The Role of String Similarity Metrics in Ontology Alignment

Michelle Cheatham and Pascal Hitzler

August 9, 2013

1 Introduction

Tim Berners-Lee originally envisioned a much different world wide web than the one we have today – one that computers as well as humans could search for the information they need [3]. There are currently a wide variety of research efforts towards achieving this goal, one of which is ontology alignment.

An ontology is a representation of the concepts in a domain and how they relate to one another. Engineering new ontologies is not a deterministic process – many design decisions must be made, and the designers' backgrounds and the application they are targeting will influence their decisions in different ways. The end result is that two ontologies that represent the same domain will not be the same. They may use synonyms for the same concept, they may be at different levels of abstraction, they may not include all of the same concepts, and they may not even be in the same language. The goal of ontology alignment is to determine when an entity in one ontology is semantically related to an entity in another ontology (for a comprehensive discussion of ontology alignment, including a formal definition, see [22]). This would allow software applications to ingest information across different websites.

There have been dozens of ontology alignment algorithms developed over the last decade. In researching past and present alignment systems, it became obvious that nearly all such systems make use of a string similarity metric. But despite the ubiquity of these metrics, there has been little systematic analysis on which string similarity metrics perform well when applied to ontology alignment. This paper seeks to fill in that gap by analyzing string similarity metrics in this domain, as well as the utility of string pre-processing approaches such as tokenization, translation, synonym lookup, and others.

This study leads naturally to a follow-up question: how much performance can we squeeze out of string-based techniques? We therefore consider how much an existing alignment algorithm can be improved by incorporating string similarity metrics that are optimized for the particular ontology matching problem at hand.

In particular, we seek to answer the following questions in this paper:

- What is the most effective string similarity metric for ontology alignment if the primary concern is precision? recall? f-measure?
- Does the best metric vary based on the nature of the ontologies being aligned?
- Does the performance of the metrics vary between classes and properties?
- Do string pre-processing strategies such as tokenization, synonym lookup, translations, normalization, etc improve ontology alignment results?
- What is the best we can do on the ontology alignment task using only string pre-processing and string similarity metrics?
- When faced with the task of aligning two ontologies, how can we automatically select which string similarity metrics and pre-processing strategies are best, without any training data available?
- How much does using optimized string similarity metrics improve an existing ontology alignment system?

2 Related Work

There has been some prior analysis of string similarity metrics in the context of ontology alignment as part of the development of a new string similarity metric designed specifically for this domain done by Stoilos and his colleagues [69]. They compared the performance of their own metric to that of Levenstein, Needleman-Wunsch (a weighted version of Levenstein), Smith-Waterman, Monge Elkan, Jaro Winkler, 3-gram, and substring on a subset of the OAEI benchmark test set. The benchmark test set is an older OAEI track that was phased out in 2010 in favor of a dynamically generated test set. Stoilos and his colleagues found that the Monge Elkan and Smith Waterman metrics performed very poorly on this task. The metric developed by the researchers performed the best. Another piece of work done in this area is a report produced by the Knowledge Web Consortium in 2004 that contained a description of a variety string (terminological) metrics applied to the problem of ontology alignment [21]. This document also discussed string pre-processing strategies such as normalization and stemming.

When the area of interest is expanded to include string similarity metric studies for other domains, we find some more interesting surveys. For instance, Branting looked at string similarity metrics as applied to the names of people, businesses, and organizations, particularly in legal cases [8]. Nine categories of name variations were identified: punctuation, capitalization, spacing, qualifiers, organizational terms, abbreviations, misspellings, word omissions, and word permutations. His work evaluated the performance of various combinations of normalization, indexing (determining which names would be compared to one another) and similarity metrics. He found that string normalization was useful for this application and that a string similarity metric that he called RWSA (described below) resulted in the best performance. In addition, Cohen, Ravikumar, and Fienberg did a very thorough analysis of string similarity metrics as applied to name-matching tasks [12]. They found that TF-IDF, Monge Elkan, and Soft TF-IDF performed well on the data sets they analyzed. In addition, they developed the SecondString Java library of string similarity metrics, which has become very widely used in the research community (including in our work here).

Some researchers have not set out to study string similarity metrics but have learned some interesting things about the topic while developing ontology alignment systems. For instance, the developers of Onto-Mapology tried Jaro, Jaro-Winkler, TF-IDF, and Monge Elkan in their alignment system and found Jaro-Winkler to have the highest performance [4], and the developers of SAMBO, which focuses on biomedical ontologies, found that a weighted sum of n-gram, edit distance, and an unnamed set metric performed better than any of those metrics alone [39]. In addition, the X-SOM developers note that the optimal combination of metrics does not vary based on the domain of the ontologies but rather based on their design characteristics [16].

While string similarity metrics are certainly not a new area of research, it remains unclear which string metric(s) are best for use in ontology alignment systems. In the OAEI competition algorithms surveyed for this work, 24 different string similarity metrics were used. In just the work cited above, Monge Elkan was found to be among the best performing metrics for name matching but among the worst performing for ontology alignment, yet several of the systems in the OAEI competition use Monge Elkan. Since nearly all alignment algorithms use a string similarity metric, more clarity in this area would be of benefit to many researchers. The work presented here expands on the previous efforts discussed above by considering a wider variety of string metrics, string pre-processing strategies, and ontology types. It also takes the work further by placing the string metrics into a complete ontology alignment system and comparing the results of that system to the current state of the art.

3 String Similarity Metrics

The Ontology Alignment Evaluation Initiative has become the primary venue for work in ontology alignment. Since 2006, participants in the OAEI competition have been required to submit a short paper describing their approach and results. All of these papers were surveyed to determine what lexical metrics were employed and what pre-processing steps were being used (or proposed). In cases where the paper was not explicit about the string similarity metric used, the code for the alignment algorithm was downloaded and examined when possible. The results of this survey are shown in appendix A.

We can group string metrics along three major axes: global versus local, set versus whole string, and perfect-sequence versus imperfect-sequence.

Global versus local refers to the amount of information the metric needs in order to classify a pair of strings as a match or a non-match. In some cases, the string metric needs to compute some information over all of the strings in one or both ontologies before it can match any strings. Such a metric is global. In other cases, the pair of strings currently being considered is all the input that is required. Such a metric is local. Global metrics can be more tuned to the particular ontology pair being matched, but that comes at the price of increased time complexity.

Perfect-sequence metrics require characters to occur in the same position in both strings in order to be considered a match. Imperfect-sequence metrics equate matching characters as long as their positions in the strings differ by less than some threshold. In some metrics, this threshold is the entire length of the string. Imperfect-sequence metrics are thought to perform better when the word ordering of labels might differ. This is common in biology-based ontologies. For instance, we would like to match leg bone with bone of the leg. Imperfect sequence metrics are more likely to identify such matches. The drawback is that they also frequently result in more false positives. For instance, the words stop and post would be a perfect match for an imperfect-sequence metric if the threshold were the entire length of the string.

Largely orthogonal to these axes lie set-based string similarity metrics. A set-based string metric works by finding the degree of overlap between the sets of tokens contained in two strings. Tokens are most commonly the words within the strings. The set-based metric must still use a basic string metric to establish if the individual tokens are equal (or close enough to be considered equal). This helper metric is often exact match, but it could be any non-set string metric. Word-based set metrics are generally thought to perform well on longer strings such as sentences or documents whereas they are assumed to give relatively high precision but low recall for shorter strings. Many ontologies have elements with short names that contain only a word or two, but ontologies in some domains may have longer labels. Also, the labels of individuals (versus classes or properties) in an ontology often have longer labels. Word-based set string similarity metrics may perform well in these situations.

The list below contains all string similarity metrics found in the review of OAEI participants and categorizes them based on the classifications described above. For set-based metrics, the underlying base metric used is given in parentheses. One combination does not contain any metrics: non-set/global/perfect-sequence. A subset of these metrics has been chosen for analysis related to various aspects of the ontology alignment problem. These metrics were chosen to reflect those most commonly used in existing alignment systems as well as to cover as fully as possible all combinations of the classification system provided. The chosen metrics are shown in bold.

- Set
    - Global
        - Perfect-sequence
            - Evidence Content (with exact)
            - **TF-IDF (with exact match)**
        - Imperfect-sequence
            - **Soft TF-IDF (with Jaro-Winkler)**
    - Local
        - Perfect-sequence
            - **Jaccard (with exact match)**
            - Overlap Coefficient (with exact)
        - Imperfect-sequence
            - RWSA
            - **Soft Jaccard (with Levenstein)**
- Non-set
    - Global
        - Perfect-sequence
            - None
        - Imperfect-sequence
            - COCLU
    - Local
        - Perfect-sequence
            - **Exact Match**
            - **Longest Common Substring**

    - Prefix
    - Substring Inclusion
    - Suffix
  - Imperfect-sequence
    - Jaro
    - **Jaro-Winkler**
    - **Levenstein**
    - Lin
    - **Monge Elkan**
    - **N-gram**
    - Normalized Hamming Distance
    - Smith Waterman
    - Smith Waterman Gotoh
    - **Stoilos**
    - String Matching (SM)

The basic idea behind each metric is explained below. The list is organized alphabetically.

3.1 COCLU

COCLU is short for Compression-based Clustering. The metric uses a Huffman tree to cluster the strings in one ontology and then matches each string in the second ontology to the appropriate cluster. Strings in the same cluster are considered equivalent. Whether to put a new string in a given cluster or create a new one is based on a distance metric called Cluster Code Difference (CCDiff), which is the difference between the summed length of the Huffman codes of all the strings in the cluster and the same with the new string added to the cluster. This has the effect of grouping together strings with the same frequent characters, regardless of the order of those characters. More information about COCLU can be found in [74].

3.2 Document Indexing

The idea behind this approach is to use existing document indexing and retrieval tools as a string similarity metric. Each entity in the second ontology to be matched is treated as a document. The content of the document varies in different approaches. Options include any combination of an entity's label, name, id, comment, neighbors, ancestors, descendants, and instances. The documents (e.g. entities) are first indexed by a standard search engine tool such as Lucene or Indri. Then entities in the first ontology to be matched are treated as search queries over the second ontology. Matches are made to the best search results, provided that the quality is above a threshold set by the user.

3.3 Exact Match

The most straightforward string similarity metric, exact match simply returns one if the two strings are identical and zero otherwise.

3.4 Evidence Content

Evidence content is a cousin of the Jaccard metric. Rather than weighting each word equally, however, words are weighted based on their evidence content, which is the negative logarithm of the frequency of the number of entities a word appears in, relative to the entire ontology. See [23] for a discussion of this metric with respect to ontology alignment.

3.5 Hamming Distance (normalized)

The Hamming distance is the number of substitutions required to transform one string into another. The normalized version divides this distance by the length of the string. This is similar to the Levenstein distance, but it only applies to strings of the same length.

3.6 Jaccard

This is a classic string similarity metric. The formula is:

$Jaccard(s1, s2) = \frac{|A \cap B|}{|A \cup B|}$

The Jaccard metric is most commonly used as a set metric, where the union of A and B refer to all of the unique words in the two strings being compared and the intersection refers to the words common to both strings (as determined by simple string equality). It is also possible to use this metric as a base rather than set metric by considering individual letters instead of words in the strings.

3.7 Jaro

This is another classic string similarity metric. The formula is:

$Jaro(s1, s2) = \frac{1}{3}(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m})$

where m is the number of matching characters and t is the number of transpositions. Two characters match if they are not further apart than $\lfloor(max(s1.length, s2.length)/2) - 1\rfloor$. Transpositions are cases where two characters match but appear in the reverse order.

3.8 Jaro-Winkler

This variation on the Jaro metric gives a preference to strings that share a common prefix. The thought is that many similar strings, particularly verbs and adjectives, have common roots but a variety of possible endings. The formula is:

$JaroWinkler(s1, s2) = Jaro(s1, s2) + (lp(1 - Jaro(s1, s2))$

where l is the length of the common prefix, up to four characters, and p is a weight for consideration of the common prefix (this must be less than 0.25 and is usually set to 0.1).

3.9 Longest Common Substring (LCS)

This metric simply normalizes the length of the largest substring that the two strings have in common. The formula is:

$LCSSim(s1, s2) = \frac{2 \cdot length(\max CommonSubstring(s1,s2))}{length(s1)+length(s2.length)}$

where $length$ returns the number of characters in a string.

3.10 Levenstein Edit Distance

This is by far the most commonly used string similarity metric in ontology alignment systems. The Levenstein edit distance is the number of insertions, deletions, and substitutions required to transform one string into another. It can be normalized by dividing the edit distance by the length of the string (either the first string, to create an asymmetric metric, or the average of the lengths of both strings). Variations on this metric weight different types of edits differently.

3.11 Lin

This metric is described in [42]. The idea behind this metric is that the similarity between two things can be assessed by taking a measure of what they have in common and dividing by a measure of the information it takes to describe them. This definition has its basis in information theory. They apply this intuition to determining string similarity using the following formula:

$$Sim(s1, s2) = \frac{2 \cdot \sum\limits_{t|tri(s1)\cap tri(s2)} \log P(t)}{\sum\limits_{t|tri(s1)} \log P(t) + \sum\limits_{t|tri(B)} \log P(t)}$$

where tri enumerates the trigrams in a string and $P(t)$ is the probability of a particular trigram occurring in a string, which is estimated by their frequencies in the words (i.e. over all of the words in the ontologies).

3.12 Monge Elkan

Monge and Elkan describe both a set-based similarity metric and a variant of the Smith-Waterman metric in their paper [47]. Different groups appear to refer to each of these as the Monge Elkan metric. The SecondString library, a Java-based implementation of many different string similarity metrics, implements the Smith-Waterman variant as Monge-Elkan, so that is what we will consider here.

