CALCULATING SEMANTIC SIMILARITY BETWEEN FACTS

Sergey Afonin, Denis Golomazov
Institute of Mechanics, Moscow State University, Michurinskij av. 1, Moscow, Russia
serg@msu.ru, denis.golomazov@gmail.com

Keywords: semantic similarity, events.

Abstract: The present paper is devoted to the calculation of semantic similarity between facts. A fact is considered as a single sentence including three parts, “what happened”, “where” and “when”. We propose a function calculating the semantic similarity and provide some experimental results justifying it.

1 Introduction

People use search engines to find information on any subject. Sometimes they search for facts. A fact is something that has happened in a certain place at a certain time. Why do people need information on facts? If they just want to check news they use news portals, not search engines. That's true, but every once in awhile people want to find additional information about a fact. For example, they have heard from a friend about some fact and want to read more about it. Or it can be a journalist creating a dossier for a person and they want to check if some rumours are true facts. Moreover, fact search provides a basis for task of revealing relations between facts and other data mining problems.

The problem is that fact search is a type of search that search engines currently cannot always handle properly. Let us consider some local news, for example, bank robbery in Livermore, California in July 2008. A user wants to find some additional information on the fact. Suppose they forgot the name of the city and ask Google with the query *California bank robbery in July 2008.* No relevant results are on the first page. The same situation occurs if we ask Google with the query *Livermore bank burglary in July 2008.* This probably happens due to the not great importance of the fact, and low page rank of the portal on which the news had been posted. The search engine ranks the page describing the event relatively low since there was not exact matching, and there were a lot of robberies in California during that period. Another problem of modern search engines is the inability to perform almost any analytics. For example, we can not get neither a list of all events in some city for some period, nor a list of cities in which bank robberies occured in July 2008, nor a list of dates on which robberies in California took place.

Fact search appears to be a more complicated task than ordinary keyword search. The main reason is that a fact can be represented in many ways, using synonyms, abbreviations, with some keywords included or omitted. The second reason is that a fact can be inferred basing on information distributed over sentences or even documents.

Let us discuss a virtual system that can perform fact search. It operates as follows. First, it crawls the web and extracts facts from web pages. Second, it lets a user enter a sentence describing a fact and returns semantically similar facts from the database. During this process it somehow calculates the “trust rate” of the fact, i.e. how likely is the fact to be true. To construct such a system, we divide the fact search task into four steps.

- Extraction of facts from large number of texts in natural language.

- Calculating semantic similarity between facts
- Calculating the trust rate of a fact.
- Efficiently perform similar facts search on a large database. For example, this can include development of index structures for facts.

In this paper we focus on the task of calculating semantic similarity between facts. We believe this task to be the cornerstone of the fact search problem. For example, the similar facts search task can be easily, though not efficiently solved with the help of a semantic similarity function and linear search. Having a semantic similarity function for facts, one can apply it for such common tasks as fact classification and clustering. Fact classification and clustering can be used for automatic classification and clustering of small portions of text, such as news headlines, twitter updates, sms etc. It can be also used for forecasting. Suppose some facts have recently occured that are semantically close to war (e.g. terrorism, provocations, threats etc.). This can be automatically considered as a warning of an upcoming war and signalize that appropriate means should be taken. We consider finding consistent patterns and forecasting of events using large-scale datasets as the further development of the system.

2 Related Work

The problem of semantic similarity calculation is a wide one. The objects to calculate similarity between can be single words, groups of words, sentences, or texts. In this paper we consider sentences of a special kind, namely that describe facts. In this section we provide a brief review of papers devoted to semantic similarity calculation task.

Semantic similarity of terms using an ontology is studied in (Maguitman et al., 2005) and (Varelas et al., 2005). The authors of the first paper state that graph-based similarity function is more efficient than traditional tree-based ones. WordNet ontology is used in the second paper.

In (Bollegala et al., 2009) semantic similarity between words is computed using automatically extracted lexical patterns. The method proposed in the paper is stated to outperform all existing web-based semantic similarity measures. Essentially, to calculate similarity, an ontology is being constructed under the hood and then a metric is applied using the ontology. This approach seems very promising and we use a similar approach in the present paper.

