Spotting Facts in the Wild

Tomasz Tylenda
Max-Planck-Institute for Informatics
Saarbrücken, Germany
ttylenda@mpi-inf.mpg.de

Yafang Wang
School of Computer Science and Technology, Shandong University, Jinan 250101, China
yafang.wang@sdu.edu.cn

Gerhard Weikum
Max-Planck-Institute for Informatics
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

Knowledge bases have become key assets for many tasks related to search and analytics. Retrieving textual evidence for the facts about an entity – in news, social media, or other web contents – is important for assessing the truth of statements, gathering statistics about the saliency of facts, and as a basis for summarizing documents or entire corpora. This paper addresses the problem of spotting occurrences of known facts in textual documents, reporting the presence or absence of facts. Our solution is based on rich dictionaries of paraphrases for entity names and binary relations, and uses heuristic rules for high precision and recall. We evaluated our method by finding facts from Freebase in a corpus of stylistically diverse biographies from the web.

1. INTRODUCTION

1.1 Motivation

Knowledge bases (KBs), like DBpedia, Freebase, or Yago, contain hundred millions of facts about entities, in the form of subject-predicate-object (SPO) triples. They are constructed by automated information extraction (IE) from structured data (e.g., Web tables or Wikipedia infoboxes) and unstructured text (e.g., news articles or social media).

As IE can hardly ever be perfect, KBs contain erroneous facts. To fix these errors, KBs rely on human curation (e.g., crowdsourcing). However, for a human to assess the correctness of a fact, it is crucial to see the fact in context, by looking at a variety of news, biographies, or postings in online communities. For example, to realize that the SPO statements `Sarah_Palin bornIn Hope_(Alaska)` and `Neil_Young diedOn 25-Aug-2012` are wrong, one needs to explore texts about these claims or alternative versions.

To support this curation process, we consider the reverse of IE, called *fact spotting* in this paper. Given an SPO triple in a KB, we aim to retrieve text documents that contain supporting evidence for the triple. Search engines give us a set of documents about the S entity for an SPO triple. This set may contain spurious matches, but is a starting point for finding evidence for the triple. This paper focuses on the issue of finding the evidence for the triple within a given document or reporting its absence.

Notions of reverse IE have been proposed earlier [5]. However, these methods are limited to the text sources from which a fact was extracted (by remembering this provenance and relying on the prior processing). In contrast, our method finds fact occurrences in textual documents that were not used for IE at all. This includes occurrences that may be too difficult for IE anyway, due to highly complex sentence structures, sophisticated co-references, or other ambiguity.

Fact spotting is beneficial for various purposes:

- *Evidence for fact credibility:* By collecting many occurrences of SPO statements in news, social media and other web contents, a user (or even an algorithm) can more easily assess the statement's truth.
- *Statistics for semantic search and analytics:* Entity search and RDF querying over heterogeneous data often needs to rank results [6, 5, 12, 25]. To this end, it is crucial to have co-occurrence frequencies for SPO components. Such statistics is also vital for named entity disambiguation [9, 29] and for lifting text analytics to the entity level [1, 4, 16].
- *Multi-document summarization:* By understanding which SPO facts are central in a document or an entire corpus of documents, we can automatically generate more informative summaries. The corpus may be a day's or week's news or a month's scholarly publications on a scientist's field of interest.

1.2 Contribution

The task addressed in this paper is as follows. Given an arbitrary text document D about an entity S and a knowledge base K with facts about S, identify the SPO facts from K that appear in D. We assume that S is given. However, this is not a strong assumption: many documents, like news articles or social-media postings are about a main entity (e.g., an event or a person or organization) that is fairly easy to detect by first running a method for named entity recognition and disambiguation (e.g., [29]) and then picking the most frequently or prominently (e.g., in a headline) mentioned entity.

