

Combining Web-based Searching with Latent Semantic Analysis to Discover Similarity Between Phrases

Sean M. Falconer¹, Dmitri Maslov², and Margaret-Anne Storey¹

¹ University of Victoria, Victoria BC V8W 2Y2, CANADA
{seanf, mstorey}@uvic.ca

² University of Waterloo, Waterloo ON N2L 3G1, CANADA
dmitri.maslov@gmail.com

Abstract. Determining semantic similarity between words, concepts and phrases is important in many areas within Artificial Intelligence. This includes the general areas of information retrieval, data mining, and natural language processing. Existing approaches have primarily focused on noun to noun synonym comparison. We propose a new technique for the comparison of general expressions that combines web searching with Latent Semantic Analysis. This technique is more general than previous approaches, as it is able to match similarities between multi-word expressions, abbreviations, and alpha-numeric phrases. Consequently, it can be applied to more complex comparison problems such as ontology alignment.

1 The problem

In many domains and applications, understanding semantic similarity, or the semantic relationships between expressions is an important problem. With some applications, one may be interested in direct synonymy, and often thesauri like Roget’s thesaurus or WordNet [9] are used to try to solve this problem. However, there are different types of semantic relationships one may be interested in. For example, besides direct synonymy, one may be interested in correlations between a term and an instance of that term (*e.g.* “Human” to “Albert Einstein”), or possibly in an association (*e.g.* “Cooking” and “Food”). Previous work has largely been task or domain specific, such as limited to English synonym matching or restricted to a domain such as medical terms. In contrast, we are interested in the general problem of calculating expression similarity and we were led to this problem through our interest in ontology alignment.

An ontology is a conceptualization of a domain [7]. This conceptualization consists of a set of terms with certain semantics and relationships [18]. Generally, the terms are related by the `is_a` relationship, however, other relationships such as `has_a` often exist. In ontology alignment, one wants to map one ontology to another by matching like terms between the two ontologies so that the ontologies can be merged or compared.

Often, ontology alignment tools combine various heuristics to determine similarity between terms. With domain specific ontologies, synonym matching is sometimes performed using databases like UMLS (Unified Medical Language System) [12]. However, it is difficult to apply synonym matching in general, as most existing techniques either depend on a specific domain of knowledge or are restricted to a particular language.

Moreover, other semantic relationships exist within the ontologies that are equally important to map and understand. For example, by understanding the `has_a` relationship, the internal structure of terms can be mapped. We are interested in human guided ontology alignment, where the alignment algorithm attempts to propose likely matches to the user. We feel that existing techniques can benefit by considering expression similarity along with their other similarity calculations when proposing possible like or related terms.

We propose a data driven approach to solving this problem, which uses the novel idea of web searching to capture relevant instances or data for the terms of interest. To compare two expressions, our algorithm first performs a search using two queries corresponding to the expressions. The results returned represent highly compact and highly relevant instances of the expression. Generally speaking, if two expressions are not related, their corresponding search results have very little in common. Alternatively, if the items are associated with each other, the texts must have certain commonality. This is based on the idea that if the two ontologies in question were used to annotate web search results, then the annotations between matching terms should have a high probability of overlapping.

The results of the web searches get processed by the SVD (Singular Value Decomposition) matrix technique [13]. This is done to change the actual number of each single word occurrence in a single document to its expected number of occurrences based on the relative importance of this word in the given document. The numeric result for semantic similarity is returned using the cosine measure between the vectors of expected word occurrences. For multiple word terms we perform a linear combination of the terms to generate a single vector representing the entire expression.

The advantage of using web search, specifically the Google search engine, is that Google has access to billions of indexed pages in various languages and on various topics. This means that the algorithm is not restricted to a single domain, a single language, and to static information that cannot be easily extended. All of these are vitally important for general ontology alignment. Also, by using only the title of search results along with the page description for a small number of results, we are able to use far less data than traditional Latent Semantic Analysis (LSA), as the data is highly relevant to the given expression. Moreover, by processing smaller amounts of information, our algorithm is able to perform these comparisons quickly, which is important for large ontologies. Finally, searches can also be guided to provide clues as to the relationship between the expressions being compared, such as synonymy versus antonymy.

1.1 Organization of the paper

The remainder of the paper is organized as follows. In the next section we discuss the set of requirements for a good semantic matcher, which is based on the analysis of common types of term matching divergences. It is followed by a summary of the related work, considered in the following two directions: synonym matchers and ontology alignment algorithms. Afterwards, we formulate our algorithm, LSA-IR, and discuss particulars of its implementation. Next, we introduce the test data sets and follow this with the section describing the experimental results. In the latter, we show our tool's performance in comparison to other techniques, discuss the results, and examine the limitations of

our approach. We conclude with a short summary and a discussion of future research directions.

2 Developing requirements

2.1 What causes alignment problems?

There are many causes to alignment problems, and in general, it is a very difficult problem to solve. Ontologies can be developed by different groups of individuals and later those groups may want to consolidate their separately built ontologies. Separately built ontologies will most likely have terminology divergences as well as structural ones. For example, in [12] the authors attempt to align two anatomy ontologies, FMA and CRM. FMA contains approximately 59,000 concepts while CRM contains approximately 24,000. Even after the authors performed some basic normalization on the concept names (e.g. remove camel case and split concatenated words), there were only 1834 string matches.

