Unsupervised Learning of the Morpho-Semantic Relationship in MEDLINE®

W. John Wilbur
National Center for Biotechnology Information / National Library of Medicine, National Institutes of Health, Bethesda, MD, U.S.A.
wilbur@ncbi.nlm.nih.gov

Abstract

Morphological analysis as applied to English has generally involved the study of rules for inflections and derivations. Recent work has attempted to derive such rules from automatic analysis of corpora. Here we study similar issues, but in the context of the biological literature. We introduce a new approach which allows us to assign probabilities of the semantic relatedness of pairs of tokens that occur in text in consequence of their relatedness as character strings. Our analysis is based on over 84 million sentences that compose the MEDLINE database and over 2.3 million token types that occur in MEDLINE and enables us to identify over 36 million token type pairs which have assigned probabilities of semantic relatedness of at least 0.7 based on their similarity as strings.

1 Introduction

Morphological analysis is an important element in natural language processing. Jurafsky and Martin (2000) define morphology as the study of the way words are built up from smaller meaning bearing units, called morphemes. Robust tools for morphological analysis enable one to predict the root of a word and its syntactic class or part of speech in a sentence. A good deal of work has been done toward the automatic acquisition of rules, morphemes, and analyses of words from large corpora (Freitag, 2005; Jacquemin, 1997; Monson, 2004; Schone and Jurafsky, 2000; Wicentowski, 2004; Xu and Croft, 1998; Yarowsky and Wicentowski, 2000). While this work is important it is mostly concerned with inflectional and derivational rules that can be derived from the study of texts in a language. While our interest is related to this work, we are concerned with the multitude of tokens that appear in English texts on the subject of biology. We believe it is clear to anyone who has examined the literature on biology that there are many tokens that appear in textual material that are related to each other, but not in any standard way or by any simple rules that have general applicability even in biology. It is our goal here to achieve some understanding of when two tokens can be said to be semantically related based on their similarity as strings of characters.

Thus for us morphological relationship will be a bit more general in that we wish to infer the relatedness of two strings based on the fact that they have a certain substring of characters on which they match. But we do not require to say exactly on what part of the matching substring their semantic relationship depends. In other words we do not insist on the identification of the smaller meaning bearing units or morphemes. Key to our approach is the ability to measure the contextual similarity between two token types as well as their similarity as strings. Neither kind of measurement is unique to our application. Contextual similarity has been studied and applied in morphology (Jacquemin, 1997; Schone and Jurafsky, 2000; Xu and Croft, 1998; Yarowsky and Wicentowski, 2000) and more generally (Means and others, 2004). String

BioNLP 2007: Biological, translational, and clinical language processing, pages 201–208,
Prague, June 2007. ©2007 Association for Computational Linguistics

similarity has also received much attention (Adamson and Boreham, 1974; Alberga, 1967; Damashek, 1995; Findler and Leeuwen, 1979; Hall and Dowling, 1980; Wilbur and Kim, 2001; Willett, 1979; Zobel and Dart, 1995). However, the way we use these two measurements is, to our knowledge, new. Our approach is based on a simple postulate: If two token types are similar as strings, but they are not semantically related because of their similarity, then their contextual similarity is no greater than would be expected for two randomly chosen token types. Based on this observation we carry out an analysis which allows us to assign a probability of relatedness to pairs of token types. This proves sufficient to generate a large repository of related token type pairs among which are the expected inflectional and derivationally related pairs and much more besides.

2 Methodology

We work with a set of 2,341,917 token types which are the unique token types that occurred throughout MEDLINE in the title and abstract record fields in November of 2006. These token types do not include a set of 313 token types that represent stop words and are removed from consideration. Our analysis consists of several steps.

