Refining Terminological Saturation using String Similarity Measures

Alyona Chugunenko[1[0000-0002-9760-3558]], Victoria Kosa[1[0000-0002-7300-8818]], Rodion Popov[1], David Chaves-Fraga[2[0000-0003-3236-2789]], and Vadim Ermolayev[1[0000-0002-5159-254X]]

[1] Department of Computer Science, Zaporizhzhya National University, Zhukovskogo st. 66, 69600, Zaporizhzhya, Ukraine
aluonac@i.ua, victoriya1402.kosa@gmail.com, rodeonpopov@gmail.com, vadim@ermolayev.com
[2] Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain
dchaves@fi.upm.es

Abstract. This paper reports on the refinement of the THD algorithm, developed in the OntoElect framework. This baseline THD algorithm used exact string matches for key term comparison. It has been refined by introducing an appropriate string similarity metric for grouping the terms having similar meaning and looking similar as text strings. To choose the most appropriate metric, several existing metrics have been cross-evaluated on the developed test set of multi-word terms in English. The rationale for creating this test set is also presented. Further, the refined algorithm for measuring terminological difference has been cross-evaluated with the baseline THD algorithm. For this cross-evaluation, the bags of terms extracted from the TIME collection of scientific papers were used. The experiment revealed that using the refined algorithm yielded better and quicker terminological saturation, compared to the baseline.

Keywords: Automated Term Extraction, OntoElect, Terminological Difference, Key Term, Linguistic Similarity Metric, Bag of Terms, Terminological Saturation.

1 Introduction

The research presented in this paper is the part of the development of the methodological and instrumental components for extracting representative (complete) sets of significant terms from the representative sub-collections of textual documents having minimal possible size. These terms are further interpreted as the required features for engineering an ontology in a particular domain of interest. Therefore, it is assumed that the documents in a collection cover a single and well circumscribed domain. The main hypothesis, put forward in this work, is that a sub-collection can be considered as representative to describe the domain, in terms of its terminological footprint, if any additions of extra documents from the entire collection to this sub-collection do not noticeably change this footprint. Such a sub-collection is further considered as complete and

therefore yields a representative bag of significant terms describing its domain. The approach to assess the representativeness does so by evaluating terminological saturation in a document (sub-)collection [1], [31].

Detecting saturation is done by measuring terminological difference (*thd*) among the pairs of the consecutive incrementally enlarged datasets, as described in Section 4. This set measure is of course based on measuring differences between individual terms. A (baseline) THD algorithm [1] has been developed and implemented in the OntoElect project[1]. This THD algorithm, however, uses a simple string equivalence check for detecting similar individual terms. The objective of the research presented in this paper was to find out if it is possible to achieve better performance in measuring terminological difference by using a proper string similarity measure to compare individual terms.

The remainder of the paper is structured as follows. Section 2 reviews the related work. Section 3 reports on the implementation of the chosen string similarity measures and selecting the proper term similarity thresholds for their use. Section 4 sketches out the approach of OntoElect for measuring *thd* and our refinement of the baseline THD algorithm. Section 5 presents the set-up and results of our evaluation experiments. Our conclusions and plans for the future work are given in Section 6.

2 Related Work

The work reported in this paper aims at improving the measures of terminological difference between the bags of terms extracted from textual documents. The improvement is sought via the proper choice and use of existing string metrics for measuring linguistic (dis)similarity between extracted terms, as opposed to the baseline THD algorithm [1] which uses text string equality measures for comparing terms. It is also the premise in our approach that the bags of terms are multi-word, extracted from plain text files, and accompanied by numeric significance (rank) values. The terms are also expected to be English. Therefore, the work related to the presented research is sought in automated term extraction (ATE) from English texts and string similarity (distance) measurement of the pairs of text strings containing one to several words.

In the majority of approaches to ATE, e.g. [2] or [3], processing is done in two consecutive phases: Linguistic Processing and Statistical Processing. Linguistic processors, like POS taggers or phrase chunkers, filter out stop words and restrict candidate terms to n-gram sequences: nouns or noun phrases, adjective-noun and noun-preposition-noun combinations. Statistical processing is then applied to measure the ranks of the candidate terms. These measures are [4] either the measures of "unithood", which focus on the collocation strength of units that comprise a single term; or the measures of "termhood" which point to the association strength of a term to domain concepts.