This metric uses the Smith-Waterman approach with a match score of -3 for mismatched characters, +5 or the same characters (case insensitive), and +3 for approximately the same characters. This approximation is a variation on the original Smith-Waterman, along with the non-linear gap penalties used – 5 for a gap start and 1 for a gap continuation. The alphabet is upper and lower case letters, digits, period, comma, space – all other characters are ignored.

Two characters are approximately equal if they fall into the same set:

- {d t}
- {g j}
- {l r}

- {m n}
- {b p v}
- {a e i o u}
- {. ,}

3.13 N-gram

This metric converts each of the strings into a set of n-grams. For instance, if one of the words is hello and n is 3, the set of n-grams wold be {hel, ell, llo}. The resulting sets for both strings are then compared using any set similarity metric (cosine similarity and Dice's coefficient are common). A variation is to have special characters to indicate prior to the start of the string and after the end of the string. Using this approach, hello would result in the set {##h, #he, hel, ell, llo, lo%, o%%}.

3.14 Overlap Coefficient

This is very similar to the Jaccard metric. The formula is:

$Overlap(s1, s2) = \frac{|A \cap B|}{\min(|A|, |B|)}$

where A is the set of tokens (either words or characters) in the first string and B is the same for the second string.

3.15 Prefix

This metric returns one if the first string is a prefix of the second, zero otherwise.

3.16 RWSA

RWSA stands for Redundant, Word-by-word, Symmetrical, Approximate. This is based on the classification system for string similarity metrics presented in [8]. Each string is indexed by the Soundex representation of its first and last words. Soundex is a phonetic encoding consisting of the first letter of the string followed by three digits representing the phonetic categories of the next three consonants, if they exist. The phonetic categories are:

1. B, P, F, V

2. C, S, K, G, J, Q, X, Z
3. D, T
4. L
5. M, N
6. R

When comparing two strings, a list of possible matches is retrieved by hashing the shorter of the two strings, and the remainder of the algorithm is run on these potential matches to find the best one and determine if it is above a threshold. These potential matches are all considered with respect to the indexing string. Both strings are broken into their component words. Two strings are considered to be a match if each word in the smaller of the two strings approximately matches a unique work in the larger string. An approximate match is one in which the edit distance is within a mismatch threshold. When computing the edit distance, there is a penalty of 1.0 for insertions, deletions and substitutions and a penalty of 0.6 for transpositions. The indexing to retrieve candidate matches may enable this metric to be used on larger ontologies than others that require each string to be compared against every other.

3.17 String Matching (SM)

This metric was developed by Alexander Maedche and Steffen Staab and is described in [43]. It is essentially a normalized Levenstein edit distance in which the difference between the length of the shorter string and the edit distance is divided by the length of the shorter string.

3.18 Smith-Waterman

This is a variant of the Needleman-Wunch metric and like that metric, it uses a dynamic programming algorithm [20]. To compare two strings, a matrix is created with the number of columns equal to the length of the first string and the number of rows equal to the length of the second string. The first row and first column are all zeros. All other elements i, j are set to the maximum of the following:

- 0
- $H(i-1, j-1) + w(s1(i), s2(j))$, for a match/mismatch where w is the weight for a match/mismatch

- $H(i-1, j) + w(deletion)$, for a deletion where $w(deletion)$ is the starting or continuing gap penalty
- $H(i, j-1) + w(insertion)$, for an insertion where $w(insertion)$ is the starting or continuing gap penalty

Once this matrix has been created, the distance between the two strings is found by starting with the highest value in the matrix and moving either up, left, or diagonally up and left, towards whichever value is highest. This is repeated until either a zero or the upper left corner of the matrix is reached. The distance is the sum of all of the values that were traversed.

3.19 Smith Waterman Gotoh

This is a variation of the Smith-Waterman metric that has affine (non-linear) gap penalties. Because the length of the gaps doesn't matter in this version (a flat penalty is assessed for elongating an existing gap), a significantly faster implementation is possible [24].

3.20 Stoilos Metric (SMOA)

This string metric was specifically developed for use in ontology alignment systems. The main idea is to explicitly consider both the commonalities and differences of the two strings being compared. The formula is:

$Sim(s1, s2) = Comm(s1, s2) - Diff(s1, s2) + JaroWinkler(s1, s2)$

$Comm(s1, s2)$ finds the longest common substring, then removes it from both strings and finds the next longest substring, and so on until none remain. Then their lengths are summed and divided by the sum of the original string lengths. The formula for this is:

$Comm(s1, s2) = \frac{2 \cdot \sum length(maxCommonSubString(s1,s2))}{length(s1) + length(s2)}$

$Diff(s1, s2)$ is computed using the following formula:

$Diff(s1, s2) = \frac{uLen(s1) \cdot uLen(s2)}{p + (1-p) \cdot (uLen(s1) + uLen(s2) - uLen(s1) \cdot uLen(s2))}$

where $uLen$ is the length of the unmatched part of the string from the first step divided by the length of the corresponding original string and p is the importance of the difference factor. The authors experimentally found 0.6 to be a good choice.

This metric ranges from -1 for completely different strings to +1 for identical strings. More information about this metric can be found in [69].

3.21 Soft Jaccard

Unlike the Jaccard metric, soft Jaccard is a set metric *only.* It must be used in conjunction with a base similarity metric. First, the base similarity metric is run on all combinations of the words in both strings. The metric counts the number of these pairs in which the base metric result is greater than some threshold. This number is then divided by the number of words in the string with the higher word count. This is summarized in the formula below:

$SoftJaccard(s1, s2, t) = \frac{|sim(A_i,B_j)>=t|}{\max(|A|,|B|)}$

where A is the set of words in the first string, B is the set of words in the second string, t is the threshold for the base similarity metric, and *sim* is that base metric. The subscript i goes from 0 to the number of words in the first string and j does the same for the second string.

3.22 Soft TF-IDF

This version of the TF-IDF metric is identical except that rather than requiring exact matches when computing the cosine similarity, words are considered matching if their similarity according to some base similarity metric is above a threshold. In our work, we have followed the lead of Cohen and his colleagues in [12] and used Jaro-Winkler as the base metric.

3.23 Substring Inclusion

This metric returns one if the first string is contained within the second and zero otherwise.

3.24 Suffix

This metric returns one if the first string is a suffix of the second one and zero otherwise.

3.25 TF-IDF/cosine

TF-IDF stands for Term Frequency – Inverse Document Frequency. It is a technique used for document indexing in information retrieval systems. The term frequency is the number of times a word appears in a document, divided by the number of words in the document.

The inverse document frequency is the logarithm of the number of documents divided by the number of documents that contain the word in question. The idea behind using this approach for ontology alignment is that it is more indicative of similarity if two entities share a word that is rare in the ontologies than if they share a common word such as "the."

When computing the metric, the term frequency and inverse document frequency for each word in each document is computed (where document here means the same as described for the document indexing metric). This must be done for both ontologies before any entities can be compared. Then to compare two strings, each word's term frequency is multiplied by its inverse document frequency, creating a vector for each string. The string similarity is then the cosine similarity of the vectors.

4 String Pre-processing Strategies

This section describes all of the pre-processing approaches that were either tried or proposed by OAEI participants. The approaches mentioned by more than two participants are shown in bold italics – these will be examined in detail. Some operations are directly beneficial to the string similarity metric, while others primarily help with returning valid synonyms/translations (and so hopefully benefit the metric indirectly).

These approaches can be divided into two major categories: syntactic and semantic. Syntactic pre-processing methods are based on the characters in the strings or the rules of the language in which the strings are written. They can generally be applied quickly and without reference to an outside data store. Semantic methods relate to the meanings of the strings. These methods generally involve using a dictionary, thesaurus, or translation service to retrieve more information about a word or phrase.

- Syntactic
  - **tokenization**
  - split compound words
  - **normalization**
  - **stemming/lemmatization**
  - **stop word removal**
  - consider part-of-speech (i.e. weight functional words less)
- Semantic
  - **synonyms**

- antonyms
- categorization
- use language tag
- **translations**
- expand abbreviations and acronyms

4.1 Abbreviations and Acronyms

Abbreviations and acronyms are particularly challenging for string similarity metrics. There have been several attempts to expand such shortcuts into their original representation by either looking them up in external knowledge sources or using language production rules. Reliable expansion of abbreviations and acronyms would be useful not just for the string metric, but also in improving synonym lookup and translations.

4.2 Antonyms

Some similarity metrics consider differences as well as commonalities. A possible strategy for such metrics is to gather antonyms from a thesaurus in the same manner that synonyms are retrieved. These can then be used to determine that two strings are not equivalent.

4.3 Categorization

In this approach, an external source containing a category hierarchy is used. Strings falling into the same category are considered more similar.

4.4 Compound Words

It is possible that splitting compound words into their constituents can improve the performance of some set-based string similarity metrics.

4.5 Language Tag

Ontology files sometimes use a language tag to specify the language of a particular string in the ontology. This can be used to avoid potentially misleading comparisons of words in different languages. It can also be used in conjunction with translations, to determine which language to translate from and which to translate to.

4.6 Normalization

The idea behind normalization is to eliminate stylistic differences between strings as much as possible. This generally involves putting all characters into either upper or lower case, replacing punctuation characters with a space, and standardizing word order, often by alphabetizing the words within the string. Normalization might also involve transliterating characters not in the Latin alphabet to their closest equivalent.

4.7 Part-of-speech

Similar in concept to stop word removal, it is possible to remove functional words such as articles, conjunctions, and prepositions from strings prior to assessing their similarity. Another possibility is to keep these words in the strings but weight them less than other words when a set-based string similarity metric is used.

4.8 Stemming/Lemmatization

Stemming attempts to eliminate grammatical differences between words due to verb tense, plurals, and other word forms by finding the root of each word in the string. This topic has been studied in computer science since the sixties, and there are many existing algorithms. Stemming is both directly useful for string metrics and helpful in synonym lookup and translation.

4.9 Stop Word Removal

Stop words are the most commonly used words in a language. The idea behind removing stop words from strings prior to computing their similarity is that very common words add little useful information. There are many lists of stop words available for different languages.

4.10 Synonyms

In this pre-processing phase, strings are supplemented with their synonyms using either a general thesaurus such as WordNet or a domain-specific one such as UMLS for the biomedical domain. Biomedical ontologies also frequently have synonyms embedded in the ontology itself. Synonym lookup is by far the most often proposed pre-processing operation, but some who have actually implemented it report that it did not improve the performance of their system (e.g. SAMBO, GeRoMeSuite/SMB).

4.11 Tokenization

Tokenization involves splitting strings into their component words. Word boundaries vary based on implementation, but often some combination of whitespace, underscores, hyphens, slashes, and lower-to-uppercase changes (to detect camelCase) is used. Tokenization is useful when comparing ontologies with different naming conventions, such as underscores versus hyphens to delineate words. This is particularly important for set-based string similarity metrics. In addition, tokenization is also needed for some other pre-processing steps, such as synonym lookup, translations, and word stemming.

4.12 Translation

Translating strings when the ontologies to be matched are known to be in different languages has been suggested for a long time, but implementation has only become common in ontology alignment systems with the introduction on the multifarm test set in the OAEI. The language tag can be used to know which languages are involved, or a sample of the words in the ontologies can be analyzed to determine the languages.

5 Experimental Setup

In this section we describe the experimental framework and metric implementations in enough detail that others can reproduce our results. In addition, the source code for these experiments can be downloaded from http://www.pascal- hitzler.de/pub/StringMetricTester.jar.

The Ontology Alignment Evaluation Initiative (OAEI)[1] was started in 2004 with the goal of making it easier for researchers to compare the results of their ontology alignment algorithms. The organizers hold a contest each year in which participants run their algorithms on a large set of ontology matching problems and compare the results based on precision, recall, and f-measure. The OAEI features several tracks to test different types of ontology matching problems, three of which were used in this work.

The conference track consists of finding equivalence relations among 16 real-world ontologies describing the same domain – conference organization. The ontologies are based on

[1]http://oaei.ontologymatching.org/

conference websites, software tools designed to support conference organization, and input from experienced conference organizers. These ontologies are all fairly small, with each one containing less than 200 classes and properties. The multifarm track consists of the ontologies from the conference track translated by native speakers into eight different languages: Chinese, Czech, Dutch, French, German, Portuguese, Russian, and Spanish (along with the original English). The goal is to align all combinations of languages. Finally, the anatomy track consists of two ontologies from the biomedical domain: one describing the anatomy of a mouse and the other the anatomy of a human. As is common for biomedical ontologies, these are significantly larger than those found in the conference track, with each containing around 3000 classes.

In order to get a sense of whether the results on the OAEI test sets generalize to similar cases, we have also run our tests on other ontology pairs of the same type. As an analog to the conference test set, we have used two BizTalk files representing the domain of purchase orders: CIDX and Excel.[2] These were converted to an OWL format using a script that simply created a class for each entity (because we are only using the labels and not any relationship information, this is sufficient). The reference alignment for this dataset was created by domain experts. In addition, native speakers have assisted us in translating these schemas into German, Portuguese, Finish, and Norwegian so that we also have an analog for the OAEI multifarm track. Finally, we have attempted to match the Gene Ontology[3] to the multifun schema[4], both of which cover topics from biomedicine (the Gene Ontology covers the general domain of genetics, while the multifun schema is a description of cell function). The multifun schema was put into an OWL format using the same procedure as the CIDX and Excel data sets. The reference alignment for this test set was also generated by domain experts.[5] The GO ontology and associated schema mappings are made possible by the work of the Gene Ontology Consortium [2].