2.1 Measuring Contextual Similarity

In considering the context of a token in a MEDLINE record we do not consider all the text of the record. In those cases when there are multiple sentences in the record the text that does not occur in the same sentence as the token may be too distant to have any direct bearing on the interpretation of the token and will in such cases add noise to our considerations. Thus we break the whole of MEDLINE into sentences and consider the context of a token to be the additional tokens of the sentence in which it occurs. Likewise the context of a token type consists of all the additional token types that occur in all the sentences in which it occurs. We used our own software to identify sentence boundaries (unpublished), but suspect that published and freely available methods could equally be used for this purpose. This produced 84,475,092 sentences over all of MEDLINE. While there is an advantage in the specificity that comes from considering context at the sentence level, this approach also gives rise to a problem. It is not uncommon for two terms to be related semantically, but to never occur in the same sentence. This will happen, for example, if one term is a misspelling of the other or if the two terms are alternate names for the same object. Because of this we must estimate the context of each term without regard to the occurrence of the other term. Then the two estimates can be compared to compute a similarity of context. This we accomplish using formulas of probability theory applied to our setting.

Let T denote the set of 2,341,917 token types we consider and let t_1 and t_2 be two token types we wish to compare. Then we define

$$\begin{aligned} p_c(t_1) &= \sum_{i \in T} p(t_1 \mid i)\, p(i) \text{ and} \\ p_c(t_2) &= \sum_{i \in T} p(t_2 \mid i)\, p(i) \end{aligned} \quad . \qquad (1)$$

Here we refer to $p_c(t_1)$ and $p_c(t_2)$ as contextual probabilities for t_1 and t_2, respectively. The expressions on the right sides in (1) are given the standard interpretations. Thus $p(i)$ is the fraction of tokens in MEDLINE that are equal to i and $p(t_1 \mid i)$ is the fraction of sentences in MEDLINE that contain i that also contain t_1. We make a similar computation for the pair of token types

$$\begin{aligned} p_c(t_1 \wedge t_2) &= \sum_{i \in T} p(t_1 \wedge t_2 \mid i)\, p(i) \\ &= \sum_{i \in T} p(t_1 \mid i)\, p(t_2 \mid i)\, p(i) \end{aligned} \quad . \qquad (2)$$

Here we have made use of an additional assumption, that given i, t_1 and t_2 are independent in their probability of occurrence. While independence is not true, this seems to be just the right assumption for our purposes. It allows our estimate of $p_c(t_1 \wedge t_2)$ to be nonzero even though t_1 and t_2 may never occur together in a sentence. In other words it allows our estimate to reflect what context would imply if there were no rule that says the same intended word will almost never occur twice in a single sentence, etc. Our contextual similarity is then the mutual information based on contextual probabilities

$$conSim(t_1,t_2) = \log\left(\frac{p_c(t_1 \wedge t_2)}{p_c(t_1)\,p_c(t_2)}\right) \tag{3}$$

There is one minor practical difficulty with this definition. There are many cases where $p_c(t_1 \wedge t_2)$ is zero. In any such case we define $conSim(t_1,t_2)$ to be -1000.

2.2 Measuring Lexical Similarity

Here we treat the two token types, t_1 and t_2 of the previous section, as two ASCII strings and ask how similar they are as strings. String similarity has been studied from a number of viewpoints (Adamson and Boreham, 1974; Alberga, 1967; Damashek, 1995; Findler and Leeuwen, 1979; Hall and Dowling, 1980; Wilbur and Kim, 2001; Willett, 1979; Zobel and Dart, 1995). We avoided approaches based on edit distance or other measures designed for spell checking because our problem requires the recognition of relationships more distant than simple misspellings. Our method is based on letter ngrams as features to represent any string (Adamson and Boreham, 1974; Damashek, 1995; Wilbur and Kim, 2001; Willett, 1979). If $t = "abcdefgh"$ represents a token type, then we define $F(t)$ to be the feature set associated with t and we take $F(t)$ to be composed of i) all the contiguous three character substrings "*abc*", "*bcd*", "*cde*", "*def*", "*efg*", "*fgh*"; ii) the specially marked first trigram $"abc!"$; and iii) the specially marked first letter $"a\#"$. This is the form of $F(t)$ for any t at least three characters long. If t consists of only two characters, say $"ab"$, we take i) $"ab"$; ii) $"ab!"$; and iii) is unchanged. If t consists of only a single character $"a"$, we likewise take i) "a"; ii) "a!"; and iii) is again unchanged. Here ii) and iii) are included to allow the emphasis of the beginning of strings as more important for their recognition than the remainder. We emphasize that $F(t)$ is a set of features, not a "bag-of-words", and any duplication of features is ignored. While this is a simplification, it does have the minor drawback that different strings, e.g., $"aaab"$ and $"aaaaab"$, can be represented by the same set of features.

