Using Similarity Metrics For Terminology Recognition

Jonathan Butters, Fabio Ciravegna
Department of Computer Science
The University Of Sheffield
Sheffield, S1 4DP, UK
E-mail: j.butters@dcs.shef.ac.uk, f.ciravegna@dcs.shef.ac.uk

Abstract

In this paper we present an approach to terminology recognition whereby a sublanguage term (e.g. an aircraft engine component term extracted from a maintenance log) is matched to its corresponding term from a pre-defined list (such as a taxonomy representing the official break-down of the engine). Terminology recognition is addressed as a classification task whereby the extracted term is associated to one or more potential terms in the official description list via the application of string similarity metrics. The solution described in the paper uses dynamically computed similarity cut-off thresholds calculated on the basis of modeling a noise curve. Dissimilar string matches form a Gaussian distributed noise curve that can be identified and extracted leaving only mostly similar string matches. Dynamically calculated thresholds are preferable over fixed similarity thresholds as fixed thresholds are inherently imprecise, that is, there is no similarity boundary beyond which any two strings always describe the same concept.

1. Introduction

Although objects are discrete, the terms people use to describe objects are usually not. Different communities can use different descriptions to refer to the same objects. Phenomena such as sublanguage (Harris, 1968) influence not only the syntax of languages but its terminology as well. Synonyms introduce a 'many-to-one' relationship between the linguistic descriptions and the concept being described. Technical domains are particularly affected by the multi-word synonimity issue. Technical terms can be very complex from a linguistic point of view; an average technical domain has thousands of official terms and several thousands of synonyms can be found. For example, (Ciravegna, 1995) discusses how in the car domain, technical lists can contain around 30,000 terms and synonymic terms can be represented in Italian via complex noun phrases up to ten words long. Studies in the biological domain have shown a similar situation (Krauthammer & Nenadić, 2004). There are a number of issues that affect terminology recognition. From simple orthographical differences (change in case, hyphenation, use of Arabic Vs Roman numerals, etc.) to more complex issues like abbreviations and acronyms, as well as the use of synonymous words. While orthographical normalisation is a fairly simple process that can be performed using some regular grammars, the more complex cases require the use of more sophisticated techniques. Soft string matching (e.g. using string metrics) returns a similarity score between two terms that are not identical. If the process is applied between one target term and a dictionary, it can return a set of candidate terms, ranked according to their matching score.

Similarity metrics can be tuned to a specific task by modifying some parameters (typically the cost of edit operation), either manually or automatically. Some automatic approaches use only positive examples (i.e. synonymous strings) to tune the cost of editing (Yeganova et al. 2004; Cohen & Minkov 2006). Other, more recent approaches use also negative examples to learn (Bilenko & Mooney 2003; McCallum et al. 2005; Tsurouka et al. 2007) hence enabling the adaption to features that discriminate negative from positive examples. Modeling the cost function is just one of the steps, though. Once the metrics has been identified, there is still the issue of applying the metrics and identifying the thresholds to discriminate spurious matches from correct matches. The use of a fixed threshold is adopted by many authors. Unfortunately a fixed threshold does not take into account the actual distribution of matches in each single test. Some terms have a number of similar terms (i.e. they will require a restrictive threshold), while others have just a few and can allow a less restrictive threshold. However, applying strict thresholds can negatively affect recall when the results of the match are very sparse.

In this paper we present an approach to terminology recognition that departs from the use of a fixed threshold. Instead, it identifies the most appropriate threshold case by case (i.e. every time a term is tested against a bank of potential terms). In practical terms, we define a statistical filter which inspects the distribution of scores of the matches and models the noise in the distribution. As noise is a random variable, it takes on a Gaussian distribution. As 100% of a normally distributed variable falls below 4 standard deviations above the mean, noise can be identified as all matches that achieve a score below the value of approximately four standard deviations, and those terms that score higher are deemed to be correct matches; hence the threshold is set at 4 standard deviations above the mean of the distribution of scores for an individual query. This means that every time a term is tested against a bank of terms, an individual optimal threshold is identified.

We applied the filter on a set of experiments in the aerospace domain where the scores returned by a standard library of distance metrics were filtered of noise.

This paper is organized as follows: section 2.1 describes the investigation carried out in selecting a suitable similarity metric capable of effectively modeling noise. Section 2.2 describes how noise is identified and removed. Section 2.3 describes a how the methodology operates on strings of various length and provides a practical example. Section 3 describes the results.

