Deriving Semantic Word Distances from Web Search Hits

Sebastian Fichtner*

Seminar Information Retrieval
Konstanz University, 2010/11
Prof. Dr. Marc H. Scholl
Advisor: Dr. Christian Grün

Abstract. This seminar paper explores the possibilities of deriving pairwise word similarities from page counts of web search engines. We motivate the problem, introduce selected papers and provide further exemplification through theoretic considerations as well as implementation- and evaluation of different measures.

1 Introduction

In the field of information retrieval we treat words as the semantic atoms of a text, because we are interested in its content, not its linguistic shape. We can agree that some words have a more similar meaning than others, but there is no natural measure for their similarity. However, there is a great need to have a function that would estimate our understanding of language by mapping word pairs on concrete distance values for further processing. Different estimators have been developed and provide benefits to almost all applications of information retrieval. Here are some example applications:

- natural language processing (syntax analyses, word sense disambiguation)
- query processing (relevance of a document for a query)
- query result expansion (for instance suggest similar)
- indexing and topic modeling
- clustering and classification of words and documents
- visualization of words, documents and corpora (positioning)

Since the first attempts in the sixties, research in information retrieval, data mining, computer linguistics and information theory has produced a vast number of methods to estimate semantic word similarity, either as its main objective or as part of a solution to a related problem. Most of all, word similarity is a crucial part of document similarity. We divide those methods into two fundamentally different groups: On the one hand, we might start at the word of interest and look-up its specific quality and relations to other words in order to logically infer from that theoretic knowledge. On the other hand, we might look at the

* sebastian.fichtner@uni-konstanz.de

whole context from a bird's eye view and apply statistical methods on text to quantify some form of word co-occurrence. Of course, different approaches can be combined, but it helps to distinguish the opposite principles.

	bottom up	top down
language data	theory	usage
instrument of reasoning	logic	statistics
analyzed property	quality	quantity

Prominent data sources for the bottom up method are WordNet, VerbNet, FrameNet, OpenThesaurus and even Wikipedia. The data source for the top down method is usually some text corpus which might be the specific text we want to analyze, a large corpus representing all our language or anything in between. The type we want to discuss here uses the internet as the corpus and a search engine to quantify the co-occurrence of words. Its idea is simple: When we search for a combination of two words, the page count reflects how often both words occur together. This co-occurrence relative to the separate occurrences of both words reflects the significance of their co-occurrence. Words that appear together in documents significantly more frequent than others must be more semantically related and this relatedness might be a useful notion of similarity. The following example was produced on google.com:

query	god	+	knowledge	+	education
results (m)	652	49	334	206	709

knowledge and *education* are significantly more similar (206m) than *knowledge* and *god* (49m). This is obvious, because the separate queries for *god* (652m) and *education* (709m) produce quite similar page counts. If we had an explicit calculation formula to turn page counts into a value for semantic distance, this method would provide certain advantages:

- As a statistical top down approach, it needs no prior linguistic knowledge and is easier to automate. Responsibility is given to the programmer and the applied methods are more clinical.
- The corpus is the largest collection of documents in english language and represents the language in general. It is equally applicable to any subject area and provides high statistical significance.
- The corpus updates itself all the time and comprehends novel meanings of words, like 'apple' referencing the company instead of the fruit.
- The corpus is always accessible and consumes no storage space, although it is huge.
- Calculations on the client side are very simple. Collecting, indexing and counting of documents is taken over by the search engine.
- If need be, the collection of page counts can be heavily parallelized and even be distributed on different computers.

In spite of its advantages, there are only few publications that explicitly pursue this idea. These are the ones that come closest:

- *A web-based kernel function for measuring the similarity of short text snippets* (Sahami and Heilman, 2006)
- *Measuring semantic similarity between words using web search engines* (Bollegala et al., 2007)
- *Using web-search results to measure word-group similarity* (Gledson and Keane, 2008)

Throughout this seminar paper, we will refer to the publication of Gledson and Keane (G&K) as a guideline and reference for our own thoughts and experiments, for it covers the idea we have just descibed most accurately. We will not present it in full detail, though.