Our approach harnesses dictionaries of paraphrases for entities and for relations [20]. For example, we know that phrase "First Lady" may refer to Michelle Obama (but also to other first ladies) and that "married to" and "on honey-

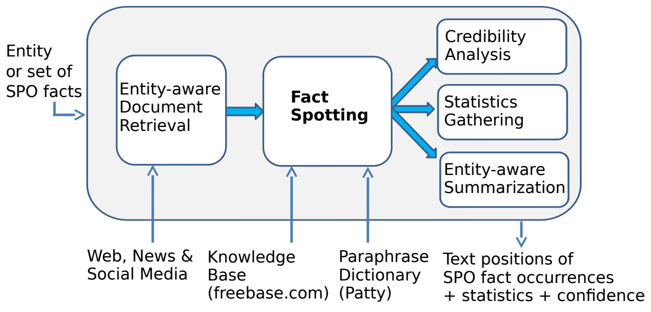

Figure 1: System Overview

moon with" are paraphrases of the `spouseOf` relation. For a given S, we enumerate the combined PO paraphrases for the KB triples that we have about S, and perform partial-matching against the input text. This gives us a superset of fact candidates, which need closer inspection, though. For pruning out false positives, we use rules on the co-occurrence of fact components: either matching P and O, or matching multiple O values for different facts about S within textual proximity.

Although this approach is heuristic in nature, it is powerful and yields fairly high precision and recall. This is demonstrated in experiments with biographies from web pages of different styles and detail.

2. SYSTEM COMPONENTS

The system environment we are working in is depicted in Fig. 1. The input to the fact spotting component is an entity (or a set of facts about it) and an arbitrary document. Our method makes use of a knowledge base and a dictionary of paraphrases to determine which facts occur in the document. The output, in the form of fact positions or aggregated statistics, can be processed in downstream applications, including truthfulness analysis and entity/fact-aware summarization.

2.1 Knowledge Base

The knowledge base used in our system is Freebase(`www.freebase.com`). It stores data as a graph, in which nodes are called topics and correspond to entities or classes. Topics are assigned ids, for example, `/m/02mjmr` for Barack Obama. Types in Freebase are containers for properties, for example, `/people/person/date_of_birth` belongs to type `/people/person`. The values of properties can be literals (e.g. `int`, `datetime`, `string`), references to other topics (`object` type), or special `compound` values, which consist of several fields, for example, `/people/person/spouse_s` contains the person one is married to, it can also contain a start date, optional end date, location of the ceremony, type of union etc.

Note that this data model is quite different from the typically used RDF model of SPO triples. For example, Freebase makes extensive use of nested values of values. We deal with this by breaking down composite facts into triples. Section 3 presents more detail.

2.2 Entity Aliases

Spotting facts requires knowledge of possible labels of entities. Such labels are provided by Freebase in form of `/type/object/name` and `/common/topic/alias` relations. The former is the unique, canonical name, similar to article titles in Wikipedia (e.g. Victoria Beckham), while the latter contains alternative names of the entity (e.g. Victoria Caroline Adams, Posh Spice).

We observed that initially our fact spotting method missed some facts, because Freebase aliases did not match the entity names used in document. For example, in one of the biographies of Arnold Schwarzenegger, *University of Wisconsin-Superior* is referred to as *University of Wisconsin* – a name which is not known to Freebase. We solve this problem by expanding the set of entity aliases with shorter versions created by dropping a single word from labels which are longer than three words. To avoid incorrect matches, we add the constraint that the new, shorter labels must not be the same as any label of a known entity. For instance, we expand the set {*Patrick Arnold Schwarzenegger*} (the son of the famous bodybuilder) with the derived labels {*Patrick Schwarzenegger, Patrick Arnold*}. Our derived set does not include *Arnold Schwarzenegger*, because this would clash with the name of another entity (his father). Name clashes still occur among labels which were not derived; for example, Clint Eastwood is the name of both the actor and his father.

2.3 Relation Patterns

Spotting the predicate of an SPO triple requires that we know which phrases can be used to express it. For example, words like *died, death*, in proximity of a city name means that a sentence expresses `diedIn` and not `bornIn` relation. If a set of relations were small, a list of such phrases could be constructed manually. While this approach could lead to high precision, it is not scalable and it is also likely to have poor recall. A better solution is to use an existing repository of relational paraphrases, like PATTY [20]. It contains 127811 paraphrases of 225 DBpedia relations and 43124 paraphrases of 25 YAGO2 relations with overall precision of 76%.