Our test framework takes the two ontologies to be aligned and compares the label of every entity in the first ontology to every entity in the second ontology. The label is first considered to be the URI of the ontology entity, with the namespace (everything before the # character) removed. In the case that this approach results in an empty or null string, the label annotation of the entity is used instead. For each pair of labels, the metric being tested is run in both directions: metric.compute(labelA, labelB) and metric.compute(labelB, labelA). These results are put into two separate two dimensional arrays. Then the stable marriage algorithm is run on these two arrays to determine the best matches between the two ontologies. This algorithm finds a mapping that ensures there are no As matched to a B where A is more similar to a different B and B is more similar to a different A. This approach is used because in the OAEI test set gold standards each entity is involved

[2]http://disi.unitn.it/~accord/Experimentaldesign.html
[3]http://www.geneontology.org/GO.database.shtml
[4]http://genprotec.mbl.edu/files/MultiFun.html
[5]http://www.geneontology.org/GO.indices.shtml

in at most one equality mapping. The version of the stable marriage algorithm used is deterministic. Finally, any mappings for which the minimum of metric.compute(labelA, labelB) and metric.compute(labelB, labelA) is less than one threshold or the maximum of those two values is less than a second threshold are thrown out. The resulting alignment is scored against the OAEI-provided gold standard in terms of precision, recall, and f-measure.

Due to the nature of the test framework, each metric requires at least two parameters: the thresholds for the similarities between the two strings (in both directions). For each test, each metric had both of the parameters initially set at 1.0. The parameters were then both decreased in steps of 0.1 until the f-measure ceased to improve. Then the first parameter was decreased in steps of 0.1 while the second was held constant. Then the second parameter was decreased in steps of 0.1 while the first was held constant. Then the first parameter was increased in steps of 0.1 and finally the second parameter was increased in steps of 0.1. This entire process was repeated as long as improvements in f-measure continued to be made. In addition, the soft set metrics (soft Jaccard and soft TF-IDF) require an additional parameter. This was initially set at 0.9 (setting it at 1.0 would have negated the soft aspect of the metrics), the test was run according to the previous description, then the third parameter was set to 0.8 and the process was repeated. This process was repetitive in some cases, but it was a reasonably thorough search of the parameter space for each metric.

5.1 Tokenization

Tokens are delimited by dash, underscore, space, camelCase, forward slash, and back slash. Each token is put into all lowercase. This is done to prevent camelCase labels from retaining a difference between tokens.

5.2 Stop Word Removal

The Glasgow IR group's stop words list is used.[6] It consists of 318 common English words. The Tokenization is done prior to stop word removal.

5.3 Stemming

Tokenization is done prior to stemming to handle cases like runningTotal. The Porter stemming algorithm is used [57].

[6]http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words

5.4 Normalization

Any whitespace is replaced by a single space. Any letters with diacritic marks, umlauts, etc are replaced with the closest corresponding letter from the English alphabet. Russian characters are transliterated using the approach specified here: http://www.translit.cc/. We were unable to transliterate the Chinese symbols. As a final step, the label is tokenized and the tokens are ordered alphabetically.

5.5 Synonyms

This test was only run on the conference and anatomy test sets. The language test set could be included in the future if appropriate electronic thesauri could be found for each language. The labels are first tokenized and then synonyms are looked up either in Wordnet, for the conference set, or in the synonyms attribute of the entity itself, for the anatomy set.

There were some differences between using Wordnet and the internal synonyms. The JWNL Java Wordnet API was used to query Wordnet. Tokenization must be done to get reasonable hit rates for synonym lookup from Wordnet. In addition, the Wordnet API does its own form of word stemming internally, and that was left in place because it is the most common way to use Wordnet in applications. A question arose on how to handle labels that are phrases when querying Wordnet. Looking up "masters thesis" returned only the synonyms for "masters" and nothing for thesis or for the phrase as a whole. The same was true when masters_thesis was used as the query (even though Wordnet will return items in a similar form, such as baseball_bat). Therefore the synonym set is generated by querying Wordnet for each token in the label and aggregating the results. So for this example, the synonym set contains all synonyms for masters plus all synonyms for thesis.

A second question concerned how to compute the overall similarity value based on the similarity values of the synonyms. After some preliminary experimentation, it was determined that the synonyms provided within the anatomy ontologies are far more specific and relevant than those arrived at by querying Wordnet for the conference ontologies. As a result, the best strategy for computing overall similarity in the anatomy case was to take the maximum value of the similarity of the label of the first entity compared to that of the label and each of the synonyms of the second entity (and the same for the other direction). For the Wordnet case the best result was obtained by employing a set similarity metric: the similarity values for the first entity's label and all of its synonyms where computed with respect to the second entity's label and all of its synonyms. All of these values were summed and divided by the size of the synonym set of the first entity, plus one for the label itself. We also tried using the same approach on the conference test set that was used on the anatomy test set, but the results were much worse than those resulting from the approach just outlined.

5.6 Translation

Only the language test set was used for this experiment. Google Translate (via the Google Translate API) is used to translate from one language to another. Google Translate can handle translations between all of the languages in the OAEI test set. Unfortunately, it is not free. The cost is \$20 per 2 million characters. It cost \$12.08 to align all of the ontology pairs in the test set once. To avoid paying that for each metric multiple times (to optimize the parameter values), the results were cached and the cache was used after the first run. This does not affect the accuracy of the metrics. The service can also detect the language of the input it is provided with – the labels of ten randomly chosen entities were submitted to the translation service to detect the language. Google Translate does some internal pre-processing involving stemming, but Google does not provide details on this.

5.7 Exact

The Java String class's startsWith method is used for this metric. The reason for using startsWith rather than the equals method is that this makes the metric asymmetric. For instance, exact.compute("leg bone", "leg") returns 1 because "leg bone" starts with "leg" while exact.compute("leg", "leg bone") returns 0 since "leg" does not start with "leg bone".

5.8 Jaccard

The SecondString library implementation of this metric is used.

5.9 Jaro-Winkler

The SecondString library implementation of this metric is used.

5.10 Longest Common Substring

This metric was coded based on the following logic: find the shorter of the two strings, for each character in that string, check to see if the longer string contains that character. If it does, find the length of the longest common substring starting with that character. The maximum of all of these lengths is then divided by the length of the first string and returned. This is an asymmetric metric.

5.11 Levenstein

The distance between the two input strings is computed using the SecondString implementation of the Levenstein metric. The distance is then normalized by dividing it by the length of the first input string (creating an asymmetric metric). In order to have 1 rather than 0 represent a perfect match, the normalized distance is then subtracted from 1.

5.12 Monge Elkan

The SecondString implementation of this metric is used.

5.13 N-gram

After some initial experimentation, n was set equal to 3 for all of these tests. This metric was coded based on the following logic: construct all trigrams for each input string, representing characters prior to the beginning of the string with '#' and those after the end of the string with '%'. Return the number of trigrams common to the two strings, divided by the number within the first string (asymmetric metric), being sure to handle duplicate trigrams correctly.

5.14 Soft Jaccard

A set of the unique words in the first string (where uniqueness is determined by a Levenstein distance less than a threshold from any word already in the set) and another set of the unique words in the second string are created. The intersection and union of these two sets are computed, and the metric returns the size of the intersection divided by the size of the union. The Levenstein distance is computed using the Second String library implementation.

5.15 Soft TF-IDF

The Second String library SoftTFIDF class was used as the basis for this implementation. The internal metric used was the Second String implementation of JaroWinkler. A SimpleTokenizer was created to split the strings into words with whitespace as the delimiter. The SoftTFIDF dictionary was created using the words from all of the labels in both ontologies. If the test considered synonyms, sets of synonymous words were maintained and only one representative word from each set was added to the dictionary. If the test considered translation, the word was first translated to the appropriate language before being added to the dictionary. A BasicStringWrapperIterator was used to train the metric.

5.16 Stoilos

This is coded based on the definition provided in the paper in which the metric was originally proposed. There was a point of confusion related to the Jaro-Winkler-based improvement factor mentioned in that paper, however. The paper states that the metric should range from -1 to +1, but with that factor in it ranges from -1 to +2. We tried both approaches when attempting to reproduce the results mentioned in the paper but achieved at best an f-measure that was around 0.1 lower than what was reported. For instance, on the 301 benchmark test with a 0.6 threshold, our tests resulted in precision = .83 and recall = .68 for an f-measure of .75 while the authors report a precision of .98, recall of .79, and f-measure of .87. The results using the Jaro-Winkler improvement factor were significantly worse. The experiments here were conducted without that factor included.

5.17 TF-IDF

The Second String library TFIDF class was used as the basis for this implementation. A SimpleTokenizer was created to split the strings into words with whitespace as the delimiter. The TFIDF dictionary was created using the words from all of the labels in both ontologies. If the test considered synonyms, sets of synonymous words were maintained and only one representative word from each set was added to the dictionary. If the test considered translation, the word was first translated to the appropriate language before being added to the dictionary. A BasicStringWrapperIterator was used to train the metric.

6 Results

In this section we review the results of the experiments described above.

6.1 OAEI Results

First, we look at the effect of the different string pre-processing strategies on precision, recall, and f-measure for the OAEI test sets. Figures 1 through 7 show the results for one string pre-processing strategy on all three OAEI datasets (conference, multifarm, and anatomy. F-measure, precision, and recall are all shown on the graphs.

Figure 1 shows the results of when no string pre-processing is employed.

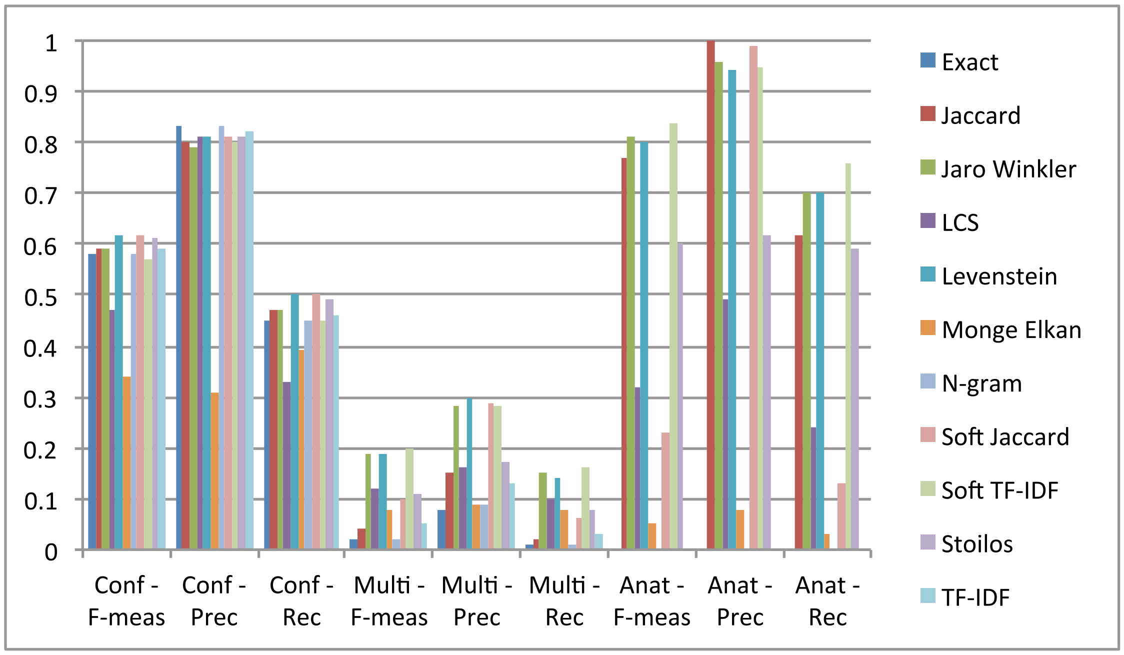

Figure 1: Results of all metrics on the OAEI test sets without any string pre-processing.

The conference dataset does not reveal much disparity among the string similarity metrics. If we leave out the Monge Elkan and Longest Common Substring metrics, which perform very poorly, we are left with very little standard deviation for either precision or recall in this test set. Also, the optimal thresholds indicate that the best approach is to look for matches that are as exact as possible.

The multifarm test set is much more challenging than the conference domain – both precision and recall are less than one fourth what they were in that case. This test set also reveals a much wider disparity among string similarity metrics. There is a large standard deviation for both precision and recall. This is likely because most ontologies, including these, contain the majority of the semantic information in labels. This intrinsic information is hidden from string similarity metrics by translation.

The exact, n-gram, and TF-IDF metrics cannot generate any matches for the anatomy test set. Furthermore, the longest common substring, Stoilos, and Monge Elkan metrics perform very poorly. For the remaining metrics, we find that this test is easier for string similarity metrics than the conference dataset. This is expected because biomed datasets usually deal with a smaller, more regular vocabulary. There is often a small set of nouns with associated modifiers. The precision for this dataset is extremely high and the standard deviation of the precision is relatively low. There is a much higher standard deviation of

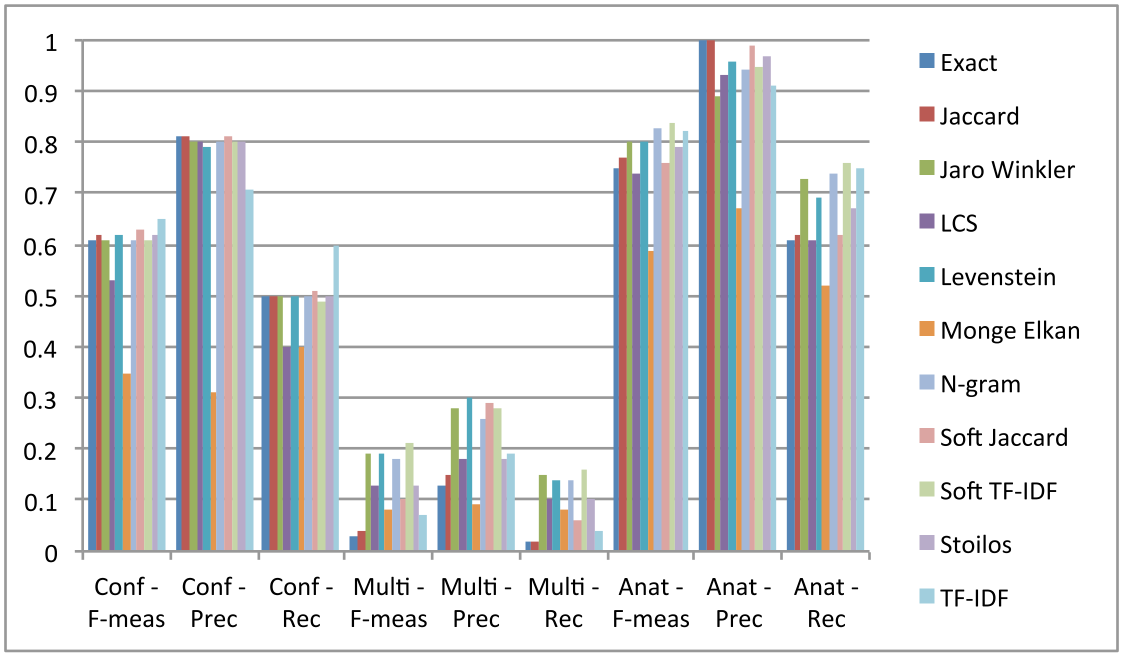

Figure 2: Results of all metrics on the OAEI test sets using tokenization.

recall. In particular, the metric with the next-to-highest precision (Soft Jaccard) has by far the lowest recall. This is a dataset where set metrics do particularly well, again because the biomed domain frequently involves phrases that can be presented in different orders. The anatomy test set is also interesting in that there is a more clear choice to be made between metrics that have good precision and those that have good recall there is not a lot of overlap between these two sets.