2. Investigation

Observations were made by conducting experiments on two datasets of aerospace terminology terms. These consisted of a list of 298 component terms automatically extracted [1] from reports written by engineering technicians and a list of 513 official engine component terms. SimMetrics[2], a Java library containing 23 string similarity metrics was used to compare the lists of terms. SimMetrics has the functionality to produce normalized similarity coefficients where all metrics output a bound result between 0 (totally dissimilar) and 1 (totally identical).

2.1. Metric Selection

The first step in terminology recognition is to select the best string similarity metrics for the case at hand. Other authors (Cohen et al. 2003) have approached the issue by investigating a selection of string similarity metrics used by various communities, including statistics, AI and databases to ascertain which similarity metrics performed most accurately when comparing strings from datasets including data such as names and addresses. Different metrics from different communities were investigated since each community had devised the problem of matching strings differently. It was found that a hybrid scheme combining TF-IDF weighting, with the Jaro-Winkler string-distance metric, was the overall best performing method in terms of speed and accuracy.

In order to select the most effective string similarity metrics, it was essential to analyze the strengths and weaknesses of the standard, unmodified string distance metrics included within SimMetrics. An experiment was devised and performed whereby every combination of the 298 unique extracted component term strings were compared using each of the metrics within SimMetrics, this resulted in ${}^{298}C_2$, = 44,253 similarity comparisons for each metric.

$$\frac{n!}{k!(n-k)!} = \frac{298!}{2!(298-2)!} = 44253$$

Equation 1: Number of string comparisons for each metric

By ordering the pairs of strings by similarity score, it was seen that only the similarities above certain values appeared to have meaningful string matches, for example the Jaro metric assigns the two strings "*p2/t2 probe*" and "*p2t2 probe*" a similarity of 0.969697, this is a meaningful match as the two strings are similar, whereas "*oil pump*" and "*starter (post sb 80-d207))*" which scores a similarity of 0.4433333 is not a meaningful match as the two strings are unrelated. The similarity value between the mostly similar and mostly dissimilar matches was named the Terminator and is a different value for each metric.

By totaling the number of pairs of strings that fell into 0.01 increments in similarity score a graphical representation of the distribution of similarity scores is given, according to the metric used these distributions were either Gaussian, Dirac or a mixture of the two.

- **Dirac distributions** occurred frequently for vector based similarity metrics such as Block distance, Jaccard similarity and Cosine similarity, and for overlap based similarity metrics such as Overlap coefficient and Dice's coefficient. Dirac distributions occurred when many pairs of strings took on relatively few similarity scores. These occur for example for pairs of strings where both strings have m words, where n words are in common, these pairs of strings would attain a similarity score of n/m no matter how long the common words are.

- **Gaussian distributions** occurred primarily for edit distance similarity metrics such as the Smith-Waterman, Jaro-Winkler and Levenshtein metrics. String similarity scores are randomly distributed and take on a Gaussian distribution due to the wide ranging edit costs incurred when comparing one dissimilar string with another.

Our approach is to identify good matches by modeling (and then filtering out) noise. As noise is generally a randomly distributed variable about a mean value, it was decided to focus a metric that provides a Gaussian distribution. This is because Gaussian distributions characterize randomly distributed variables; hence they are able to model the behavior of noise. This decision is in accordance with what proposed by (Cohen et al, 2003) who suggests a TF-IDF weighting scheme with the Jaro-Winkler metric (i.e. a Gaussian distribution); however, we focused on the use of the Levenshtein metric, because an investigation into the Jaro-Winkler metric concluded that this metric gave consistently relatively high scores to unrelated strings. The Terminator value for the Jaro-Winkler metric was 0.637 and the mean similarity score was 0.51 (Figure 1).

[1] Using a separate Named Entity Recognition tool

[2] SimMetrics available at
http://www.dcs.shef.ac.uk/~sam/simmetrics.html

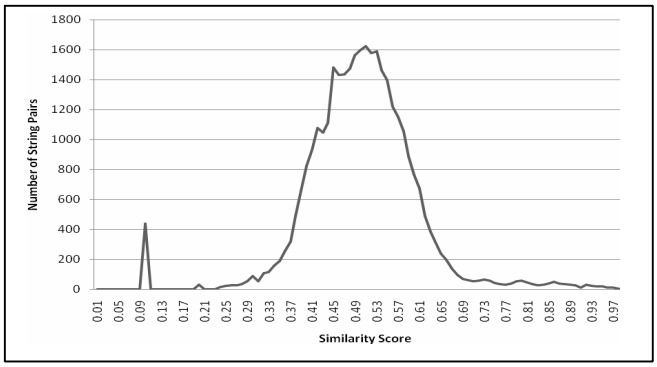

Figure 1: Jaro-Winkler distribution

Compared with the Levenshtein metric where the Terminator for the same set of strings was 0.33 with a mean similarity of 0.17 (Figure 2):