PATTY patterns contain: *1*) words, *2*) POS tags, which are placeholders for words of the right part-of-speech, *3*) wildcards, denoting any sequence of words and *4*) <type>, which is a placeholder for an entity of type <type>. For example, a pattern *<person>'s [adj] voice * <song>* matches "Amy Winehouse's soft voice in Rehab".

Our system uses a simplified form of PATTY pattern. First, we drop the leading and trailing wildcards such as *[det]* and filter out the patterns which still contain them in the middle. The result is a set of word sequences. Then, to speed up matching, we eliminate redundancy. For instance, *studied with* makes *who studied with* redundant, therefore we drop those patterns which are superstrings of other patterns. Our notion of superstring respects words boundaries, so *child* is not a substring of *children.*

PATTY knows only paraphrases for relations from DBpedia and YAGO; so to use it with our data set we had to map them to Freebase relations. Since the number of relations is not prohibitively large we performed this step manually. An alternative and feasible approach would be to re-run the PATTY construction algorithms with Freebase relations. In addition to PATTY, we also use Freebase relation names to generate patterns. Since all relation names follow a similar structure, we simply take two trailing parts; for example `/theater/theater_role/play` yields `theater role` and `play`.

Algorithm 1: Fact Spotting.

```
atoms = $\emptyset$            // atomic parts of all facts
for $f \in facts$ do
    atoms = atoms $\cup$ atoms(f)
for $a \in atoms$ do
    matches[a] = $\emptyset$ // maps atoms to text positions
    if $a$ is an object then
        for $l \in getObjectLabels(a)$ do
            matches[a].addAll(occurrences(l, text))
    else if $a$ is predicate then
        for $p \in pattyPatterns(a)$ do
            matches[a].addAll(occurrences(p, text))
    else if $a$ is value then
        matches[a] = occurrences(a, text)
foundFacts = $\emptyset$
for $f \in facts$ do
    if predicate and object found in proximity then
        foundFacts.add(f)
    if object and a sibling found in proximity then
        foundFacts.add(f)
return foundFacts
```

3. FACT SPOTTING

3.1 Freebase Data Processing

The basic building blocks of facts are simple values and predicate names. Simple values are literals and references to Freebase topics. Predicate names occur in two places: as relations names, and as attribute names in compound values. We refer to simple values and predicate names as atoms.

A compound value is a non-empty sequence of identifiers and predicate names $\{\langle name, value\rangle\}_{i=1}^{k}$. A fact is an $\langle S, P, O\rangle$ triple consisting of a subject, predicate name and object. The subject is always a reference to a Freebase topic, whereas the object is a simple or compound value.

A compound fact $\langle S, P, \{\langle name, value\rangle\}_{i=1}^{k}\rangle$ can be decomposed into a set of simple facts: $\{\langle S, P_i, value_i\rangle\}_{i=1}^{k}$. For example a fact `David_Beckham spouse {<from, 1999>, <spouse, Victoria_Beckham>}` will be decomposed into two facts: `David_Beckham spouseFrom 1999` and `David_Beckham spouse Victoria_Beckham`. Decomposing facts simplifies both spotting and evaluation of the results. In general, such decomposition is not reversible. If a person is married more than once, we would not be able to tell which date refers to which spouse. We fix this problem by simply remembering which triples belong together. In the following sections, we call two atoms *siblings* if they occur in the same compound value (this also applies to attribute names and values).

3.2 Spotting Algorithm

Our fact spotting method assumes that it is presented with a text document about a single entity and that all the facts stated in the document are about this entity. While biographies do contain facts about other entities, they do not pose a problem, because they are uncommon, and different from the facts about the entity we are interested in. The assumption simplifies spotting SPO triples to spotting PO pairs.

For efficiency, fact spotting is divided into two phases. We noticed that facts often share their atomic parts, such as relation names and objects, therefore we first find the matches to atomic objects, and then use them to find matches of whole facts.