2 Distance Functions

How will we calculate the actual distance between two words w_A and w_B? For this purpose, web search results for the queries w_A, w_B and $w_A \wedge w_B$ are modeled as sets of hits A, B and $A \cap B$ with hit counts $a = |A|$, $b = |B|$ and $i = |A \cap B|$. Now if we take (a, b, i) as the input vector to our distance function, the outcome

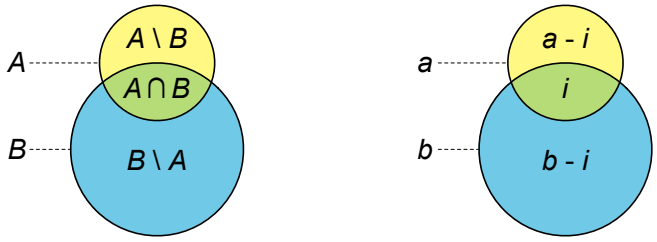


Fig. 1: *modeling search results as sets*

shouldn't change when the vector is scaled, meaning that the function would be homogeneous of degree 0. What matters are only the relations between a, b and i, which can be determined by only two ratios:

$$x = i/a = P(h \in B \mid h \in A)$$
$$y = i/b = P(h \in A \mid h \in B)$$

These are the conditional probabilities, that a search hit h for one word, is also in the result set of the other one. Because both ratios have the size of the intersection as their numerator, they can either both be 0 or both be greater than 0: $x > 0 \iff y > 0$. So the normalized distance function would be of the form

$$d(w_A, w_B) = d(x, y) \ :]0,1] \times]0,1] \cup \{(0,0)\} \to [0,1]$$

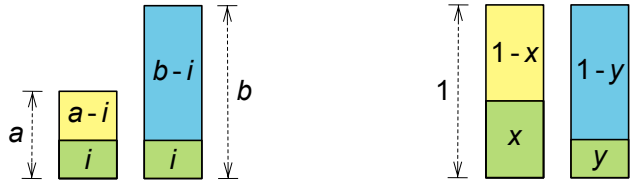


Fig. 2: *normalization of hit counts*

2.1 Criteria

To find a sound model for distance, we need criteria the distance function has to fulfill. The most common criteria is that it has to be a metric:

1. commutation:
   $d(w_A, w_B) = d(w_B, w_A)$
2. limits:
   $d(w_A, w_B) = 0 \iff x = y = 1$
   $d(w_A, w_B) = 1 \iff x = y = 0$
3. triangle inequality:
   $d(w_A, w_B) + d(w_B, w_C) \geq d(w_A, w_C)$

In addition to the metric property we need further criteria to match our intuitive understanding of similarity. Note, that x and y both indicate directly how much both words have in common:

4. none saturation:
   $d(x, y) > d(x + t, y)$ with $0 < t \leq \bar{x}$
5. complementarity:
   $d(x, y) < d(x - t, y + t)$ with $x \leq y$ and $0 \leq x - t < y + t \leq 1$
6. differentiability:
   $d(x, y)$ is differentiable in x and y

None saturation means, that similarity must increase, when one of both values increases. Complementarity demands, that x and y are not perfect substitutes, but complement each other, meaning that for a constant sum $x + y$ similarity must be greater when x and y are more equal. Natural distance would also be differentiable, because the way in which similarity adjusts to small changes of x or y would not change abruptly.

In the following sections, we will explain the distance function used by G&K as well as some other selected ones, before we compare them all regarding our six criteria.

2.2 Gledson & Keane

G&K calculated distance on small word groups. They order the words by decreasing page count, and then determine the page count that each word together with

all preceding words would produce. What they get are two arrays of decreasing page counts, one for single- and one for combined queries. For these arrays the gradients of their best fitting lines are calculated. Finally, the difference (not a ratio!) of these gradients is taken as the distance measure. It reflects, how much smaller the intersections are in comparison to single word result sets. However, the value is not normalized. If we would double all retrieved page counts, distance would double as well, which means that more popular words appear more distant. When we reduce this distance calculation to the special case of word pairs, the function would be $d_0(w_A, w_B) = min(a - i, b - i)$. In order to make this comparable to others, we normalize it:

