goal is to support search applications. In particular, Biperpedia should enable tasks such as parsing a user query, recovering the semantics of columns of Web tables, and recognizing when sentences in text refer to attributes of entities. Traditional ontologies have been used for search in the past (e.g., [15, 23]). However, traditional ontologies tend to be brittle in the sense that they contain a *single* way of modeling the world, including a single name for any class, entity or attribute. Hence, supporting search applications with an ontology can be tricky because mapping a query or a text snippet to the ontology can be arbitrarily hard.

Biperpedia includes a set of constructs that facilitates query and text understanding. In particular, Biperpedia attaches to every attribute a set of common misspellings of the attribute, its synonyms (some which may be approximate), other related attributes (even if the specific relationship is not known), and common text phrases that mention the attribute (in the spirit of [20]).

In addition to bootstrapping from Freebase itself, Biperpedia extracts new attributes from two sources: the query stream and text on the Web. The query stream is a source of frequently asked attributes, but the challenge is to recognize the attribute seeking queries among all the others. Web text is even broader in coverage than the query stream, but as previous work on open information extraction has shown [9], can be a source of many meaningless attributes. This paper makes the following contributions.

- We describe a new kind of ontology that is tailored for search applications by adding constructs that enable interpreting Web queries and text.
- We develop algorithms for extracting attributes for the ontology from the query stream and from Web text. In particular, we develop a method for extracting attributes from text with high precision, by using distant supervision to learn text extractors that are seeded attributes we extract from the query stream and Freebase.
- We describe a novel algorithm that uses verbs with which attributes are mentioned to classify attributes into categories of numeric (e.g., GDP), atomic (e.g., CAPITAL), and narrative (e.g., HISTORY). Such a classification is invaluable while curating attributes into an existing schema, for further filtering attributes in Biperpedia, and for downstream applications like fact extraction and question answering.
- We describe an algorithm for attaching attributes to the most appropriate class in a given class hierarchy. Such an algorithm is crucial in order to contribute attributes from Biperpedia into other ontologies.
- We demonstrate the utility of Biperpedia by showing that it can help interpret the semantics of Web tables. We describe a schema matching algorithm that leverages Biperpedia and increases the number of tables we can interpret by more than a factor of 4 compared to what is possible with the schema of Freebase alone.
- We experimentally validate Biperpedia's high precision and recall for attributes of interest. In particular, we obtain over 0.5 precision for the top 5000 attributes. Among those attributes, only 1% exist in Freebase and 1% in DBpedia [2].

The paper is organized as follows. Section 2 defines our problem setting, and Section 3 describes the architecture of Biperpedia. Section 4 describes how we extract attributes from the query stream, and Section 5 describes how we extract additional attributes from text. Section 6 describes how we merge the attribute extractions and enhance the ontology with synonyms. Section 7 evaluates the attribute quality. Section 8 describes an algorithm for placing attributes in the hierarchy. Section 9 describes how we use Biperpedia to improve our interpretation of Web tables. Section 10 describes related work, and Section 11 concludes.

2. PROBLEM DEFINITION

The goal of Biperpedia is to find schema-level attributes that can be associated with classes of entities. For example, we want to discover CAPITAL, GDP, LANGUAGES SPOKEN, and HISTORY as attributes of COUNTRIES. Biperpedia is not concerned with the values of the attributes. That is, we are not trying to find the specific GDP of a given country.

Class hierarchy: We assume a given set of *classes* of entities, such as COUNTRIES or US_PRESIDENTS (note that class names always begin with a larger font letter). These classes include the types in Freebase and additional classes that identify subsets of Freebase types. For our purposes, we assume that (1) the classes are of high quality (i.e., they correspond to natural sets of entities to model in the world), (2) for each class we have a set of instances (e.g., France is a country). We also have a subclass hierarchy imposed on the set of classes (e.g,. US_PRESIDENTS is a subclass of POLITICIAN), but the subclass hierarchy may be incomplete. Furthermore, siblings in the hierarchy are not always of equal stature (e.g., under the class LOCATIONS we could have an important subclass such as TOURIST_ATTRACTIONS and an uninteresting one such as SPORTS_TEAMS_LOCATIONS).

Name, domain class, and range: The name of an attribute in Biperpedia is a string with one or more tokens, such as POPULATION or LIFE EXPECTANCY FOR WOMEN. Each attribute has a domain class, i.e., the set of entities for which the attribute is defined. Multiple classes may have attributes that have the same name. For example, POPULATION is an attribute of the class LOCATIONS and of the class BIOLOGICAL_BREEDS. Hence, the combination of class name and attribute name uniquely defines an attribute.

The domain of an attribute merely specifies that the attribute is applicable to instances of that class. However, attributes are often attached to many classes. For example, the attribute POPULATION is applicable to over 2500 classes (including all subclasses of LOCATIONS). Hence, Biperpedia also tries to identify the *best classes* to which to attach an attribute (Section 8).

Biperpedia attaches a range with every attribute. The range specifies the classes or data types to which the values of the attribute should belong. For example, the range of CAPITAL is CITIES, while the range of LIFE EXPECTANCY is a real number. We do not address the problem of computing ranges in this paper except for simple cases.

Synonyms and misspellings: The main goal of Biperpedia is to be able to identify mentions of attributes in user-generated content such as queries and text. Users will obviously not necessarily refer to an attribute with the same name that Biperpedia has for it, and they have also grown accustomed to search engines automatically fixing common spelling mistakes. To support as wide of a recall as possible, Biperpedia attaches to each attribute a set of common misspellings and a set of synonyms that Biperpedia believes refer to the same attribute (Section 6). It is important to note that misspellings and synonyms depend on the class. For example, MOTER is a misspelling of MOTHER for the class PERSON, while it is a misspelling of MOTOR for the class CARS.

Related attributes and mentions: It is also important that Biperpedia can recognize closely related attributes. For example, Biperpedia should know that MOTHER is a *subset* of PARENT, that FIRST NAME is a *part* of FULL NAME, and that RURAL POPULATION is a *component* of POPULATION. Such knowledge can help a search engine answer a broader set of queries, and retrieve relevant Web tables when they contain more detailed data. For example, it is common that a user query specifies an attribute abstractly (e.g., POPULATION), and there is a high-quality table that contains component attributes such as RURAL POPULATION and URBAN POPULATION. Biperpedia does not currently identify all possible relationships and collapses all these relations to sub/super attributes.

In the same vein, in order to facilitate recognition of mentions of attributes in text and queries, Biperpedia also attaches to every attribute a set of sentence patterns and query patterns that mention the values of that attribute (see Figure 1). In this respect, Biperpedia is very similar to the Patty System [20]. We do not discuss the problems of detecting related attributes and mentions in this paper.

Provenance: Since Biperpedia's attributes are extracted from multiple sources, we attach the set of data sources in which the attribute was discovered (`Freebase, QueryStream, Text`) and a few source-specific provenance information (described in Sections 4 and 5). When we merge two attributes we deem to be synonymous, we preserve the provenance of each of the synonyms.

Differences from a traditional ontology: At the heart of the differences between Biperpedia and a manually created ontology is the fact that Biperpedia attempts to find attributes that people consider relevant to entities in the way they are expressed in queries and text. In contrast, design considerations may dictate that a manually curated ontology model an attribute in more complex ways. For example, ARCHITECTURAL STYLE is an attribute that Biperpedia considers relevant to the class MUSEUMS. However, the same attribute is modeled in Freebase as a path: MUSEUMS has an attribute BUILDINGS OCCUPIED that has a set of buildings, and the class BUILDINGS has an attribute ARCHITECTURAL STYLE. The Freebase design is more modular, but in queries and in text people still refer to the architectural style of museums, and our goal is to recognize such mentions. Of course, Biperpedia may still have the attribute BUILDINGS OCCUPIED for MUSEUMS and ARCHITECTURAL STYLE for BUILDINGS. To support more complex reasoning that involves traversing paths in the ontology, Biperpedia will need to identify the redundant paths it contains. In general, combining the benefits of a best-effort ontology such as Biperpedia and a manually created ontology presents exciting directions for future research.

Evaluation: Given our goals, we evaluate the quality of Biperpedia in two ways. First, given an attribute A that Biperpedia proposes for class C, is A a reasonable attribute for instances of the class C? Second, we evaluate whether Biperpedia is useful in applications. In this paper we show that Biperpedia is an indispensable tool for understanding the semantics of Web tables.