Problems also develop between different versions of an ontology and these versions may later need to be compared or merged. Terminologies may have been enhanced or replaced between versions, making the merge or comparison difficult to perform.

In Table 1 we characterize some of the types of divergences or problems that may occur. This characterization was partly inspired by the divergences discussed in [22], which dealt with data divergences between different data sources where objects are represented by terse English phrases. These problems help drive our requirements which we discuss next.

Table 1. Characterization of term matching divergences

Type	Example/Explanation
Structural differences	Differences in hierarchy or internal structure
Missing information	Information contained in ontology not available in other.
Misspellings	
Dropped, transposed substitutions	MICROSOFT <i>vs</i> MICOSOFT, MICROSOFT <i>vs</i> MICORSOFT, MICROSOFT <i>vs</i> MIKROSOFT
Common misspellings	APPARENT <i>vs</i> APPARANT
Variant spellings	MODELLING <i>vs</i> MODELING
Phonetic spelling	THEIR <i>vs</i> THERE
Synonyms	DOG <i>vs</i> CANINE
Abbreviations	FEMALE <i>vs</i> FML
Acronyms	HTML <i>vs</i> HYPERTEXT MARKUP LANGUAGE
Spelling, punctuation, spacing and casing	COLOR <i>vs</i> COLOUR ISSN <i>vs</i> I.S.S.N PUBLISHED_BY <i>vs</i> PUBLISHED BY <i>vs</i> "Published by"
Prefixes	xyzHUMAN <i>vs</i> HUMAN
Root forms	VALVES <i>vs</i> VALVE, RECEIVE <i>vs</i> RECEPTION
Alpha/Numeric	PI <i>vs</i> 3.141592
Technical/Common	DOG <i>vs</i> CANIS FAMILIARIS
Word order	THESIS PHD <i>vs</i> PHD THESIS

2.2 Requirements

In Table 1 we characterize different types of data divergences that can occur between two related ontologies. We use these characterizations to help build a set of requirements that any expression comparison technique for terminology alignment must satisfy.

- **Multi-lingual.** Any expression comparison technique cannot be restricted to a single language.
- **Multi-domain.** Expression comparison cannot be restricted to a single domain, as this is far too limiting for general approaches. The technique must be able to potentially support any domain.
- **Multi-word expressions.** Must be able to support comparison between multiple word expressions.
- **Dynamic.** Must be dynamic, in that we do not allow it to be restricted to static data that cannot be easily extended.
- **Multiple-relation comparison.** Cannot restrict similarity comparison to simply synonyms, other relationships are also important.
- **Misspellings:** Must be able to handle the various types of misspellings described in the ontology divergence table.
- **General expressions.** Must be able to compare similarities between various types of expressions, *e.g.* alpha/numeric, non-noun phrases, abbreviations, acronyms, *etc.*

3 Related work

In this section we briefly discuss related work. This discussion is partitioned into several topics that appear in separate subsections. In the first, we discuss work that has been conducted on the comparison of synonyms. In the second, we discuss existing ontology alignment techniques and discuss how our technique relates to existing approaches. Finally, we compare the set of existing synonym matching techniques to our list of requirements.

3.1 Synonym matching approaches

There have been various approaches applied to the problem of matching synonyms. Latent Semantic Analysis (LSA) [10], Roget Thesaurus [9], statistical techniques such as PMI-IR [23], product rule [24], and WordNet similarity measurements [9], [16] have all been applied to this problem. The LSA technique attempts to collect statistics about the relative frequency of a word and its neighboring words. It is based on the assumption that two words are similar if they have similar neighboring content words. One of the limitations of LSA is that it depends on preprocessing a large set of text to build a static representation of words and their relative frequencies. Extending this representation requires re-running the preprocessing. Thus, terms that do not appear in the preprocessed text cannot be later compared. We discuss this method in more detail in the description of our algorithm.

Similar to LSA, PMI-IR is based on co-occurrence of words, however, it computes this in a completely different manner. Rather than preprocessing documents and computing relative frequencies, it applies proximity based web searching and Pointwise Mutual Information (PMI). PMI comes from Information Theory, and it is a technique for calculating the similarity between two words or possibly a word and a document. In PMI-IR, web searching is applied to calculate statistics about the words in question. Two searches are carried out on the Altavista search engine and the number of $hits(query1)$ for the query is used in the calculation. For example, if we wanted to compare $term1$ to $term2$, PMI-IR would perform the following searches and calculation.

$$sim(term1, term2) = \frac{hits(term1 \text{ NEAR } term2 \text{ AND NOT } ((term1 \text{ OR } term2) \text{ NEAR "not"}))}{hits(term2 \text{ AND NOT } (term2 \text{ NEAR "not"}))}$$

The term NEAR is a keyword in the Altavista search engine that only reports sites where the two words separated by the NEAR keyword appear within 10 words of each other.