$$d_0(x, y) = min(\bar{x}, \bar{y})$$

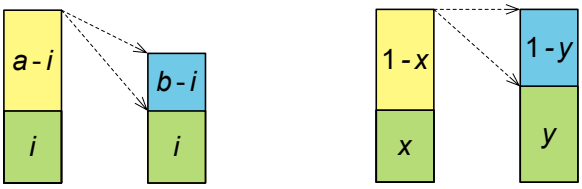


Fig. 3: *normalizing the distance of Gledson & Keane for word pairs*

This function however only fulfills criteria (1), which is trivial. In the following section we will look for a sound model of distance in the sense of our six criteria.

2.3 Other Distance Functions

The easiest way to map our "partial" similarities x and y to one distance value is to use their sum or product:

$$d_1(x, y) = 1 - \frac{1}{2}(x + y)$$
$$d_2(x, y) = 1 - \sqrt{x * y}$$
$$d_3(x, y) = \sqrt{\bar{x} * \bar{y}}$$

Because we want to have a more concrete size, we further define a measure in terms of probability. The probability, that a search hit h for any of two query words is actually relevant to both, reflects their similarity:

$$d_4(x, y) = 1 - P(h \in A \cap B \mid h \in A \cup B)$$
$$= 1 - \frac{i}{a + b - i} = 1 - \frac{1}{\frac{1}{x} + \frac{1}{y} - 1}$$

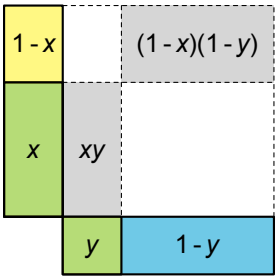


Fig. 4: *products reflecting similarity and distance*

Still, because this is normalized by the size of the union of both results, which is more like a sum, it should be brought down to the intrinsic order of magnitude. We do this by normalizing with the average size of both result sets:

$$d_5(x,y) = 1 - \frac{i}{(a+b)/2} = 1 - \frac{2}{\frac{1}{x} + \frac{1}{y}}$$

We can apply vector based distance measures as well, because actually x and y describe probability vectors. Let W be a random variable over our query words w_A and w_B and let S be a random variable over subsets s_{ij} of all documents. s_{ij} with $i, j \in \{0, 1\}$ is the biggest set of documents (search hits) that is relevant to w_A if $i = 1$ and relevant to w_B if $j = 1$. We build the probability distribution for the joint variable (S, W) where w_A and w_B are equally important and therefore equally "probable":

		W		
		w_A	w_B	$\sum$
S	$s_{00} = \setminus A \setminus B$	0	0	0
	$s_{01} = B \setminus A$	0	$\bar{y}/2$	$\bar{y}/2$
	$s_{10} = A \setminus B$	$\bar{x}/2$	0	$\bar{x}/2$
	$s_{11} = A \cap B$	$x/2$	$y/2$	$(x+y)/2$
	$\sum$	0.5	0.5	1

We now represent both words by their conditional probability vectors:

$$\boldsymbol{w}_A = \begin{pmatrix} P(s_{00}|w_A) \\ P(s_{01}|w_A) \\ P(s_{10}|w_A) \\ P(s_{11}|w_A) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \bar{x} \\ x \end{pmatrix} \qquad \boldsymbol{w}_B = \begin{pmatrix} P(s_{00}|w_B) \\ P(s_{01}|w_B) \\ P(s_{10}|w_B) \\ P(s_{11}|w_B) \end{pmatrix} = \begin{pmatrix} 0 \\ \bar{y} \\ 0 \\ y \end{pmatrix}$$

Both words together are represented by the absolute probability vector:

$$\boldsymbol{w} = \begin{pmatrix} P(s_{00}) \\ P(s_{01}) \\ P(s_{10}) \\ P(s_{11}) \end{pmatrix} = \begin{pmatrix} 0 \\ \bar{y}/2 \\ \bar{x}/2 \\ (x+y)/2 \end{pmatrix} = \frac{\boldsymbol{w}_A + \boldsymbol{w}_B}{2}$$