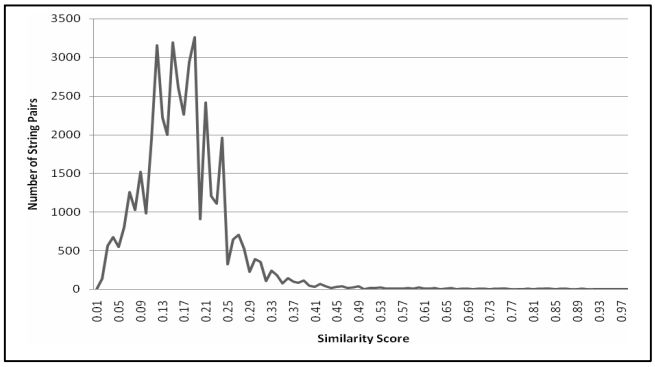

Figure 2: Levenshtein distribution

This means that differentiation between a meaningless match score and a meaningful match score is less apparent with the Jaro-Winkler metric than that for the same set of strings using the Levenshtein metric. This is seen when a search is performed using the string "*oil pressure transmitter*". The mean similarity score using the Jaro-Winkler metric was 0.5918. The same search string attained a mean similarity of 0.2438 with the Levenshtein metric. By analyzing the similarities assigned to string pairs by the Jaro-Winkler metric it was seen that there was little difference in similarity between the highest score of a meaningless result and the score of the lowest meaningful result.

The Levenshtein metric also has the additional advantages of being able to model most of the requirements for terminology recognition in the aerospace domain, i.e.:

- It does not give excessively low scores to pairs of strings that include terms that only differ by one or two characters but otherwise may be related; for example "*air valve*" and "*air valves*" score 0.5 with the Overlap Coefficient, but score 0.9 with the Levenshtein.

- It gives inherent preference to word order, for example although "*oil filter*" and "*filter oil*" contain the same two words, they refer to two very different concepts. The Levenshtein metric assigns a similarity score of 0.199.

- It allows the greatest flexibility as more specific similarity metrics can be used for post processing.

2.2. Noise Detection and Removal

As expected, it was noted experimentally that meaningless matches formed a Gaussian noise curve, while meaningful matches are located above the noise curve closer to a similarity of 1. However, it was also noted that the peak of the curve occurred within a wide similarity boundary depending on the relative length of the strings being compared, this makes the application of a fixed cut-off threshold ineffective because:

- A threshold that is set too high would always filter out the noise curve, but would risk being too restrictive and could also filter out poorly scoring relevant matches. This would provide high precision but low recall.

- A threshold that is set too low would let a large number of relevant matches through, but would also let through a portion of the noise curve. This leads to low precision.

Our proposal is therefore to apply dynamic thresholds that are sensitive to the location and geometry of the noise curve. A statistical approach such as this is preferable over setting an arbitrary cut off as it scales well with the size of the source data. When the top ten matches from a candidate list of some hundreds of terms are considered, it is likely that there will be less than ten relevant matches, meaning that all relevant matches are returned along with some irrelevant matches. However, when the list is much larger, for example 1000, it is likely that there will be more than 10 significant matches meaning that some relevant matches are missed. It may also be the case that there are no relevant matches, this means all results will constitute a Gaussian distribution and would therefore be disregarded with the proposed method.

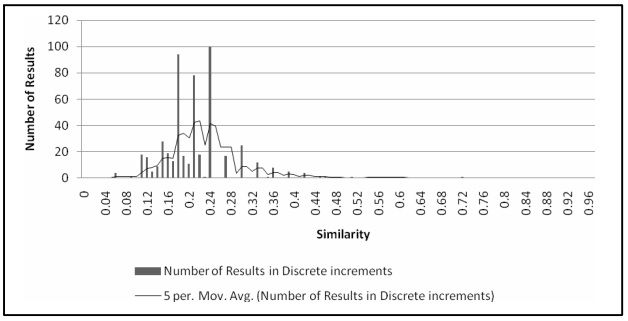

Figure 3: Results distribution for "*fuel pump drive and idler housing*"

The discrete results distribution graphs produced by Levenshtein metric (Example Figure 3) can be modeled with a Gaussian distribution by the following equations:

$$Z = \frac{x - \mu}{\sigma}$$

Equation 2: Standard Normal Random variable Z

$$\mu = \frac{\sum x}{n}$$

Equation 3: Mean, μ

$$\sigma = \sqrt{\frac{\sum \left(x - \bar{x}\right)^2}{n}}$$

Equation 4: Standard Deviation σ

Because it is only the edit costs associated with transforming one randomly dissimilar string into another that causes the Gaussian noise distribution, by detecting and removing the randomly distributed similarity scores the significant and more meaningful matches will remain.