Similar to PMI-IR, another approach called LC-IR (local context-information retrieval) [8], makes use of the Altavista search engine. While PMI-IR measures similarity based on the proximity of the two words in question, LC-IR restricts searches to results where the two words are exactly adjacent to each other. By adding this restriction, the calculation and search queries are simplified and LC-IR slightly outperforms PMI-IR on the standard synonym data sets.

The product rule is a hybrid approach that uses several individual similarity modules and combines their results into a single calculation. For individual modules, it makes use of LSA, PMI-IR, synonym lists, and a Connector that uses Google summary pages to estimate stem-choice similarity. Each module is trained on a training set and a weight for that module is calculated based on the results from the training set. The weight represents the likelihood that the answer given by the module is correct and is used by the product rule when combining the individual calculations. On the TOEFL synonym test, the product rule has the best known result.

Thesauri are sometimes used for calculating semantic similarity. In the Roget Thesaurus work, paths between individual words are used to calculate similarity. The further apart the words, the smaller the similarity. Similar approaches have been applied to WordNet. These techniques are both limited to calculating synonymy, and WordNet primarily supports only noun to noun comparison. Also, with thesauri, many concepts that could appear in an ontology may not be present due to the thesauri language or domain. Moreover, alpha/numeric comparisons such as PI to 3.14159, are generally not supported.

3.2 Ontology alignment

As in synonym comparison, there has been a variety of approaches used to automatically or semi-automatically perform ontology alignments. For example, Euzenat *et al.* discuss over 20 different algorithms/tools in [6]. Some of the most widely used methods are based on heuristic techniques.

Heuristics are generally applied at two different levels, the first being to compute syntactic similarity and the second is to measure structural similarity. Both Chimera [1] and PROMPT [14] use syntactic similarities to make suggestions to the user. They first run an ontology alignment algorithm that attempts to find exact matches on concept names, prefixes, suffixes, or word roots. They then use the user’s feedback about the suggestions to make further suggestions based on structural similarities.

Structural similarity can be partitioned into two classes: internal and external structure [6]. Internal structural comparisons measure similarity between concept properties, such as cardinality, range, and symmetry. On the other hand, external structural comparisons attempt to find similarities in the `is_a` relationship structure between the two ontologies. Many hybrid approaches exist, such as QOM [5], which employ a large number of heuristics and use a weighted sum to normalize all similarity measurements into a single metric.

Another, less widely used approach is the so called instance-based or instance-level approach [2]. Here, concepts are compared based on their instances rather than their representation. An instance is an actual value of a concept, for example, an instance of a concept “Professor”, would be an actual Professor, such as Dr. Donald Knuth. Concept similarity can then be measured by comparing shared instances. Another way to measure the similarity for an instance-based approach is to apply machine learning techniques to build classifiers for concepts. The Glue system is an example of this, it builds learning classifiers for concepts and then evaluates the joint probability distribution of the assigned instances [6].

The semantic similarity technique that we are proposing is more closely related to the instance-based approach to ontology alignment. However, as mentioned, ontology alignment is a very difficult problem and it is unlikely that a single approach is general enough to capture all possible matching scenarios. Tools like PROMPT could be combined or extended to use our proposed technique in order to improve the suggestions made to the user.

3.3 Requirements table

In Table 2, we compare the previously discussed synonym matchers to our list of term matching requirements. Although each approach partially fulfills the requirements, none of the approaches completely satisfy every criteria that we established as important for semantic comparison in term alignment problem. Of course, this is not terribly surprising as none of these approaches were designed with these requirements in mind, they were designed to solve synonym tests. PMI-IR is the closest to satisfying our requirements, and could possibly be extended to meet the requirements, however the authors did not discuss multiple word or multiple types of comparison as this was not the focus of their work. Also, PMI-IR is not publically available so we could not verify whether this extension is possible. Altavista has also changed considerably over the last few years and PMI-IR may not work with the latest Altavista. Finally, on standard synonym tests, the method we developed (LSA-IR) significantly outperforms PMI-IR. LSA-IR is described in the next section.

Table 2. Requirement satisfaction for existing work

Technique	Multi-lingual	Multi-domain	Multi-word	Dynamic	Multi-comparison	Misspellings	General
LSA	√	√	√	×	√	×	√
Roget	×	√	×	×	×	×	×
PMI-IR	√	√	*	√	*	√	√
LC-IR	√	√	×	√	×	√	√
WordNet Sim	×	√	×	×	×	×	×
Product rule	√	×	√	√	×	√	√
LSA-IR	√	√	√	√	√	√	√

√ - satisfied, × - not satisfied, * - could possibly be extended, but not discussed

4 The algorithm

Our algorithm combines web searching and LSA, so following the tradition of PMI-IR and LC-IR, we refer to our technique as LSA-IR. By adapting LSA to use web search results rather than a static collection of text, the method is dynamic and not limited to preprocessed data. Also, since the number of results is kept small, and therefore the corresponding text to process is relatively small, LSA can be performed quickly and immediately. Moreover, limiting LSA to processing smaller portions of text helps improve LSA accuracy for certain applications. The authors of [15] introduced a two stage LSA algorithm for finding word aliases. The authors found that when LSA is applied to a general text collection, the most similar words to any given word often do not correspond to proper aliases. In the two stage approach, the authors perform LSA a second time, this time instead of using the general text documents, the documents are constructed by extracting small windows of text that surround the words of interest. That is, the second stage only considers the local context around occurrences of words rather than the global context with respect to the entire document. This extension to traditional LSA greatly improved the performance for extracting word aliases. Similarly, in our tool, since we only process link titles and Google’s content summary, we are essentially considering the local context of the word.