The spotting algorithm is presented in Alg. 1. We start by extracting the atoms from the facts about the current entity. The atoms are the simple values, relation names and compound attribute names. Subsequently, we find the occurrences of facts in text, and store their locations.

In the simplest case we find matches to the predicate P and the object O, and if such matches occur close enough in the text, we assume that the SPO fact was found. In our implementation "close enough" means that the beginnings of matches of the predicate and object value have to be at most 200 characters apart. Oftentimes, it is possible to find an occurrence of the object, but we do not have a match for the relation. In such a case we consider the presence of a sibling atom as a cue that we indeed spotted the fact.

4. EVALUATION

4.1 Setup

We collected a set of 233 names of football (i.e., soccer) players and actors. Queries following a pattern `<name> biography` were submitted to the Bing search engine. In the results we identified a set of 10 web sites that feature biographies of famous people, with highly varying styles across sites. We made sure that the biographies are not copies of Wikipedia articles and are not simple lists of facts without narrative structure. We kept only the results from those web sites and obtained a set of 848 biographies of 233 entities.

For evaluating recall of fact spotting we manually annotated a set of biographies of 5 football players and 5 actors. The annotations were created with respect to Freebase, that is, we read the biographies and selected the facts or parts of facts which were indeed mentioned in the text. This constitutes the *ground-truth* against which our method's results are compared. The fact spotting algorithm decomposes Freebase fact to obtain simple SPO triples, and we follow the same procedure with the ground-truth annotations.

4.2 Results

The precision, recall, and F1 scores of our fact spotting method are reported in Tbl. 1. The aggregated values over all test entities are micro-averages, calculated by comparing the spotted facts against the overall set of facts for all test cases. Tbl. 1 also shows the absolute numbers of facts in Freebase (KB), in the ground-truth annotations (GT), and found by our method (correct and incorrect).

Our F1 scores are close to 60%; we consider this result to be fairly good, given the difficulty of the task. Our method is also fairly fast: processing all 848 biographies (i.e., not just the annotated samples for which we have ground-truth) took only 134 seconds in total.

By looking only at absolute numbers, Freebase seems to contain way more facts than typical biographies on the web. However, this is a bit misleading, as many facts in Freebase have compound values and contain non-salient attributes, e.g., the fact that Elizabeth Taylor received the Oscar for Cat on a Hot Tin Roof in 1958 has the additional information that it was during 31st Academy Awards ceremony. Our decomposition into SPO triples thus reflects a large number

Entity	Domain	Prec. %	Recall %	F1	In KB	In GT	Found
Arnold Schwarzenegger	movies.msn.com	48.9	48.9	0.489	298	45	45
Clint Eastwood	movies.msn.com	51.0	53.1	0.520	306	49	51
David Beckham	www.history-of-soccer.org	65.9	65.9	0.659	265	41	41
David Beckham	www.thebiographychannel.co.uk	49.2	62.5	0.550	265	48	61
Elizabeth Taylor	www.thebiographychannel.co.uk	76.1	77.8	0.769	245	45	46
Gianluigi Buffon	www.history-of-soccer.org	75.0	66.7	0.706	58	18	16
Jodie Foster	movies.msn.com	53.8	46.7	0.500	204	30	26
Oliver Kahn	www.history-of-soccer.org	84.6	42.3	0.564	68	26	13
Pelé	www.history-of-soccer.org	50.0	83.3	0.625	105	6	10
Pelé	www.biography.com	66.7	71.4	0.690	105	14	15
Woody Allen	www.thebiographychannel.co.uk	50.0	17.6	0.261	344	17	6
Zinedine Zidane	www.biography.com	55.2	69.6	0.615	132	23	29
Aggregated		58.8	58.3	0.585	2395	362	359

Table 1: Precision and recall on annotated data.

of facts. At the same time we observed that many of Elizabeth Taylor's movies are not captured in Freebase at all, but are prominently mentioned in biographies.