3. THE BIPERPEDIA SYSTEM

The Biperpedia extraction pipeline is shown in Figure 2. At a high level, the pipeline has two phases. In the first phase, we extract attribute candidates from multiple data sources, and in the second phase we merge the extractions and enhance the ontology by finding synonyms, related attributes, and the best classes for attributes. The pipeline is implemented as a FlumeJava pipeline [6].

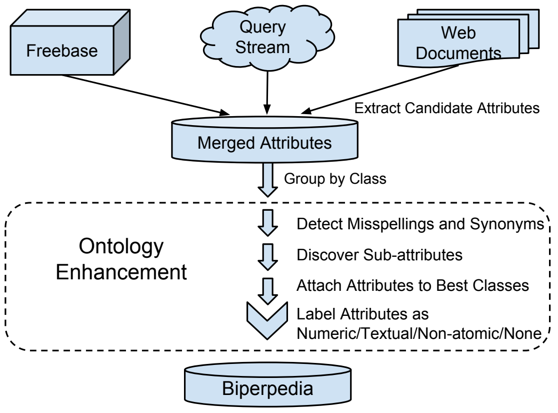
Freebase
Query Stream
Web Documents
Extract Candidate Attributes
Merged Attributes
Group by Class
Ontology Enhancement
Detect Misspellings and Synonyms
Discover Sub-attributes
Attach Attributes to Best Classes
Label Attributes as Numeric/Textual/Non-atomic/None
Biperpedia

Figure 2: Biperpedia extraction pipeline.

Attribute extraction: The pipeline begins by extracting attribute candidates from multiple sources, including Freebase, the query stream, and Web text.

The extraction from Freebase proceeds as follows (note that in Freebase, classes are referred to as types and attributes are referred to as properties). We iterate over all the types and retrieve each of its attached properties as an attribute in Biperpedia. For each property, we store the name, the type of range, and the English description if one exists. We also attach Freebase properties to their sub-types. For example, POPULATION is a property of LOCATIONS in Freebase, but we attach it also to COUNTRIES and to CITIES. The extractions from the query stream and from Web text are more involved and we describe them in Section 4 and Section 5, respectively.

We note that we also experimented with extracting attributes from Web tables [4]. However, we found that the vast majority of the attributes in Web tables are very context sensitive. As we show in Section 9, we can use Biperpedia to better interpret the attributes in Web tables.

Ontology enhancement: Once the extractions are completed, we merge the sets of attribute candidates and we index them by the domain class (e.g., we collect all attributes of COUNTRIES). Hence, the following steps are done within a single domain class.
Misspellings: we begin by identifying which attributes are common misspellings of good attribute names. Our experiments show that misspellings account for 4% of the attributes, but some tend to occur very frequently and so end up being highly-ranked attributes (e.g., FALG was one of the top attributes queried for COUNTRIES). Naturally, misspellings are much more common in the query stream than in other sources.
Synonyms: Next, Biperpedia identifies synonyms among attribute names (see Section 6). Our experiments show that 32% of the (spell-corrected) attributes can be considered synonyms of others.

Note that in principle, we could try to detect misspellings and synonyms at query time when the ontology is used to interpret a query. However, storing misspellings and synonyms in Biperpedia offers the benefit of analyzing many past queries. In addition, misspellings and synonyms do not tend to change very often, hence analyzing the query in real-time does not provide significant benefit.
Sub-attributes: We currently use two heuristics to identify sub-attributes: (1) if we find sufficient evidence on the Web (using techniques such as [14]) that "A ISA B", where both A and B are attributes, we deem attribute A to be a sub-attribute of B (e.g., MOTHER is a PARENT). (2) if we find a pair of attributes where one includes a modifier on the other (e.g., RURAL POPULATION and POPULATION), we deem the first to be a sub-attribute of the second. Although noisy, these heuristics produce useful results in practice.
Best class: Biperpedia processes each attribute in turn and tries to find the best classes in the hierarchy to attach it to.
Categorization as numeric/textual/non-atomic: Biperpedia categorizes each attribute as *numeric* (e.g. COFFEE PRODUCTION), *atomic-but-textual* (e.g. POLICE-CHIEF), non-atomic (e.g. HISTORY), or none of the above. These labels are useful for, for example, manually curating or extracting facts for only atomic attributes, or inferring measurement units and ranges for numeric attributes.

4. QUERY STREAM EXTRACTION

The query stream is an excellent source for attributes that reflect users' interests. The main challenge in mining the query stream is that queries that involve attributes of entities are mixed in with other queries, and it is hard to tease them apart. Biperpedia extracts attributes from a set of 36 billion anonymized unique queries in the following steps:

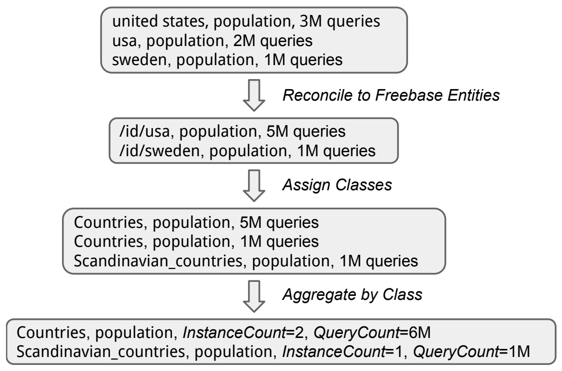
united states, population, 3M queries
usa, population, 2M queries
sweden, population, 1M queries
Reconcile to Freebase Entities
/id/usa, population, 5M queries
/id/sweden, population, 1M queries
Assign Classes
Countries, population, 5M queries
Countries, population, 1M queries
Scandinavian_countries, population, 1M queries
Aggregate by Class
Countries, population, InstanceCount=2, QueryCount=6M
Scandinavian_countries, population, InstanceCount=1, QueryCount=1M

Figure 3: Query stream extraction

Find candidate attributes: Initially, we consider all queries of the form *"what is the* A *of* E*"* to find candidate attribute names A. For example, we may find the query *"what is the population of france"* in the query stream. We denote by $\mathcal{A}$ all the values of A found.

Next, we consider all the queries of the form "A E" or "E A", where A $\in \mathcal{A}$ and E has been tagged as an entity by an entity recognizer [12]. Note that since this short form is much more common in queries, this step will match many more queries than in the first step. In particular, we will find attributes A applying to many more entities, which will be crucial for the rest of our pipeline.

We output a set of triples of the form (A, E, f), where A $\in \mathcal{A}$ is a candidate attribute name, E is an entity string, and f is the number of times the query "A E" or "E A" appeared in the query stream.

Reconcile to Freebase: The goal of this step is to lift attribute mentions from instances to classes so we can reason about them in aggregate. Specifically, for every pair (C, A), where C is a class and A is a candidate attribute name, we compute two values:

- $InstanceCount$(C, A): the number of distinct entities E of C for which queries of the form "A E" or "E A" were mentioned in the query stream, and
- $QueryCount$(C, A): the total number of queries in the stream of the form "A E" or "E A" where E refers to an instance of C.

$InstanceCount$(C, A) indicates how prevalent the attribute A is for instances of C, whereas $QueryCount$(C, A) indicates how popular the attribute A is in the query stream. Together, these values indicate whether a particular A is even an attribute at all – candidate A's with low counts are typically noise. For each triple, (A, E, f), we proceed as follows (see Figure 3).

- Reconcile E with an entity E in Freebase. This step is inherently heuristic: in some cases we will match e to the wrong entity and in others we will not find a match at all. Our recognizer returns a ranked list of candidate entities and we conservatively choose the first one. There are cases where we miss valid mappings (e.g., for *apple* the recognizer will return both the company and the fruit but we choose only the company). However, we can afford to be conservative because even if we miss a few entities it will not degrade the schema-level extractions. There are plenty of other companies who will not be confused with fruit names, and so properties of companies will be found reliably.
- Given the value of E, we look up all the classes in Freebase, $C_1, \ldots, C_n$, in which E is known to be a member of. For each pair (C_i, A) we increase $InstanceCount$(C, A) by 1 and add C to $QueryCount$(C, A).

Remove co-reference mentions: A common pattern in search queries is to follow an entity by a qualifier. For example, many users query *barack obama president* (and do so for every president of the USA!) As a result, we could extract PRESIDENT as a top attribute of the class US_PRESIDENTS, which is clearly undesirable.

To filter such extractions, we look for evidence in Web text that suggests that BARACK OBAMA *is* a president. We run a standard coreference resolution algorithm [13] to find if the strings "barack obama" and "president" co-refer to the same entity sufficient number of times in the text corpus. For coreferring string pairs, we do not increase the counters $InstanceCount$(C, A) and $QueryCount$ (C, A). Instead, we maintain a counter $CoReferences$(C, A) to track the number of co-reference occurrences.