The input for our algorithm is a pair of ASCII character strings *String1* and *String2* along with optional information that indicates the type of comparison to be made. An example of the type of comparison may be to find the similarity between strings interpreting them as synonyms, antonyms, or homonyms. Such a comparison is guided by a user who points the search in a specific direction. For instance, when one looks for a synonym dependence between two expressions, the query entries are “*String1* synonym” and “*String2* synonym”. Similarly, with antonym dependence, the query entries are “*String1* antonym” and “*String2* antonym”. Both the “synonym” and “antonym” context help guide the search when the type of comparison is known.

As a more concrete example, consider the situation where we want to compare “dog” to “canine” as synonyms. We can guide the type of search results by adding “synonym” to the search queries, thus searching for pages containing “dog synonym” and “canine synonym”. This context acts as a heuristic which restricts the search results to certain types of websites. Our technique is based on the assumption that similar expres-

sions should yield similar search results, and we calculate this similarity by inspecting the local context as generated by the results.

Moreover, other restrictions are possible, such as restricting the search to a specific web site, ignoring web pages with certain words, or forcing exact phrase matches. While we did not experiment with the other guidelines for the search, it is in our future plans to do so. If no comparison type is specified we assume a general association comparison must be made and prepare the search queries as pure input strings “*String1*” and “*String2*”.

The next step is using a web search engine to search for the texts related to the prepared search strings. Search engines have access to billions of indexed web pages, and this provides a very rich data set to work with. Moreover, almost any concept can be found in a variety of different contexts and possible meanings. Our choice of the search engine is Google, which is based partly on the free availability of their search API and also due to Google currently being the most widely used search engine [20]. The result of the search is truncated at the first K hits with all being treated as equal. As discussed in the next section, in our experiments we use $K = 75$. From our initial experiments, this number of search results provides enough generality to find how the terms are being used, and yet is small enough to allow for fast data analysis.

Each of the K search results is treated as a separate document and the total number of documents in both searches equals $2K$. We next stem the words in all $2K$ documents using Porter’s Java implementation [17] and delete the most popular words carrying minimal information (stop words). Our list of stop words consists of around 400 words and includes such words as “the”, “a”, “of”, “in”, *etc.* The remaining words are used to form a matrix in which there are $2K$ columns that correspond to each of the returned documents and N strings with each corresponding to a single stemmed word that appears at least once in the set of documents. The column vectors are then normalized, meaning we treat all the documents as equal.

The matrix element $m_{i,j}$, where $1 \leq i \leq N$ and $1 \leq j \leq 2K$ is the number of occurrences of the i -th word in the j -th document of the first web search if $j \leq K$ and in the $(j - K)$ -th document of the second web search when $j > K$. We weight each element, $m_{i,j}$, as $\log(m_{i,j} + 1)$. This is a common practice in information processing applications [3]. We next process this matrix using the *Singular Value Decomposition* (SVD).

SVD [10], [13] is a popular and efficient technique for text analysis. Its input is a matrix M of word occurrences in the documents constructed as discussed above. This matrix gets decomposed into the product of three matrices $M = U D V^T$, where matrices U and V have orthogonal columns such that $U^T U = V V^T = I$ and D is a diagonal matrix. Keeping a percentage of the largest diagonal elements in matrix D transforms it to D' . Next, we construct the matrix M' as a product $M' := U D' V^T$. Matrix M' differs from the original matrix M . A nice property of M' is that an element $m'_{i,j}$ estimates the number of occurrences of the i -th word in the j -th document. The optimal number of dimensions that must be kept for the best performance is unknown and must be determined through experiments. In our evaluations we chose to keep 70% of the largest coefficients in the diagonal matrix. The details of how we arrived at this percentage are explained in the next section.

Once the SVD transformation is completed, we measure the similarity between the two input strings *String1* and *String2*. In the case of single word items we compare the vector of the expected word occurrences of *String1* with the vector of the expected occurrences of *String2*. The vectors are constructed using matrix M' by considering the strings corresponding to the words *String1* and *String2*. We use the cosine measure defined as follows

$$\cos(\bar{a}, \bar{b}) := \frac{(\bar{a} \cdot \bar{b})}{\|\bar{a}\| * \|\bar{b}\|}.$$

The cosine measurement defines if the two vectors point in the same direction (meaning if the items are equally used in the similar contexts) without caring which of the two vectors is longer (meaning which of the two search strings is a more commonly used expression/concept/term). When the cosine is close to zero this means that the two items are rarely used in the same contexts and thus we consider such items unrelated.

In the case when the strings *String1* and *String2* are composed with multiple words we first stem each word and delete common words from both strings. This results in the construction of strings *String1'* and *String2'*. If a word appears in both *String1'* and *String2'*, we apply a weight to this word in order to reduce its importance. For example, if we are interested in the similarity between “computer network” and “computer monitor”, we care more about the words “network” and “monitor”, than “computer” since this term appears in both strings. Then, we add the vectors corresponding to the expected occurrences of all words in *String1'*, and do the same with the words from *String2'*. Finally, the result is returned as a cosine between these summary vectors.