Figure 2 shows the same results using tokenization.

On the conference dataset, there were improvements in the recall of most metrics (a large one for LCS and TF-IDF) with little to no decrease in precision, except for TF-IDF. The recall of TF-IDF went from average to much better than any of the other metrics on this dataset with tokenization. This improvement in recall seems primarily due to the use of underscores as word separators in some ontologies and camelCase in others.

For the multifarm test set there were no improvements in the best-performing metrics due to tokenization.

After tokenization, the exact, n-gram, and TF-IDF metrics are capable of producing results on the anatomy dataset whereas without it they could not. In fact, exact now has perfect precision. The recall of most metrics was improved slightly or left unchanged. In the case

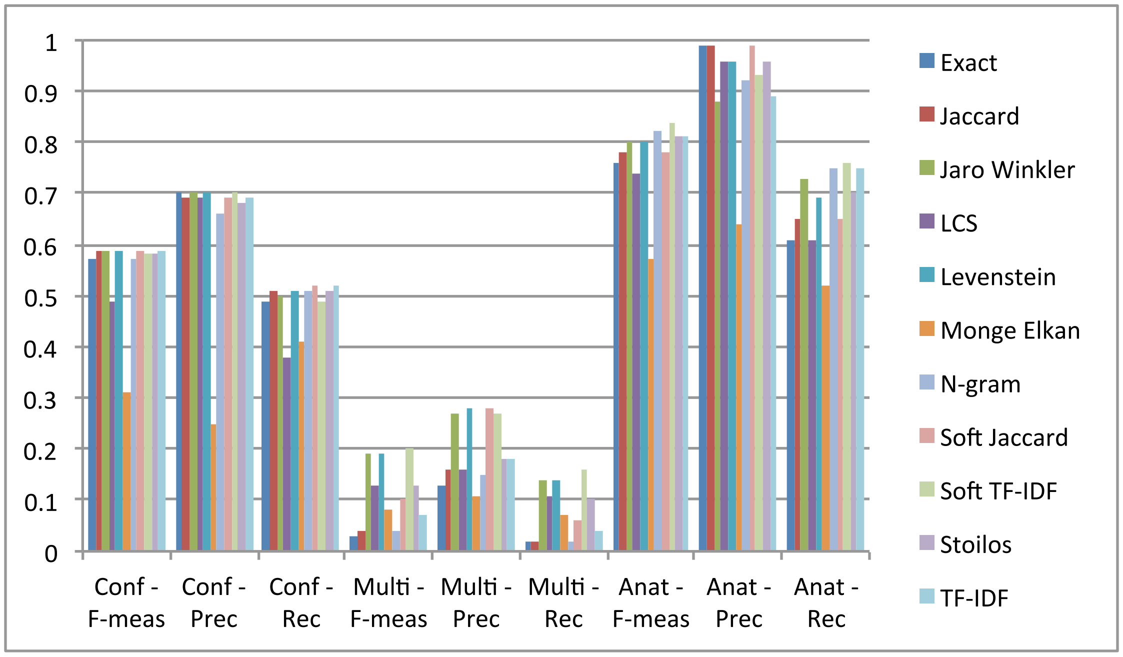

Figure 3: Results of all metrics on the OAEI test sets using stemming.

of the n-gram metric this was significant enough to make it one of the best-performing metrics in terms of recall. These changes were enough to change the set of top performing metrics in terms of f-measure as well.

In general, it makes sense to perform tokenization as a pre-processing step – it improves overall performance (especially recall) slightly, particularly for metrics that involve exact-match.

Figure 3 shows that when compared with tokenization as a baseline, stemming does not improve performance on any of the test sets (it actually slightly hurts precision on the conference set). The only exception is that recall of the Soft TF-IDF metric is improved by 27% on the conference test set, and that metric becomes the best choice for that test set in terms of recall. Recall of the n-gram metric on the anatomy test set is hurt badly enough to move it out of the best-performing metrics for that category.

Removing stop words lowers precision on the conference test set and has essentially no effect on the other two test sets when compared to tokenization (figure 4).

Figure 5 shows that normalization had little effect on the conference test set. Performance in terms of both precision and recall on the languages test set greatly improved with normalization (mostly due to the transliteration). Normalization had little effect on the

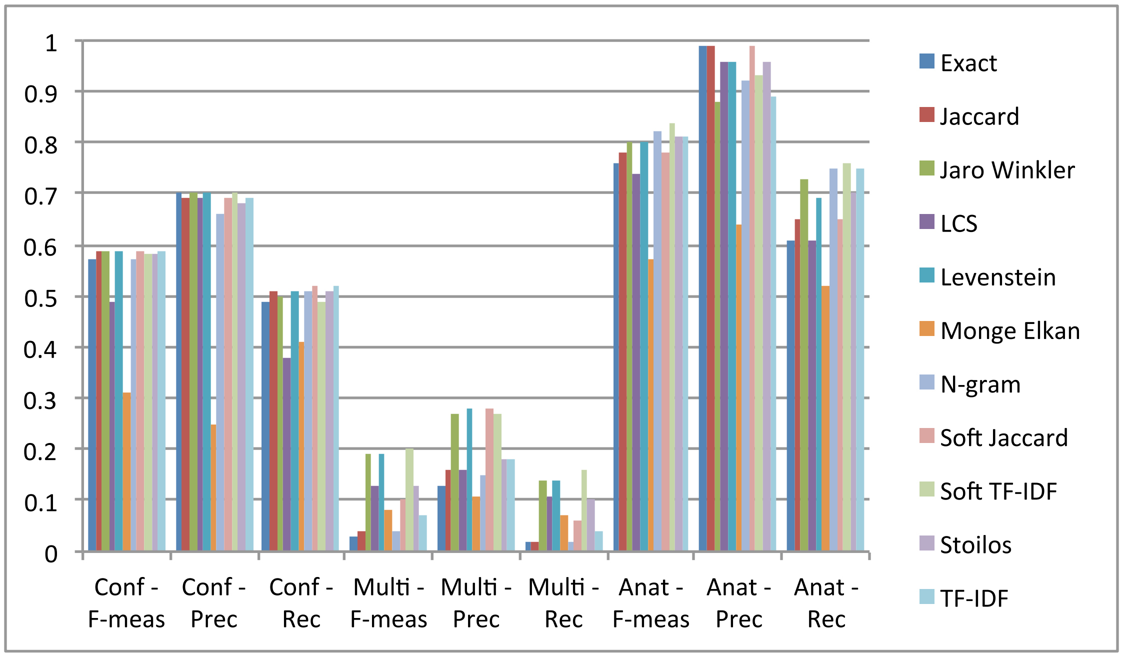

Figure 4: Results of all metrics on the OAEI test sets using stop word removal.

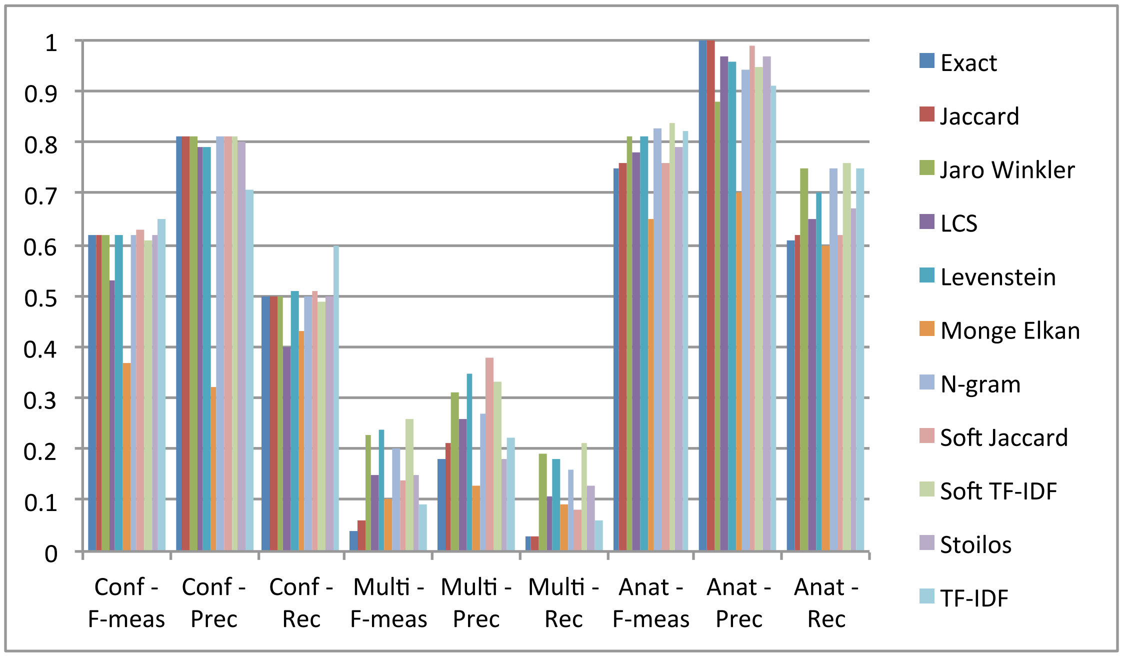

Figure 5: Results of all metrics on the OAEI test sets using normalization.

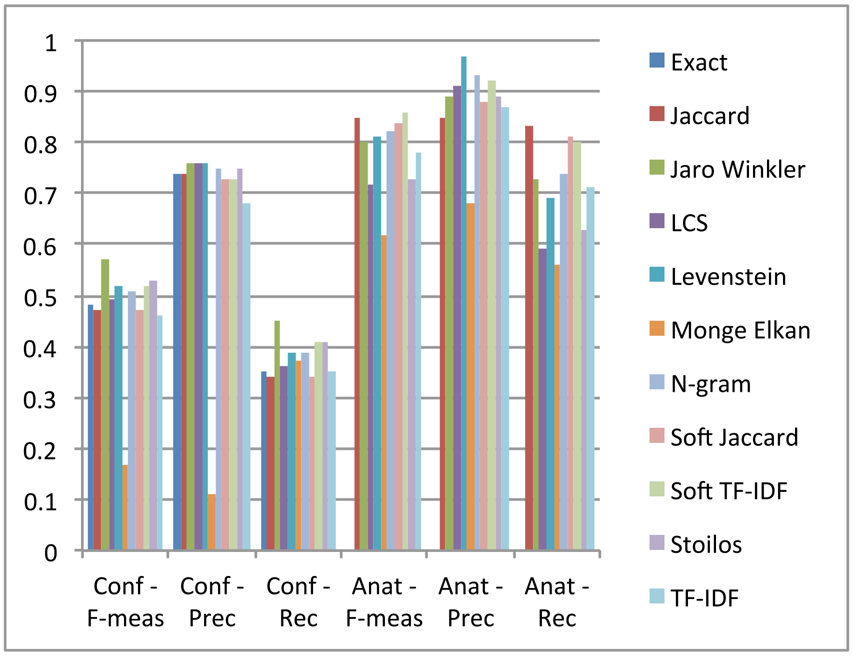

Figure 6: Results of all metrics on the OAEI test sets using synonyms.

precision or recall of any metric on the anatomy test set.

Figure 6 shows the results when synonyms are considered. This hurts both precision and recall for all metrics on the conference test set. On the anatomy test set, the precision of all metrics was either flat or worse than for tokenization alone, but the recall of several metrics improved enough to raise the f-measure when synonyms were considered.

Translations result in huge improvements in both precision and recall for all metrics on the multifarm test set, both over tokenization alone and over normalization (see figure 7). There is a wide variation in the performance of metrics in this configuration on this test set. Also, the metrics with good precision have mediocre recall and vice versa.