4.3 Discussion

We analyzed the cases where our fact spotting method falls short. One of the most common causes of false positives (spotting a fact which is not in the text) are incorrect patterns for relations. For instance, PATTY's patterns for award nomination contain the word *film* which is fairly common in biographies of actors. If we spot it next to a film name it makes us mistakenly believe that the fact is stated there. False negatives (missing out on facts that actually occur), on the other hand, are often caused by missing labels; for instance, `Arnold_Schwarzenegger nationality Austria` is missed because the biography uses the adjective *Austrian*, which in not in the label set for Austria.

5. RELATED WORK

IE and Relational Patterns: There is ample work on information extraction for constructing knowledge bases (see the tutorials [3, 24] and references given there). Many methods make use of textual patterns that express relations, such as "on honeymoon with" for the spouseOf relation. However, there is only few projects that have systematically compiled pattern repositories [7, 20, 18, 11]. Among these, the largest publicly available resource is PATTY [20], hence our choice for this current paper.

Reverse IE: The most notable work on providing provenance sources for the facts that result from IE is [17]. The method works by retaining the text documents from which facts were extracted. This forms the basis for interactive exploration and corroboration of a knowledge base. "Witness documents" for a fact are ranked based on statistical techniques. However, this approach is limited to the documents that were indeed processed by the IE workflow; contents retrieved "from the wild" and never seen by the IE process cannot be handled. There is also work on explaining relationships between entities [8]. However, the underlying information is the knowledge base itself; there is no usage of textual contents.

Truth Finding: Motivated by the goal of assessing the truth of doubtful or debated statements in Web contents, various projects have developed methods for identifying source documents that support or refute a given statement [28, 15, 21, 19, 14]. Although fact spotting can also contribute to this end, it is quite different from that prior work on truth finding. Truth finding takes a single SPO fact (or other form of statement) as its input and computes statistical aggregates over the retrieved sources, whereas we aim to find the exact positions of as many facts as possible in a given text.

Question Answering: The problem of answering questions has been addressed by various systems. One of the most successful is IBM's Watson [10]. One particular subproblem of QA is answer validation, that is deciding whether a text snippet supports a given answer to a question [22].

Entity and Fact Ranking: The information retrieval community has intensively worked on entity ranking for search results (e.g., [2]). There is much less work on ranking facts, one of the most notable projects being [6] in the context of applying IR techniques to queries over RDF graphs. The ranking was based on automatically compiled statistics for SPO facts, derived from the full text of Wikipedia. However, [6] simplified this task by discovering and counting only individual entities in their S and O roles, and then combining these counts by assuming independence. The P component of an SPO fact was not considered at all.

Entity Summarization: The task of summarizing an entity (or a document about an entity) in terms of its most salient facts has been addressed from a variety of angles like visualization and interactive exploration [25, 23], faceted search [13], the generation of timelines [26], and the generation of knowledge excerpts [27]. In all of this work, the set of relevant facts is given upfront; there is no need for discovering textual statements about facts in documents.

6. CONCLUSION AND FUTURE WORK

We presented a viable solution for the fact spotting problem, the reverse of information extraction. Our method is relatively simple and very light-weight; this enables scaling it out to Web-scale collections of highly heterogeneous text contents.

Our future work includes investigating the use of fact spotting for automated truth assessment and for entity-aware summarization of documents or entire text collections. To this end, we study models for scoring the confidence in spotting a fact.

7. REFERENCES

[1] R. A. Baeza-Yates, J. Masanès, and M. Spaniol. Proceedings of the 4th temporal web analytics workshop. In *TempWeb'14*, 2014.

[2] K. Balog, Y. Fang, M. de Rijke, P. Serdyukov, and L. Si. Expertise retrieval. *Found. Trends Inf. Retr.*, 6(2–3):127–256, Feb. 2012.

[3] D. Barbosa, H. Wang, and C. Yu. Shallow information extraction for the knowledge web. *ICDE*, 0:1264–1267, 2013.

[4] T. Cheng, K. Chakrabarti, S. Chaudhuri, V. R. Narasayya, and M. Syamala. Data services for e-tailers leveraging web search engine assets. In *ICDE*, pages 1153–1164, 2013.