5 Data sets

In this section we describe the data sets used in our experiments. The first data set consists of two biomedical ontologies, the NCI (National Cancer Institute) thesaurus and SNOMED-CT (The Systematized Nomenclature of Medicine). The second data set consists of three popular synonym tests that have been used by previous researchers.

5.1 Ontology data

As mentioned, the ontology data comes from the NCI thesaurus and SNOMED-CT. The authors of [19] used an automated heuristical approach to map ontological terms from the NCI thesaurus and SNOMED-CT to unstructured keyword data within the Stanford Tissue Microarray Database (TMAD). The authors were interested in being able to represent the TMAD data in a structured way, such as in an ontology. They were able to successfully map 80% of the TMAD unstructured annotations. The authors mention that the mapping from the NCI thesaurus to TMAD and SNOMED-CT to TMAD could be used to align terms from the NCI thesaurus to SNOMED-CT.

Using the data generated in [19], we manually extracted 60 expression pairs. Each pair consists of one expression from the NCI thesaurus and its corresponding equivalent expression from SNOMED-CT. The expression pairs were classified based on the types

of divergence discussed in Table 1. The primary type of divergence is “Missing Information”, that is, one ontology used a more descriptive expression for the term but the two expressions had some overlap in the terms used. The data set also contains abbreviation matches, such as “Malignant_Peripheral_Nerve_Sheath_Tumor” to “MPNST”, synonym matches, such as “Neoplasm” to “Tumor”, spelling variations, punctuation changes, and changes in the word order. There are no exact string matches.

In the test, we first normalize the expressions by downcasting all characters, replace underscores with a space, and trim whitespace. We then compute similarities between all pairs of expressions, that is, we compute all $60 \times 60 (= 3600)$ expression similarities. Afterwards, we greedily select matches by pairing off the two expressions with the highest similarity, then we eliminate these two expressions from further consideration, and continue until all expressions have a match.

5.2 Synonym data sets

Although we did not develop LSA-IR with the specific purpose of synonym mining, we chose to evaluate it against some standard synonym data sets in order to compare with existing methods. The data sets used were 80 questions from TOEFL (Test of English as a Foreign Language) [10], 50 questions from ESL (English as a Second Language) [21], and 300 questions from RDWP (Reader’s Digest Word Power Game) [4].

The TOEFL and RDWP tests have the same structure. For each question they consist of a single word and four possible matches. One of the four possibilities is a synonym of the given word. The ESL test is slightly different, each question consists of a sentence where the word of interest is used in some context, and there are also four possible matches.

For both the TOEFL and RDWP questions, we compare the given word to all four possibilities and choose the word that yields the highest similarity value. The Google search was guided by using a context of “synonym”, that is, to compare “*String1*” to “*String2*”, we perform the following two Google searches, “*String1* synonym” and “*String2* synonym”. The similarity is returned as the median score between this search and searches “*String1*” and “*String2*”.

With the ESL data set, we first preprocess the questions to extract a single word context. We do this by first removing stop words from the questions, and then for each word w_i near the word of interest w , we compute $sim(w, w_i)$. Here *near* is any word within three words of w . The context becomes the word w_i with the highest similarity value to w . As an example, consider the ESL question “Don’t forget to vacuum the [rug] before they come home.” The word of interest, $w = \text{“rug”}$, and the term “vacuum” has the highest degree of similarity to “rug” according to LSA-IR, thus this becomes the context. For each of the four possible synonyms, we compute the similarity between the word of interest and the possible answer using the “synonym” context added together the similarity between the extracted context with the word of interest and the possible answer. The answer with the highest similarity is chosen as the correct synonym.

6 Algorithm parameters

In the discussion of the algorithm we introduced two constants, the number of search queries K , and the percentage of the diagonal elements of the matrix D that we keep. We must now choose these parameters for optimal performance. Our intention is to keep the number of search results as small as possible such that the amount of processing is kept small, yet yields a good performance. The smaller K is, the less the interaction with the web, and the less data that needs to be processed afterwards. Secondly, we adjust the number of diagonal elements to be kept. In adjusting the number of dimensions, we are primarily concerned with the quality of the results. This is because the percentage of diagonal elements kept does not greatly affect the speed of our algorithm.

6.1 Setting the parameters

We conducted an experiment using half of the TOEFL synonym data to find the best values for K and P . Using $10 \leq K \leq 120$ with an increment of 10 and $0.4 \leq P \leq 0.8$ with an increment of 0.05, we evaluated the TOEFL results for every $\{K, P\}$ combination. The results are shown in Figure 1. The height map represents the accuracy or percentage evaluated to be correct. Results fluctuate from a low of 0.7 to a high of 0.85. The high of 0.85 was achieved 3 times, with $K = 70, P = 0.7$ and with $K = 80, P = 0.7$ and $P = 0.75$. Based on this result, for all further experiments, we fixed $K = 75$ and $P = 0.7$.