The most prominent vector distance is the eucleadian distance which we normalize by $\sqrt{2}$:

$$d_6(x,y) = \sqrt{\frac{(x-y)^2 + \bar{x}^2 + \bar{y}^2}{2}}$$

Another possibility is inspired by the Needleman-Wunsch algorithm for global sequence alignment. We try to align the sets A and B using the probability distributions given by x and y. The length of the path that connects opposite corners of the square in Figure (5) reflects the distance between w_A and w_B:

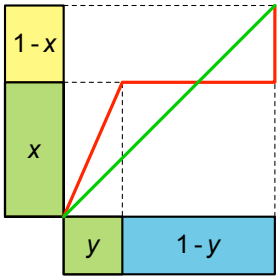


Fig. 5: *deriving distance from path length*

The path length is calculated as a sum of vector lengths:

$$\sqrt{2} \leq \sum_{i,j \in \{0,1\}} \left| \begin{pmatrix} P(s_{ij}|w_A) \\ P(s_{ij}|w_B) \end{pmatrix} \right| \leq 2$$

Mapping this path length to the interval $[0,1]$ and simplifying its formula leads to the distance function

$$d_7(x,y) = \frac{\sqrt{x^2+y^2} + \bar{x} + \bar{y} - \sqrt{2}}{2 - \sqrt{2}}$$

Though this a descriptive way to define distance, geometric length cannot be its natural unit of measurement. That is why we add another function that is based on entropy:

$$\begin{aligned} d_8(x,y) &= H(S) - H(S|W) \\ &= H(\boldsymbol{w}) - (0.5 \cdot H(\boldsymbol{w}_A) + 0.5 \cdot H(\boldsymbol{w}_B)) \\ &= H\left(\frac{\boldsymbol{w}_A + \boldsymbol{w}_B}{2}\right) - \frac{H(\boldsymbol{w}_A) + H(\boldsymbol{w}_B)}{2} \end{aligned}$$

To explain the nature of this idea, is beyond the scope of this seminar paper, but let's just say, that d_8 measures how much uncertainty is added, when we don't distinguish between our query words anymore and only consider the marginal distribution of S. Note that the entropy value is naturally in $[0,1]$, because we compare exactly two words.

2.4 Comparison

Which distance function is best? The following table gives a comparison:

distance	short name	criteria broken	comment on the dimension
d_0	G&K	2, 3, 4, 5, 6	hard to interpret
d_1	sum	5	average probability
d_2	product_sim		harmonic average of probabilities
d_3	product_dist	5	harmonic average of probabilities
d_4	intersection_uni		probability, too high
d_5	intersection_avg		pseudo probability
d_6	eucleadian	4, 5	geometric length
d_7	path		geometric length
d_8	entropy		entropy / bit / uncertainty

The diagrams in Figure (6) show cross sections through all functions. It is plain to see, that they are quiet different. We conclude, that only d_2, d_5, d_7 and d_8 behave as we would expect from a distance function.