Given that each string is represented by a set of features, it remains to define how we compute the similarity between two such representations. Our basic assumption here is that the probability $p(t_2 \mid t_1)$, that the semantic implications of t_1 are also represented at some level in t_2, should be represented by the fraction of the features representing t_1 that also appear in t_2. Of course there is no reason that all features should be considered of equal value. Let F denote the set of all features coming from all 2.34 million strings we are considering. We will make the assumption that there exists a set of weights $w(f)$ defined over all of $f \in F$ and representing their semantic importance. Then we have

$$p(t_2 \mid t_1) = \sum_{f \in F(t_1) \cap F(t_2)} w(f) \Big/ \sum_{f \in F(t_1)} w(f). \tag{4}$$

Based on (4) we define the lexical similarity of two token types as

$$lexSim(t_1,t_2) = \left(p(t_2 \mid t_1) + p(t_1 \mid t_2)\right)/2 \tag{5}$$

In our initial application of *lexSim* we take as weights the so-called inverse document frequency weights that are commonly used in information retrieval (Sparck Jones, 1972). If $N = 2{,}341{,}917,$ the number of token types, and for any feature f, n_f represents the number of token types with the feature f, the inverse document frequency weight is

$$w(f) = \log\left(\frac{N}{n_f}\right). \tag{6}$$

This weight is based on the observation that very frequent features tend not to be very important, but importance increases on the average as frequency decreases.

2.3 Estimating Semantic Relatedness

The first step is to compute the distribution of $conSim(t_1,t_2)$ over a large random sample of pairs of token types t_1 and t_2. For this purpose we computed $conSim(t_1,t_2)$ over a random sample of 302,515 pairs. This resulted in the value -1000, 180,845 times (60% of values). The remainder of the values, based on nonzero $p_c(t_1 \wedge t_2)$ are distributed as shown in Figure 1.

Let τ denote the probability density for $conSim(t_1,t_2)$ over random pairs t_1 and t_2. Let $Sem(t_1,t_2)$ denote the predicate that asserts that t_1 and t_2 are semantically related. Then our main assumption which underlies the method is

Postulate. For any nonnegative real number r

$$Q = \{conSim(t_1,t_2) \mid lexSim(t_1,t_2) > r \wedge \neg Sem(t_1,t_2)\} \quad (7)$$

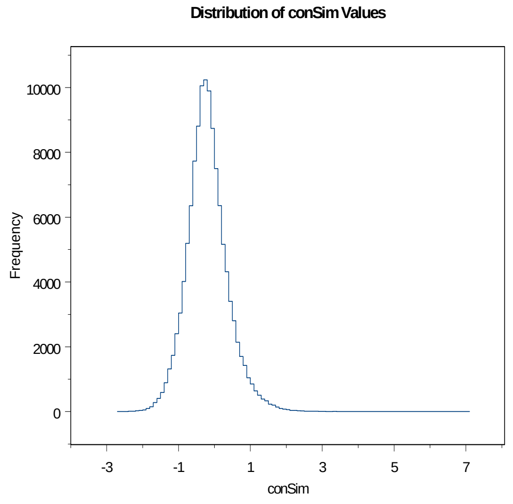

Figure 1. Distribution of *conSim* values for the 40% of randomly selected token type pairs which gave values above -1000, i.e., for which $p_c(t_1 \wedge t_2) > 0$.

has probability density function equal to τ.

This postulate says that if you have two token types that have some level of similarity as strings ($lexSim(t_1,t_2) > r$) but which are not semantically related, then $lexSim(t_1,t_2) > r$ is just an accident and it provides no information about $conSim(t_1,t_2)$.