For "unithood", the metrics are used such as mutual information [5], log likelihood [6], t-test [2], [3], the notion of 'modifiability' and its variants [7], [3]. The metrics for "termhood" are either term frequency-based (unsupervised approaches) or reference corpora-based (semi-supervised approaches). The most used frequency-based metrics

[1] https://www.researchgate.net/project/OntoElect-a-Methodology-for-Domain-Ontology-Refinement

are TF/IDF (e.g. in [8], [9]), weirdness [10] which compares the frequency of a term in the evaluated corpus with that in the reference corpus, domain pertinence [11]. More recently, hybrid approaches were proposed, that combine "unithood" and "termhood" measurements in a single value. A representative metric is c/nc-value [12]. C/nc-value-based approaches to ATE have received their further evolution in many works, e.g. [2], [11], [13] to mention a few.

Linguistic Processing is organized and implemented in a very similar fashion in all ATE methods, except some of them that also include filtering out stop words. Stop words could be filtered out also at a cut-off step after statistical processing. So, in our review and selection we look at the second phase of Statistical Processing only. Statistical Processing is sometimes further split in two consecutive sub-phases of term candidate scoring, and ranking. For term candidates scoring, reflecting its likelihood of being a term, known methods could be distinguished by being based on (c.f. [8]) measuring occurrences frequencies (including word association), assessing occurrences contexts, using reference corpora, e.g. Wikipedia [14], topic modelling [15], [29].

Perhaps the most cited paper that compares string similarity (distance) metrics is [17]. In their cross-evaluation aimed at finding the proper metric for approximate name matching in databases, the authors of [17] used two metric functions based on edit distance: Levenstein distance [18]; and Monger-Elkan distance [19] metrics. Among the metrics based on other principles, they also mentioned Jaro [20], Jaro-Winkler [21] metrics; token-based Jaccard similarity index [22], TF/IDF based cosine similarity and several other corpus-based metrics.

The authors of [23] also acknowledge that there is a rich set of string similarity measures available in the literature, including character n-gram similarity [24], Levenstein distance [15], Jaro-Winkler measure [21], Jaccard similarity [22], tf-idf based cosine similarity [25], and Hidden Markov Model-based measure [26].

To the best of our knowledge, none of the published techniques in ATE use text string similarity (distance) measures to group linguistically similar terms. This is done in the work presented in this paper. Furthermore, none of the techniques, except OntoElect [1], [16], use terminological saturation measures to minimize the sets of documents necessary for extracting the bags of terms which represent a domain.

3 Implementation of String Similarity Measures and the Choice of Term Similarity Thresholds

From the variety of metrics, mentioned above, due to the specifics of our task of the approximate comparison of short strings containing a few words, we filtered out those: (i) that require long strings or sets of strings of a considerably big size; (ii) that are computationally hard. We also tried to keep the representatives of all kinds of string metrics in our short list as much as it was possible. As a result, we formed the following list of measures to be considered for further use:

- Levenstein distance, Hamming distance [27], Jaro similarity, and Jaro-Winkler similarity – edit distance based syntactic measures

- Jaccard similarity index – a token based measure
- Sørensen-Dice coefficient [28] – a bi-gram comparison based measure

Among those, Levenstein and Hamming distances appeared to be the least appropriate in our context due to their limitations. Levenstein returns an integer number of required edits, while the rest of the measures return normalized reals. So, it has not been clear if normalizing Levenstein would really make the result comparable to the other measures in a way to use the same term similarity threshold. Hamming is applicable only to the strings of equal lengths. So, adding spaces to the shorter string would really lower the precision of measurement. Therefore, it has finally been decided to use Jaro, Jaro-Winkler, Jaccard, and Sørencen-Dice for implementation and cross-evaluation in our work. Further, it is briefly explained how should the selected measures be computed and referred to their implementation code. After that, it is explained how term similarity thresholds have been chosen for these implemented measures.

Jaro similarity sim_j between two strings S_1 and S_2 is computed (1) as the minimal number of one character transforms to be done to the first term (string) for getting the second string in the compared pair.

$$sim_j = \begin{cases} 0, & if \quad m = 0 \\ 1/3 * (\frac{m}{|S_1|} + \frac{m}{|S_2|} + \frac{m-t}{m}) & otherwise \end{cases}, \tag{1}$$

where: $|S_1|$, $|S_2|$ are the lengths of the compared strings; m is the number of the matching characters; and t is the half of the number of transposed characters. The characters are matching if they are the same and their distance from the beginning of the string differs by no more than $\lfloor max(|S_1|, |S_2|)/2 \rfloor - 1$. The number of matching but having different sequence order symbols is the number of transposed characters.