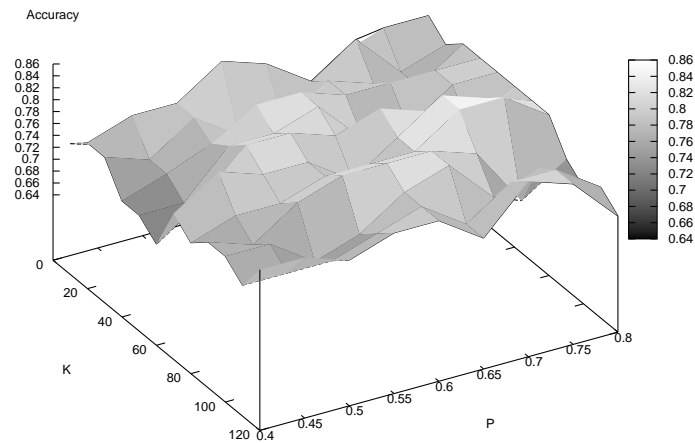


Fig. 1. 3D surface diagram of parameter setting experiment.

7 Experiment results

In the following two sub-sections we illustrate the results of the application of our tool to the data sets discussed previously. When applicable, we compare the results of these experiments with that of other techniques.

7.1 Ontology test

Table 3 describes the results from our experiment of matching terms from the NCI thesaurus to SNOMED-CT. The “Baseline” represents the results one would expect from random guessing. We included results computed using standard syntactical matching techniques, Levenstein distance [11] and substring distance [6]. We used the same normalization and greedy approach for both syntactical measurements. As previously mentioned, a large portion of the aligned terms were classified as type “Missing information”. Thus, a large number of the expressions contains overlapping substrings, which is most likely why the substring distance measurement achieved greater than 50% accuracy. However, both syntactical measurements were unable to match more complex (semantic) equivalences such as synonymity, so unsurprisingly LSA-IR significantly outperformed syntactical measurements. Moreover, even the incorrect matches computed by LSA-IR made semantic sense, that is, the matched terms were related, but they were not the most related as determined by our data set.

Table 3. Ontology matching experiment results

Technique	NCI to SNOMED-CT
Baseline	< 2.00%
Levenstein distance	$\frac{29}{60} = 48.33\%$
Substring distance	$\frac{32}{60} = 53.33\%$
LSA-IR	$\frac{53}{60} = 88.33\%$

7.2 Synonym tests

In this experiment we test our technique on some standard synonym data sets. The results are shown in Table 4. Again, the “Baseline” represents the results one would expect from random guessing.

LSA-IR has an overall result of 85.10%, which is the highest overall accuracy. Unfortunately the Product rule, which essentially solved TOEFL, does not have published results for ESL and RDWP. The Product rule would most likely outperform LSA-IR on these two data sets as it is a hybrid approach as opposed to a single technique like LSA-IR. However, we suspect that LSA-IR more efficiently computes similarities than the Product rule since the Product rule combines four different computational approaches.

Table 4. Synonym experiment results

Technique	TOEFL	ESL	RDWP	Total
Baseline	$\frac{20}{80} = 25.00\%$	$\frac{12.5}{50} = 25.00\%$	$\frac{75}{300} = 25.00\%$	25.00%
LSA	$\frac{51.5}{80} = 64.40\%$	-	-	-
Product rule	$\frac{78}{80} = 97.50\%$	-	-	-
PMI-IR	$\frac{59}{80} = 73.75\%$	$\frac{37}{50} = 74.00\%$	$\frac{216.83}{300} = 72.30\%$	72.80%
Roget's Thesaurus	$\frac{63}{80} = 78.75\%$	$\frac{41}{50} = 82.00\%$	$\frac{223}{300} = 74.30\%$	76.00%
LC-IR	$\frac{65}{80} = 81.30\%$	$\frac{39}{50} = 78.00\%$	$\frac{224.33}{300} = 74.80\%$	76.40%
LSA-IR	$\frac{71}{80} = 88.75\%$	$\frac{39}{50} = 78.00\%$	$\frac{256}{300} = 85.33\%$	85.10%

7.3 Recommendation test

In our final test, we used the RDWP synonym data set again, but we randomly removed 50 of the correct answers and replaced them with a random incorrect answer. Moreover, this time, LSA-IR only made a match recommendation if the similarity between the question word and its most similar answer was above a threshold value of ω . In our test, we experimented with a variety of ω values.

This test becomes a more complicated version of the original, as now LSA-IR must not only determine the correct synonym, but it must also determine if a synonym for the word exists at all. This is an important test because in many matching problems, like ontology alignment, it is very likely that not every term has an appropriate match. We do not wish to make bad suggestions just because it is the only suggestion we can make.

We used the standard Information Retrieval evaluation metrics, precision and recall, as defined in [5] for ontology alignment.