[5] S. Elbassuoni, K. Hose, S. Metzger, and R. Schenkel. Roxxi: Reviving witness documents to explore extracted information. *Proc. VLDB Endow.*, 3(1-2):1589–1592, Sept. 2010.

[6] S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, and G. Weikum. Language-model-based ranking for queries on rdf-graphs. In *CIKM*, pages 977–986, Hongkong, China, 2009.

[7] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP '11*, Edinburgh, Scotland, UK, July 27-31 2011.

[8] L. Fang, A. D. Sarma, C. Yu, and P. Bohannon. Rex: Explaining relationships between entity pairs. *PVLDB*, 5(3):241–252, Nov. 2011.

[9] P. Ferragina and U. Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM*, pages 1625–1628, New York, NY, USA, 2010.

[10] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. A. Welty. Building watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010.

[11] R. Gupta, A. Halevy, X. Wang, S. Whang, and F. Wu. Biperpedia: An ontology for search applications. In *PVLDB*, 2014.

[12] G. Kasneci, S. Elbassuoni, and G. Weikum. Ming: mining informative entity relationship subgraphs. In *CIKM*, pages 1653–1656, New York, NY, USA, 2009.

[13] C. Li, N. Yan, S. B. Roy, L. Lisham, and G. Das. Facetedpedia: Dynamic generation of query-dependent faceted interfaces for wikipedia. In *WWW*, pages 651–660, New York, NY, USA, 2010. ACM.

[14] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava. Truth finding on the deep web: is the problem solved? In *PVLDB'13*, pages 97–108, 2013.

[15] X. Li, W. Meng, and C. Yu. T-verifier: Verifying truthfulness of fact statements. In *ICDE '11*, pages 63–74, Washington, DC, USA, 2011.

[16] Y. Li, F. Reiss, and L. Chiticariu. Systemt: A declarative information extraction system. In *ACL (System Demonstrations)*, pages 109–114, 2011.

[17] S. Metzger, S. Elbassuoni, K. Hose, and R. Schenkel. S3k: Seeking statement-supporting top-k witnesses. In *CIKM'11*, pages 37–46, Glasgow, UK, 2011.

[18] A. Moro and R. Navigli. Wisenet: Building a wikipedia-based semantic network with ontologized relations. In *CIKM '12*, pages 1672–1676, New York, NY, USA, 2012. ACM.

[19] N. Nakashole and T. M. Mitchell. Language-aware truth assessment of fact candidates. In *ACL*, 2014.

[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] J. Pasternack and D. Roth. Latent credibility analysis. In *WWW '13*, pages 1009–1020, 2013.

[22] Á. Rodrigo, A. Peñas, and F. Verdejo. Overview of the answer validation exercise 2008. In *CLEF*, pages 296–313, 2008.

[23] G. Sobczak, M. Pikuła, and M. Sydow. Agnes: A novel algorithm for visualising diversified graphical entity summarisations on knowledge graphs. In *ISMIS*, pages 182–191. Springer Berlin Heidelberg, 2012.

[24] F. Suchanek and G. Weikum. Knowledge harvesting from text and web sources. *ICDE*, pages 1250–1253, 2013.

[25] M. Sydow, M. Pikuła, and R. Schenkel. The notion of diversity in graphical entity summarisation on semantic knowledge graphs. *J. Intell. Inf. Syst.*, 41(2):109–149, 2013.

[26] T. A. Tuan, S. Elbassuoni, N. Preda, and G. Weikum. Cate: context-aware timeline for entity illustration. In *WWW (Companion Volume)*, pages 269–272, 2011.

[27] T. Tylenda, M. Sozio, and G. Weikum. Einstein: Physicist or vegetarian? Summarizing semantic type graphs for knowledge discovery. In *WWW 2011*, pages 273–276, Hyderabad, India, 2011. ACM.

[28] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. In *KDD '07*, pages 1048–1052, New York, NY, USA, 2007. ACM.

[29] M. A. Yosef, J. Hoffart, I. Bordino, M. Spaniol, and G. Weikum. Aida: An online tool for accurate disambiguation of named entities in text and tables. *PVLDB*, 4(12):1450–1453, 2011.