Jaro-Winkler similarity measure $sim_{j\text{-}w}$ refines Jaro similarity measure sim_j by using a prefix scale value p which assigns better ratings to the strings that match from their beginnings for a prefix length l. Hence, for the two strings S_1 and S_2 it is computed as shown in (2).

$$sim_{j\text{-}w} = sim_j + l*p*(1 - sim_j), \tag{2}$$

where l is the length of a common prefix (up to a maximum of 4 characters); p is a constant scaling factor for how much the similarity value is adjusted upwards for having common prefixes (up to 0.25, otherwise the measure can become larger than 1; [21] suggests that p=0.1).

Sometimes Winkler's prefix bonus $l*p*(1 - sim_j)$ is given only to the pairs having Jaro similarity higher that a particular threshold. This threshold is suggested [21] to be equal to 0.7.

Jaccard similarity index sim_{ja} is a similarity measure for finite sets, characters in our case. It is computed, for the two strings S_1 and S_2, as the ratio between the cardinalities of the intersection and union of the character sets in S_1 and $S2$ as shown in (3).

$$sim_{ja} = (|S_1| \cap |S_2|) / (|S_1| \cup |S_2|) \tag{3}$$

Finally, Sørensen-Dice coefficient, regarded as a character string similarity measure, is computed by counting identical character bi-grams in S_1 and S_2 and relating these to the overall number of bi-grams in both strings – as shown in (4).

$$sim_{sd} = 2n_{\equiv} / (n_{S_1} + n_{S_2}), \tag{4}$$

where: $n_{\equiv}$ is the no of bi-grams found in S_1 and also in S_2; n_{S_1}, n_{S_2} are the numbers of all bi-grams in S_1 and S_2.

The functions for all four string similarity measures have been implemented[2] in Python 3.0 and return real values within [0, 1].

For the proper use of those functions it is however necessary to determine what would be a reasonable threshold to distinguish between (semantically) similar and not similar terms. For determining that, the following cases in string comparison need to be taken into account:

- Character **strings are fully the same** – **Full Positives (FP)**. This case clearly falls into similar (the same) terms.
- Character **strings are very different** and the terms in these strings carry **different semantics** – **Full Negatives (FN)**. This case is also clear and is characterized by low values of similarity measures.
- Character **strings are partially the same** and the terms in these strings carry the **same or similar semantics** – **Partial Positives (PP)**.

The terms in such strings are similar, though it may not be fully clear. The following are different categories of terms that bring us about this case: words in the terms have different endings (e.g. plural/singular forms); different delimiters are used (e.g. "-", or "–", or " - "); a symbol is missing, erroneously added, or misspelled (a typo); one term is a sub-string of the other (e.g. subsuming the second); one of the strings contains unnecessary extra characters (e.g. two or three spaces instead of one, or noise).

- Character **strings are partially the same** but the terms in these strings carry **different semantics** – **Partial Negatives (PN)**

The terms in such strings are different, though it may not be fully clear. The following are the categories that bring us about this case: the terms carried by the compared strings differ by a few characters, but have different meanings (e.g. "deprecate" versus "depreciate"); the compared terms have common word(s) but fully differ in their meanings (e.g. "affect them" versus "effect them"). These false positives are the hardest case to be detected.

The test set of term pairs falling into the cases described above has been manually developed[3]. For each pair of terms in this test set all four string similarity measures have been computed.

---

[2] These functions are publicly available at: https://github.com/EvaZsu/OntoElect

[3] The test set and computed term similarity values are publicly available at https://github.com/EvaZsu/OntoElect/blob/master/Test-Set.xls

We have computed the average values of all four similarity measures for each category using all the test set term pairs falling into this category. The results are given in Table 1.