The next step is to consider a pair of real numbers $0 \le r_1 < r_2$ and the set

$$S(r_1,r_2) = \{(t_1,t_2) \mid r_1 \le lexSim(t_1,t_2) < r_2\} \quad (8)$$

they define. We will refer to such a set as a *lexSim* slice. According to our postulate the subset of $S(r_1,r_2)$ which are pairs of tokens without a semantic relationship will produce *conSim* values obeying the τ density. We compute the *conSim* values and assume that all of those pairs that produce a *conSim* value of -1000 represent pairs that are unrelated semantically. As an example, in one of our computations we computed a slice $S(0.7,0.725)$ and found the lexSim value -1000 produced 931,042 times. In comparing this with the random sample which produced 180,845 values of -1000, we see that

$$931,042/180,845 = 5.148 \quad (9)$$

So we need to multiply the frequency distribution for the random sample (shown in Figure 1) by 5.148 to represent the part of the slice $S(0.7,0.725)$ that represents pairs not semantically related. This situation is illustrated in Figure 2. Two observations are important here. First, the two curves match almost perfectly along their left edges for *conSim* values below zero. This suggests that sematically related pairs do not produce *conSim* scores below about -1 and adds some credibility to our assumption that semantically related pairs do not produce *conSim* values of -1000. The second observation is that while the higher graph in Figure 2 represents all pairs in the *lexSim* slice and the lower graph all pairs that are not semantically related, we do not know which pairs are not semantically related. We can only estimate the probability of any pair at a particular *conSim* score level being semantically related. If we let Ψ represent the upper curve coming from the *lexSim* slice and Φ the lower curve coming from the random sample, then (10) represents the probability

$$p(x) = \frac{\Psi(x) - \Phi(x)}{\Psi(x)} \quad (10)$$

that a token type pair with a *conSim* score of x is a semantically related pair. Curve fitting or regression methods can be used to estimate p. Since it is reasonable to expect p to be a nondecreasing function of its argument, we use isotonic regression to make our estimates. For a full analysis we set

$$r_i = 0.5 + i \times 0.025 \quad (11)$$

and consider the set of *lexSim* slices $\{S(r_i, r_{i+1})\}_{i=0}^{20}$ and determine the corresponding set of probability functions $\{p_i\}_{i=0}^{20}$.

2.4 Learned Weights

Our initial step was to use the IDF weights defined in equation (6) and compute a database of all non-identical token type pairs among the 2,341,917 token types occurring in MEDLINE for which $lexSim(t_1,t_2) \ge 0.5$. We focus on the value 0.5 because the similarity measure *lexSim* has the

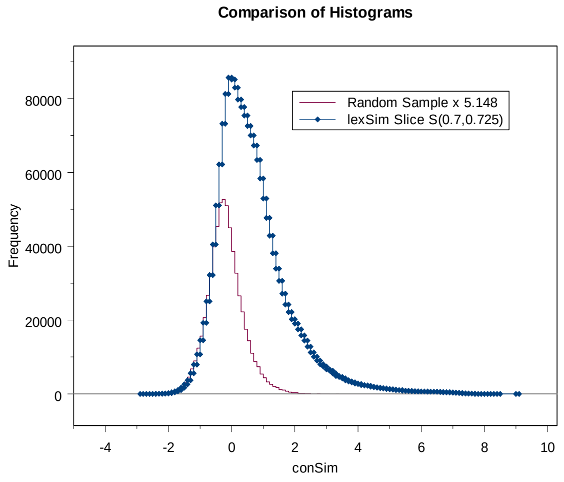

Figure 2. The distribution based on the random sample of pairs represents those pairs in the slice that are not semantically related, while the portion between the two curves represents the number of semantically related pairs.

property that if one of t_1 or t_2 is an initial segment of the other (e.g., 'glucuron' is an initial segment of 'glucuronidase') then $lexSim(t_1,t_2) \ge 0.5$ will be satisfied regardless of the set of weights used. The resulting data included the *lexSim* and the *conSim* scores and consisted of 141,164,755 pairs. We performed a complete slice analysis of this data and based on the resulting probability estimates 20,681,478 pairs among the 141,164,755 total had a probability of being semantically related which was greater than or equal to 0.7. While this seems like a very useful result, there is reason to believe the IDF weights used to compute *lexSim* are far from optimal. In an attempt to improve the weighting we divided the 141,164,755 pairs into C_{-1} consisting of 68,912,915 pairs with a *conSim* score of -1000 and C_1 consisting of the remaining 72,251,839 pairs. Letting $\vec{w}$ denote the vector of weights we defined a cost function