100% of a randomly distributed variable will fall below approximately 4 standard deviations above the mean; therefore the value for which scores below should be disregarded can be calculated with Equation 5.

$$CutOffSimilarity = \left(4 \cdot \sqrt{\frac{\sum \left(x - \bar{x}\right)^2}{n}}\right) + \left(\frac{\sum x}{n}\right)$$

Equation 5: Cut off similarity is four standard deviations above the mean

For example, consider the sublanguage term "*external gearbox drive shaft and shroud*", when compared against the terms in the official list, the distribution of similarity scores is shown in Figure 4.

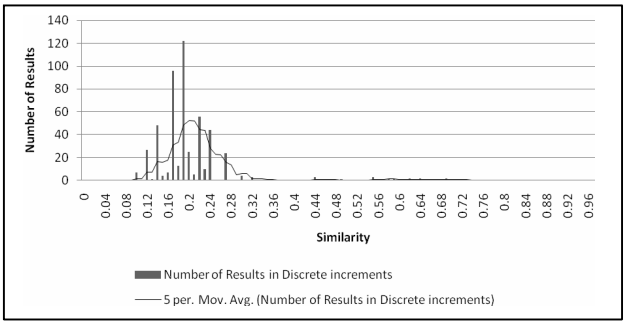

Figure 4: Discrete results distribution term "*external gearbox drive shaft and shroud*"

The Gaussian curve for term "*external gearbox drive shaft and shroud*" has standard deviation σ = 0.085362 and mean μ = 0.210: by applying equation 4 it is possible to derive that:

$$CutOffSimilarity = \left(4 \times 0.085362\right) + \left(0.210\right) = 0.514$$

By setting the cut off similarity to 0.551 noise matches are severely attenuated, the list of candidate strings in the official term list is reduced from 513 to the 10 most likely and meaningful matches.

2.3. Shorter Terms

To investigate the performance of automatically calculating the cutoff threshold with shorter strings, the list of 298 component strings was split into individual words (minimum string length was 1, maximum string length 48), these individual words were then compared in turn against the original list of component strings and for each set of comparisons the mean similarity, standard deviation and word length was recorded. This is summarized in Figures 5 and 6.

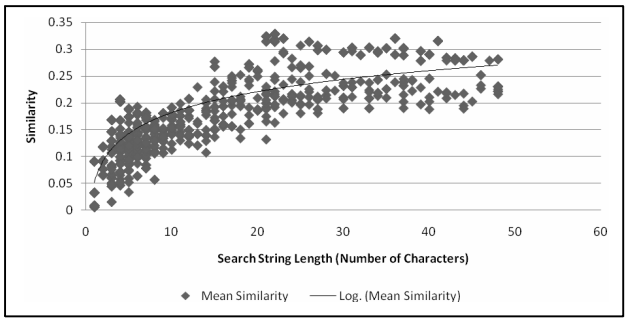

Figure 5: Mean similarity scores for strings of various lengths

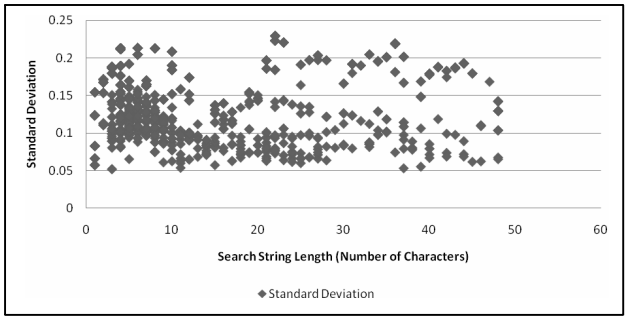

Figure 6: Standard Deviations for strings of various lengths

As can be seen in Figure 5, the mean similarity initially increases as larger search strings are used. This is because shorter single word search strings, such as "lp", "*air*" and "*fuel*", only constitute a small percentage of longer and more typical component terms such as "*lp axial compressor*", "*oil cooled air vanes*" and "*fuel metering unit*". As longer and more realistic search strings are used (such as "*support assy of lp turbine exhaust*"), the mean similarity became fairly constant and was located between approximately 0.17 and 0.325.