Table 1: Average similarity measure values for different categories of term pairs from the test set

Case / Category	**Items in Test Set**	**Sørensen-Dice**	**Jaccard**	**Jaro**	**Jaro-Winkler**
Different strings (FN)	**6**	**0.03**	**0.45**	**0.55**	**0.55**
Identical strings (FP)	**3**	**1.00**	**1.00**	**1.00**	**1.00**
Similar Semantics (PP)	**32**	**0.71**	**0.72**	**0.63**	**0.70**
- Unnecessary (extra) characters	7	0.8401	0.8820	0.8714	0.8784
- Common parts (words)	6	0.7122	0.7280	0.6375	0.7043
- Typos	6	0.7797	0.8637	0.8863	0.9220
- Different delimiters	6	0.7860	0.8473	0.9125	0.9442
- Different endings	7	0.8911	0.9135	0.9410	0.9590
Different Semantics (PN)	**18**	**0.89**	**0.89**	**0.89**	**0.91**
- Common parts (words)	11	0.4336	0.5221	0.6161	0.6408
- Very few character differences	7	0.8826	0.8845	0.8914	0.9059
Total:	**59**				

Term similarity thresholds have to be chosen such that full and partial negatives are regarded as not similar, but full and partial positives are regarded as similar. Hence, for the case of partial positives, the thresholds have to be chosen as minimal of all the case categories, and for the partial negatives – as the maximal of all the case categories. The values of case thresholds are shown bolded in Table 1 and provide us with the margins for relevant threshold intervals in our experiments. These intervals have been evenly split in four points as presented in Table 2. The requirements for partial positives and negatives unfortunately contradict to each other. For example, if a threshold is chosen to filter out partial negatives, also some of the partial positives will be filtered out. Therefore, subsuming that partial negatives are rare, it has been decided to use the thresholds for partial positives.

Table 2: Term similarity thresholds chosen for experimental evaluation

	Term Similarity Thresholds			
	Min	*Ave-1*	*Ave-2*	*Max*
Sørensen-Dice	0.71	0.76	0.83	0.89
Jaccard	0.72	0.77	0.83	0.89
Jaro	0.63	0.72	0.80	0.89
Jaro-Winkler	0.70	0.77	0.84	0.91

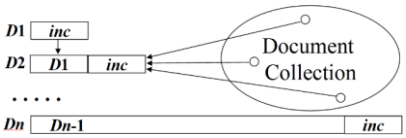
4 OntoElect and the Refinement of the THD Algorithm

OntoElect, as a methodology, seeks for maximizing the fitness of the developed ontology regarding what the domain knowledge stakeholders think about the domain. Fitness

is measured as the stakeholders' "votes" – a measure that allows assessing the stakeholders' commitment to the ontology under development – reflecting how well their sentiment about the requirements is met. The more votes are collected – the higher the commitment is expected to be. If a critical mass of votes is acquired (say 50%+1, which is a simple majority vote), the ontology is considered to satisfactorily meet the requirements.

Unfortunately, direct acquisition of requirements from domain experts is not very realistic as they are expensive and not really willing to do the work falling out of their core activity. So, we focus on the indirect collection of the stakeholders' votes by extracting these from high quality and reasonably high impact documents authored by the stakeholders.

An important feature to be ensured for knowledge extraction from text collections is that the dataset needs to be representative to cover the opinions of the domain knowledge stakeholders satisfactorily fully. OntoElect suggests a method to measure the terminological completeness of the document collection by analyzing the *saturation* of terminological footprints of the incremental slices of the document collection [1]. The full texts of the documents from a retrospective collection are grouped in datasets in the order of their timestamps. As pictured in Fig. 1a, the first dataset *D*1 contains the first portion (*inc*) of documents. The second dataset *D*2 contains the first dataset *D*1 plus the second incremental slice (*inc*) of documents. Finally, the last dataset *Dn* contains all the documents from the collection.



(a)

Term	c-value
temporal	6428.5
time	3950
temporal logic	2558
logic	2186.5
interval	2011
model	1957
formula	1750
relations	1613
temporal representation	1604
relation	1574.5
constraints	1557
reasoning	1549
....	

(b)

Fig. 1: (a) Incrementally enlarged datasets in OntoElect; (b) an example of a bag of terms extracted by UPM Term Extractor [30].

At the next step of the OntoElect workflow the bags of multi-word terms *B*1, *B*2, …, *Bn* are extracted from the datasets *D*1, *D*2, …, *Dn*, using UPM Term Extractor software [30], together with their *significance* (*c-value*) scores. Please see an example of an extracted bag of terms extracted in Fig. 1b.

At the subsequent step, every extracted bag of terms *Bi*, *i* = 1, …, *n* is processed as follows:

- **Normalized scores** are computed for each individual term: *n-score* = *c-value* / max(*c-value*)
- **Individual term significance threshold** (*eps*) is computed to cut off those terms that are not within the majority vote. The sum of *n-scores* having values above *eps* form the majority vote if this sum is higher that ½ of the sum of all *n-scores*.

