Adaptive String Similarity Metrics for Biomedical Reference Resolution

Ben Wellner[†*]
[*]The MITRE Corporation
202 Burlington Rd
Bedford MA 01730
wellner@mitre.org

José Castaño[†] **and James Pustejovsky**[†]
[†]Computer Science Department
Brandeis University
Waltham MA 02454
{jcastano,jamesp}@cs.brandeis.edu

Abstract

In this paper we present the evaluation of a set of string similarity metrics used to resolve the mapping from strings to *concepts* in the UMLS MetaThesaurus. String similarity is conceived as a single component in a full Reference Resolution System that would resolve such a mapping. Given this qualification, we obtain positive results achieving 73.6 F-measure (76.1 precision and 71.4 recall) for the task of assigning the correct UMLS concept to a given string. Our results demonstrate that adaptive string similarity methods based on Conditional Random Fields outperform standard metrics in this domain.

1 Introduction

1.1 String Similarity and Reference Resolution

String similarity/matching algorithms are used as a component in *reference resolution* algorithms. We use reference resolution in a broad sense, which includes any of the following aspects:

a. Intra-document noun phrase reference resolution.

b. Cross-document or corpus reference resolution.

c. Resolution of entities found in a corpus with databases, dictionaries or other external knowledge sources. This is also called semantic integration, e.g., (Li et al., 2005), reference grounding, e.g., (Kim and Park, 2004) or normalization, e.g., (Pustejovsky et al., 2002; Morgan et al., 2004).

The last two aspects of reference resolution are particularly important for information extraction, and the interaction of reference resolution with information extraction techniques (see for example Bagga (1998)). The extraction of a particular set of entities from a corpus requires reference resolution for the set of entities extracted (e.g., the EDT task in ACE[1]), and it is apparent that there is more variation in the cross-document naming conventions than in a single document.

The importance of edit distance algorithms has already been noticed, (Müller et al., 2002) and the importance of string similarity techniques in the biomedical domain has also been acknowledged, e.g., (Yang et al., 2004).

String similarity/matching algorithms have also been used extensively in related problems such as Name databases and similar problems in structured data, see (Li et al., 2005) and references mentioned therein.

The problem of determining whether two similar strings may denotate the same *entity* is particularly challenging in the biomedical literature. It has already been noticed (Cohen et al., 2002) that there is great variation in the naming conventions, and noun phrase constructions in the literature. It has also been noticed that bio-databases are hardly ever updated with the names in the literature (Blaschke

[1]http://www.nist.gov/speech/tests/ace/

et al., 2003). A further complication is that the actual *mentions* found in text are more complex than just *names* - including descriptors, in particular. Finally, ambiguity (where multiple entities have the same name) is very pervasive in biomedicine.

In this paper we investigate the use of several string similarity methods to group together *string mentions* that might refer to the same *entity* or *concept*. Specifically, we consider the sub-problem of assigning an unseen mention to one of a set of existing unique entities or concepts, each with an associated set of known synonyms. As our aim here is focusing on improving string matching, we have purposely factored out the problem of ambiguity (to the extent possible) by using the UMLS MetaThesaurus as our data source, which is largly free of strings that refer to multiple entities. Thus, our work here can be viewed an important piece in a larger normalization or reference resolution system that resolves ambiguity (which includes filtering out mentions that don't refer to any entity of interest).

The experiments reported on in this paper evaluate a suite of robust string similarity techniques. Our results demonstrate considerable improvement to be gained by using *adaptive* string similarity metrics based on Conditional Random Fields customized to the domain at hand. The resulting best metric, we term SoftTFIDF-CRF, achieves 73.6 F-measure on the task of assigning a given string to the correct concept. Additionally, our experiments demonstrate a tradeoff between efficiency and recall based on *q*-gram indexing.

2 Background

2.1 Entity Extraction and Reference Resolution in the Biomedical Domain

Most of the work related to reference resolution in this domain has been done in the following areas: a) Intra-document Reference resolution, e.g (Castaño et al., 2002; Lin and Liang, 2004) b) Intra-document Named entity recognition (e.g Biocreative Task 1A (Blaschke et al., 2003), and others), also called classification of biological names (Torii et al., 2004) c) Intra-document alias extraction d) cross-document Acronym-expansion extraction, e.g., (Pustejovsky et al., 2001). e) Protein names resolution against database entries in SwissProt, *protein name grounding*, in the context of a relation extraction task (Kim and Park, 2004). One constraint in these approaches is that they use *several patterns* for the string matching problem. The results of the protein name grounding are 59% precision and 40% recall. The Biocreative Task 1B task challenged systems to *ground* entities found in article abstracts which contain mentions of genes in Fly, Mouse and Yeast databases. A central component in this task was resolving ambiguity as many gene names refer to multiple genes.