In (Turney and Littman, 2005) and (Turney, 2006) an algorithm for choosing analogous pairs of words is provided. For example, for the pair "mason-stone" it yields "carpenter-wood".

A vast number of researches, e.g. (Pantel and Lin, 2002), (Mccarthy et al., 2004), (Purandare and Pedersen, 2004) are devoted to the problem of determining senses of a word and clustering them. A promising paper is (Udani et al., 2005) in which an unsupervised method for clustering noun senses using Web search results is presented. The algorithm allows to determine real-world senses rather than dictionary ones and yields high accuracy (over 80%). A thorough review of algorithms for determining word senses is provided in (Navigli, 2009).

It should be noted that all of the papers mentioned are devoted to the task of calculating similarity between terms and between texts. In paper (Islam and Inkpen, 2008) authors approach the problem of calculating similarity between sentences and short texts. They combine corpus-based methods with string similarity function (namely, a modified version of the Longest Common Subsequence algorithm). A particularly relevant paper is (Li et al., 2006) in which the task of calculating semantic similarity between sentence is considered. The algorithm proposed by the authors uses information from a structured lexical database and from corpus statistics.

To conclude the brief review of related work it should be mentioned that there are a lot of papers considering semantic similarity between terms, texts, and sentences. In this paper we consider sentences of a special kind, namely representing an event, or fact, that occured in some place at some time. We have not found any papers devoted to this particular problem, though some algorithms described in relevant papers can be applied and there exist works on event description detection and classification, e.g. (Naughton et al., 2010).

3 Semantic Similarity Between Facts

We consider facts consisting of three parts: "what happened, where and when", so a fact F is a triple $F = (what, where, when)$. Our goal is a function $S(F_1, F_2)$ that calculates semantic similarity between facts F_1 and F_2. The function S takes values between 0 and 1, and higher value means higher similarity. In this section we discuss properties that such function should satisfy.

First of all, let us note that two facts should be trated similar if all their components are pairwise similar. It seems unlikely that there exists a "universal" semantic similarity function suitbale for all three part, so three separate functions s_t, s_r, and s_n mesauring semantic similarity of wha**t**, whe**r**e and whe**n** parts, respectively, should be defined. The fact similarity fuinction S should mix up values of these three function. Note that in general functions s_t, s_r, and s_n should use all components of the compared facts.

As we assume that one of facts, say F_1, is user's query and that this query describe the same fact as F_2, but with different lexical means, we will classify possible reasons for facts description mismatching.

Synonimy, acronyms, abbreviations etc. Two facts may be described differently due to synonymy, acronyms, abbreviations or a slang. For example, *A theft in X bank* describes the same fact as *A larceny in X bank*, and *armed robbery* may be replaced by a slang word *blagging*.

Underspecification. Quite often desriptions of geographic objects contain specifications like *small-town X* that can be omitted in a query. It seems that in most cases descriptive words like *large* or *small* may be simply dropped without loosing any information about the fact itself. Nevertheless in some cases, such as *small city Moscow*, a discriptive word may be used for distinguishing of the defined object from some other (well-known) object.

Vertical taxonomy relations. By vertical taxonomy relation we mean hyponym-hypernym relation. A user formulating the query may have only a fuzzy knowledge about the fact she is looking for. For example, she may not be aware of the pricese crime hapend, or the date, or place. If a query requests for a robbery in *a small California town* the fact describing a robbery in *Livermore, CA* should be considered as relevant. Similarly, hypernym relation may exist between what-parts of two facts, e.g. *burglary – larceny – crime*, and when-part, e.g. *June – summer*.

We expect that this type of mismatching will be one of the most frequent in real applications.

Horizontal taxonomy relations. This type correspond to the case when a user provided information on the same level of abstraction, but it does not match the fact precisely. Two facts refer to differnt concepts (e.g. *robbery – burglary*), but these concepts share a common hypernym (*crime*). Similarly, toponyms like *Livermore* and *Hartford* may be considered similar if they have similar description (in this example both names correspond to small towns in California). Clearly that not every pair of words having the same hypernym are similar. For example, both *murder* and *stealing* are crimes, but facts *A muder in an X's office* and *A stealing in an X's office* are not similar. This means that some additional constraints should be applied. For example, both words may be required to be similar in the sense of the next type.