Output attribute candidates: Biperpedia keeps any pair (C, A) for which $InstanceCount$(C, A) is above a pre-determined threshold. The provenance of the attribute includes the counters $Instance$-$Count$(C, A) and $QueryCount$(C, A).

5. EXTRACTION FROM WEB TEXT

Text on the Web offers an even richer source of attributes for Biperpedia, and is in many ways complementary to attributes extracted from queries. While queries indicate attributes that people are asking about, text provides more of an encyclopedic overview of an entity (e.g., Wikipedia, news articles, press releases, product brochures, etc.). For example, while many people query for the current MAYOR of a town, relatively much fewer query for its FIRE-CHIEF. In addition, attribute extraction from text also serves as a way to corroborate attributes extracted from queries and other sources. In this section we describe how we extract attributes for Biperpedia from text. The main technical challenge we face is that extraction from text can be extremely noisy. Our main technical contribution in this section is to show that we can use the extractions from the query stream and Freebase to train a learner a high-quality text extractor.

But before we describe our extraction process, it is instructive to step back and look at two broad questions: (1) Can we manually define a handful of patterns by hand, like we did for queries, obviating the need for designing an extractor?, and (2) Since our extraction goal is open-domain, can we simply leverage existing open information extraction systems instead of designing a new extractor?

The answer to the first question is no, and we provide empirical evidence later in the section that we do need lots of extraction patterns in practice. The intuitive reason is that in document text, attributes might be co-expressed with entities in lots of different ways, as opposed to query text, which is short and allows only a few possibilities of attribute expression.

The answer to the second question is also no, and warrants the following detailed discussion.

Can we leverage existing open-domain systems? As mentioned earlier, Biperpedia's goal is schema-level attribute extraction while most existing open-domain extraction systems output facts. In principle, one can use the output of a conventional open-domain extractor to emit attributes instead of attribute values. For example, one can extract many instances of (PERSON, STARRED IN, MOVIE) and posit that STARRED IN is an attribute applicable to PERSON. However, such an approach has a few limitations, that are best illustrated in the context of various state of the art open-domain extractors.

The ReVerb system [9, 10] extracts binary relationships like "starred in" and "was elected to" between entities. Each relation is constrained to be a verb phrase, or satisfy a verb-centric regular expression on the parts-of-speech tags. This is restrictive as many attributes like CULTURE or POLICE-CHIEF are unnatural to represent via verb-phrases. Secondly, it is non-trivial to deduplicate verb phrases and canonicalize them to a noun-phrase form. For example, we need to be able to normalize STARRED IN, IS THE STAR OF, and HAS ACTED IN to the noun-phrase form LEAD ACTOR, while excluding close phrases like STARRED which incidentally expresses the reverse attribute.

The OLLIE system [18] is an improvement on ReVerb, as it also induces noun-phrase patterns, and therefore can extract values for attributes like POLICE-CHIEF. While this is better, this still does not properly handle cases like *"Brazil's coffee production rose by 5%"*. In this sentence, the extracted relation would be "rose by" between "Brazil's coffee production" and "5%", instead of the desired relation between Brazil and coffee-production. It is simpler to directly learn the pattern that connects Brazil and coffee production.

Last, the NELL system [5] has a limited set of $\approx$ 600 relations (including inverses), so one cannot get a huge attribute set out of it.

In light of these issues, we take the approach of directly learning patterns that connect an entity to an attribute. Both the entity and the attribute are assumed to be noun-phrases, which bypasses the issue of deduplicating verb forms of attributes to a canonical noun phrase form. Note that deduplication is required even in the case of noun-phrase attributes (e.g., LEAD ACTOR and STAR ACTOR should be the same), but here we use the synonym detection system described in Section 6.

Since we already have high-quality attributes extracted from Freebase and from the query stream, it is natural to apply distant supervision to seed the extraction of attributes from text.

5.1 Extraction via distant supervision

Biperpedia extracts attributes from text by inducing a set of (entity, attribute) extraction patterns and applying them to a text corpus. These patterns leverage a set of standard natural language processing (NLP) primitives. For example, once we have identified noun phrases in the text, we can apply the lexical pattern "A of E", where A and E match noun phrases, and represent an attribute and entity respectively. Similarly, the parse pattern "E poss> A" identifies the second noun phrase as the attribute once the text has been processed by a dependency parser (here the dependency label *poss* refers to 'possessive'). We note that the induced patterns would need to be extended in order to extract *values* for the attributes.

We use *distant supervision* [19] and high-quality attributes extracted from Freebase and the query stream to induce extraction patterns from text. Distant supervision has been successfully used in various large scale extraction tasks, e.g., [19, 27]. In distant supervision, one is not given a supervised corpus (as that would be very expensive to obtain); instead one is provided with a knowledge base (KB) containing sample facts of the kind that we wish to extract. In our case, KB is created from the top attributes already extracted by Biperpedia. We then assume that if a pair of related entities in this KB is seen in a sentence, then that occurrence expresses the corresponding relation. As an example, consider the left-half of Figure 4. Say we know that COFFEE PRODUCTION is an attribute of `Brazil`. Then if we see the text *"The coffee production of Brazil rose by 5%"*, we can posit that the lexical pattern 'The A of E' and the parse pattern 'A prep> of pobj> E' are candidate patterns that connect an attribute to an entity.

We obtain an aggregated list of candidate patterns sorted by frequency, along with the number of known attributes they cover. Table 1 shows some of the top lexical and parse patterns thus induced. We select all patterns induced by at least 10 unique existing (entity, attribute) pairs, and that fire at least 10 times in a held-out subset of the text corpus.

Attribute extraction using induced patterns: We begin by applying the following natural language pre-processors to all the text: parts-of-speech (POS) tagger, dependency parser, noun phrase seg-

Pattern	Example
E A	[Google] (CEO) Larry Page
A , E	Larry Page, (CEO), [Google]
(E = his) A	[his] (wife)
E 's A	[Google] 's (CEO)
A of E	(CEO) of [Google]
A in E	(urban population) in [Kingston]
A of the E	(captain) of the [Australian cricket team]
(E = its) A	[its] (fire-chief)
A at E	(CEO) at [Google]
A for E	(spokesman) for [gun control]
E nn> A	[Eagles] (coach) Al Skinner
E poss> A	[Susan Sarandon] 's (partner)
(E = his) <poss A	[his] (wife)
A prep> of pobj> E	(CEO) of [Google]
A prep> in pobj> E	(MVP) in the [National League]
A appos> E	Steve Jobs, former (CEO), [Apple]
(E = her) <poss A	[her] (husband)
A prep> with pobj> E	(hitting coach) with the [Yankees]

Table 1: Top few induced lexical (top-table) and parse patterns (bottom table) for attribute-extraction, along with an example.

menter, named entity recognizer, coreference resolver, and entity resolver. These steps are necessary so we can define specific extraction patterns as illustrated earlier. The coreference resolver resolves pronouns and nominals to entities, thereby increasing coverage. For example in the text *"John Smith lives in London. [His] (wife)..."*, the coreference resolver will tell us that 'His' and 'John Smith' are the same entity, so 'wife' is really an attribute of 'John Smith'. The entity resolver maps mentioned entities to Freebase, enabling us to generalize from individual extractions to classes as we did with the query stream.

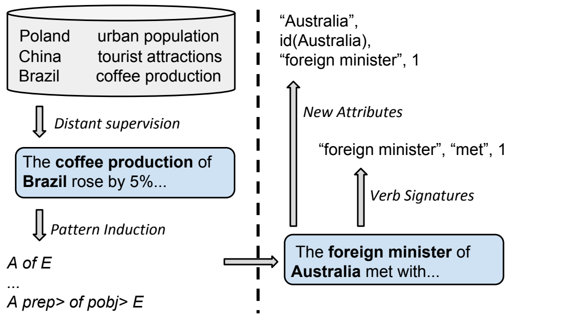
Poland urban population
China tourist attractions
Brazil coffee production
Distant supervision
The coffee production of Brazil rose by 5%...
Pattern Induction
A of E
...
A prep> of pobj> E
"Australia", id(Australia), "foreign minister", 1
New Attributes
"foreign minister", "met", 1
Verb Signatures
The foreign minister of Australia met with...

Figure 4: Distant supervision and subsequent extraction.

The induced patterns are then applied using lexical and parse pattern matchers on a large NLP-processed web-crawl corpus (right half of Figure 4). Matches where the attribute is not a nominal (i.e., common noun) are discarded. This step outputs a set of extraction tuples of the form (A, E, E-ID, f). These tuples have the same form as the extractions from the query stream, except that E-ID is the Freebase id of E. These tuples are processed in the exact same way as the query stream.