2.2 String Similarity and Ambiguity

In this subsection consider the string similarity issues that are present in the biology domain in particular. The task we consider is to associate a string with an existing *entity*, represented by a set of known strings. Although the issue of ambiguity is present in the examples we give, it cannot be resolved by using string similarity methods alone, but instead by methods that take into account the context in which those strings occur.

The protein name *p21* is ambiguous at least between two entities, mentioned as *p21-ras* and *p21/Waf* in the literature. A biologist can look at a set of descriptions and decide whether the strings are ambiguous or correspond to any of these two (or any other entity).

The following is an example of such a mapping, where *R* corresponds to *p21-ras*, *W* to *p21(Waf)* and *G* to another entity (the gene). Also it can be noticed that some of the mappings include subcases (e.g., R.1).[2]

String Form	Entity
ras-p21 protein	R
p21	R/W
p21(Waf1/Cip1)	W
cyclin-dependent kinase-I p21(Waf-1)	W
normal ras p21 protein	R
pure v-Kirsten (Ki)-ras p21	R.1
wild type p21	R/W
synthetic peptide P21	R/W.2
p21 promoter	G
transforming protein v-p21	R.3
v-p21	R.3
p21CIP1/WAF1	W
protein p21 WAF1/CIP1/Sd:1	W

Table 1: A possible mapping from strings to entities.

[2]All the examples were taken from the MEDLINE corpus.

If we want to use an external knowlege source to produce such a mapping, we can try to map it to concepts in the UMLS Methatesaurus and entries in the SwissProt database.

These two entities correspond to the concepts C0029007 (p21-Ras) and C0288472 (p21-Waf) in the UMLS Methathesaurus. There are 27 strings or *names* in the UMLS that map to C0288472 (Table 2):

oncoprotein p21	CAP20
CDK2-associated protein 20 kDa	MDA 6
Cdk2 inhibitor	WAF1 CIP1
Cdk-interacting protein	cdn1 protein
CDK-Interacting Protein 1	CDKN1A
CDKN1 protein	Cip1 protein
Cip-1 protein	mda-6 protein
Cyclin-Dependent Kinase Inhibitor 1A	p21
p21 cell cycle regulator	p21(cip1)
p21 cyclin kinase inhibitor	p21(waf1-cip1)
Pic-1 protein (cyclin)	p21-WAF1
senescent cell-derived inhibitor protein 1	protein p21
CDKN1A protein	WAF1 protein
WAF-1 Protein	

Table 2: UMLS strings corresponding to C0288472

There are 8 strings that map to concept C0029007 (Table 3).

Proto-Oncogene Protein p21(ras)	p21(c-ras)
p21 RAS Family Protein	p21 RAS Protein
Proto-Oncogene Protein ras	c-ras Protein
ras Proto-Oncogene Product p21	p21(ras)

Table 3: UMLS strings corresponding to C0029007

It can be observed that there is only one exact match: *p21* in C0288472 and Table 1. It should be noted that *p21*, is not present in the UMLS as a possible string for C0029007. There are other close matches like *p21(Waf1/Cip1)* (which seems very frequent) and *p21(waf1-cip1)*.

An expression like *The inhibitor of cyclin-dependent kinases WAF1 gene product p21* has a high similarity with *Cyclin-Dependent Kinase Inhibitor 1 A* and *The cyclin-dependent kinase-I p21(Waf-1)* partially matches *Cyclin-Dependent Kinase*

However there are other mappings which look quite difficult unless some context is given to provide additional clues (e.g., *v-p21*).

The SwissProt entries **CDN1A_FELCA**, **CDN1A_HUMAN** and **CDN1A_MOUSE** are related to *p21(Waf)*. They have the following set of common description names:

Cyclin-dependent kinase inhibitor 1, p21, CDK-interacting protein 1.[3]

There is only one entry in SwissProt related to *p21-ras*: Q9PSS8_PLAFE: with the description name *P21-ras protein* and a related gene name: *Ki-ras*.