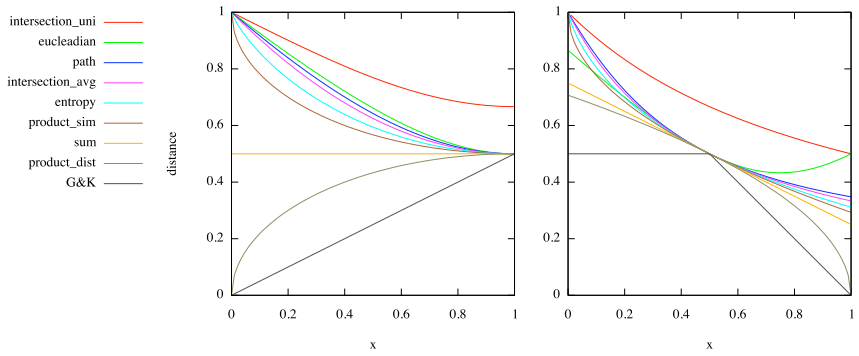
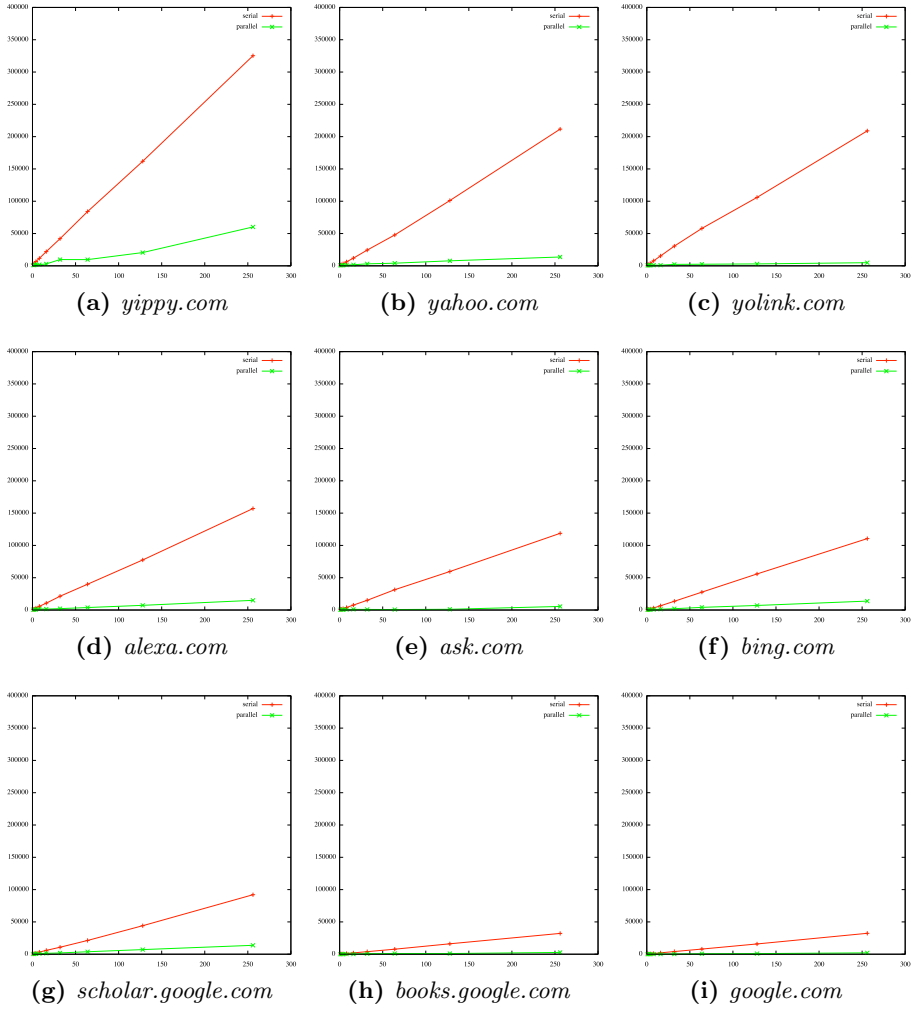


Fig. 6: *cross section through distance functions* $d(\frac{x}{2}, 1 - \frac{x}{2})$ *and* $d(x, \frac{1}{2})$

3 Implementation

G&K used the web APIs of YAHOO, MSN and Google. We did not use such services, because they are different for every search engine, not always available and often restricted in several ways. So we implemented the previously discussed distance functions using C++ and Qt. A URL was build with the query words and requested via basic network functions. Page counts were then parsed from the HTML code of the result page. We tested nine public search engines, of which three appeared unfeasible with this method. Google Scholar and YAHOO refused to show more results after a while, because they detected automated requests. Yippy.com gave us no results for certain words that we wanted to query

in the evaluation step. The following engines were used successfully: alexa.com, ask.com, bing.com, books.google.com, google.com and yolink.com.