6.2 Comparative Results

The next set of graphs contains the results of the same tests on the non-OAEI data sets. These were conducted in order to determine whether the results presented above are specific to the OAEI data sets or if they carry over to other ontologies of the same general type. Figure 8 shows the f-measure of the best-performing metric using each pre-processing strategy while figure 9 shows the f-measure of each metric using the best-performing pre-processing strategy. The same information is shown for the analogous OAEI data set for

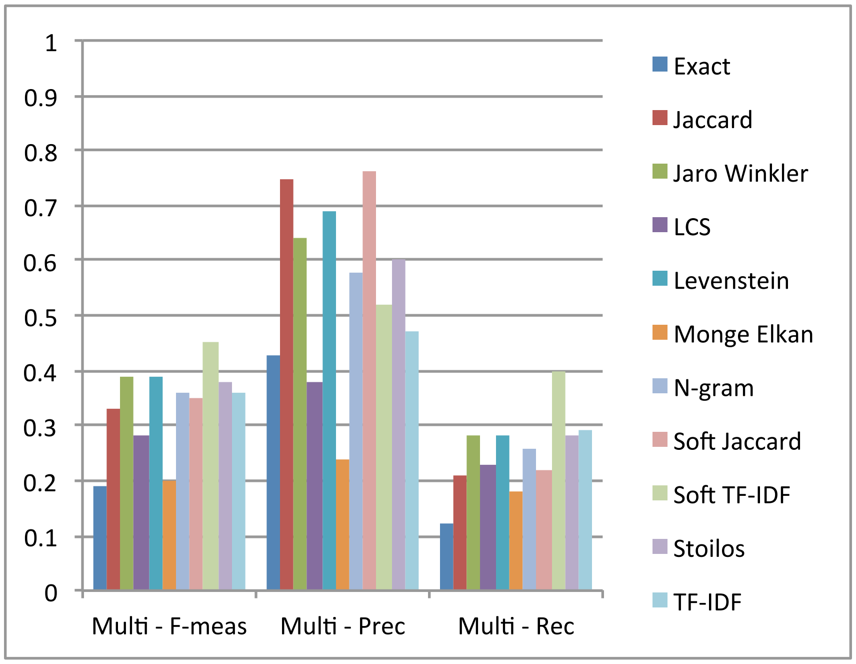

Figure 7: Results of all metrics on the OAEI test sets using translations.

comparative purposes. Variations in the absolute heights of the bars between analogous data sets are to be expected because the overall difficulty of matching a particular ontology pair may vary considerably – what we are looking for is the same general shape of the bars for the adjacent sets (or a clear understanding of any differences).

For the most part the pre-processing strategies exhibit similar behavior on the analogous data sets, as shown on figure 8. The only exception is that stemming improves performance on the Genes test set but not on Anatomy. Further analysis shows that this difference turns out to not be significant, however. The TF-IDF metric is the top-performing metric on the Genes test set. It turns out that many of the entities in the Genes dataset that are successfully matched using stemming contain the word transport or transporter. When stemming is used, these two words are considered the same and the TF-IDF metric weights them less due to a more frequent occurrence in the ontologies, thereby allowing more correct results. In short, a single lucky break has resulted in a rather noticeable variation in performance.

The significantly smaller sizes of the non-OAEI test sets cause more variability in metric performance (i.e. because there is a small number of matches, a metric is more heavily rewarded if it gets lucky on a particular match and more heavily penalized if it does not). However, we see from figure 9 that choosing a string similarity metric is less important for

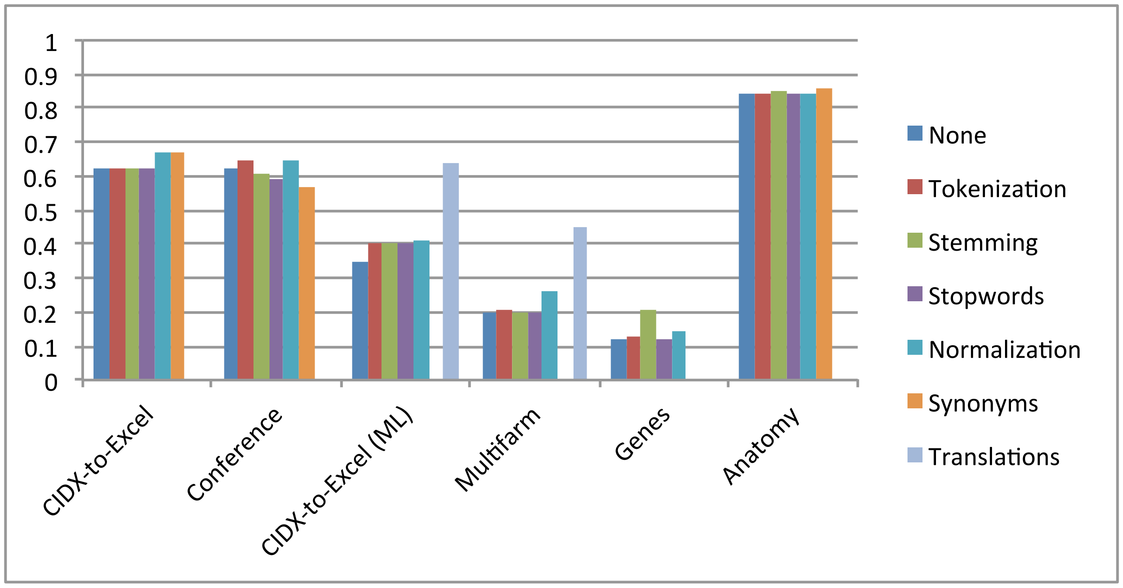

Figure 8: F-measures of the best-performing metric on all test sets for all string pre-processing strategies.

standard ontologies because performance varies little among metrics. This is not the case for the multilingual and biomedical ontologies. In addition, we see that choosing a string similarity metric based on its performance on the OAEI test sets leads to good relative performance on analogous ontology matching problems.

6.3 Classes vs Properties

Others have found that human experts have a more difficult time agreeing on when properties match than on classes [43]. We seek here to determine if string similarity metrics also have particular difficulty with properties. In this section we look at the performance of the metrics on classes versus properties for the Conference and Multifarm data sets. There are no matching properties in the Anatomy test set.

From figures 10 and 11 it is evident that string similarity metrics perform much worse on properties than on classes. This suggests that more work should be done in this area in the future. It appears from an empirical analysis of the results here that properties are particularly challenging for ontology alignment systems for several reasons. Properties frequently involve verbs, which can appear in a wider variety of forms than nouns (in addition to plurality/conjugation, verbs vary by tense). There are also often more functional words, such as articles and prepositions, in property names. Also, there are generally more

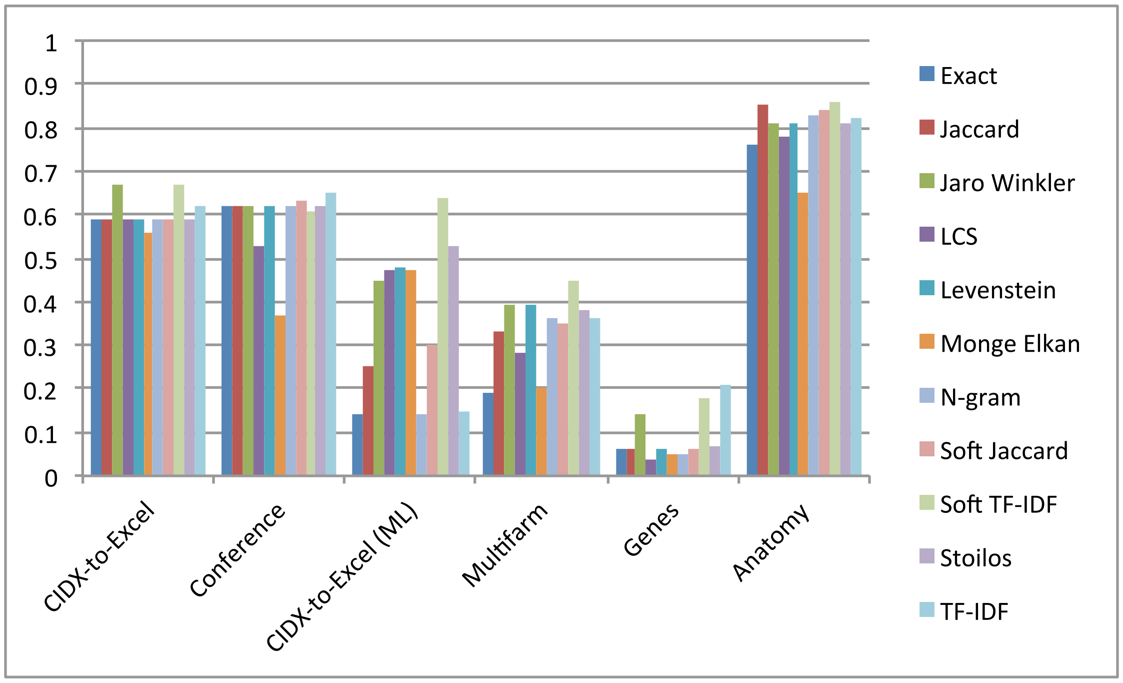

Figure 9: F-measures of all metrics on all test sets using the best-performing string pre-processing strategy.

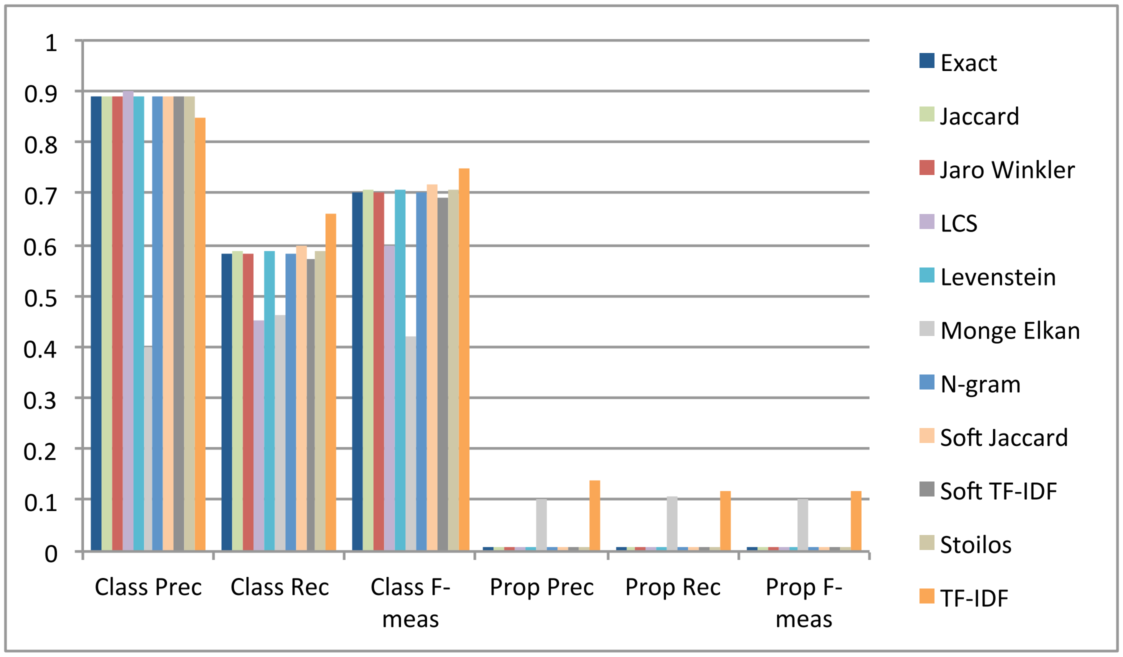

Figure 10: F-measures of all metrics on the classes and properties in the conference dataset using string tokenization.

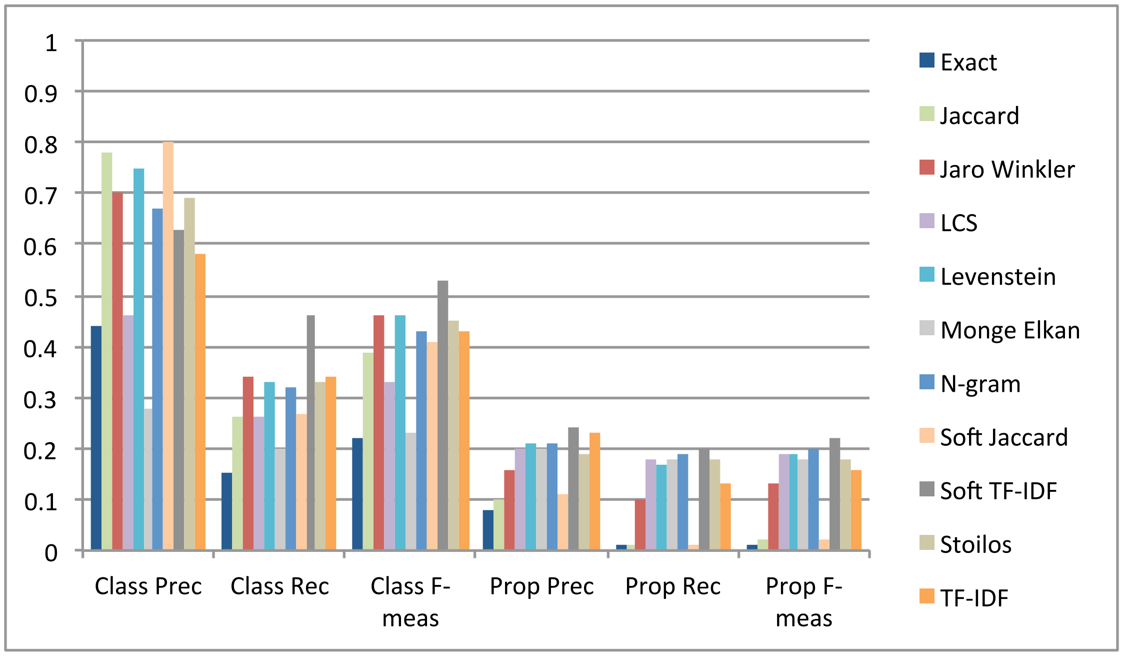

Figure 11: F-measures of all metrics on the classes and properties in the multifarm dataset using string tokenization.