$$\Lambda(\vec{w}) = \sum_{(t_1,t_2)\in C_1} -\log\left(lexSim(t_1,t_2)\right) + \sum_{(t_1,t_2)\in C_{-1}} -\log\left(1-lexSim(t_1,t_2)\right) \qquad (12)$$

and carried out a minimization of Λ to obtain a set of learned weights which we will denote by $\vec{w}_0$. The minimization was done using the L-BFGS algorithm (Nash and Nocedal, 1991). Since it is important to avoid negative weights we associate a potential $v(f)$ with each ngram feature f and set

$$w(f) = \exp(v(f)). \qquad (13)$$

The optimization is carried out using the potentials.

The optimization can be understood as an attempt to make *lexSim* as close to zero as possible on the large set C_{-1} where $conSim = -1000$ and we have assumed there are no semantically related pairs, while at the same time making *lexSim* large on the remainder. While this seems reasonable as a first step it is not conservative as many pairs in C_1 will not be semantically related. Because of this we would expect that there are ngrams for which we have learned weights that are not really appropriate outside of the set of 141,164,755 pairs on which we trained. If there are such, presumably the most important cases would be those where we would score pairs with inappropriately high *lexSim* scores. Our approach to correct for this possibility is to add to the initial database of 141,164,755 pairs all additional pairs which produced a $lexSim(t_1,t_2) \ge 0.5$ based on the new weight set $\vec{w}_0$. This augmented the data to a new set of 223,051,360 pairs with *conSim* scores. We then applied our learning scheme based on minimization of the function Λ to learn a new set of weights $\vec{w}_1$. There was one difference. Here and in all subsequent rounds we chose to define C_{-1} as all those pairs with

$conSim(t_1,t_2) \le 0$ and C_1 those pairs with $conSim(t_1,t_2) > 0$. We take this to be a conservative approach as one would expect semantically related pairs to have a similar context and satisfy $conSim(t_1,t_2) > 0$ and graphs such as Figure 2 support this. In any case we view this as a conservative move and calculated to produce fewer false positives based on *lexSim* score recommendations of semantic relatedness. We actually go through repeated rounds of training and adding new pairs to the set of pairs. This process is convergent as we reach a point where the weights learned on the set of pairs does not result in the addition of a significant amount of new material. This happened with weight set $\vec{w}_4$ and a total accumulation of 440.4 million token type pairs.

Table 1. Number of token pairs and the level of their predicted probability of semantic relatedness found with three different weight sets.

Weight Set	Prob. Semantically Related ≥ 0.7	Prob. Semantically Related ≥ 0.8	Prob. Semantically Related ≥ 0.9
$\vec{w}_4$	36,173,520	22,381,318	10,805,085
Constant	34,667,988	20,282,976	8,607,863
IDF	31,617,441	18,769,424	8,516,329

3 Probability Predictions

Based on the learned weight set $\vec{w}_4$ we performed a slice analysis of the 440 million token pairs on which the weights were learned and obtained a set of 36,173,520 token pairs with predicted probabilities of being semantically related of 0.7 or greater. We performed the same slice analysis on this 440 million token pair set with the IDF weights and the set of constant weights all equal to 1. The results are given in Table 1. Here it is interesting to note that the constant weights perform substantially better than the IDF weights and come close to the performance of the $\vec{w}_4$ weights. While the $\vec{w}_4$ predicted about 1.5 million more relationships at the 0.7 probability level, it is also interesting to note that the difference between the $\vec{w}_4$ and constant weights actually increases as one goes to higher probability levels so that the learned weights allow us to

Table 2. A table showing 30 out of a total of 379 tokens predicted to be semantically related to 'lacz' and the estimated probabilities. Ten entries are from the beginning of the list, ten from the middle, and ten from the end. Breaks where data was omitted are marked with asterisks.