It can be seen in Figure 6, that the standard deviation did not vary greatly, and was distributed fairly uniformly between the values of 0.05 and 0.25. This showed that the same basic distribution shape is seen in all searches.

2.3.1. Practical Effect of Shorter Strings

The effects searching for successively longer strings have on the Gaussian distribution are now examined.

The discrete results distribution, where the normally distributed results, mean similarity, calculated cut-off similarity and any similarity scores above the cut off similarity are indicated as in Figure 7.

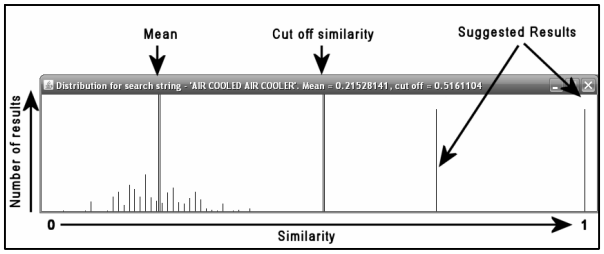

Figure 7: Discrete Results Distribution legend

The mean is located at the first full height double line and the cut off similarity is located at the second full height double line. Suggested results (results whose similarity falls above the cut off similarity) are shown by lines that are ¾ height. In Figure 7, the search string was "*air cooled air cooler*", the mean is shown to be 0.2153, cut off similarity is 0.516 (shown in the window title) and there are two unique suggested results, "*air cooled air cooler*" and "*fuel cooled air cooler*".

To investigate the effects of string length on the distribution of noise and number of suggested results, searches for "*air*", "*air cooled*", "*air cooled air*" and "*air cooled air cooler*" were made.

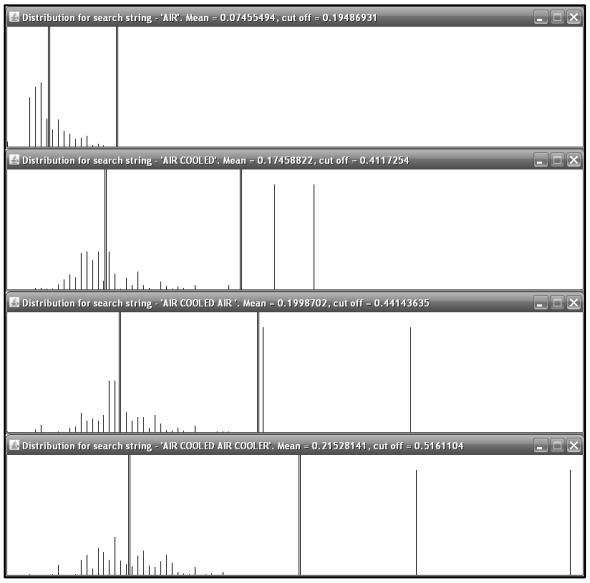

Figure 8: Similarity distributions and automatically calculated cut-off thresholds for search strings of various lengths

The top window in Figure 8 shows the results distribution for the search string "*air*". The resultant matches all scored a similarity below 0.194. All the results scored low similarities as the search string "*air*" contains 3 characters, most of the strings in the official terms list are multi-word strings with a mean length of 32 characters, and hence the string "*air*" is a very small percentage of any of the strings in the pool and would score a low similarity. In the top window it can be seen that there were several results close to the automatically calculated cut-off value marker of 0.194. Short strings in which "*air*" is a substring would score highly, for example the string "*air tube*" scores highly with a similarity of 0.182.

In the second window, the search string "*air cooled*" was searched for. The mean has increased to 0.175 and the automatically calculated cut-off to 0.411. The results distribution now demonstrates a strong Gaussian distribution pattern. Two results were above the cut-off value, the highest matching of these was "*air cooled air cooler*" and the lower "*fuel cooled oil cooler*".

The third window shows the distribution for "*air cooled air*". The noise distribution has become more defined. The mean and cut off values have increased. There are still two results above the cut off value and these are the same results as those for the search term "*air cooled*", however one result has gained similarity and one has lost similarity. This can be expected as the search string "*air cooled air*" constitutes a larger percentage of "*air cooled air cooler*" than the previous search string ("*air cooled*"), and a smaller percentage of "*fuel cooled oil cooler*".