(a) *yippy.com* **(b)** *yahoo.com* **(c)** *yolink.com*

(d) *alexa.com* **(e)** *ask.com* **(f)** *bing.com*

(g) *scholar.google.com* **(h)** *books.google.com* **(i)** *google.com*

Fig. 7: *performance in mili seconds dependent on the number of queries*

It became clear, that efficiency also had to be addressed: If we want to apply our distance measure to the clustering of document corpora, pair wise distances on a set of thousands of words would have to be calculated. Google requests have a delay of around 100ms, so the queries for all distance calculations on 10000 words would take about 58 days. Since our client computer would just be waiting for network responses most of the time, we parallelized that task using

multithreading. In our implementation many requests are send to the search engine before the first got answered. The diagrams in Figure (7) show for each tested search engine, how efficiency improves with parallelization. Red and green graphs show the performances of serial and parallel querying. It can be seen, that performance also depends on the chosen search engine. Google seems to be fastest.

4 Results

G&K used the TC-353 data set by Finkelstein et al. (2001) to evaluate their distance measure. This data set contains 353 word pairs and their similarity rated by humans. It is frequently used as test data for word similarity measures. To evaluate a novel measure, its correlation with the user ratings of the TC-353 data set is calculated. G&K achieved the following correlation coefficients:

search engine	correlation with TC-353
Yahoo	0.37
Live-Search (MSN)	0.4
Google	0.119

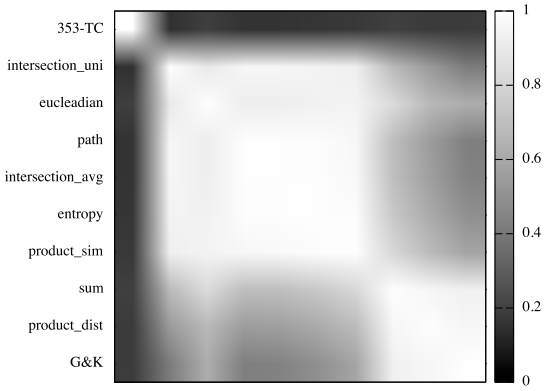
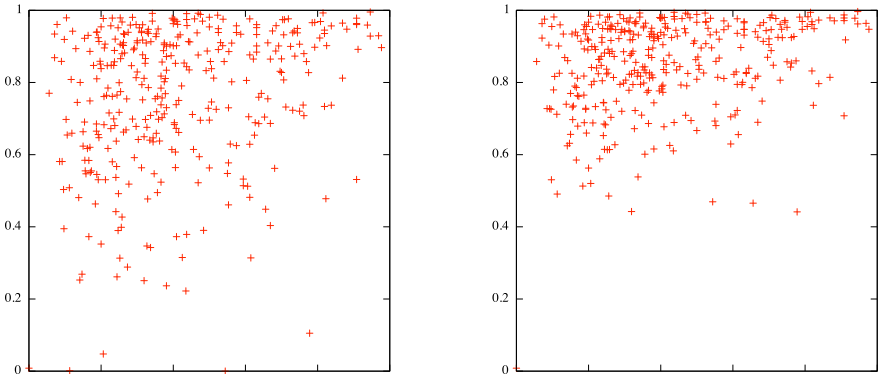


Fig. 8: *correlations between distance functions and test data on google.com*

They used two other test data sets as well. The TC-353 however is larger and newer then the others and is available on the internet, so we based our little evaluation on it and use it to compare our results with those of G&K. The matrix in Figure (8) shows the correlations between all previously discussed distance functions and the TC-353 collection on google.com. The matrices for other search engines are different in interesting details, but the bigger picture is the same for all of them: The correlation with the test data is quiet small. Correlations between different distance functions reveal, that there are two groups (or clusters) of distance functions as indicated by the bright blocks.

The scatter plots in Figure (9) illustrate correlations in more detail. Each point is a word pair from the TC-353 collection. The x-axis indicates the test data distance, the y-axis indicates a distance function. Figure (9a) demonstrates

the G&K distance function. Note, that our implementation produced a correlation twice as high as G&K originally declared in their paper, even with their own distance function. There are two possible reasons for this: First, we normalized their distance function, which might introduce some improvement. Second, we don't rely on Google's web service API, but parse the page counts that a normal user would actually see.