General similarity. Both types of taxonomy relations described above are special cases of semantic similarity between terms. We have separeted them into specific classes because if such relations exist then one can expect strong semantic relation between corresponding facts. Nevertheless, the facts may be similar even if taxonomic relations are not present. For instance, one can expect that facts about *robbery of a bank* and *shooting in a bank* are similar.

4 Evaluation

Semantic similarity between two facts $F_1 = (what_1, where_1, when_1)$ and $F_2 = (what_2, where_2, when_2)$ is calculated using the following formula:

$$S(F_1, F_2) = \min\{s_t(t_1, t_2), s_r(r_1, r_2), s_n(n_1, n_2)\}$$

This function simply return the worst mismatch of what-, where-, and when- parts of the two facts. The functions s_t, s_r, s_n are defined in sections 4.1, 4.2, 4.3, respectively. All these functions are context-less: the similarity measure of one part does not depend on other parts of facts.

4.1 The *What* Part

It should be noted that we consider the *what* parts consisting of terms that are single words. First, we propose the following list of possible types of semantic relations between terms.

- *vertical taxonomy relations*, or *hypernymy*, e.g. Livermore, CA – small town in California, July – summer, armed robbery – robbery – larceny – felony – crime);
- *horizontal taxonomy relations*, i.e. terms that have a common direct hypernym, e.g. Livermore – Hartford, robbery – burglary).

- *other semantic relations.*

The formula for calculation of semantic similarity between two terms $what_1$ and $what_2$ is the following.

$$\begin{aligned} s_t(t_1, t_2) \quad = \quad & \max\{s_{vert}(t_1, t_2) \times C_1, \\ & s_{horiz}(t_1, t_2) \times C_2, \\ & s_{stat}(t_1, t_2) \times C_3\}, \end{aligned}$$

where the function s_{vert} calculates the estimation of the fact that one of the terms $what_1$, $what_2$ is a hypernym of the other one. The function s_{horiz} calculates the estimation of the fact that terms $what_1$, $what_2$ have a common hypernym (i.e. they have a horizontal taxonomy relation). The function s_{stat} calculates the semantic similarity between terms $what_1$ and $what_2$ using a statistical method. C_1, C_2, and C_3 are weight coefficients that help implement the idea that vertical taxonomy relation is more important than horizontal taxonomy relation and the latter is more important than "default" semantic similarity calculated statistically. In our experiments we choose $C_1 = 1, C_2 = 0.8, C_3 = 0.8$.

The functions s_{vert}, s_{horiz}, and s_{stat} are defined in sections 4.1.2, 4.1.3, and 4.1.1, respectively.

4.1.1 Statistical Semantic Similarity Function

As function s_{stat} we use *Normalized Google Distance* (Cilibrasi and Vitanyi, 2007) to statistically calculate semantic similarity between individual terms. We did not use Google because of access limits and chose the YahooBOSS API[1].

4.1.2 Hypernyms Estimation

The goal is to estimate likeliness of the fact that one of two terms $what_1$ and $what_2$ is a hypernym of the other one. We use an idea that if one term t_1 is a hypernym of the other term t_2, then hyponyms of t_1 should be semantically close to t_2.

We use the following formula to estimate vertical taxonomy relation between $what_1$ and $what_2$.

$$s_{vert}(t_1, t_2) = \max\{hyper(t_1, t_2), hyper(t_2, t_1)\}, \tag{1}$$

where

$$hyper(what_1, what_2) = \frac{M}{k}, \tag{2}$$

where M is the maximum number of points-neighbours from a single cluster when we classify the point $what_2$ using clustered terms that are candidates for being hyponyms of $what_1$. Let us describe the algorithm in detail.