In the final window, the search string "*air cooled air cooler*" was searched for. The mean and cut off value once again increased. One result scored a perfect match. The second significant match ("*fuel cooled oil cooler*") had a slightly higher score than the previous search string; this is because some of the new characters added to the search string are common.

3. Results

To evaluate this approach, the 298 extracted sublanguage terms were compared a list of 513 official terms. Only 112 of the extracted terms had similar enough terms in the official list to have relevant terms automatically suggested. An exhaustive manual examination of the official list showed that the remaining 186 extracted terms did not have corresponding terms in the official list (the official list contains 513 terms from just one section of one model of engine).

Recall is the ratio of the number of correct (relevant) terms returned to the total number of correct terms in the official term list. Recall gives an indication as to how much of the total relevant information has been retrieved.

Precision is the ratio of the number of correct (relevant) terms returned to the total number of irrelevant and relevant terms retrieved. It is usually expressed as a percentage. Precision gives an indication of how much of the information retrieved from a particular search is relevant.

Precision and recall for all search terms was calculated for n = 1 to 5, where n is number of suggested results. This was done to simulate how the system would perform if it were to be used automatically (where n=1) as well as if a user was allowed to choose results from a list of up to five suggestions. Calculating precision considering the first n results is called 'Precision at n'.

n	Recall at n	Precision at n
1	86.84%	86.84%
2	90.13%	88.82%
3	92.67%	89.04%
4	98.67%	91.56%
5	99.40%	92.08%

Table 1: Recall and Precision at n = 1 to 5.

The experiment showed that five suggestions were enough to return 99.40% of the relevant components and that 92.08% of these would be correct. This number of suggestions would be sufficiently low for a knowledgeable user to quickly select the relevant components from the suggestions. A larger number of suggestions would increase the time taken to select the relevant terms without significant increase in recall.

4. Conclusions

The results show that the method of selecting most relevant string similarities by disregarding those that form a noise pattern is effective in automatically detecting only the most similar terms and also finding sublanguage terms for which no corresponding official term exists. By allowing a system based on this method to suggest up to a maximum of five results 99.40% of relevant terms are returned, furthermore as the cut off similarity is automatically set on each query there are often less than five results returned, as there are often less than five relevant terms in the official list.

Although the datasets used in the experiments discussed in this paper belong in the aerospace domain, the resultant approach and methodology for Terminology Recognition can be applied to virtually any field requiring the comparison of strings. The authors have successfully applied the methodology to the recognition of people's names.

5. References

Bernardini, S & Baroni, M. (2005). Spotting translationese: A corpus-driven approach using support vector machines. *Proceedings of Corpus Linguistics 2005*, page 9.

Bilenko, M. and Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. *In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, pages 39–48.

Ciravegna, F. (1995). Understanding Messages in a Diagnostic Domain. *In Information Processing and Management*, Vol. 31, No. 5, pp 687-701, 1995

Cohen, W. Ravikumar, P. Fienberg, S. (2003). A Comparison of String Metrics for Matching Names and Records. pages 1, 2, 3.

Cohen, W. W. and Minkov, E. (2006). A graph-search framework for associating gene identifies with documents. *BMC Bioinformatics*, 7(440).

Fang, H., Murphy, K., Jin, Y., Kim, J. S., and White, P. S. (2006). Human gene name normalization using text matching with automatically extracted synonym dictionaries. *In Proceedings of BioNLP'06*

Harris, Z. (1968). Mathematical Structures of Language. *John Wiley & Sons, New York*.

Krauthammer, M., Nenadić, G.. (2004). Term Identification in the Biomedical Literature, *Journal of Biomedical Informatics, Volume 37, Issue 6,*

Levenshtein, V. I. (1965). Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1(1), 8–17.

Martin, L.E. (1990). Knowledge Extraction. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 252--262.

McCallum, A., Bellare, K., and Pereira, F. (2005). A conditional random field for discriminatively-trained finite-state string edit distance. *In Proceedings of Conference on Uncertainty in AI (UAI).*

Tsuruoka, Y,. McNaught, J. Tsujii, J. Ananiadou, S. (2007). Learning string similarity measures for gene/protein name dictionary look-up using logistic regression. *In Bioinformatics Advance Access*.

Winkler, W. E. (1999). The state of record linkage and current research problems. *Technical report, Statistical Research Division, U.S. Bureau of the Census*.

Yeganova, L., Smith, L., and Wilbur,W. J. (2004). Identification of related gene/protein names based on an hmm of name variations. *Computational Biology and Chemistry.*