It should be noted that SwissProt classifies, as different entities, the proteins that refer to different organisms. The UMLS MetaThesaurus, on the other hand, does not make this distinction. Neither is this distinction always present in the literature.

3 Methods for Computing String Similarity

A central component in the process of normalization or reference resolution is computing string similarity between two strings. Methods for measuring string similarity can generally be broken down into character-based and token-based approaches.

Character-based approaches typically consist of the edit-distance metric and variants thereof. Edit distance considers the number of edit operations (addition, substitution and deletion) required to transform a string s_1 into another string s_2. The Levenstein distance assigns unit cost to all edit operations. Other variations allow arbitrary costs or special costs for starting and continuing a “gap” (i.e., a long sequence of adds or deletes).

Token-based approaches include the Jaccard similarity metric and the TF/IDF metric. The methods consider the (possibly weighted) overlap between the tokens of two strings. Hybrid token and character-based are best represented by SoftTFIDF, which includes not only exact token matches but also close matches (using edit-distance, for example). Another approach is to perform the Jaccard similarity (or TF/IDF) between the *q*-grams of the two strings instead of the tokens. See Cohen et al. (2003) for a detailed overview and comparison of some of these methods on different data sets.

[3]There are two more description names for the human and mouse entries. The SwissProt database has also associated Gene names to those entries which are related to some of the possible names that we find in the literature. Those gene names are: *CDKN1A, CAP20, CDKN1, CIP1, MDA6, PIC1, SDI1, WAF1, Cdkn1a, Cip1, Waf1.* It can be seen that those names are incorporated in the UMLS as protein names.

Recent work has also focused on automatic methods for adapting these string similarity measures to specific data sets using machine learning. Such approaches include using classifiers to weight various fields for matching database records (Cohen and Richman, 2001). (Belenko and Mooney, 2003) presents a generative, Hidden Markov Model for string similarity.

4 An Adaptive String Similarity Model

Conditional Random Fields (CRF) are a recent, increasingly popular approach to sequence labeling problems. Informally, a CRF bears resemblance to a Hidden Markov Model (HMM) in which, for each input position in a sequence, there is an observed variable and a corresponding hidden variable. Like HMMs, CRFs are able to model (Markov) dependencies between the hidden (predicted) variables. However, because CRFs are conditional, discriminatively trained models, they can incorporate arbitrary overlapping (non-independent) features over the entire input space — just like a discriminative classifier.

CRFs are log-linear models that compute the probability of a state sequence, $\vec{s} = (s_1, s_2, ..., s_T)$, given an observed sequence, $\vec{o} = (o_1, o_2, ..., o_T)$ as:

$$P(\vec{s}|\vec{o}) = \frac{1}{Z_{\vec{o}}} \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(s_{t-1}, s_t, \vec{o}, t)\right)$$

where the f_k are arbitrary feature functions, the λ_k are the model parameters and $Z_{\vec{o}}$ is a normalization function.

Training a CRF amounts to finding the λ_k that maximize the conditional log-likelihood of the data.

Given a trained CRF, the inference problem involves finding the most likely state sequence given a sequence of observations. This is done using a slightly modified version of the Viterbi algorithm (See Lafferty et al. (2001) more for details on CRFs).

4.1 CRFs for String Similarity

CRFs can be used to measure string similarity by viewing the observed sequence, $\vec{o}$, and the state sequence, $\vec{s}$, as sequences of characters. In practice we are presented with two strings, q_1, and q_2 of possibly *differing* lengths. A necessary first step is to align the two strings by applying the Levenstein distance procedure as described earlier. This produces a series of edit operations where each operation has one of three possible forms: 1) $\epsilon \rightarrow o_j$ (addition), 2) $s_i \rightarrow o_j$ (substitution) and 3) $s_i \rightarrow \epsilon$ (deletion). The observed and hidden sequences are then derived by reading off the terms on the right and left-hand sides of the operations, respectively. Thus, the possible state values include all the characters in our domain plus the special null character, ϵ.