Figure 5 shows the yield of the top induced extraction patterns. Although we induce more than 2500 patterns, we see that the top-200 patterns account for more than 99% of the extractions. At the same time we observe that no pattern covers more than 6% of the extractions, which means that we indeed need a pattern-induction pipeline to learn the large variety of patterns, and we cannot simply operate with a small set of hand-written patterns.

5.2 Attribute classification

Given the huge number of attributes we can extract from Web text, we still need a method for selecting the ones we add to Biperpedia. Given the anticipated uses of Biperpedia (e.g., additions

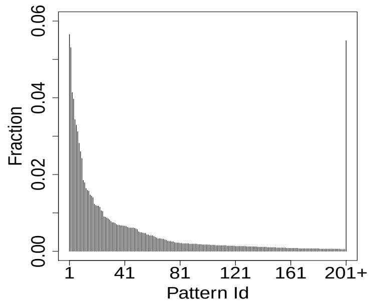

Figure 5: Extraction yield of patterns. X-axis plots patterns in decreasing order of extraction yield, and the y-axis plots the yield as a fraction of total.

to Freebase, question answering, and table interpretation), we focus Biperpedia on *atomic* attributes with clearly defined values. In particular, our goal is to distinguish *atomic-numeric* attributes (e.g., COFFEE PRODUCTION), and *atomic-textual* attributes (e.g., POLICE-CHIEF) from non-atomic attributes such as CULTURE and HISTORY. Classifying attributes as atomic versus non-atomic also enables future efforts to detect meta-data such as ranges and units, and ultimately extraction of attribute values.

The technical challenge is to classify attributes without having to extract values for them (which is an even harder problem). The algorithm we describe below demonstrates another benefit of the natural language processing we perform. Specifically, since our text corpus is already pre-processed, we have the dependency parse information for every sentence in the corpus. Since each attribute is a noun phrase, we know the verb in the sentence that it is a grammatical subject of (as specified by the *nsubj* dependency label). We found that the list of most frequent verbs for an attribute are highly informative for our classification task. For example, by looking at the text *"Brazil's coffee production increased by 5%"*, we can infer that COFFEE PRODUCTION is a numeric attribute since the verb-lemma 'increase' is positively correlated with the presence of numeric attributes. Similarly, by looking at the text *"New York's police-chief resigned today..."*, we can infer that POLICE-CHIEF is not a numeric attribute as the verb-lemma 'resign' is negatively correlated with numeric attributes.

We note that our classification is not exhaustive as there are other kinds of attributes that are not fully covered by our three categories e.g., phone-numbers, dates, misspellings, discrete-valued attributes, etc. We club all of them in a fourth category called *other*, which is a less structured 'background' category. In what follows we describe how we learn three independent binary classifiers for our three more-structured categories of interest.

Features. Our list of raw features is just the set of top-k verbs that are *nsubj* parents of the attribute in our text corpus, where k is set via cross-validation. These are extracted from the text along with the attribute (see Figure 4). It is possible to enrich this feature space further by adding extra linguistic cues like adjectives, modifiers, and special clauses attached to the attribute, and this is something that we wish to explore in future work.

Table 2 shows the top few verbs for some sample attributes. As illustrated earlier, numeric attributes like COFFEE PRODUCTION and ENROLLMENT are strongly associated with verb-lemmas like *increase, decline, drop,* etc. At the same time, popular verbs like *have, say, become* are present in every category, and will ultimately be given low weights during training.

However, using raw lexical terms as features is prone to overfitting due to the huge vocabulary size, as well as unknown features at test time, as we only have a limited training corpus. Therefore we employ the fairly standard feature hashing trick to reduce the verb vectors to a standard space with a preset dimensionality d. More concretely, each verb v_i is hashed to dimension $h(v_i)$ mod d, where h is a hash-function. Hashing has the extra advantage that ones does not need to lug around a lexicon of features; instead only the hash function h and d need to be stored.

Classifier. The final hashed features thus obtained are combined using a logistic regression model, whose weights are learnt using a small manually labeled corpus of attributes. The model is regularized using both L1 and L2 costs, i.e., we optimize the following training objective:

$$\min_{\mathbf{W}} \sum_i \mathbf{W}^T \cdot \mathbf{F}(x_i, y_i) + \lambda_1 ||\mathbf{W}|| + \lambda_2 ||\mathbf{W}||^2 \qquad (1)$$

where $\mathbf{W}$ is the weight vector to be trained, $\mathbf{F}(x_i, y_i)$ is the hashed feature vector for training attribute x_i labeled as $y_i \in \{-1, 1\}$, and $\lambda_1,\ \lambda_2$ are the L1/L2 hyperparameters set using cross-validation. This objective is optimized using a standard off-the-shelf second-order solver. We emphasize that separate models (i.e., $\mathbf{W}$ vectors) are learnt for the three categories of interest, and that the hyperparameters $\lambda_1,\ \lambda_2,\ d,\ k$ are separately tuned for each classifier via grid search and cross-validation.

Experiments. We trained and evaluated our three classifiers using 5-fold cross validation on a manually labeled corpus of 1212 attributes. These attributes belong to 16 varied classes so as to avoid any bias during training. We compare our classifiers against a simple baseline that always assigns the majority label (positive or negative) to each attribute. Our goal is to see how much the verb-signatures help in classifying the attributes over this baseline.

Table 3 lists the classification results for the three tasks, with best settings of the hyperparameters. The two metrics of interest are: our precision/recall/F1 over just the positive label (i.e., labeling an attribute as numeric, atomic-textual, or non-atomic, respectively, in the three tasks), and our overall accuracy on both positive and negative labels. We see that our classifiers can identify atomic attributes (both numeric and textual) with a fairly high positive-F1 score as well as overall accuracy. Numeric attributes in particular are very easy to identify with verb signatures, with an overall accuracy of 93.9%. Non-atomic attributes on the other hand, are harder, with a lower F1-score, but even there the verb-signatures help us in improving over the baseline by almost 10 absolute points.

We observe that the three tasks require varying lengths of verb signatures (i.e., k). While only 50 verbs are enough to identify numeric attributes, other attributes need a lot more verb evidence to make a decision. Figure 6 plots the F1 of the positive label and the overall accuracy for the three classifiers as we vary k. We see that for all the classifiers, the performance increases up to a point, then starts decreasing/flattening after a particular k. This is expected as the system initially benefits from seeing more informative verbs, and then degrades once it is fed more and more verbs that lack discriminatory power. Overall we observe that 50–200 verbs are sufficient to classify an attribute with a fairly good accuracy.

Category	# of Verbs	P(+)	R(+)	F1(+)	Base Acc	Our Acc
Numeric	50	86.5	78.8	82.5	81.6	93.9
Atomic-textual	100	83.3	86.4	84.8	59.2	81.7
Non-atomic	200	69.3	57.1	62.8	77.6	86.0

Table 3: Attribute classification results. (+) means that the metric is only for the positive class.

Attribute	Type	Top associated verbs
coffee production	numeric	fall, decline, rise, drop, increase, reach, climb, have, expand, grow, come, total
urban population	numeric	grow, increase, exceed, reach, rise, double, expand, surpass, have, continue, jump, outnumber, pass
enrollment	numeric	increase, grow, drop, rise, decline, fall, continue, jump, double, begin, go, reach, stand
police chief	atomic-textual	say, tell, resign, announce, accuse, confirm, have, warn, quit, ask, call, take
headquarters	atomic-textual	say, remain, have, move, receive, become, open, collapse, announce, provide, issue, locate,
protagonist	atomic-textual	have, say, find, go, win, work, make, do, come, take, become, ask, struggle, fall
abbreviation	atomic-textual	stand, say, mean, become, refer, have, come, appear, feature, use, reveal, derive, go, begin
doctrine	non-atomic	say, apply, require, make, have, provide, teach, hold, become, seek, allow, mean, call
history	non-atomic	show, say, make, date, go, include, have, begin, suggest, give, tell, come, prove, mean
culture	non-atomic	have, say, change, become, make, come, go, allow, continue, encourage, need, seem
employee benefits	non-atomic	include, say, recognize, offer, continue, increase, account, decrease, match, terminate
family structure	non-atomic	include, become, weaken, crumble, prove, continue, base, have, collapse, say, undergo, mean, make

Table 2: Verb-signatures for sample attributes in Biperpedia. For each attribute, the verbs are represented in lemma form, and sorted in decreasing order of count.