Precision is the measurement of correctly retrieved mappings in proportion to the total number of found mappings.

$$p = \frac{\#correct_found_mappings}{\#found_mappings}$$

Recall is the measurement of correctly retrieved mappings in proportion to existing mappings in the data set.

$$r = \frac{\#correct_found_mappings}{\#existing_mappings}$$

Finally, as is typically done in Information Retrieval, we combine the two metrics into an f -measure.

$$f = \frac{2pr}{p+r}$$

The results are shown in Table 5. We varied ω from 0.1 to 0.4 and recorded the precision and recall for each of these values. As a baseline, we used $\omega = 0.0$. With this threshold, LSA-IR always chooses the term with the highest similarity as the synonym

regardless of the value of this measurement. In this case, the recall should be maximized while the precision should be minimized, which the experimental results confirm. The best overall performance, based on the f -measure, occurred with $\omega = 0.2$. With this value, we achieved a very high precision and still correctly found 165 results, which is 83.76% of our original 197 correctly found results. With higher ω values, as expected, our precision continues to climb, however our recall value is much too low.

Table 5. Recommendation test results

ω	Precision	Recall	f -measure
Baseline (0.0)	$\frac{197}{300} = 65.67\%$	$\frac{197}{250} = 78.80\%$	0.7164
0.1	$\frac{197}{299} = 65.89\%$	$\frac{197}{250} = 78.8\%$	0.7177
0.2	$\frac{165}{183} = 90.16\%$	$\frac{165}{250} = 66.00\%$	0.7621
0.3	$\frac{73}{76} = 96.05\%$	$\frac{73}{250} = 29.20\%$	0.4478
0.4	$\frac{76}{27} = 96.43\%$	$\frac{27}{250} = 10.80\%$	0.1942

Figure 2 displays the plot of the f -measure versus the different ω values.

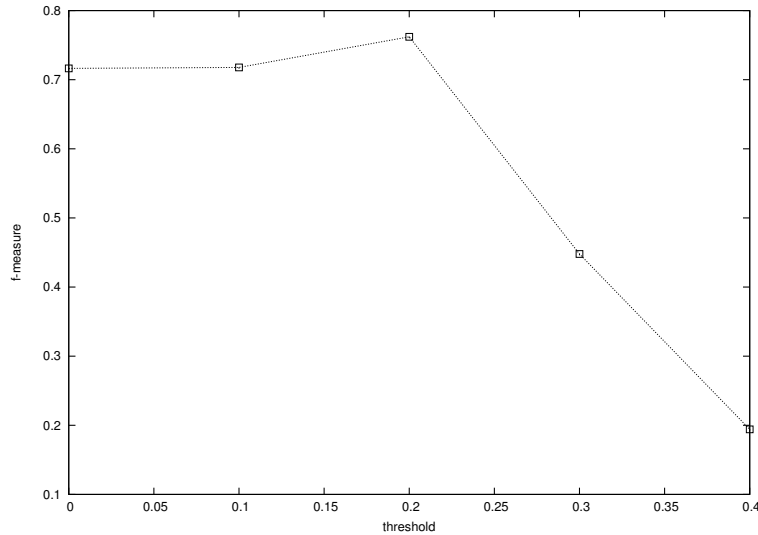


Fig. 2. Plot of the f -measure versus threshold value ($0.0 \leq \omega \leq 0.4$).

7.4 Discussion of results

We are primarily interested in user-guided ontology alignment, and the results of semantic comparison of the terms from the NCI thesaurus to SNOMED-CT ontologies

indicate that LSA-IR is potentially beneficial for computing suggestions during the ontology alignment process. In particular, for ontology alignment it would be extremely helpful to users to be able to suggest matches such as “Olfactory_Neuroblastoma” to “Esthesioneuroblastoma”, which many syntactical matchers are not able to recommend.

Although synonym matching is not the primary purpose of our tool, the results of the synonym comparison experiment helped verify the usefulness of the tool for mining strictly synonyms. As discussed, in comparison to previous approaches, LSA-IR had the best overall performance.

In the final experiment, we wanted to test the precision and recall of LSA-IR. To our knowledge, this is the first time this kind of test has been used to evaluate a semantic comparator. Previous research has not reported precision results. However, we feel this is a critical test if our tool is to be used in applications like user-guided ontology alignment. We do not want to make inappropriate suggestions to the user simply because it is the only suggestion we can make. There must be a balance between precision and recall, as we do not want to suggest too many incorrect matches and we also do not want to exclude too many possibly correct matches. Our results indicate that our method is able to correctly identify matches with high precision, and still produce a large number of suggested matches. The results from each of these preliminary experiments are very encouraging and indicate to that we should continue experimenting with this approach.

7.5 Limitations

LSA-IR suffers from several potential limitations. Firstly, since the method depends on the Google search index, it inherits certain limitations that are intrinsic with Google. For example, although Google has billions of pages indexed, it cannot possibly index everything, thus, certain word contexts may be missing or not present within the first 75 Google results. Moreover, we are also limited by the quality of the search results returned by Google. Ideally, we would like to be able to extract the local context of the words of interest from each search result, but performing this extraction from each page would be too costly. In order to bypass this computational cost, we rely on the Google summary and page title, but the proper context of the words of interest may not be present in this result information.

8 Conclusions and future work

In this paper we presented an automated, data-driven tool for semantic comparison of phrases. Our tool extends traditional LSA by adapting it to work with web-based search results. By using web search results, we are able to perform comparisons between many types of expressions that previously has been difficult to perform using a single tool. Previous web-based comparison tools have concentrated on computing statistics based on the number of search results returned, whereas with LSA-IR, we concentrated on actually analyzing the text of these returned results.