Feature Description	Variables
State uni-gram	(s_i)
State bi-gram	(s_{i-1}, s_i)
Obs. uni-gram; state uni-gram	(o_i, s_i)
Obs. bi-gram; state uni-gram	(o_{i-1}, o_i, s_i)
Obs. is *punctuation* and state uni-gram	(o_i, s_i)
Obs. is a *number* and state uni-gram	(o_i, s_i)

Table 4: Features used for string similarity

We employ a set of relatively simple features in our string similarity model described in Table 4. One motivation for keeping the set of features simple was to determine the utility of string similarity CRFs without spending effort designing domain-specific features; this is a primary motivation for taking a machine learning approach in the first place. Additionally, we have found that more specific, discriminating features (e.g., observation tri-grams with state bi-grams) tend to reduce the performance of the CRF on this domain - in some cases considerably.

4.2 Practical Considerations

We discuss a few practical concerns with using CRFs for string similarity.

The first issue is how to scale CRFs to this task. The inference complexity for CRFs is $O(s^2 t)$ where s is the size of the vocabulary of states and t is the number of input positions. In our setting, the number of state variable values is very large - one for each character in our alphabet (which is on the order of 40 or more including digits and punctuation). Moreover, we typically have very large training sets largely due to the fact that $\binom{n}{2}$ training pairs are derivable from an equivalence class of size n.

Given this situation, standard training for CRFs becomes unwieldy, since it involves performing inference over the entire data set repeatedly (typically a few hundred iterations are required to converge).

As such, we resort to an approximation: Voted Perceptron training (Collins, 2002). Voted Perceptron training does not involve maximizing log-likelihood, but instead updates parameters via stochastic gradient descent with a small number of passes over the data.

Another consideration that arises is given a pair of strings, which one should be considered the "observed" sequence and which one the "hidden" sequence.

Another consideration that arises is given a pair of strings, which string should be considered the "observed" sequence and which the "hidden" sequence?[4] We have taken to always selecting the longest string as the "observed" string, as it appears most natural, though that decision is somewhat arbitrary.

A last observation is that the probability assigned to a pair of strings by the model will be reduced geometrically for longer string pairs (since the probability is computed as a product of t terms, where t is the length of the sequence). We have taken to normalizing the probabilities by the length of the sequence roughly following the approach of (Belenko and Mooney, 2003).

A final point here is that it is possible to use Viterbi decoding to find the n-best hidden strings given *only* the observed string. This provides a mechanism to generate domain-specific string alterations for a given string ranked by their probability. The advantage of this approach is that such alterations can be used to expand a synonym list; exact matching can then be used greatly increasing efficiency. Work is ongoing in this area.

5 Matching Procedure

Our matching procedure in this paper is set in the context of finding the concept or entity (each with some existing set of known strings) that a given string, s, is referring to. In many settings, such as the BioCreative Task 1B task mentioned above, it is necessary to match large numbers of strings against the lexicon - potentially every possible phrase in a large number of documents. As such, very fast matching times (typically on the order of milliseconds) are required.

[4]Note that a standard use for models such as this is to find the most likely hidden sequence given *only* the observed sequence. In our setting here we are provided the hidden sequence and wish to compute it's (log-)probability given the observed sequence.

Our method can be broken down into two steps. We first select a reasonable candidate set of strings (associated with a concept or lexical entry), $S = s_1, s_2, ..., s_n$, reasonably similar to the given string s using an efficient method. We then use one of a number of string similarity metrics on all the pairs: $\langle s, s_1\rangle, \langle s, s_2\rangle, ...\langle s, s_n\rangle$

The set of candidate strings, $s_1, s_2, ..., s_n$ is determined by the *q-gram match ratio*, which we define as:

$$qRatio(s, s_i) = 1 - \frac{|ng(s) \cap ng(s_i)|}{|ng(s) \cup ng(s_i)|}$$

where $ng(x) = \{y|$ such that y is a q-gram of $x\}$. This set is retrieved very quickly by creating a q-gram index: a mapping between each q-gram and the strings (entries) in which it occurs. At query time, the given string is broken into q-grams and the sets corresonding to each q-gram are retrieved from the index. A straightforward computation finds those entries that have a certain number of q-grams in common with the query string s from which the ratio can be readily computed.

Depending on the setting, three options are possible given the returned set of candidates for a string s:

1. Consider s and s_i equivalent where s_i is the most similar string
2. Consider s and s_i equivalent where s_i is the most similar string and $sim(s, s_i) \geq T$, for some threshold T
3. Consider s and s_i equivalent for *all* s_i where $sim(s, s_i) \geq T$, for some threshold T

In the experiments in this paper, we use the first criterion since for a given string, we know that it should be assigned to exactly one concept (see below).