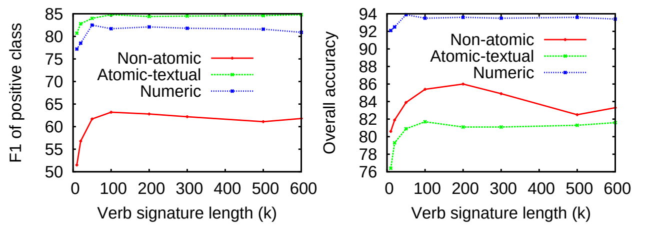

Figure 6: Attribute classification performance vs verb-signature size.

6. SYNONYM DETECTION

By nature, people refer to the same attribute in different ways in queries and in Web text. For example, the TOURIST ATTRACTIONS and TOURIST SPOTS of a country are synonymous. Furthermore, the query stream contains many spelling mistakes that users assume the search engine will fix for them automatically. The quality of Biperpedia critically depends on its ability to recognize synonyms and misspellings. We note that both misspellings and synonyms are dependent on the class to which the attribute is attached.

Detecting synonyms and misspellings deserves a detailed treatment on their own, and is not a major contribution in this paper. For completeness, we describe how Biperpedia addresses these problems. In both cases we use internal implementations of techniques proposed for query expansion [7] and spell correction.

For spell correction, we rely on the search engine. Given an attribute A of a class C, we examine the spell corrections that the search engine would propose for the query "C A". If one of the corrections is an attribute A' of C, then we deem A to be a misspelling of A'. For example, given the attribute WRITTER of class BOOKS, the search engine will propose that *books writer* is a spell correction of *books writter*.

To detect synonyms, we train an SVM classifier that uses the following features on a pair of attributes A_1 and A_2: (1) Jaro-Winkler text similarity between attributes, (2) whether one attribute is a sub-attribute of the other, (3) whether the two attributes are known to be antonyms in WordNet [11], and (4) similarity of query expansion. Specifically, we compare the query expansion results of the queries "C A_1" and "C A_2". High overlap in the sets of expansions is a signal that A_1 and A_2 may be synonyms. Based on experiments that we do not report in this paper, we estimate the precision of the SVM-based synonymizer to be 0.87.

7. ATTRIBUTE QUALITY

In this section, we evaluate the quality of Biperpedia attributes and compare it to DBpedia [2], which is an ontology that is automatically extracted from InfoBoxes in Wikipedia.

7.1 Experimental setting

The current version of Biperpedia is built using a hierarchy of slightly over 10,000 classes. After extraction, merging synonyms and attaching attributes only to the best classes, we ended up with 1.6M class-attribute pairs with 67,000 distinct attribute names. (We note that the step of attaching attributes to best classes reduces the number of class-attribute pairs by almost 50%.) As seen in Table 4, Biperpedia is about 30 times bigger than DBpedia. We extracted all the attributes from Freebase, but limited our extraction from the query stream to 8000 attributes per class and only an additional 4000 from text that were not extracted from the query stream. We note that the number of attributes extracted from Freebase accounts for less than 6% of the total number of attributes.

Statistics	Biperpedia	DBpedia
Number of classes	10,309	529
Number of class-attribute pairs	1.6M	52,380
Average number of attributes per class	269	99
Number of unique attributes	67K	2,379

Table 4: Biperpedia and DBpedia statistics.

We experiment with 10 representative classes (Table 5), chosen so some are broad and others are narrow. We also show the number of attributes in corresponding DBpedia classes if they exist.

Class	No. of attributes	
	Biperpedia	DBpedia
FILMS	7.2K	44
COUNTRIES	7K	186
UNIVERSITIES	4.8K	21
NEWSPAPERS	4.3K	10
HOTELS	4.2K	17
COMEDY_FILMS	5.1K	n/a
US_PRESIDENTS	3.9K	n/a
UK_COUNTRIES	3.5K	n/a
SPORTS_CARS	1.6K	n/a
AMUSEMENT_RIDES	0.8K	n/a

Table 5: 10 classes for evaluation.

7.2 Overall quality

To measure precision, we manually labeled 100 randomly chosen attributes out of the top-k for several values of k (unless $k \leq 100$, in which case we labeled all of them). We then used majority voting among 3 evaluators to determine whether an attribute is good or bad for this class. In Table 6 we measure the precision with two different rankings of the attributes: **Rank by Query** – ranks the attributes of a class by the number of instances of that class for which the attribute appeared in the query stream, and **Rank by Text** does the same for appearances in text. The **Precision** column specifies the fraction of attributes that were labeled as good. As k increases, the precision drops, but is still over 0.5 at 5000. We note that previous work [16, 21] reported precision only as far as $k = 50$ and the precision reported at $k = 50$ is slightly lower than what we obtain for $k = 1000$. The results also suggest that there are many more good attributes beyond 5000 that could be added to Biperpedia, but we need more focused techniques to identify them. The **Freebase** column shows the fraction of good attributes that are also found in Freebase, which drops rapidly. One striking observation is that 99% of attributes at $k = 5000$ are new for Freebase, which brings out the importance of Biperpedia for schema augmentation. Similarly, the **DBpedia** column shows the fraction of good attributes that are also in DBpedia (using the 5 classes in Table 5 that exist in DBpedia), which also drops as k increases.

The precision of the attributes when ranked by text is a bit lower initially than when ranked by the query stream. This can be explained by the fact that extraction from the query stream can be made more precise than from general text. However, at the lower ranks the precision starts being similar. While we do not show this in the table, it is interesting to note that the precision for attributes that appear in *both* the query stream and text is similar for the small values of k, but *higher by up to 8%* for $k = 5000$. This happens because when an attribute appears very frequently in one of the sources, it does not need any corroboration, but as the frequency decreases, corroboration becomes a factor.

Top-k	Rank by Query			Rank by Text		
	Precision	Freebase	DBpedia	Precision	Freebase	DBpedia
10	0.98	0.4	0.06	0.88	0.25	0.24
50	0.95	0.14	0.09	0.76	0.15	0.11
100	0.91	0.11	0.06	0.7	0.08	0.06
500	0.79	0.04	0.03	0.64	0.05	0.03
1000	0.72	0.02	0.02	0.6	0.02	0.02
2000	0.6	0.02	0.01	0.57	0.01	0.02
5000	0.52	0.01	0.01	0.54	0.01	0.01

Table 6: Attribute quality. The Freebase (DBpedia) column shows the ratio of good attributes also in Freebase (DBpedia).

Table 7 shows an interesting relationship between an attribute's rank (k) and its number of synonyms. As k increases, the synonyms per attribute decreases. For example, the class UNIVERSITIES has 2.8 synonyms per top-10 attributes, but only 0.75 synonyms per top-1000 attributes. Hence, the highly-ranked attributes tend to be expressed in many forms.

Class	Top-10	Top-100	Top-1000
FILMS	2.3	1.61	1.13
COUNTRIES	1	1.9	1.05
UNIVERSITIES	2.8	1.46	0.75
NEWSPAPERS	2.6	1.45	0.88
HOTELS	2.9	1.5	0.88
COMEDY_FILMS	2.5	1.5	0.98
US_PRESIDENTS	1.8	1.36	0.75
UK_COUNTRIES	1.8	1.34	0.88
SPORTS_CARS	2.2	0.92	0.44
AMUSEMENT_RIDES	0.9	1.01	0.37

Table 7: Avg. # synonyms per attribute.

7.3 Discussion

The analysis of Biperpedia reveals several important observations. First, the most significant reason for losses in precision is caused by errors in mapping strings to entities in Freebase. For example, for classes such as COUNTRIES or US_PRESIDENTS where entity linking is easier, we obtain precision at 5000 that is 20-30% higher, while the precision for classes such as FILMS and HOTELS, where there is a much higher frequency of erroneously linking strings to entities, is lower by that amount. Fortunately, improving entity linking is a subject of several ongoing efforts. In the same vein, Biperpedia does not extract attributes for classes of objects whose instances are not notable enough to be in Freebase, such as SPOONS. To extract attributes for such classes, Biperpedia can still use the same extraction patterns, but should not attempt to map the entity in the pattern to Freebase, but rather assume that each mention is to an *anonymous* entity of the appropriate class.

Second, we note that many of the highly queried attributes are the non-concrete ones (e.g., CULTURE, HISTORY). This can be explained by the fact that users start their explorations of a topic with such queries. We also note that there are classes with which many more attributes can be associated than others. For example, geographical locations tend to have many attributes because quantities are often measured with respect to geography.