- We execute the query "* is a $what_1$" in a search engine and take the most frequent nouns from it, i.e. nouns that occur more times than the average number of occurences of nouns in the abstracts retrieved. We have obtained the set B of possible hyponyms of $what_1$.
- We calculate semantic distances between words from B. To accomplish this we use the function s_{stat}.
- We cluster the points from B using the graph cut clustering algorithm.
- We take k nearest neighbours of $what_2$ (again, using the s_{stat} distance function) and calculate M as the maximum number of neighbours from a single cluster.
- We calculate the estimation of fact that $what_1$ is a hypernym of $what_2$ using the formula $hyper(what_1, what_2) = \frac{M}{k}$.
- We calculate $hyper(what_2, what_1)$ using the same procedure and take maximum of two values.

4.1.3 Common Hypernym Estimation

We calculate the estimation of a fact that the terms $what_1$ and $what_2$ have a common hypernym that is semantically close to them, using the following formula.

$$s_{horiz} = \max_{h_1 \in H_1, h_2 \in H_2} s_{stat}(h_1, h_2), \tag{3}$$

where H_1 and H_2 are sets of possible hypernyms of $what_1$ and $what_2$, respectively. The sets H_1 and H_2 are constructed as follows.

- We execute the query "A is a *" in a search engine and take the nouns that occur in search results more often than the average value of occurences of nouns in the results.
- For each term t obtained we calculate the function $hyper(t, what_1)$ and take those terms t for which the value of the function $hyper$ is more than the average value of the function for all terms, thus obtaining the set H_1.
- We get the set H_2 using the same procedure for $what_2$.

[1]http://developer.yahoo.com/search/boss

4.2 The *Where* Part

The specificity of the *Where* part of a fact is that it usually contains geographical labels that can be mapped to latitude/longitude coordinates. We calculate the Let w_1, w_2 be the strings representing the *where* parts of two facts F_1, F_2. The semantic similarity is between w_1 and w_2 is calculated using the following formula.

$$s(w_1, w_2) = 1 - \frac{\min\limits_{g_1 \in G(w_1), g_2 \in G(w_2)} dist(g_1, g_2)}{MAX\_DIST},$$

where G(w) is a set of all geographical objects matching the string w, $dist(g_1, g_2)$ is the great-circle distance between geographical objects g_1 and g_2, and MAX_DIST is the maximum distance between two points on the Earth surface, which is about 20018 km and the two points are cities Esmeraldas in Ecuador and Padang in Indonesia. The great-circle distance is the shortest distance between any two points on the surface of a sphere measured along a path on the surface of the sphere. We calculate it using latitude/longitude coordinate of objects.

Roughly the algorithm can be described as follows.

- Get all possible latitude/longitude coordinates of w_1 and w_2;
- Calculate the great-circle distance between each pair of coordinates g_1 and g_2 where g_1, g_2 takes all possible coordinates of w_1 and w_2, respectively.
- Take the minimum distance calculated. It is assumed to be the distance between w_1 and w_2.
- Calculate the similarity using the calculated distance.

To implement the algorithm we use Google Maps API[2] and python package geopy[3].

4.3 The *When* Part

To calculate semantic similarity between two strings w_1, w_2 representing the *where* parts of two facts F_1, F_2, we use the following idea. We map the strings into dates and then calculate relative time interval between the dates applying some normalization. We use the following formula.

$$s(w_1, w_2) = 1 - \frac{d(w_1, w_2)}{d(w_1, w_2) + \min\{d(w_1, D), d(w_2, D)\}},$$

where $d(w_1, w_2)$ is the time interval (in seconds) between two dates matching the strings w_1 and w_2. D is the date representing the current moment. Normalization by D is used to implement the idea that one-year interval 1000 years ago should be considered less that the same interval nowadays, e.g. between January 1, 2009 and January 1, 2010.

We use the date parser from the dateutil python package[4]. It is rather smart and correctly parses such strings as “summer 2008”, “approximately 11.15” etc.

4.4 Experimental Results

To evaluate the function proposed, we ran the following experiment. We took a sentence from the news. Then we manually extracted a fact from it and manually extracted some more or less similar facts from the news. Then we calculated the similarity matrix of these facts. These facts are presented in Table 1. Corresponding similarity matrices for when, where and what parts are presented in Tables 2, 3, and 4, respectively.