Probability Semantic Relation	Token 1	Token 2
0.973028	lacz	'lacz
0.975617	lacz	010cblacz
0.963364	lacz	010cmvlacz
0.935771	lacz	07lacz
0.847727	lacz	110cmvlacz
0.851617	lacz	1716lacz
0.90737	lacz	1acz
0.9774	lacz	1hsplacz
0.762373	lacz	27lacz
0.974001	lacz	2hsplacz
***	***	***
0.95951	lacz	laczalone
0.95951	lacz	laczalpha
0.989079	lacz	laczam
0.920344	lacz	laczam15
0.903068	lacz	laczamber
0.911691	lacz	laczatttn7
0.975162	lacz	laczbg
0.953791	lacz	laczbgi
0.995333	lacz	laczbla
0.991714	lacz	laczc141
***	***	***
0.979416	lacz	ul42lacz
0.846753	lacz	veroicp6lacz
0.985656	lacz	vglacz1
0.987626	lacz	vm5lacz
0.856636	lacz	vm5neolacz
0.985475	lacz	vtkgpedeltab8rlacz
0.963028	lacz	vttdeltab8rlacz
0.993296	lacz	wlacz
0.990673	lacz	xlacz
0.946067	lacz	zflacz

predict over 2 million more relationships at the 0.9 level of reliability. This is more than a 25% increase at this high reliability level and justifies the extra effort in learning the weights.

Table 3. A table showing 30 out of a total of 96 tokens predicted to be semantically related to 'nociception' and the estimated probabilities. Ten entries are from the beginning of the list, ten from the middle, and ten from the end. Breaks where data was omitted are marked with asterisks.

Probability Semantic Relation	Token 1	Token 2
0.727885	nociception	actinociception
0.90132	nociception	actinociceptive
0.848615	nociception	anticociception
0.89437	nociception	anticociceptive
0.880249	nociception	antincociceptive
0.82569	nociception	antinoceiception
0.923254	nociception	antinociceptic
0.953812	nociception	antinociceptin
0.920291	nociception	antinociceptio
0.824706	nociception	antinociceptions
***	***	***
0.802133	nociception	nociceptice
0.985352	nociception	nociceptin
0.940022	nociception	nociceptin's
0.930218	nociception	nociceptine
0.944004	nociception	nociceptinerg
0.882768	nociception	nociceptinergic
0.975783	nociception	nociceptinnh2
0.921745	nociception	nociceptins
0.927747	nociception	nociceptiometric
0.976135	nociception	nociceptions
***	***	***
0.88983	nociception	subnociceptive
0.814733	nociception	thermoantinociception
0.939505	nociception	thermonociception
0.862587	nociception	thermonociceptive
0.810878	nociception	thermonociceptor
0.947374	nociception	thermonociceptors
0.81756	nociception	tyr14nociceptin
0.981115	nociception	visceronociception
0.957359	nociception	visceronociceptive
0.862587	nociception	withnociceptin

A sample of the learned relationships based on the $\vec{w}_4$ weights is contained in

Table 2 and Table 3. The symbol 'lacz' stands for a well known and much studied gene in the E. coli bacterium. Due to its many uses it has given rise to myriad strings representing different aspects of molecules, systems, or methodologies derived from or related to it. The results are not typical of the inflectional or derivational methods generally found useful in studying the morphology of English. Some might represent misspellings, but this is not readily apparent by examining them. On the other hand 'nociception' is an English word found in a dictionary and meaning "a measurable physiological event of a type usually associated with pain and agony and suffering" (Wikepedia). The data in Table 3 shows that 'nociception' is related to the expected inflectional and derivational forms, forms with affixes unique to biology, readily apparent misspellings, and foreign analogs.

4 Discussion & Conclusions

There are several possible uses for the type of data produced by our analysis. Words semantically related to a query term or terms typed by a search engine user can provide a useful query expansion in either an automatic mode or with the user selecting from a displayed list of options for query expansion. Many misspellings occur in the literature and are disambiguated in the token pairs produced by the analysis. They can be recognized as closely related low frequency-high frequency pairs. They may allow better curation of the literature on the one hand or improved spelling correction of user queries on the other. In the area of more typical language analysis, a large repository of semantically related pairs can contribute to semantic tagging of text and ultimately to better performance on the semantic aspects of parsing. Also the material we have produced can serve as a rich source of morphological information. For example, inflectional and derivational transformations applicable to the technical language of biology are well represented in the data.