Finally, we realized that one of the strengths of Biperpedia is extracting multi-token attributes. These attributes tend to be richer and also tend to appear often in Web tables. To extract additional such attributes, we implemented the following rule, explained by example. If POPULATION or RURAL POPULATION are highly ranked attributes from the query stream, then we infer that other attributes like URBAN POPULATION that share the same syntactic root (i.e., 'population') are also good, even if their support is not high. For each class in Table 5 we evaluated a random subset of 100 multi-token attributes sampled from *beyond the first 5000 attributes* from Web text, and we obtained an average precision of 0.67. This is better than the precision of 0.64 over the first 500 attributes from Web text (Table 6). Using this method, we were able to approximately double the size of Biperpedia while maintaining its average precision. In general, we believe that Biperpedia demonstrates that there are many more good attributes to be mined beyond the top 5000, but we will need more focused techniques to get them.

8. FINDING BEST CLASSES

As mentioned earlier, Biperpedia attaches an attribute to *every* class in the hierarchy to which it is relevant. This is appropriate when we want to verify whether an attribute is relevant to a particular class, but if we wish to create a more modular ontology or find attributes that can be contributed to Freebase, we need to find the *best* classes to which to attach an attribute.

The challenge in finding the best classes is illustrated by the following example. Suppose we want to assign the attribute BATTERY LIFE to classes in the hierarchy shown in Figure 7. The top-most class CONSUMER_PRODUCTS is too broad because not all consumer products (e.g., SHOES) have batteries. The leaf classes SLR_DIGITAL_CAMERAS and COMPACT_DIGITAL_CAMERAS, on the other hand, are too specific because any digital camera has a battery. As a result, the class DIGITAL_CAMERAS can be considered as a best class for BATTERY LIFE. Using a similar argument, COMPUTER_PERIPHERALS (a sibling class of DIGITAL_CAMERAS) can also be viewed as a best class for BATTERY LIFE. Alternatively, if BATTERY LIFE applies to the vast majority of the sub-classes of CONSUMER_PRODUCTS, we may now want to attach it to CONSUMER_PRODUCTS. Hence, the challenge is to find the classes that are not too general, but not too specific either, and trading off parsimony of the ontology.

The key idea underlying the algorithm we describe below is that we compute a support for each attribute within a class. We then compare the support of the attribute in the class with its support in parent classes and in sibling classes to make a decision on the best placement of the attribute. Importantly, the notion of support enables us to consider multiple ways to make the placement decision.

Our algorithm makes the placement decision for each attribute independently of others. An interesting future direction is to use previous attribute placement decisions to drive subsequent ones.

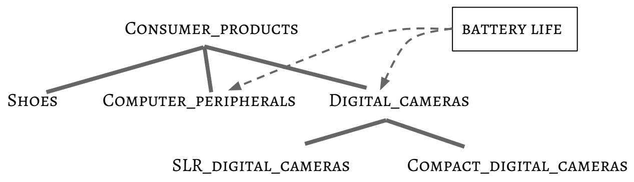

Figure 7: Class hierarchy and attribute placement

8.1 Placement Algorithm

Algorithm 1 computes the best classes, O_{A}, for an attribute A. Informally, for every attribute A, the algorithm traverses, in a bottom-up fashion, each tree of classes for which A has been marked as relevant (i.e., each tree that contains class C where (C, A) is extracted). For each pair of class and attribute, (C, A), we compute the *support* for A in C, $S(\text{C, A})$. The support is computed from the provenance. For example, for an attribute extracted from the query stream, the support is the ratio between the number of instances of C that have A and the maximal number of instances for any attribute of C.

$$S_{\text{query}}(\text{C, A}) = \frac{InstanceCount(\text{C, A})}{\max\limits_{\text{A*}}\{InstanceCount(\text{C, A*})\}}$$

The support from text extractions is computed similarly, and the support for attributes from Freebase is 1. We define $S(\text{C,A})$ to be the maximal support it gets from any of the sources.

The key decision the algorithm needs to make is the following. When there are several siblings with sufficient support, should they all be in O_{A}, or should we continue up the class hierarchy. To address this challenge, the algorithm computes a *diversity* measure for the siblings. If there is little diversity among the support for the siblings, we continue up the tree. If there is significant diversity, i.e., only a few of the siblings have sufficient support, we output these siblings. The diversity is defined as follows.

$$D(\text{C}_1, \ldots, \text{C}_n, \text{A}) = \begin{cases} \frac{1}{n-1}\sum\limits_{i=1}^{n} \frac{\max\limits_{j=1,\ldots,n}\{S(\text{C}_j,\text{A})\} - S(\text{C}_i,\text{A})}{\max\limits_{j=1,\ldots,n}\{S(\text{C}_j,\text{A})\}} & n > 1, \\ 0 & n = 1 \end{cases}$$

where $\text{C}_1, \ldots, \text{C}_n$ are sibling classes. When the diversity is above a threshold θ (Line 9 in Algorithm 1), we add to O_{A} all the siblings whose support is a factor of α more than the highest among the siblings. In our example, for attribute BATTERY LIFE, the diversity for sibling classes under CONSUMER_PRODUCTS will be high because some consumer products have batteries, but others do not. Hence, the sibling classes that have high support will be added to $O_{\text{BATTERY LIFE}}$. We tuned θ separately in our experiments.

8.2 Evaluation

We applied Algorithm 1 on the attributes of Biperpedia and describe the result of evaluating the assignments of 50 attributes onto 1100 classes. For each assignment, we asked 3 evaluators to specify whether the assignment is (1) correct, (2) should be to the immediate parent, (3) to one of its immediate children, (4) to classes other than the immediate parent/children in the same hierarchy, or (5) completely erroneous. We call an assignment *exact* if an attribute is assigned to one of its best classes and *approximate* if it is exact or is assigned to immediate parents or children of best classes. Our results consider several precision measures:

Algorithm 1: Place attribute in class hierarchy

```
   input  : An attribute A and a forest of classes $T$.
   output: $O_A$: the classes to which we attach A.
1  $O_A \leftarrow \emptyset$;
2  foreach C $\in$ T.Classes s.t. (C, A) is extracted do
3      Add the roots of $T$ that contain C to $relevantRoots$;
4  foreach R $\in relevantRoots$ do
5      $t \leftarrow$ tree with root R;
6      foreach C $\in$ t.Classes in post order do
7          if C is not leaf node then
8              Compute $S(C_i, A), i = 1, \ldots, n, C_i \in C.children$;
9              $S_{max} \leftarrow \max_{i=1,\ldots,n} S(C_i, A)$;
10             if $D(C_1, \ldots, C_n, A) > \theta$ then
11                 $O_A \leftarrow O_A \cup \{C_j: S(C_j, A) > \alpha \cdot S_{max}\}$;
12             else if C = $root$ then
13                 Add C to $rootList$;
14 foreach ROOT $\in rootList$ do
15     if $S(ROOT, A) > \alpha \cdot \max_{R \in rootList} \{S(R, A)\}$ then
16         $O_A \leftarrow O_A \cup$ ROOT;
17 return $O_A$;
```

- $\mathcal{M}_{exact}$: ratio of number of exact assignments to all assignments.
- $\mathcal{M}_{approx}$: ratio of number of approximate assignments to all assignments. Note that an approximate assignment is still valuable because a human curator would only have to consider a small neighborhood of classes to find the exact match.

$\mathcal{M}_{exact}^{filtered}$ (resp. $\mathcal{M}_{approx}^{filtered}$) is the same as $\mathcal{M}_{exact}$ (resp. $\mathcal{M}_{approx}$) but applied only to attributes deemed good in the experiments in Section 7.

To demonstrate the benefits of the diversity index, we also compare with a strawman algorithm that traverses the tree bottom up using the following rule when considering a node n with sufficient support. When the support for the parent of n is similar to that of n, or there are 5 (tuned to perform best in the experiment) or more children with similar support to n, the algorithm continues up the tree. Otherwise, the attribute is assigned to n and any of its siblings that have sufficient support.

The evaluation results, shown in Table 8, demonstrate that the best results for our algorithm are obtained when we set θ to be 0.9 (we show only a few selected values from the ones we experimented with), and that the algorithm outperforms the strawman algorithm by more than 50% on both unfiltered attributes and filtered attributes. Even on the unfiltered attributes, our algorithm shows a precision of 72% for approximate matches, which offers an excellent starting point for importing Biperpedia attributes into a curated ontology. $\mathcal{M}_{approx}$ and $\mathcal{M}_{approx}^{filtered}$ at $\theta = 1.0$ are not computed because under this extreme case, attribute A will be assigned to and only to every relevant root, which has no immediate parent. In the experiments, the value of α is set to 0.1, but we note that α (Line 10 in Algorithm 1) mostly affects the number of classes returned, rather than the precision of the results.