One can see that for very short descriptions the results are meaningful. For example, the most relevant neighbors for term *shootout* are *robbery* and *armed robbery*, and the closest term for the term *anniversary* is *festival.*

5 Conclusion and Future Work

In this paper we described the task of fact search and proposed a function for calculating semantic distance between facts that are represented by a single sentence and consist of three parts (what, where, and when). Some experimental results are provided, justifying the proposed function.

Future work includes applying some methods from ontology theory, further develop the comparison function, and extending what-parts from one word to more complex types. We believe that the function calculating semantic similarity can be significantly improved using ontologies. The problem is that ontology construction is a resource consuming procedure and one can not expect that a “universal” ontology covering all possible domains will be constructed in the nearest future.

[2]http://code.google.com/apis/maps

[3]http://code.google.com/p/geopy

[4]http://labix.org/python-dateutil

ID	What	Where	When
1	robbery	Livermore	28 July 2008
2	burglary	California	July 2008
3	deposit	Fremont	November 2, 2007
4	anniversary	small town in California	summer 2007
5	shootout	California	January 3, 1997
6	crime	Hartford	August 27, 2007
7	kill	West Yorkshire, England	February 21, 2010
8	wine country festival	Livermore	2008
9	traffic	on main street in Pleasanton	Tuesday August 13, 2008
10	armed robbery	901 S. Main St. in Hartford, KY	On Friday July 13, 2007 at approximately 11:15 A.M.

Table 1: Test facts.

ID	1	2	3	4	5	6	7	8	9	10
28 July 20	1.00	0.99	0.72	0.63	0.14	0.67	0.17	0.95	0.97	0.64
July 2008	0.99	1.00	0.72	0.63	0.14	0.67	0.17	0.95	0.96	0.65
November 2	0.72	0.72	1.00	0.87	0.19	0.93	0.12	0.75	0.70	0.89
summer 200	0.63	0.63	0.87	1.00	0.22	0.93	0.11	0.66	0.61	0.98
January 3,	0.14	0.14	0.19	0.22	1.00	0.20	0.02	0.14	0.13	0.21
August 27,	0.67	0.67	0.93	0.93	0.20	1.00	0.11	0.70	0.65	0.95
February 2	0.17	0.17	0.12	0.11	0.02	0.11	1.00	0.16	0.17	0.11
2008	0.95	0.95	0.75	0.66	0.14	0.70	0.16	1.00	0.92	0.67
Tuesday Au	0.97	0.96	0.70	0.61	0.13	0.65	0.17	0.92	1.00	0.63
Friday Jul	0.64	0.65	0.89	0.98	0.21	0.95	0.11	0.67	0.63	1.00

Table 2: Similaity between when-parts.

ID	1	2	3	4	5	6	7	8	9	10
Livermore	1.00	0.99	1.00	1.00	0.99	0.79	0.58	1.00	1.00	0.85
California	0.99	1.00	0.99	0.99	1.00	0.80	0.58	0.99	0.99	0.86
Fremont	1.00	0.99	1.00	1.00	0.99	0.79	0.58	1.00	1.00	0.85
small town	1.00	0.99	1.00	1.00	0.99	0.80	0.58	1.00	1.00	0.86
California	0.99	1.00	0.99	0.99	1.00	0.80	0.58	0.99	0.99	0.86
Hartford	0.79	0.80	0.79	0.80	0.80	1.00	0.74	0.79	0.79	0.93
West Yorks	0.58	0.58	0.58	0.58	0.58	0.74	1.00	0.58	0.58	0.68
Livermore	1.00	0.99	1.00	1.00	0.99	0.79	0.58	1.00	1.00	0.85
on main st	1.00	0.99	1.00	1.00	0.99	0.79	0.58	1.00	1.00	0.85
901 S. Mai	0.85	0.86	0.85	0.86	0.86	0.93	0.68	0.85	0.85	1.00

Table 3: Similaity between where-parts.