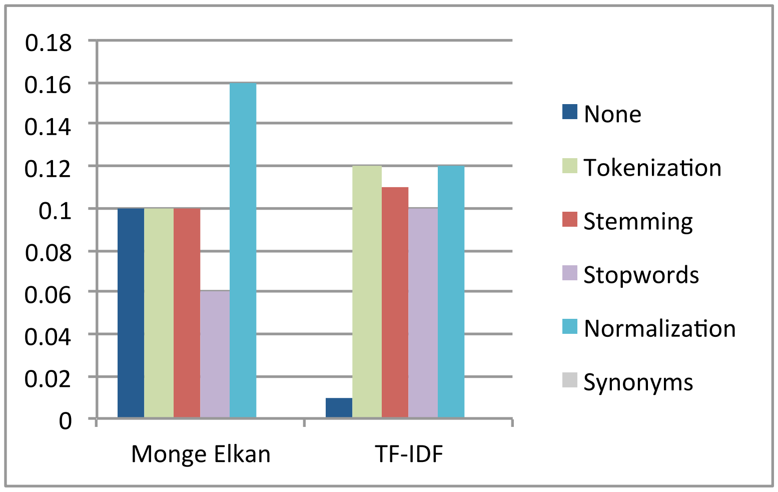

Figure 12: F-measures of Monge Elkan and TF-IDF on properties in the conference dataset for all of the string pre-processing strategies.

common synonyms available for the (often very generic) verbs in property names than the (often more specific) nouns in class names. We therefore thought that stemming, stop word removal, or synonym lookup might be effective when matching properties. However, that turned out not to be the case. Figure 12 shows the effect of various pre-processing strategies in combination with the two metrics that performed the best on properties for the conference test set: Monge Elkan and TF-IDF. Tokenization is required for the TF-IDF metric to work because it is a global set metric. Normalization improved the performance of Monge Elkan but not TF-IDF. Analysis of the results seems to indicate this is because putting the words into alphabetical order reduced the number of gap penalties for matching properties in Monge Elkan. This had no effect for TF-IDF because set metrics are not sensitive to word order.

It is curious that properties are more easily matched on the mulitfarm data set. This data set consists of exactly the same ontologies as the conference set, just translated into a variety of languages. It will be interesting to explore what is going on there, but the assistance of native speakers of some of the other languages will likely be required.

The above results were collected using the best thresholds found by optimizing the f-measure on the overall alignment problem (both classes and properties). In addition, we wanted to determine whether it was helpful to choose different thresholds for classes and properties. Figures 13 and 14 show the best results achieved for property matching on both the conference and multifarm data sets when the thresholds were optimized based solely on the f-measure for properties. The precision, recall, and f-measure when the thresholds were optimized for overall f-measure are reproduced here for ease of comparison. The results

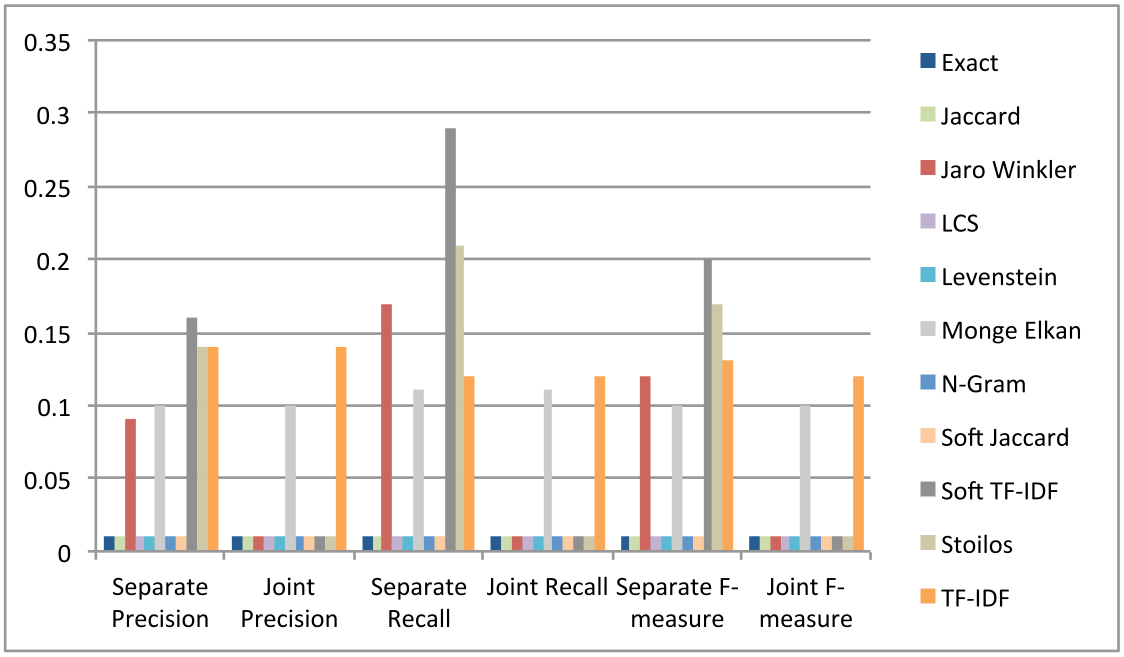

Figure 13: F-measures of all metrics using tokenization on the conference dataset when the thresholds were optimized once for classes and properties together versus separately for properties.

are better than in the previous case, indicating that for these data sets there is value in selecting different similarity metric thresholds for class and property comparisons.

7 Analysis

The results of the different metrics on the test sets reveal a potential trap for developers of ontology alignment systems. Results on the conference test set, which is representative of many real-world ontologies, do not reveal much difference in the performance of the metrics in terms of f-measure. Basically, if you pick any string similarity metric and set the threshold high (i.e. between .9 and 1.0) then the results will be near optimal. However, the other test sets reveal that all string metrics are not created equal – performance of different metrics on the multi-lingual and biomedical test sets varied considerably. Choosing a string metric for use on these alignment tasks involves a significant impact on precision and recall. The moral of the story is that when choosing a string metric for use in an ontology alignment algorithm, one should consider the characteristics of the ontologies being aligned and whether precision or recall is more important for the algorithm. Below

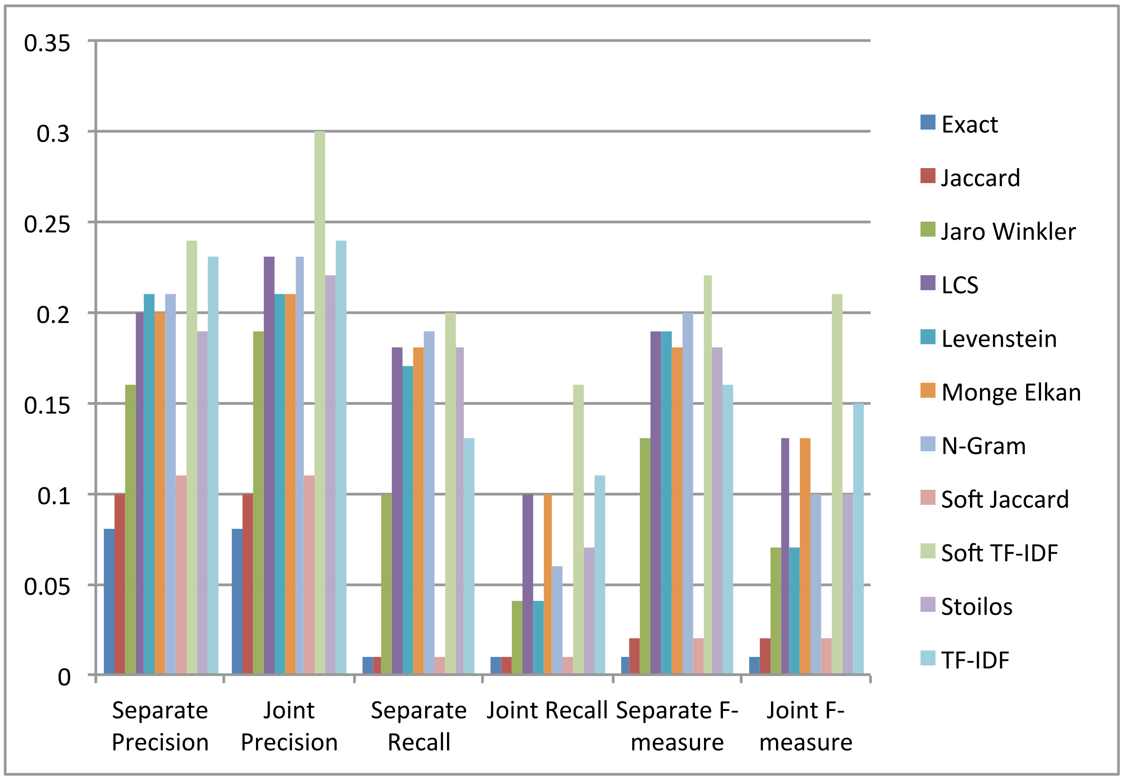

Figure 14: -measures of all metrics using tokenization on the multiform dataset when the thresholds were optimized once for classes and properties together versus separately for properties.

are some general guidelines:

- Standard ontology
  - Precision: All but Monge Elkan
  - Recall: TF-IDF
  - F-measure: All but Monge Elkan and LCS
- Multilingual
  - Precision: Soft Jaccard, Jaccard
  - Recall: Soft TF-IDF
  - F-measure: Soft TF-IDF
- Biomedical
  - Precision: Levenstein
  - Recall: Jaccard, Soft Jaccard, Soft TF-IDF
  - F-measure: Soft TF-IDF, Jaccard, Soft Jaccard

Of the pre-processing strategies analyzed, few were beneficial. Tokenization is useful if the naming conventions differ between the ontologies (camelCase versus underscores to separate words, for example). Translation is very helpful when ontologies involve multiple languages. If translation is not available, normalization can be useful for multilingual ontology pairs, particularly if one of the languages uses a non-Latin alphabet and can be transliterated. Synonyms can be useful (particularly with respect to recall) for biomedical ontologies, where the synonyms are often embedded in the ontologies themselves.

Class labels are significantly easier for string metrics to match than are property labels. Performance can be improved by using different thresholds for classes and properties. It would be helpful to look into this further by examining what enables some metrics to do better than others and potentially develop a new metric that emphasizes these strengths further when it comes to property labels.

8 String-centric Ontology Alignment

With this analysis of string similarity metrics as applied to ontology alignment, we now turn to the question of how far we can get using only these metrics. To answer this question we developed a very simple ontology alignment algorithm. This algorithm works in the same way as our test framework – comparing every label in the first ontology to every label

Metric	Precision	Recall	F-measure
YAM++	0.81	0.69	0.75
LogMap	0.82	0.58	0.68
StringsOpt	**0.85**	**0.55**	**0.67**
StringsAuto	**0.79**	**0.57**	**0.66**
Optima	0.62	0.68	0.65
CODI	0.74	0.57	0.64
GOMMA	0.85	0.47	0.61
Wmatch	0.74	0.50	0.60
WeSeE	0.76	0.49	0.60
Hertuda	0.74	0.50	0.60
MaasMatch	0.63	0.57	0.60
LogMapLt	0.73	0.50	0.59
HotMatch	0.71	0.51	0.59
Baseline 2	0.79	0.47	0.59
ServOMap	0.73	0.46	0.56
Baseline 1	0.80	0.43	0.56
ServOMapLt	0.88	0.40	0.55
MEDLEY	0.54	0.50	0.52
ASE	0.63	0.43	0.51
MapSSS	0.50	0.51	0.50
AUTOMSv2	0.67	0.36	0.47
AROMA	0.33	0.48	0.39

Table 1: Results of the StringsOpt alignment algorithm together with the competitors from the OAEI 2012 competition on the conference data set

in the second and using the stable marriage algorithm to find the best mappings. The difference is that here we run the algorithm repeatedly: first with a high-precision metric and then with a high-recall metric. If a mapping in the second pass involves an entity that has already been used in the previous pass then it is ignored. Because string metrics were found to perform extremely poorly on properties, this approach does not attempt to match those (e.g. any property matches in the reference alignment are automatically false negatives). For the anatomy test set, the approach used here first runs the high precision and high recall metrics on the entity labels themselves, and then considers synonyms. For this proof-of-concept, the algorithm is hardcoded with the optimal metrics and thresholds for the particular test set under consideration. The results are shown in tables 1, 2, and 3, along with the results of the OAEI 2012 competitors. This approach is labeled StringsOpt, to indicate that this is the optimal configuration of the framework for each alignment task.

Metric	Prec diff	F-meas diff	Rec diff	Prec same	F-meas same	Rec same
AUTOMSv2	0.49	0.17	0.10	0.69	0.11	0.06
GOMMA	0.29	0.32	0.36	0.63	0.40	0.29
MEDLEY	0.16	0.10	0.07	0.34	0.14	0.09
WeSeE	0.61	0.42	0.32	0.90	0.42	0.27
Wmatch	0.22	0.22	0.22	0.43	0.18	0.11
YAM++	0.50	0.42	0.36	0.91	0.64	0.49
StringsOpt	**0.58**	**0.40**	**0.31**	**0.90**	**0.38**	**0.24**
StringsAuto	**0.64**	**0.39**	**0.28**	**0.93**	**0.26**	**0.15**

Table 2: Results of the StringsOpt alignment algorithm together with the competitors from the OAEI 2012 competition on the multifarm data set

Using only optimized string similarity metrics achieves an f-measure of .67 on the conference data set, which makes it the third highest performer. The strings only approach also significantly outperforms the baselines, which are unrefined string metrics (Baseline 1 uses string equality and Baseline 2 is the same but with dashes, underscores and the word has removed from strings prior to comparison). For the multifarm test set, only the results of metrics that have been designed to handle multilingual alignment problems are shown. These results are divided into two groups: alignments of the same ontologies in different languages (labeled same) and alignments of different ontologies in different languages (labeled different). The strings only approach scores third when aligning different ontologies and fourth when aligning the same ontologies, using f-measure as the quality metric. This approach ties for fourth on the anatomy test set.