Algorithm	$\mathcal{M}_{exact}$	$\mathcal{M}_{exact}^{filtered}$	$\mathcal{M}_{approx}$	$\mathcal{M}_{approx}^{filtered}$
Strawman	0.36	0.45	0.66	0.83
$\theta = 0.5$	0.41	0.55	0.63	0.84
$\theta = 0.7$	0.47	0.61	0.68	0.88
$\theta = 0.9$	0.56	0.71	0.72	0.91
$\theta = 1.0$	0.4	0.52	n/a	n/a

Table 8: Attribute placement precision.

Analyzing the errors of our algorithm illustrates an interesting challenge for future work. Our diversity measure is highly dependent on peoples' interest in specific entities. For example, users frequently query about the brothers of presidents, but not of people with other government titles. As a result, the diversity of the class GOVERNMENT_TITLE becomes large, and the algorithm assigns the attribute BROTHER to the class PRESIDENTS instead of GOVERNMENT_TITLES. However, GOVERNMENT_TITLES is the parent class of PRESIDENTS and should be a better class (though not the ultimate assignment) for BROTHER. A possible solution is to consider the frequency of mentions for classes in the computation of the diversity index.

9. INTERPRETING WEB TABLES

Ultimately, Biperpedia is useful if it can improve search applications. In this section we show that Biperpedia considerably improves our ability to recover the semantics of Web tables.

There are millions of high-quality HTML tables on the Web with very diverse content. In addition to answering many user queries on search engines, combining these tables in interesting ways can yield novel data sets and insights. Several recent efforts have focused on harnessing such tables [1, 4, 17, 25, 26]. One of the major challenges with Web tables is to understand the attributes that are represented in the tables. For example, the table in Figure 8 displays the GROSS TONNAGE and LOAD CAPACITY of SHIPS. In this section, we show how to *interpret* a Web table with Biperpedia. Interpreting a Web table refers to the process of attaching with it Biperpedia attributes. We may attach a Biperpedia attribute to a specific column (which we refer to as *column mapping*), or attach an attribute to the entire table (*table mapping*) if we cannot pinpoint the precise column, or if the attribute name is not represented by a single column. If we can attach Biperpedia attributes to a table, then the search engine can give much higher weight to the page containing the table when a keyword query contains those attributes, since that attribute plays a more significant role on the page than arbitrary appearances elsewhere. As a result, the table can be retrieved for keyword queries more precisely.

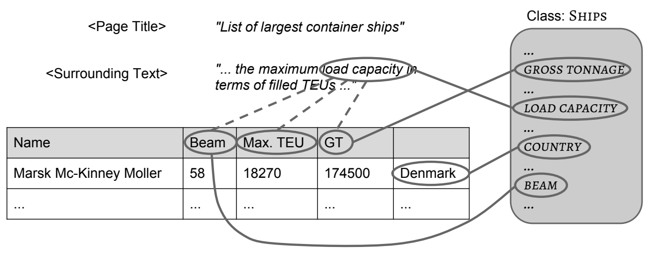

Figure 8: Column and table mappings.

Interpreting Web tables also serves as an indication for the recall of Biperpedia. In particular, we show that with Biperpedia we can increase the number of tables we interpret by more than a factor of 4 compared with Freebase.

9.1 Mapping Algorithm

We describe our algorithm for computing column and table mappings, which, in principle, is a variant on the schema matching problem for which there is a plethora of literature [8]. The main aspect distinguishing our problem setting from standard schema matching is that we are not given as input a pair of schemas and their corresponding tables. Instead, the attributes that best describe the table may be in the surrounding text or the title of the page. Surrounding text is particularly important for interpreting Web tables that have no schema. Hence, our algorithm's task is to match Biperpedia attributes to either column headers or token n-grams in the surrounding text and page title. We proceed in two steps.

(1) Preprocess: Given a table with headers $H_1 \dots, H_n$, and a header $H \in H_1 \dots, H_n$, we create a set of strings S_H with which to match. The set includes H as well as:

- if H has more than 3 tokens, we add to S_H all 2-grams and 3-grams in H.
- if H includes the token `or` or `and`, we add to S_H the text preceding and following that token.
- if the values in H's column are all members of a class C, we add the name of C to S_H.

We also add the following strings to S_H, but matches to these are considered only if we cannot find matches to the above:

- if H has multiple tokens, add its acronym by concatenating the first letters of its words.
- we add the root of the syntactic parse of the attribute (e.g. POPULATION is the root of RURAL POPULATION).

We also create a table-level set of strings T from all 2-grams and 3-grams in the text in the title of the page or the text immediately surrounding the table.

(2) Match: We assume that every table is associated with a class C in the hierarchy that describes the entities in its subject column. If we do not have a subject column, or our prediction of the class is of low confidence, we ignore the table.

We consider all the attributes that Biperpedia has for the class C and try to match their names with the elements of T and of $S_{H_1}, \dots, S_{H_n}$. We use Jaro-Winkler string similarity to find approximate matches. If we find matches in one of the S_H's we output them as column matches, and if we find matches in T we output them as table matches.

Example: In Figure 8, the class of the subject column is SHIPS. The column "Beam" maps directly to the attribute BEAM, and the column "GT" maps to GROSS TONNAGE as an acronym. During the preprocessing, the fourth column will be annotated with the label COUNTRY and will map to the attribute COUNTRY. The attribute LOAD CAPACITY will be output as a table mapping. In fact, the columns "Beam", "Max. TEU", and "GT" do describe the load capacity of a ship. The example in Figure 9 illustrates a case where Biperpedia finds useful mappings for a table with *no* schema.

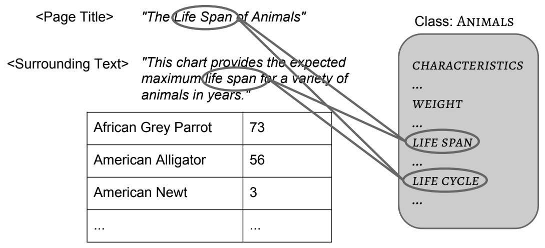

Figure 9: Table mappings for a table with no schema.

9.2 Evaluation

We evaluate the accuracy of our mapping algorithm on approximately 200 tables extracted from the Web. We chose tables that have relatively high quality as opposed to ones in which HTML tables are used to render non-tabular content. We used the subject-column annotator of WebTables and the class associated with it [24]. In addition to the full set of tables, we experiment with (1) a set of Wikipedia tables that are typically of higher quality and have more columns, and (2) a set of tables with no schema (i.e., a traditional schema matching algorithm would return nothing).

To evaluate column matching, we measure precision (the percent of correct matches) and recall (the fraction of columns that got correct mappings). However, not all columns have the same importance and many of the benefits of our mappings are in the table mappings. To measure that, we posit that each table has one or more *representative* attributes. Intuitively, these are the attributes that should cause the table to be retrieved for a query because they capture the essence of the table. For example, the attribute LOAD CAPACITY can be considered representative for the table in Figure 8 while LIFE SPAN is representative for the table in Figure 9. The representative attributes need not correspond to a single column in the table. We asked our evaluators to judge whether the mappings capture what they considered the representative attributes of a table.

9.2.1 *Interpretation Quality*

Table 9 shows the results of the quality evaluation, based a majority voting of 3 evaluators. The **Representative** column shows the number of tables for which at least one correct representative attribute was found. The **Overall (P/R)** column shows the average precision/recall over all mappings. The **Avg. P/R per table** columns compute the precision/recall per table and then averages over all the tables. For the Full dataset, 46% of all mappings are correct (51% per table), and 75% of all the columns are covered by an attribute (78% per table). We are able to capture the essence of the tables using representative attributes 82% of the time.

Dataset	#tables	Representative	Overall (P/R)	Avg. P/R per table
Full	193	158 (82%)	0.46 / 0.75	0.51 / 0.78
Wikipedia	102	83 (81%)	0.44 / 0.73	0.47 / 0.75
No Schema	27	21 (78%)	0.44 / 0.58	0.46 / 0.57

Table 9: Mapping quality.

Considering the Wikipedia tables alone, the precision and recall are slightly lower for two reasons. First, the Wikipedia tables tend to have more complicated surrounding text leading to some false mappings. This shortcoming can be addressed by using IR techniques to compute weights for words in the surrounding text based on their importance. Second, the Wikipedia tables tend to have more columns and therefore it is harder to map them all correctly.