There is the possibility of improving on the methods we have used, while still applying the general approach. Either a more sensitive *conSim* or *lexSim* measure or both could lead to superior results. While it is unclear to us how *conSim* might be improved, it seems there is more potential with *lexSim*. *lexSim* treats features as basically independent contributors to the similarity of token types and this is not ideal. For example the feature 'hiv' usually refers to the human immunodeficiency virus. However, if 'ive' is also a feature of the token we may well be dealing with the word 'hive' which has no relation to a human immunodeficiency virus. Thus a more complicated model of the lexical similarity of strings could result in improved recognition of semantically related strings.

In future work we hope to investigate the application of the approach we have developed to multi-token terms. We also hope to investigate the possibility of more sensitive *lexSim* measures for improved performance.

Acknowledgment This research was supported by the Intramural Research Program of the National Center for Biotechnology Information, National Library of Medicine, NIH, Bethesda, MD, USA.

References

Adamson, G. W., and Boreham, J. 1974. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. Information Storage and Retrieval, 10: 253-260.

Alberga, C. N. 1967. String similarity and misspellings. Communications of the ACM, 10: 302-313.

Damashek, M. 1995. Gauging similarity with n-grams: Language-independent categorization of text. Science, 267: 843-848.

Findler, N. V., and Leeuwen, J. v. 1979. A family of similarity measures between two strings. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1: 116-119.

Freitag, D. 2005. Morphology Induction From Term Clusters, 9th Conference on Computational Natural Language Learning (CoNLL): Ann Arbor, Michigan, Association for Computational Linguistics.

Hall, P. A., and Dowling, G. R. 1980. Approximate string matching. Computing Surveys, 12: 381-402.

Jacquemin, C. 1997. Guessing morphology from terms and corpora, in Belkin, N. J., Narasimhalu, A. D., and Willett, P., editors, 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Philadelphia, PA, ACM Press, p. 156-165.

Jurafsky, D., and Martin, J. H. 2000. Speech and Language Processing: Upper Saddle River, New Jersey, Prentice Hall.

Means, R. W., Nemat-Nasser, S. C., Fan, A. T., and Hecht-Nielsen, R. 2004. A Powerful and General Approach to Context Exploitation in Natural Language Processing, HLT-NAACL 2004: Workshop on Computational Lexical Semantics Boston, Massachusetts, USA, Association for Computational Linguistics.

Monson, C. 2004. A framework for unsupervised natural language morphology induction, Proceedings of the ACL 2004 on Student research workshop: Barcelona, Spain, Association for Computational Linguistics.

Nash, S. G., and Nocedal, J. 1991. A numerical study of hte limited memory BFGS method and hte truncated-Newton method for large scale optimization. SIAM Journal of Optimization, 1: 358-372.

Schone, P., and Jurafsky, D. 2000. Knowledge-free induction of morphology using latent semantic analysis, Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning - Volume 7: Lisbon, Portugal, Association for Computational Linguistics.

Sparck Jones, K. 1972. A statistical interpretation of term specificity and its application in retrieval. The Journal of Documentation, 28: 11-21.

Wicentowski, R. 2004. Multilingual Noise-Robust Supervised Morphological Analysis using the Word-Frame Model, SIGPHON: Barcelona, Spain, Association for Computational Linguistics.

Wilbur, W. J., and Kim, W. 2001. Flexible phrase based query handling algorithms, in Aversa, E., and Manley, C., editors, Proceedings of the ASIST 2001 Annual Meeting: Washington, D.C., Information Today, Inc., p. 438-449.

Willett, P. 1979. Document retrieval experiments using indexing vocabularies of varying size. II. Hashing, truncation, digram and trigram encoding of index terms. Journal of Documentation, 35: 296-305.

Xu, J., and Croft, W. B. 1998. Corpus-based stemming using cooccurrence of word variants. ACM TOIS, 16: 61-81.

Yarowsky, D., and Wicentowski, R. 2000. Minimally supervised morphological analysis by multimodal alignment, Proceedings of the 38th Annual Meeting on Association for Computational Linguistics: Hong Kong, Association for Computational Linguistics.

Zobel, J., and Dart, P. 1995. Finding approximate matches in large lexicons. Software-Practice and Experience, 25: 331-345.