We demonstrated the usefulness of our approach by conducting experiments on several different data sets. Specifically, we showed how LSA-IR was useful for matching

equivalent expressions in two biomedical ontologies, the NCI thesaurus and SNOMED-CT. We also tested the tool on standard synonym benchmarks that had been used by previous researchers and although synonym matching was not our primary goal, LSA-IR outperformed earlier tools that were built specifically to discover synonymy. Finally, we conducted an experiment to test how well LSA-IR can not only choose a correct synonym from a list, but also how well it can determine whether a synonym exists at all.

In the future, we plan to incorporate our semantic comparator into a user-guided ontology alignment tool. Further research is required on how to distinguish between `is_a` and `has_a` relationships, as simply guiding the search via a “is a” or “has a” context will most likely not be enough to distinguish the two concepts.

We also believe LSA-IR could be useful for automated data annotation between ontologies and unstructured data, similar to how it is done in [19]. We are interested in extending our tool to help automated keyword extraction for scientific papers. Many existing approaches to this problem use word frequency counts, however, authors of research papers often use semantically equivalent expressions when discussing a single topic or idea, thus the frequency counts may not be accurate.

Acknowledgements

Firstly, we would like to thank Nigam Shah, Daniel L. Rubin, Kaustubh S. Supekar and Mark A. Musen for use of their ontology matching data. We would also like to thank Thomas Landauer and the LSA group at the University of Colorado at Boulder for providing us with the TOEFL test questions.

This work was supported in part by the National Center for Biomedical Ontology, under roadmap-initiative grant U54 HG004028 from the National Institutes of Health, and by a PDF grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. McGuinness D., Rice J. Fikes R., and Wilder S. An environment for merging and testing large ontologies. 2000.
2. AnHai Doan, Pedro Domingos, and Alon Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50(3).
3. S. Dumais. Enhancing performance in latent semantic indexing. Technical report, 1990.
4. M. Lewis (ed.). Readers digest. 158(932, 934, 935, 936, 937, 938, 939, 940), 159(944, 948), 2000-2001.
5. M. Ehrig and S. Staab. Qom - quick ontology mapping. In *Proc. of the Third International Semantic Web Conference (ISWC2004)*, 2004.
6. Jérôme Euzenat, Thanh Le Bach, Paolo Bouquet Jesús Barrasa, Jan De Bo, Rose Dieng-Kuntz, Marc Ehrig, Manfred Hauswirth, Mustafa Jarrar, Rubén Lara, Diana Maynard, Amedeo Napoli, Giorgos Stamou, Heiner Stuckenschmidt, Pavel Shvaiko, Sergio Tessaris, Sven Van Acker, and Ilya Zaihrayeu. State of the art on ontology. deliverable d2.2.3, 2004.
7. T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):23–28, 1993.

8. Derrick Higgins. Which statistics reflect semantics? rethinking synonymy and word similarity. In *International Conference on Linguistic Evidence*, 2004.
9. M. Jarmasz and S. Szpakowicz. Roget's thesaurus and semantic similarity. In *Proceedings of Conference on Recent Advances in Natural Language Processing (RANLP 2003)*, pages 212–219, September 2003.
10. T.K. Landauer and S.T. Dumais. A solution to plato's problem: The latent semantic analysis: Theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240, 1997.
11. I. Levenstein. Binary codes capable of correction deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
12. Peter Mork and Philip A. Bernstein. Adapting a generic match algorithm to align ontologies of human anatomy. In *20th International Conference on Data Engineering*. IEEE, 2004.
13. J. C. Nash. *Compact Numerical Methods for Computers: Linear Algebra and Function Minimization*. Adam Hilger, Bristol, England, 2nd edition, 1990.
14. N. Noy and M. Musen. The prompt suite: Interactive tools for ontology merging and mapping. Technical report.
15. Tim Oates, Vinay Bhat, Vishal Shanbhag, and Charles Nicholas. Using latent semantic analysis to find different names for the same entity in free text. In *Proceedings of the 4th international workshop on Web information and data management*, 2002.
16. T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet::similarity - measuring the relatedness of concepts, 2004.
17. M. F. Porter. Java implementation of porter's algorithm, 2000.
18. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, New Jersey, 1995.
19. Nigam H. Shah, Daniel L. Rubin, Kaustubh S. Supekar, and Mark A. Musen. Ontology-based annotation and query of tissue microarray data. In *AMIA 2006 (under review)*, 2006.
20. Danny Sullivan. Nielsen netratings search engine ratings. <http://searchenginewatch.com/showPage.html?page=2156451>, 2006. Last visited: 08-15-2006.
21. D. Tatsuki. Basic 2000 words - synonym match 1, 1998.
22. David R. Throop. Reconciler: Matching terse english phrases, 2004.
23. P. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001)*, pages 491–502, September 2001.
24. P. Turney, M.L. Littman, J. Bigham, and V. Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proceedings of International Conference on Recent Advances in Natural Language Processing (RANLP-03)*, September 2003.