6 Experiments and Results

6.1 Data and Experimental Setup

We used the UMLS MetaThesaurus for all our experiments for three reasons: 1) the UMLS represents a wide-range of important biomedical concepts for many applications and 2) the size of the UMLS (compared with BioCreative Task 1B, for example) promotes statistically significant results as well as sufficient training data 3) the problem of ambiguity (multiple concepts with the same name) is largely absent in the UMLS.

The UMLS is a taxonomy of medical and clinical concepts consisting of 1,938,701 lexical entries (phrase strings) where each entry belongs to one (or, in very rarely, more than one) of 887,688 concepts. We prepared the data by first selecting only those lexical entries belonging to a concept containing 12 or more entries. This resulted in a total of 129,463 entries belonging to 7,993 concepts. We then divided this data into a training set of 95,167 entries and test set of 34,296 entries where roughly 70% of the entries for each concept were placed in the training set and 30% in the test set. Thus, the training set and test set both contained some string entries for each of the 7,993 concepts. While restricting the number of entries to 12 or more was somewhat arbitrary, this allowed for at least 7 (70% of 12) entries in the training data for each concept, providing sufficient training data.

The task was to assign the correct concept identifier to each of the lexical entries in the test set. This was carried out by finding the most similar string entry in the training data and returning the concept identifier associated with that entry. Since each test instance must be assigned to exactly one concept, our system simply ranked the candidate strings $s_1, s_2..., s_n$ based on the string similarity metric used. We compared the results for different maximum q-gram match ratios. Recall that the q-gram match mechanism is essentially a filter; higher values correspond to larger candidate pools of strings considered by the string similarity metrics.

We used six different string similarity metrics that were applied to the same set of candidate results returned by the q-gram matching procedure for each test string. These were **TFIDF**, **Levenstein**, **q-gram-Best**, **CRF**, **SoftTFIDF-Lev** and **SoftTFIDF-CRF**. **TFIDF** and **Levenstein** were described earlier. The **q-gram-Best** metric simply selects the match with the lowest q-gram match ratio returned by the q-gram match procedure described above[5]. The **SoftTFIDF-Lev** model is the SoftTFIDF metric described earlier where the secondary metric for similarity between pairs of tokens is the Levenstein distance.

	Precision	Recall	F-measure
SoftTFIDF-CRF(0.5)	0.761	0.714	0.736
SoftTFIDF-Lev(0.5)	0.742	0.697	0.718
CRF(0.6)	0.729	0.705	0.717
q-gram Best(0.475)	0.714	0.658	0.685
Levenstein(0.4)	0.710	0.622	0.663
TFIDF(0.325)	0.730	0.576	0.644

Table 5: Maximum F-measure attained for each string similarity metric, with corresponding precision and recall values. The numbers in parentheses indicate the q-gram match value for which the highest F-measure was attained.

The **CRF** metric is the CRF string similarity model applied to the entire strings. This model was trained on pairs of strings that belonged to the same concept in the training data, resulting in 130,504 string pair training instances. The **SoftTFIDF-CRF** metric is the SoftTFIDF method where the secondary metric is the CRF string similarity model. This CRF model was trained on pairs of tokens (not entire phrases). We derived pairs of tokens by finding the most similar pairs of tokens (similarity was determined here by Levenstein distance) between strings belonging to the same concept in the training data. This resulted in 336,930 string pairs as training instances.

6.2 Results

We computed the precision, recall and F-measure for each of the string similarity metrics across different q-gram match ratios shown in Fig. 1. Both a precision *and* recall error is introduced when the top-returned concept id is incorrect; just a recall error occurs when no concept id is returned at all - i.e. when the q-gram match procedure returns the empty set of candidate strings. This is more likely to occur when for lower q values and explains the poor recall in those cases. In addition, we computed the *mean reciprocal rank* of each of the methods. This is computed using the ranked, ordered list of the concepts returned by each method. This scoring method as-

[5]This is essentially the Jaccard similarity metric over q-grams instead of tokens