Finally, we considered a set of tables that have no schema row, or that WebTables was not able to identify a schema row. For these tables, no conventional schema matching technique can provide any interpretation. In a sense, these are the hardest tables to interpret in the corpus. By generating table mappings, however, we are still able to find representative attributes for 78% of the tables and obtain significant precision and recall values, albeit slightly lower than the Wikipedia tables. This experiment demonstrates that with a large ontology of attributes, we are able to find the ones that describe Web tables even when they have no schema.

9.2.2 *Comparison with Freebase*

The next question is how many of these mappings are due to Biperpedia versus the attributes that were already in Freebase. The results in Table 10 show that the vast majority are due to the additional attributes of Biperpedia. The first set of columns shows the number of mappings to Biperpedia attributes, the number that were mapped to Freebase attributes, and the ratio between them. For example, when mapping the Full dataset, 142 of the 807 correct mappings are to Freebase attributes, so Biperpedia increases the coverage by a factor of 5.7 compared to Freebase. The second set of columns show these numbers for mappings to representative attributes. For the representative attributes the improvement factor is even higher because the representative attributes are more complicated and thus less likely to be found in Freebase. For the tables without schema, the relative coverages are even higher.

Dataset	Mappings			Rep. Attributes		
	All	Freebase	Rel. Cov.	All	Freebase	Rel. Cov.
Full	807	142	5.7	188	23	8.2
Wikipedia	412	95	4.3	99	12	8.3
No Schema	67	3	22	25	0	n/a

Table 10: Relative coverage compared to Freebase attributes.

9.2.3 *Error Analysis*

The top part of Table 11 shows that the two most frequent causes of error of our algorithm are the noisy token n-grams in the surrounding text and page title that are mapped to attributes unrelated to the HTML table. We can reduce this error by using better IR/NLP techniques for prioritizing the n-grams that are more “relevant” to the table. The last cause is the various incorrect string matchings against column headers using query expansion. Here, a more judicious usage of query expansion may reduce the errors.

Cause	Count
Incorrect Mappings	
Noisy surrounding text	400 (41%)
Noisy page title	308 (32%)
Column header match error	261 (27%)
Missed Representative Attributes	
Table is too specific	21 (60%)
Not enough information	5 (14%)
Evaluator disagreement	4 (11%)
Biperpedia too small	3 (9%)
Missed relevant phrases in context	2 (6%)

Table 11: Interpretation error analysis.

The bottom half of Table 11 considers the errors in detecting representative attributes. The largest cause of error is when the table description involves another constant (e.g., the winners of the Ballon D’Or Award in different years). It is conceivable that our class hierarchy should contain a class BALLON D’OR WINNERS, but no matter how detailed, the class hierarchy will always have gaps. The second cause is when there is simply not enough information in the column headers or context to make a comparison with attributes. Here, external links into the page may provide additional evidence. The third cause is where the three evaluators do not agree on any representative attributes, but some of the attributes are indeed representative. The fourth cause, insufficient coverage of Biperpedia, applies only in 9% of the cases. The last cause is when the attribute name is not a 2-gram or 3-gram and more sophisticated noun-phrase recognition is required.

10. RELATED WORK

We have already touched on several related works throughout the paper. We focus here on other attribute extraction efforts. Pasca et al. [21, 22] were the first to explore the use of query stream data to generate attributes for entities, and their main result was that the query stream yields 45% more accurate extractions of attributes than text. While our results are consistent with that observation, we go further and show that the query stream can be used to seed the extractions from text. In addition, we address the problems of misspelling and synonymy that arise with extraction from the query stream and other sources. The scale of Biperpedia is considerably larger than previous efforts. In particular, the precision they report at rank 50 (which is the largest rank they consider) is slightly lower than what we report at rank 1000. Finally, we show that a broad collection of attributes is beneficial for interpreting HTML tables.

Lee et al. [16] addresses a related problem of determining how *typical* a class C is given an attribute A, or how typical an attribute A is given a class C. In contrast, we are interested in building an ontology with attributes, where we identify synonyms, misspellings, and relationships between pairs of attributes. They also have a pipeline for extracting attributes from queries and from Web text, but there are several key differences. First, they use two patterns to extract attributes from text, while we have shown that to obtain a sizeable and high-quality ontology we need to employ hundreds of patterns. Second, they extract attributes independently from the sources, while we use the high-quality extractions from the query stream to seed our text extraction. They also perform *concept level* extractions which we do not, e.g., using patterns such as *the acidity of wine* to extract attributes for the class WINES. The precision they report at rank 50 is the same level we report at rank 1000. Lee et al. mention interpreting Web tables as a motivation for their work, but do not report applying their attributes to this task.

11. CONCLUSIONS

We described Biperpedia, an ontology of binary attributes that extends Freebase with extractions from the query stream and from Web text. The key idea underlying our extraction is to use the high-quality attributes from Freebase and the query stream to seed extraction from text. We demonstrated Biperpedia's utility by showing that it enables interpreting over a factor of 4 more Web tables than is possible with Freebase. We are currently adding high-quality attributes from Biperpedia to Freebase.

In addition to improvements and extensions to our class hierarchy and to the algorithms for resolving strings to entities that immediately benefit Biperpedia, we are pursuing two main directions. First, we are developing methods for classifying different relationships between pairs and attributes and discovering them from Web text. Second, we are interested in mining a *grammar for complex attribute names*. For example, we would like to be able to recognize that INCREASE IN TOTAL ASIAN POPULATION describes a change in the attribute ASIAN POPULATION. Such interpretation is crucial for understanding a large portion of high-quality Web data.

Finally, from the perspective of a search application, our algorithms can be applied to any query stream with possibly different results. For example, applying our approach on the query stream from mobile devices will emphasize different attributes than the general query stream, and create an ontology more suited to the search needs of that traffic.

Acknowledgements

We thank Manas Joglekar for his help in evaluation, and we thank Tara McIntosh, Fernando Pereira, and Dekang Lin for comments on earlier versions.

12. REFERENCES

[1] M. D. Adelfio and H. Samet. Schema extraction for tabular data on the web. *PVLDB*, 2013.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*, pages 722–735, 2007.

[3] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*, pages 1247–1250, 2008.

[4] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *PVLDB*, 1(1):538–549, 2008.

[5] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[6] C. Chambers, A. Raniwala, F. Perry, S. Adams, R. R. Henry, R. Bradshaw, and N. Weizenbaum. Flumejava: easy, efficient data-parallel pipelines. In *PLDI*, pages 363–375, 2010.

[7] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *WWW*, pages 325–332, 2002.

[8] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[9] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open information extraction: The second generation. In *IJCAI*, pages 3–10, 2011.

[10] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545, 2011.

[11] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[12] J. R. Finkel, T. Grenager, and C. D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, 2005.

[13] A. Haghighi and D. Klein. Simple coreference resolution with rich syntactic and semantic features. In *EMNLP*, pages 1152–1161, 2009.

[14] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545, 1992.

[15] J. Lee, J.-K. Min, and C.-W. Chung. An effective semantic search technique using ontology. In *WWW*, pages 1057–1058, 2009.

[16] T. Lee, Z. Wang, H. Wang, and S.-W. Hwang. Attribute extraction and scoring: A probabilistic approach. In *ICDE*, pages 194–205, 2013.

[17] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *PVLDB*, 3(1):1338–1347, 2010.

[18] Mausam, M. Schmitz, S. Soderland, R. Bart, and O. Etzioni. Open language learning for information extraction. In *EMNLP-CoNLL*, pages 523–534, 2012.

[19] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL*, pages 1003–1011, 2009.

[20] N. Nakashole, G. Weikum, and F. M. Suchanek. Patty: A taxonomy of relational patterns with semantic types. In *EMNLP-CoNLL*, pages 1135–1145, 2012.

[21] M. Pasca and B. V. Durme. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI*, pages 2832–2837, 2007.

[22] M. Pasca, B. V. Durme, and N. Garera. The role of documents vs. queries in extracting class attributes from text. In *CIKM*, pages 485–494, 2007.

[23] T. Tran, P. Cimiano, S. Rudolph, and R. Studer. Ontology-based interpretation of keywords for semantic search. In *ISWC/ASWC*, pages 523–536, 2007.

[24] P. Venetis, A. Y. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the web. *PVLDB*, 4(9):528–538, 2011.

[25] J. Wang, H. Wang, Z. Wang, and K. Q. Zhu. Understanding tables on the web. In *ER*, pages 141–155, 2012.

[26] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *SIGMOD Conference*, pages 97–108, 2012.

[27] L. Yao, S. Riedel, and A. McCallum. Collective cross-document relation extraction without labelled data. In *EMNLP*, pages 1013–1023, 2010.