It is evident that these results compare very well with the current state-of-the-art in ontology alignment systems, but ***this is not an apples-to-apples comparison*** because this approach is not generic (due to the hard-coded metrics based on the test set). The next step is to add some means of selecting the appropriate string metrics and thresholds at runtime. Our goal is to develop a method that is fully autonomous and does not rely on any training data. We have started out with some basic observations based on the results we have gathered above:

- Precision does not vary widely among string similarity metrics for most standard ontologies.
- TF-IDF has high recall for most standard ontologies.
- When many entities in an ontology contain multiple words, it is best to use set-based string similarity metrics. Soft Jaccard and Soft TF-IDF often perform particularly well in these cases.

Metric	Precision	Recall	F-measure
GOMMA-bk	0.92	0.93	0.92
YAM++	0.94	0.86	0.90
CODI	0.97	0.83	0.89
StringsOpt	**0.88**	**0.87**	**0.88**
LogMap	0.92	0.85	0.88
GOMMA	0.96	0.80	0.87
StringsAuto	**0.86**	**0.84**	**0.85**
MapSSS	0.94	0.75	0.83
WeSeE	0.91	0.76	0.83
LogMapLt	0.96	0.73	0.83
TOAST*	0.85	0.76	0.80
ServOMap	1.00	0.64	0.78
ServOMapLt	0.99	0.64	0.78
HotMatch	0.98	0.64	0.77
AROMA	0.87	0.69	0.77
StringEquiv	1.00	0.62	0.77
Wmatch	0.86	0.68	0.76
Optima	0.85	0.58	0.69
Hertuda	0.69	0.67	0.68
MaasMatch++	0.43	0.78	0.56

Table 3: Results of the StringsOpt alignment algorithm together with the competitors from the OAEI 2012 competition on the anatomy data set

- Thresholds need to be lower when recall is of more concern than precision.
- Thresholds need to be lower when synonyms or translations are involved.

Based on these insights, we have developed an analysis module that runs before our main alignment algorithm to select the string metrics. This analysis module examines an ontology to find the answers to three simple questions:

- Is the ontology in English?
- What is the average number of words per entity label (after tokenization)?
- Does the ontology contain embedded synonyms?

The implementation of the analysis module is straightforward. The language of the ontology is determined by selecting ten entity labels and concatenating these into a single string, which is then used in a call to the translate function of the Google Translate API with English as the target language. This call is made with no value for the source language parameter, which causes Google to return its best guess as to the source language along with the translation. If the language is something other than English, the English translation is used in the remaining steps. The calculation of the average number of words is straightforward. Synonym detection is done simply by checking the input files for mention of the word synonym. The analysis module only considers classes - properties and instances are ignored.

Table 4 shows the average number of words per label for each data set. From this we see that the labels in biomedical ontologies are typically made up of more words than those in standard ontologies. Also interesting is that the number of words per label is slightly greater for the multi-lingual version of a test set than the same ontologies in English (i.e. the metrics are higher for multifarm than for conference and for cidx-to-excel (ML) than for plain cidx-to-excel). This seems to be because Google Translate sometimes produces a multiword phrase instead of a single word when performing translations.

Based on these results of the analysis module and whether precision or recall is currently of interest in the alignment process, a string metric is chosen. This is currently done using a hard-coded set of rules, but more research remains to be done in this area. When precision is the primary concern, it doesn't matter too much which metric we choose for most standard ontologies. We have decided to use Jaro-Winkler. In cases where multiple words per label are involved, word ordering often confuses the results. We therefore use the Soft Jaccard metric in these cases, with Levenstein as the base metric. When recall is the primary focus, we use TF-IDF for ontologies with predominantly one word per label and Soft TF-IDF for those with mostly multi-word labels. The thresholds used are shown in the decision tree below. In general, if translations or synonyms are involved then lower thresholds are appropriate. Note that these rules do not break cleanly among the different OAEI test sets – they are based on underlying features of the ontologies to be matched.

- Precision
  - Less than two words per label

    Jaro-Winkler 1, 1
  - Two or more words per label
    - Synonyms

      Soft Jaccard .2, .5 with Levenstein .9 base metric
    - No synonyms

      Soft Jaccard 1, 1 with Levenstein .8 base metric
- Recall
  - Less than two words per label

Test set	Words per label
conference	1.85
cidx-to-excel	1.57
multifarm	2.24
cidx-to-excel (ML)	1.61
anatomy	2.64
genes	4.11

Table 4: Comparison of data sets based on word length and number of words per label

TF-IDF .8, .8

- Two or more words per label
  - Synonyms

    Soft TF-IDF .5, .8 with Jaro-Winkler .8 base metric
  - Different Languages

    Soft TF-IDF 0, .7 with Jaro-Winkler .9 base metric
  - Other

    Soft TF-IDF .8, .8 with Jaro-Winkler .8 base metric

We have added this automatic metric selection step to our approach. The results for this are shown in tables 1, 2 and 3 under StringsAuto. We have also added it to MapSSS, an existing ontology alignment system [10]. The results for the version of MapSSS using these optimized metrics and string pre-processing strategies are compared with the results of the original system in Table 5. The deeper insight into string labels has significantly improved the performance of MapSSS on the conference test set and marginally improved it on the anatomy test set. The extremely large gains on the multiform test set are due to the inclusion of translation as a string pre-processing strategy.

9 Conclusions and Future Work

For some types of ontologies, the performance of different string similarity metrics varies greatly in terms of both precision and recall. It is important to be cognizant of this when selected a string metric for a particular use. This paper has established guidelines to assist researchers in making this selection. In addition, we have found that many string pre-processing strategies commonly used, such as stop word removal and word stemming are in many cases unhelpful and in some cases counter-productive. We have presented

Test Set	Measure	Original	Improved	Improvement	OAEI 2012
Conference	Precision	0.5	0.73	46%	Tied 11th
	Recall	0.51	0.57	12%	Tied 4th
	F-measure	0.5	0.64	28%	Tied 4th
Anatomy	Precision	0.94	0.86	-8%	Tied 14th
	Recall	0.75	0.84	12%	4th
	F-measure	0.83	0.85	2%	6th
Multifarm	Precision diff	0.08	0.45	463%	4th
	Recall diff	0.04	0.28	600%	4th
	F-measure diff	0.05	0.35	547%	3rd
	Precision same	0.97	0.96	-1%	1st
	Recall same	0.5	0.25	-50%	4th
	F-measure same	0.66	0.40	-40%	Tied 3rd

Table 5: Results of the original and improved MapSSS alignment algorithm on the OAEI 2012 data sets

data on which pre-processing strategies are useful in particular situations. In addition, we have developed a basic system to automatically select an appropriate string similarity metric for a given pair of ontologies at runtime. Finally, we have applied this technique to an existing ontology alignment algorithm and quantified the improvement in performance. Because nearly all ontology alignment algorithms make use of string similarity metrics, this work can similarly be integrated into other existing alignment algorithms and is therefore directly relevant to many researchers in this field.

There are several paths for future work based on the idea of pushing string similarity metrics as far as they can go in terms of ontology alignment. A first step is to develop a string similarity metric that performs better on properties. Another possibility is to create a string-based structural metric by considering the similarity between the labels of all of an entity's neighbors with those of another entity. For biomedical ontologies, it might also be possible to use string metrics to find subsumption relations in addition to equivalencies since many labels in these ontologies are of the form noun and modifiers+noun (e.g. vein and pulmonary vein). In terms of the work presented in this paper, the results should be validated for more pairs of ontologies. Also, the analysis module for metric selection can be made more flexible.

References

[1] Samur Araujo, Arjen de Vries, and Daniel Schwabe. Serimi results for oaei 2011. *Ontology Matching*, page 212, 2011.

[2] Michael Ashburner et al. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

[3] Tim Berners-Lee, Mark Fischetti, and Michael L Foreword By-Dertouzos. *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*. HarperInformation, 2000.

[4] Wayne L Bethea, Clayton Fink, and John Beecher-Deighan. Jhu/apl onto-mapology results for oaei 2006. In *Ontology Matching*, page 144, 2006.

[5] Jurgen Bock, Carsten Danschel, and Matthias Stumpp. Mappso and mapevo results for oaei 2011. In *Proc. 6th ISWC workshop on ontology matching (OM), Bonn (DE)*, pages 179–183, 2011.

[6] Jürgen Bock, Peng Liu, and Jan Hettenhausen. Mappso results for oaei 2009. In *OM*. Citeseer, 2009.

[7] Gosse Bouma. Cross-lingual dutch to english alignment using eurowordnet and dutch wikipedia. In *Proceedings of the 4th International Workshop on Ontology Matching, CEUR-WS*, volume 551, pages 224–229. Citeseer, 2009.

[8] L Karl Branting. A comparative evaluation of name-matching algorithms. In *Proceedings of the 9th International Conference on Artificial Intelligence and Law*, pages 224–232. ACM, 2003.

[9] Silvana Castano, Alfio Ferrara, and Gianpaolo Messa. Results of the hmatch ontology matchmaker in oaei 2006. In *Ontology Matching*, page 134, 2006.

[10] Michelle Cheatham. MapSSS results for OAEI 2011. In *Proceedings of the ISWC 2011 Workshop on Ontology Matching*, pages 184–190, 2011.

[11] Watson Wei Khong Chua and Jung-Jae Kim. Eff2match results for oaei 2010. *Ontology Matching*, page 150, 2010.

[12] William W Cohen, Pradeep Ravikumar, Stephen E Fienberg, et al. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, volume 47, 2003.

[13] Isabel Cruz, Flavio Palandri Antonelli, Cosmin Stroe, Ulas C Keles, and Angela Maduko. Using agreementmaker to align ontologies for oaei 2009: Overview, results, and outlook. In *Proceedings of the ISWC 2009 Workshop on Ontology Matching*, pages 135–146. Citeseer, 2009.

[14] Isabel F Cruz, Cosmin Stroe, Michele Caci, Federico Caimi, Matteo Palmonari, Flavio Palandri Antonelli, and Ulas C Keles. Using agreementmaker to align ontologies for oaei 2010. In *ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings*, volume 689, pages 118–125, 2010.

[15] Isabel F Cruz, Cosmin Stroe, Federico Caimi, Alessio Fabiani, Catia Pesquita, Francisco M Couto, and Matteo Palmonari. Using agreementmaker to align ontologies for oaei 2011? In *ISWC international workshop on ontology matching (OM)*, volume 814, pages 114–121, 2011.

[16] Carlo Curino, Giorgio Orsi, and Letizia Tanca. X-som results for oaei 2007. In *Proceedings of the Second International Workshop on Ontology Matching*, pages 276–285. Citeseer, 2007.

[17] Jérôme David. Aroma results for oaei 2008. In *Proc. 3rd ISWC workshop on ontology matching (OM)*, pages 128–131, 2008.

[18] Jérôme David. Aroma results for oaei 2011. *Ontology Matching*, page 122, 2011.

[19] Jean-François Djoufak-Kengue, Jérôme Euzenat, Petko Valtchev, et al. Ola in the oaei 2007 evaluation contest. In *Proc. 2nd ISWC 2007 workshop on ontology matching (OM)*, pages 188–195, 2007.

[20] Richard Durbin. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[21] Jérôme Euzenat et al. State of the art on ontology alignment. *Knowledge Web Deliverable D*, 2:2–3, 2004.

[22] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*, volume 18. Springer Heidelberg, 2007.

[23] Sylvain Gaudan, A Jimeno Yepes, Vivian Lee, Dietrich Rebholz-Schuhmann, et al. Combining evidence, specificity, and proximity towards the normalization of gene ontology terms in text. *EURASIP Journal on Bioinformatics and Systems Biology*, 2008, 2008.

[24] Osamu Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705–708, 1982.

[25] Jorge Gracia, Jordi Bernad, and Eduardo Mena. Ontology matching with cider: evaluation report for oaei 2011. *Ontology Matching*, page 126, 2011.

[26] Jorge Gracia and Eduardo Mena. Matching with cider: Evaluation report for the oaei 2008. In *3rd Ontology Matching Workshop (OM?08) at the 7th International Semantic Web Conference (ISWC?08), Karlsruhe, Germany*, 2008.

[27] Fayçal Hamdi, Brigitte Safar, Nobal Niraula, Chantal Reynaud, et al. Taxomap in the oaei 2009 alignment contest. In *The Fourth International Workshop on Ontology Matching*, 2009.

[28] Fayçal Hamdi, Brigitte Safar, Nobal B Niraula, and Chantal Reynaud. Taxomap alignment and refinement modules: Results for oaei 2010. *Ontology Matching*, page 212, 2010.

[29] Faycal Hamdi, Haifa Zargayouna, Brigitte Safar, and Chantal Reynaud. Taxomap in the oaei 2008 alignment contest, ontology alignment evaluation initiative (oaei) 2008 campaign-int. In *Workshop on Ontology Matching*, 2008.

[30] Wei Hu, Jianfeng Chen, Gong Cheng, and Yuzhong Qu. Objectcoref & falcon-ao: results for oaei 2010. *Ontology Matching*, page 158, 2010.

[31] Wei Hu, Gong Cheng, Dongdong Zheng, Xinyu Zhong, and Yuzhong Qu. The results of falcon-ao in the oaei 2006 campaign. In *Ontology Matching*, page 124, 2006.

[32] Wei Hu, Yuanyuan Zhao, Dan Li, Gong Cheng, Honghan Wu, and Yuzhong Qu. Falcon-ao: Results for oaei 2007. In *OM*, 2007.

[33] Jakob Huber, Timo Sztyler, Jan Noessner, and Christian Meilicke. Codi: Combinatorial optimization for data integration–results for oaei 2011. *Ontology Matching*, page 134, 2011.

[34] Yves R Jean-Mary, E Patrick Shironoshita, and Mansur R Kabuka. Asmov: Results for oaei 2010. *Ontology Matching*, 126, 2010.