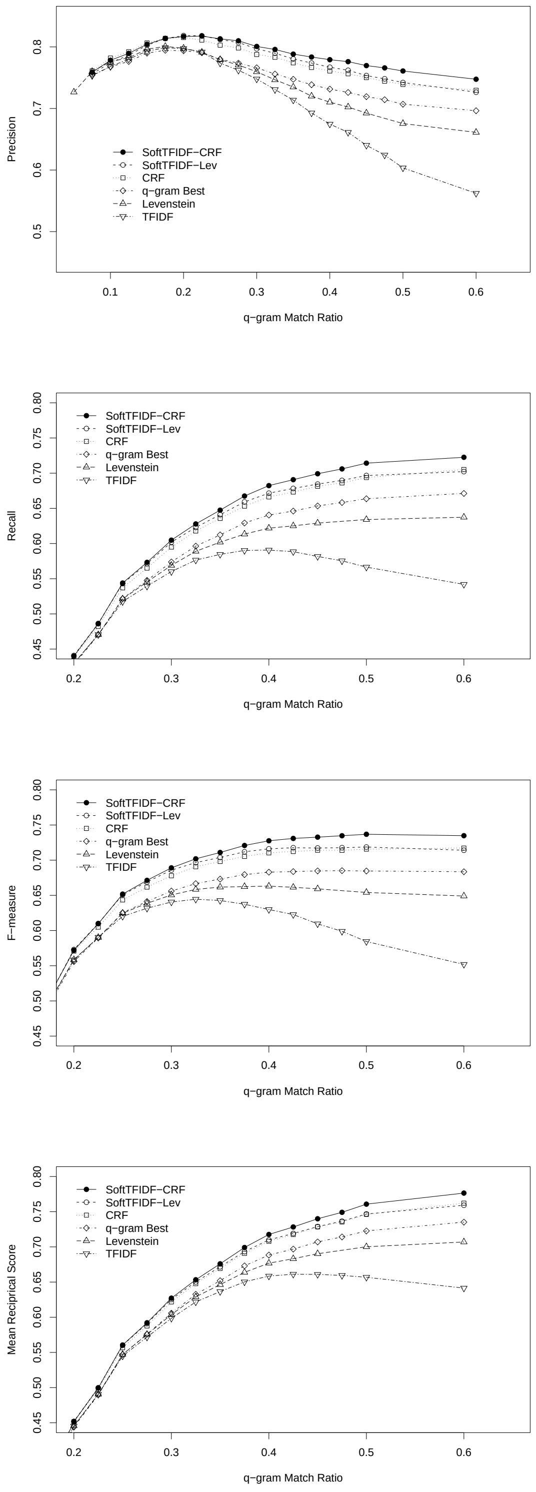

Figure 1: Precision, Recall, F-measure and Mean Reciprocal Rank comparisions for each string similarity metric across different q-gram match ratios.

signs a score of $1/R$ for each test instance where R is the position in the ranked list at which the correct concept is found. For example, by returning the correct concept as the 4th element in the ranked list, a method is awarded $1/4 = 0.25$. The *mean reciprocal rank* is just the average score over all the test elements.

As can be seen, the **SoftTFIDF-CRF** string-similarity metric out-performs all the other methods on this data set. This approach is robust to both word order variations and character-level differences, the latter with the benefit of being adapted to the domain. Word order is clearly a critical factor in this domain[6] though the **CRF** metric, entirely character-based, does surprisingly well - much better than the Levenstein distance. The **q-gram-Best** metric, being able to handle word order variations and character-level differences, performs fairly.

The graphs illustrate a tradeoff between efficiency and accuracy (recall). Lower q-gram match ratios return fewer candidates with correspondingly fewer pairwise string similarities to compute. Precision actually peaks with a q-gram match ratio of around 0.2. Recall tapers off even up to high q-gram levels for all metrics, indicating that nearly 30% of the test instances are probably too difficult for any string similarity metric. Error analysis indicates that these cases tend to be entries involving synonymous "nicknames". Acquiring such synonyms requires other machinery, e.g., (Yu and Agichtein, 2003).

7 Conclusions

We have explored a set of string similarity metrics in the biological domain in the service of reference resolution. String similarity is only one parameter to be considered in this task. We presented encouraging results for assigning strings to UMLS concepts based solely on string similarity metrics — demonstrating that adaptive string similarity metrics show significant promise for biomedical text processing. Further progress will require a system that 1) utilizes context of occurrence of respective strings for handling ambiguity and 2) further improves recall

[6]Inspection of the data indicates that the purely character-based methods are more robust than one might think. There are at least 8 strings to match against for a concept and it is likely that at least one of them will have similar word order to the test string.

through expanded synonyms.