ID	term	1	2	3	4	5	6	7	8	9	10
1	robbery	1	0.51	0.05	0.03	0.	0.25	0.88	0.	0.02	0.69
2	burglary	0.51	1	0.01	0.05	0.	0.04	0.92	0.	0.09	0.42
3	kill	0.05	0.01	1	0.	0.	0.09	0.93	0.	0.	0.11
4	deposit	0.09	0.05	0.	1	0.	0.	0.	0.	0.	0.
5	anniversary	0.	0.	0.	0.	1	0.	0.	0.25	0.	0.
6	shootout	0.25	0.04	0.09	0.	0.	1	0.01	0.12	0.09	0.32
7	crime	0.88	0.92	0.93	0.	0.	0.01	1	0.04	0.76	0.95
8	festival	0.	0.	0.	0.	0.25	0.12	0.04	1	0.07	0.
9	traffic	0.025	0.09	0.	0.	0.	0.09	0.76	0.07	1	0.02
10	armed robbery	0.69	0.46	0.10	0.01	0.	0.33	0.95	0.	0.08	1

Table 4: Similaity between what-parts.

Another direction of future work is to further detail the "what" part of the fact. For example, one can apply the subject-predicate-object ("who-did-what") model of knowledge representation, that is, again, extensively used in ontologies.

Comparing parts of facts we did not take into account the context, i.e. other parts. For example, if two facts occured in the 19th century, their "what" parts may be compared in a different way than if they occured in the 21th century. Or if the "what" part of the facts is about politics, we may compare the "where" parts in some special way. This is also a possibility for improvement of the algorithm. Another way is to create smarter functions that compare parts of facts. For instance, a function comparing "where" parts could distinguish geographical names from some abstract descriptions, e.g. *in an American school* or *in a small town near the west coast* and compare them somehow. The problem of geographical names meaning several places (e.g. there are at least five cities named Moscow) is also a complex one.

REFERENCES

Bollegala, D., Matsuo, Y., and Ishizuka, M. (2009). A relational model of semantic similarity between words using automatically extracted lexical pattern clusters from the web. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 803–812, Morristown, NJ, USA. Association for Computational Linguistics.

Cilibrasi, R. L. and Vitanyi, P. M. B. (2007). The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383.

Islam, A. and Inkpen, D. (2008). Semantic text similarity using corpus-based word similarity and string similarity. *ACM Trans. Knowl. Discov. Data*, 2(2):1–25.

Li, Y., McLean, D., Bandar, Z. A., O'Shea, J. D., and Crockett, K. (2006). Sentence similarity based on semantic nets and corpus statistics. *IEEE Trans. on Knowl. and Data Eng.*, 18(8):1138–1150.

Maguitman, A. G., Menczer, F., Roinestad, H., and Vespignani, A. (2005). Algorithmic detection of semantic similarity. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 107–116, New York, NY, USA. ACM.

Mccarthy, D., Koeling, R., Weeds, J., and Carroll, J. (2004). Finding predominant word senses in untagged text. In *In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 280–287.

Naughton, M., Stokes, N., and Carthy, J. (2010). Sentence-level event classification in unstructured texts. *Information Retrieval*, 13(2):132–156.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):1–69.

Pantel, P. and Lin, D. (2002). Discovering word senses from text. In *In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 613–619.

Purandare, A. and Pedersen, T. (2004). Senseclusters - finding clusters that represent word senses. In Susan Dumais, D. M. and Roukos, S., editors, *HLT-NAACL 2004: Demonstration Papers*, pages 26–29, Boston, Massachusetts, USA. Association for Computational Linguistics.

Turney, P. D. (2006). Similarity of semantic relations. *Computational Linguistics*, 32:379–416.

Turney, P. D. and Littman, M. L. (2005). Corpus-based learning of analogies and semantic relations. In *Machine Learning*.

Udani, G., Dave, S., Davis, A., and Sibley, T. (2005). Noun sense induction using web search results. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research* *and development in information retrieval*, pages 657–658, New York, NY, USA. ACM.

Varelas, G., Voutsakis, E., Raftopoulou, P., Petrakis, E. G., and Milios, E. E. (2005). Semantic similarity methods in wordnet and their application to information retrieval on the web. In *WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 10–16, New York, NY, USA. ACM.