(a) *G&K: correlation 0.24* **(b)** *Sum: correlation 0.25*

Fig. 9: *correlations with test data on google.com*

Still, the visible correlation is not convincing. With perfect correlation, all points would lie on some ascending straight line. We need to look at the actual word pairs, to fully understand what we measure and why it hardly correlates with our test data. Because the entropy distance fulfills all criteria and has an adequate unit of measurement, we use it for this comparison. The following table contains the 5 most similar and the 5 most distant word pairs according to our entropy distance on google.com:

distance	w_A	w_B
0.465994	school	center
0.490776	football	basketball
0.495980	Mexico	Brazil
0.516599	news	report
0.559127	football	soccer
...	...	...
0.997003	food	rooster
0.997763	cup	artifact
0.997820	water	seepage
0.998217	sign	recess
0.998265	Arafat	Jackson

We already can see the kind of relatedness that we measure. The most "similar" words are most likely to appear together in a web document. "school center" and "news report" are even terms of their own. This relatedness is expressive and might be useful in some context. To understand how it differs from the "similarity" declared by by the test data set, we have to look at the greatest and smallest "errors", that our distance measure is making according to the test data. We take the absolute difference between test data and entropy distance as an error indicator:

difference	w_A	w_B
...	...	...
0.844436	furnace	stove
0.844556	fuck	sex
0.852852	environment	ecology
0.855167	Maradona	football
0.864044	boy	lad
0.877912	gem	jewel
0.880851	magician	wizard
0.882535	journey	voyage
0.904411	dollar	buck
0.910870	midday	noon

difference	w_A	w_B
...	...	...
-0.054647	stock	CD
-0.028305	stock	phone
0.012364	sugar	approach
0.017053	king	cabbage
0.025341	professor	cucumber
0.043268	chord	smile
0.050790	noon	string
0.054204	start	year
0.054472	rooster	voyage
0.057229	month	hotel
...	...	...

Obviously our distance doesn't capture the closeness of words in the test data. The greatest "errors" are made on very related words, whereas the smallest "errors" occur with unrelated words. It must be said, that this comparison is distorted by the fact, that our distance values are of course higher in general. However, for the correlation coefficients this higher average doesn't matter, and the scatter plots support the observation, that most "errors" are made by assigning high distance values to very related words.

5 Conclusion

Two technical issues were not tackled in this work: Some search engines that have access to an immense corpus like Google does, partly estimate page counts. It must be made clear, how the page counts are created and what they mean, otherwise the whole idea stands on fragile grounds and doesn't really work. Apart from that, it is a tricky task to avoid detection of automated requests and be fast at the same time. The idea to let the application pretend to be a browser, is left for future implementation.

We learned, that there are many different definitions and measures of word distance, and each of them is based on its own understanding of similarity or respectively defines it. It makes no sense, to take one arbitrary definition as the reference or gold standard for a different one. The data set that we used as our test data, like many others have done, is especially problematic. It cannot be used as a distance measure, because it isn't clearly defined. Every person who

contributed to the test values has his own understanding of word relatedness and only 353 word pairs were rated, so those values are far more imprecise than any computational distance function. Moreover, the criteria of the word pair selection are crucial but unclear.

Whether an application can benefit from the advantages of a certain method, depends on the notion of similarity or relatedness that is appropriate for its aim and context. The method we followed throughout this work measures how often two words appear together in the same web document. Its notion of "similarity" further depends on the indexing algorithms of the search engine. Our page count distance measure would be very useful for the utilized retrieval system itself, to expand results, suggest similar documents, suggest query words, cluster results, browse through the data base and the like. But other applications might find this measure of relatedness interesting as well.

Bibliography

[1] Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. Measuring semantic similarity between words using web search engines. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 757–766, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: http://doi.acm.org/10.1145/1242572.1242675.

[2] Revisited Lev Finkelstein, Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept. *ACM Transactions on Information Systems*, 20:116–131, 2001.

[3] Ann Gledson and John Keane. Using web-search results to measure word-group similarity. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 281–288, Morristown, NJ, USA, 2008. Association for Computational Linguistics. ISBN 978-1-905593-44-6.

[4] Mehran Sahami and Timothy D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 377–386, New York, NY, USA, 2006. ACM. ISBN 1-59593-323-9. doi: http://doi.acm.org/10.1145/1135777.1135834.