[35] Ernesto Jiménez-Ruiz, Antón Morant, and Bernardo Cuenca Grau. Logmap results for oaei 2011. *Ontology Matching*, page 163, 2011.

[36] Marouen Kachroudi, Essia Ben Moussa, Sami Zghal, and Sadok Ben. Ldoa results for oaei 2011. *Ontology Matching*, page 148, 2011.

[37] Ching-Chieh Kiu and Chien-Sing Lee. Ontodna: Ontology alignment results for oaei 2007. In *OM*, 2007.

[38] Konstantinos Kotis, Alexandros G Valarakos, and George A Vouros. Automs: Automated ontology mapping through synthesis of methods. In *Ontology Matching*, page 96, 2006.

[39] Patrick Lambrix, He Tan, and Qiang Liu. Sambo and sambodtf results for the ontology alignment evaluation initiative 2008. In *Proceedings of the Third International Workshop on Ontology Matching*, pages 190–198, 2008.

[40] Juanzi Li, Jie Tang, Yi Li, and Qiong Luo. Rimom: A dynamic multistrategy ontology alignment framework. *Knowledge and Data Engineering, IEEE Transactions on*, 21(8):1218–1232, 2009.

[41] Yi Li, Juan-Zi Li, Duo Zhang, and Jie Tang. Result of ontology alignment with rimom at oaei'06. In *Ontology Matching*, page 181, 2006.

[42] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, volume 1, pages 296–304. San Francisco, 1998.

[43] Alexander Maedche and Steffen Staab. Measuring similarity between ontologies. In *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 251–263. Springer, 2002.

[44] Ming Mao and Yefei Peng. Prior system: Results for oaei 2006. In *Ontology Matching*, page 173, 2006.

[45] Ming Mao and Yefei Peng. The prior+: Results for oaei campaign 2007. In *OM*, 2007.

[46] Sabine Massmann, Daniel Engmann, and Erhard Rahm. Coma++: Results for the ontology alignment contest oaei 2006. In *Ontology Matching*, page 107, 2006.

[47] Alvaro E Monge and Charles Elkan. The field matching problem: Algorithms and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 267–270, 1996.

[48] Miklos Nagy, Maria Vargas-Vera, and Enrico Motta. Dssim-ontology mapping with uncertainty. 2006.

[49] Miklos Nagy, Maria Vargas-Vera, and Enrico Motta. Dssim-managing uncertainty on the semantic web. 2007.

[50] Miklos Nagy, Maria Vargas-Vera, Piotr Stolarski, and Enrico Motta. Dssim results for oaei 2008. In *Proceedings of the Third International Workshop on Ontology Matching*, pages 147–159, 2008.

[51] Duy Hoa Ngo, Zohra Bellahsene, Remi Coletta, et al. Yam++–results for oaei 2011. In *ISWC'11: The 6th International Workshop on Ontology Matching*, volume 814, pages 228–235, 2011.

[52] Sławomir Niedbała. Owl ctxmatch in the oaei 2006 alignment contest. *Ontology Matching*, page 165, 2006.

[53] Xing Niu, Shu Rong, Yunlong Zhang, and Haofen Wang. Zhishi. links results for oaei 2011. *Ontology Matching*, page 220, 2011.

[54] Jan Noessner and Mathias Niepert. Codi: Combinatorial optimization for data integration–results for oaei 2010. *Ontology Matching*, page 142, 2010.

[55] Roelant Ossewaarde. Simple library thesaurus alignment with silas. In *OM*, 2007.

[56] Catia Pesquita, Cosmin Stroe, Isabel Cruz, and Francisco M Couto. Blooms on agreementmaker: results for oaei 2010. *Ontology Matching*, page 134, 2010.

[57] Martin F Porter. An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14(3):130–137, 1980.

[58] Christoph Quix, Avigdor Gal, Tomer Sagi, and David Kensche. An integrated matching system: Geromesuite and smb–results for oaei 2010. *Ontology Matching*, page 166, 2010.

[59] Christoph Quix, Sandra Geisler, David Kensche, and Xiang Li. Results of geromesuite for oaei 2008. In *Proceedings of the Third International Workshop on Ontology Matching*, pages 160–166, 2008.

[60] Quentin Reul and Jeff Z Pan. Kosimap: ontology alignments results for oaei 2009. In *Proceedings of the ISWC 2009 Workshop on Ontology Matching*, pages 177–185. Citeseer, 2009.

[61] M. Sabou and J. Gracia. Spider: Bringing non-equivalence mappings to oaei. In *Proceedings of the Third International Workshop on Ontology Matching*, 2008.

[62] Fatiha Saıs, Nobal Niraula, Nathalie Pernelle, and Marie-Christine Rousset. Ln2r–a knowledge based reference reconciliation system: Oaei 2010 results. *Ontology Matching*, page 172, 2010.

[63] Frederik C Schadd and Nico Roos. Maasmatch results for oaei 2011. *Ontology Matching*, page 171, 2011.

[64] Md Hanif Seddiqui and Masaki Aono. Alignment results of anchor-flood algorithm for oaei-2008. In *Proceedings of Ontology Matching Workshop of the 7th International Semantic Web Conference, Karlsruhe, Germany*, pages 120–127, 2008.

[65] Md Hanif Seddiqui and Masaki Aono. Anchor-flood: results for oaei 2009. In *Proceedings of the ISWC 2009 Workshop on Ontology Matching*, pages 127–134. Citeseer, 2009.

[66] Guohua Shen, Lantao Jin, Ziyue Zhao, Zhe Jia, Wenmin He, and Zhiqiu Huang. Omreasoner: using reasoner for ontology matching: results for oaei 2011. *Ontology Matching*, page 197, 2011.

[67] Vassilis Spiliopoulos, Alexandros G Valarakos, George A Vouros, and Vangelis Karkaletsis. Sema: Results for the ontology alignment contest oaei 2007. In *OM*, 2007.

[68] Heiko Stoermer and Nataliya Rassadko. Results of okkam feature based entity matching algorithm for instance matching contest of oaei 2009. *Ontology Matching*, page 200, 2009.

[69] Giorgos Stoilos, Giorgos Stamou, and Stefanos Kollias. A string metric for ontology alignment. In *The Semantic Web–ISWC 2005*, pages 624–637. Springer, 2005.

[70] William Sunna and Isabel F Cruz. Using the agreementmaker to align ontologies for the oaei campaign 2007. In *OM*. Citeseer, 2007.

[71] He Tan and Patrick Lambrix. Sambo results for the ontology alignment evaluation initiative 2007. In *OM*, 2007.

[72] Uthayasanker Thayasivam and Prashant Doshi. Optima results for oaei 2011. In *Proc. of 6th OM Workshop*, pages 204–211, 2011.

[73] Quang-Vinh Tran, Ryutaro Ichise, and Bao-Quoc Ho. Clusterbased similarity aggregation for ontology matching. In *Proc. 6th ISWC workshop on ontology matching (OM), Bonn (DE)*, pages 142–147, 2011.

[74] Alexandros G Valarakos, Georgios Paliouras, Vangelis Karkaletsis, and George Vouros. A name-matching algorithm for supporting ontology enrichment. In *Methods and Applications of Artificial Intelligence*, pages 381–389. Springer, 2004.

[75] Peng Wang. Lily results on seals platform for oaei 2011. In *Proc. of 6th OM Workshop*, pages 156–162, 2011.

[76] Peng Wang and Baowen Xu. Lily: the results for the ontology alignment contest oaei 2007. In *Proceedings of the Second International Workshop on Ontology Matching*, pages 179–187. Citeseer, 2007.

[77] Peng Wang and Baowen Xu. Lily: Ontology alignment results for oaei 2008. In *Proceedings of the Third International Workshop on Ontology Matching*, pages 167–175, 2008.

[78] Song Wang, Gang Wang, and Xiaoguang Liu. Results of nbjlm for oaei 2010. *Ontology Matching*, page 187, 2010.

[79] Zhichun Wang, Xiao Zhang, Lei Hou, Yue Zhao, Juanzi Li, Yu Qi, and Jie Tang. Rimom results for oaei 2010. *Ontology Matching*, 195, 2010.

[80] Peigang Xu, Yadong Wang, Liang Cheng, and Tianyi Zang. Alignment results of sobom for oaei 2010. In *OM*, 2010.

[81] Boutheina Ben Yaghlane and Najoua Laamari. Owl-cm: Owl combining matcher based on belief functions theory. In *OM*. Citeseer, 2007.

[82] Haifa Zargayouna, Brigitte Safar, Chantal Reynaud, et al. Taxomap in the oaei 2007 alignment contest. In *Proceedings of The Second International Workshop on Ontology Matching (OM'07)*, pages 268–275, 2007.

[83] Sami Zghal, Marouen Kachroudi, Sadok Ben Yahia, and Engelbert MEPHU. Oacas: results for oaei 2011. *Ontology Matching*, page 190, 2011.

[84] Sami Zghal, Sadok Ben Yahia, Engelbert Mephu Nguifo, Yahya Slimani, et al. Soda: an owl-dl based ontology matching system. In *OM*, 2007.

[85] Songmao Zhang and Olivier Bodenreider. Nlm anatomical ontology alignment system. results of the 2006 ontology alignment contest. In *Ontology Matching*, page 153, 2006.

[86] Songmao Zhang and Olivier Bodenreider. Hybrid alignment strategy for anatomical ontologies: Results of the 2007 ontology alignment contest. In *OM*, 2007.

[87] Xiao Zhang, Qian Zhong, Juanzi Li, Jie Tang, Guotong Xie, and Hanyu Li. Rimom results for oaei 2008. In *Proceedings of the 3rd International Workshop on Ontology Matching*, volume 431, page 182, 2008.

A OAEI String Metric Survey

Algorithm	Lexical Metrics	Pre-processing
AgreementMaker [70, 13, 14, 15]	edit distance, LCS, TF-IDF	stemming, stop words, synonyms, normalization
Anchor-flood [64, 65]	JaroWinker, SMOA, SM	tokenization, abbrev/acronym expansion, stop words, categorization (WordNet)
AROMA [17, 18]	JaroWinkler, exact match	stemming
ASMOV [34]	exact match, Levenstein, set similarity metric for comments	tokenization, normalization, synonyms, part-of-speech
AUTOMS [38]	COCLU	
BLOOMS [56]	exact match, Jaccard, evidence content	tokenization, stop words, synonyms, normalization, stemming, spelling variants (proposed)
CIDER [26, 25]	exact match, Levenstein	normalization, synonyms
Cluster-based similarity aggregation [73]	edit distance, TF-IDF/cosine similarity	
CODI [54, 33]	Levenstein, cosine, Jaro-Winker, Smith-Waterman Gotoh, overlap coefficient, Jaccard	tokenization, normalization, stop words
COMA++ [46]		
Cross-lingual Dutch to English alignment [7]		stemming, split compound words, synonyms

DSSim [48, 49, 50]	Monge-Elkan, Jaccard, Jaro-Winkler	synonyms, split compound words, language tag, translations, abbrev expansion
Eff2Match [11]	exact match	tokenization, stemming, synonyms, normalization
Falcon-AO/ObjectCoref [31, 32, 30]	SMOA, edit distance	synonyms (proposed), translations (proposed)
GeRoMeSuite/SMB [59, 58]	Levenstein, Jaro-Winkler, SMOA, soft TF-IDF	synonyms
HMatch [9]		
Hybrid alignment strategy for anatomical ontologies [86]	exact match	synonyms, normalization
JHU/APL Onto-Mapology [4]	Jaro-Winkler, 2-gram, document indexing	stop words
KOSIMap [60]	Jaro-Winkler, Q-gram, SMOA, Monge-Elkan	stop words, stemming
LDOA [36]	Levenstein, Jaro-Winkler, soft Jaccard	
LILY [76, 77, 75]	Levenstein, edit distance	
LN2R [62]	soft Jaccard	synonyms (proposed)
LogMap [35]	exact match, SMOA	synonyms, alternate words forms (i.e. reverse stemming)
MaasMatch [63]	document vectors	stemming, stop words
MapPSO [6, 5]	SMOA, TF-IDF	
NBJLM [78]	exact match	synonyms

NLM Anatomical Ontology Alignment System [85]	exact match	normalization, synonyms, stemming
OACAS [83]	Levenstein, Q-gram, Jaro-Winkler	
OKKAM [68]	Levenstein	
OLA [19]	LCS, normalized Hamming distance, edit distance	tokenization, synonyms
OMReasoner [66]	edit distance, prefix, suffix	split compound words (proposed)
OntoDNA [37]	Levenstein	normalization, stop words
Optima [72]	Smith-Waterman	
OWL-CM [81]	edit distance	
OWL-CtxMatch [52]		synonyms
PRIOR/PRIOR+ [44, 45]	cosine similarity, Levenstein, document indexing	synonyms (proposed), translations (proposed)
RiMOM [41, 40, 87, 79]	edit distance, KNN, TF-IDF/cosine distance	tokenization, stemming, stop words, synonyms
SAMBO/SAMBOdtf [71, 39]	n-gram, edit distance	tokenization, stemming, stop words, synonyms
SEMA [67]	COCLU	synonyms
SERIMI [1]	RWSA	tokenization, normalization
SILAS [55]	exact match	
SOBOM [80]	edit distance, SMOA	

SODA [84]	Jaro-Winkler, Monge-Elkan	
Spider [61]		
TaxoMap [82, 29, 27, 28]	Lin's similarity measure, exact match, substring inclusion	stop words, part-of-speech, translation, synonyms (proposed)
X-SOM [16]	Jaro, Levenstein	
YAM++ [51]	Levenstein, Smith-Waterman, Jaro, Jaro-Winkler, Monge-Elkan, prefix, suffix, LCS, SMOA	
Zhishi.links [53]	exact match	