Future work should also consider the *dependent* nature (via transitivity) of reference resolution. Comparing a test string against *all* (current) members of an equivalence class and considering multiple, similar test instances simultaneously (McCallum and Wellner, 2003) are two directions to pursue in this vein.

8 Acknowledgements

We thank Dave Harris, Alex Morgan, Lynette Hirschman and Marc Colosimo for useful discussions and comments. This work was supported in part by MITRE Sponsored Research 51MSR123-A5.

References

A. Bagga. 1998. *Coreference, cross-document coreference, and information extraction methodologies*. Ph.D. thesis, Duke University. Supervisor-Alan W. Biermann.

M. Belenko and R. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Datamining*, pages 39–48, Washington D.C.

C. Blaschke, L. Hirschman, A. Yeh, and A. Valencia. 2003. Critical assessment of information extraction systems in biology. *Comparative and Functional Genomics*, pages 674–677.

J. Castaño, J. Zhang, and J. Pustejovsky. 2002. Anaphora resolution in biomedical literature. In *International Symposium on Reference Resolution*, Alicante, Spain.

William Cohen and Jacob Richman. 2001. Learning to match and cluster entity names. In *ACM SIGIR-2001 Workshop on Mathematical/Formal Methods in Information Retrieval*, New Orleans, LA, September.

K. Bretonnel Cohen, Andrew Dolbey, George Acquaah-Mensah, and Lawrence Hunter. 2002. Contrast and variability in gene names. In *Proceedings of the Workshop on Natural Language Processing in the Biomedical Domain*, pages 14–20, Philadelphia, July. Association for Computational Linguistics.

W. Cohen, P. Ravikumar, and S. Fienburg. 2003. A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation*.

Michael Collins. 2002. Discriminative training methods for hidden markove models: Theory and experiments with perceptron algorithms. In *EMNLP 2002*.

Yu H, Hatzivassiloglou V, Friedman C, Rzhetsky A, and Wilbur W. 2002. Automatic extraction of gene and protein synonyms from medline and journal articles. In *Proc AMIA Symposium*, pages 919–23.

Jung-Jae Kim and Jong C. Park. 2004. Bioar: Anaphora resolution for relating protein names to proteome database entries. In Sanda Harabagiu and David Farwell, editors, *ACL 2004: Workshop on Reference Resolution and its Applications*, pages 79–86, Barcelona, Spain, July. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

X. Li, P. Morie, and D. Roth. 2005. Semantic integration in text: From ambiguous names to identifiable entities. *AI Magazine. Special Issue on Semantic Integration*.

Y. Lin and T. Liang. 2004. Pronominal and sortal anaphora resolution for biomedical literature. In *Proceedings of ROCLING XVI: Conference on Computational Linguistics and Speech Processing*, Taipei, Taiwan.

Andrew McCallum and Ben Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 79–86, Acapulco, Mexico, August.

A. Morgan, L. Hirschman, M. Colosimo, A. Yeh, and J. Colombe. 2004. Gene name identification and normalization using a model organism database. *Journal of Biomedical Informatics*, (6):396–410.

Christoph Müller, Stefan Rapp, and Michael Strube. 2002. Applying co-training to reference resolution. In *ACL*, pages 352–359.

J. Pustejovsky, J. Castaño, B. Cochran, M. Kotecki, and M. Morrell. 2001. Automatic extraction of acronym-meaning pairs from medline databases. In *Proceedings of Medinfo, London*.

J. Pustejovsky, J. Castaño, J. Zhang, R. Sauri, and W. Luo. 2002. Medstract: creating large-scale information servers from biomedical texts. In *Proceedings of the Workshop on Natural Language Processing in the Biomedical Domain*, pages 85–92, Philadelphia, July. Association for Computational Linguistics.

M. Torii, S. Kamboj, and K. Vijay-Shanker. 2004. Using name-internal and contextual features to classify biological terms. *Journal of Biomedical Informatics*, pages 498–511.

X. Yang, G. Zhou, J. Su, and C. L. Tan. 2004. Improving noun phrase coreference resolution by matching strings. In *Proceedings of 1st Internation Joint Conference of Natural Language Processing*, pages 326–333.

H. Yu and E. Agichtein. 2003. Extracting synonymous gene and protein terms from biological literature. *Bioinformatics*, pages 340–349.