- The **cut-off** at *n-score* < *eps* is done
- The result is saved in *Ti*

After this step only significant terms, whose *n-scores* represent the majority vote, are retained in the bags of terms. *Ti* are then evaluated for saturation by measuring pairwise terminological difference between the subsequent bags *Ti* and *Ti*+1, $i = 0, \ldots, n\text{-}1$. So far it has been done by applying the baseline THD algorithm[4] [1] presented in Fig. 2.

```
Algorithm THD. Compute Terminological Difference between Bags of Terms
Input:
 Ti, Ti+1 – the bags of terms with grouped similar terms.
      Each term Ti.term is accompanied with its T.n-score.
      Ti, Ti+1 are sorted in the descending order of T.n-score.
 M –  the name of the string similarity measure function to compare terms
 th – the value of the term similarity threshold from within [0,1]
Output: thd(Ti+1, Ti), thdr(Ti+1, Ti)
1.  sum := 0
2.  thd := 0
3.  for k := 1, |Ti+1|
4.    sum :=  sum + Ti+1.n-score[k]
5.    found : = .F.
6.    for m := 1, |Ti|
7.       if (Ti+1.term[k] = Ti.term[m]) if (M(Ti+1.term[k], Ti.term[m], th))
8.           then
9.             thd += |Ti+1.n-score[k] - Ti.n-score[m]|
10.            found := .T.
11.   end for
12.   if (found = .F.) then thd += Ti+1.n-score[k]
13. end for
14. thdr := thd  / sum
```

Fig. 2: THD algorithm [1] for measuring terminological difference in a pair of bags of terms. It uses string equalities for comparing terms and therefore needs to be refined as outlined by the rounded rectangles. The refined THD has two more input parameters (*M* and *th*) and uses *M* for comparing terms (line 7) instead of checking the equality of character strings.

In fact, THD accumulates, in the *thd* value for the bag *Ti*+1, the *n-score* differences if there were the same terms in *Ti* and *Ti*+1. If there was no the same term in *Ti*, it adds the *n-score* of the orphan to the *thd* value of *Ti*+1. After *thd* has been computed, the relative terminological difference *thdr* receives its value as *thd* divided by the sum of *n-scores* in *Ti*+1.

Absolute (*thd*) and relative (*thdr*) terminological differences are computed for further assessing if *Ti*+1 differs from *Ti* more than the individual term significance threshold *eps*. If not, it implies that adding an increment of documents to *Di* for producing *Di*+1 did not contribute any noticeable amount of new terminology. So, the subset *Di*+1 of the overall document collection may have become terminologically saturated. However, to obtain more confidence about the saturation, OntoElect suggests that some

[4] The baseline THD algorithm is implemented in Python and is publicly available at https://github.com/bwtgroup/SSRTDC-modules/tree/master/THD

more subsequent pairs of *Ti* and *Ti*+1 are evaluated. If stable saturation is observed, then the process of looking for a minimal saturated sub-collection could be stopped.

Our task was to modify the THD algorithm in a way to allow finding not exactly the same but sufficiently similar terms by applying string similarity measures with appropriate thresholds – as explained in the previous Section 3. For that, the preparatory similar term grouping step has been introduced to avoid duplicate similarity detection.

For each of the compared bags of terms *Ti* and *Ti*+1 the similar term grouping (STG) algorithm is applied at this preparatory step – see Fig. 3.

```
Algorithm STG. Group similar terms in the bag of terms
Input:
 T  – a bag of terms. Each term T.term is accompanied with its
      T.n-score. T is sorted in the descending order of T.n-score.
 M – the name of the string similarity measure function to compare
     terms
 th – the value of the term similarity threshold from within [0,1]
Output: T with grouped similar terms
1.  sum := 0
2.  for k = 1,|T|
3.    term :=  T.term[k]
4.    n-score := T.n-score[k]
5.    count := 1
6.    for m = k+1,|T|
7.        if M(term, T.term[m], th)
8.           then
9.             n-score += T.n-score[m]
10.            count += 1
11.            remove(T[m])
12.   end for
13.   T.n-score[k] := n-score / count
14. end for
```

Fig. 3: Similar Term Grouping (STG) algorithm

After term grouping is accomplished for both bags of terms, the refined THD algorithm (Fig 2 – rounded rectangles) is performed to compute the terminological difference between *Ti* and *Ti*+1.

5 Cross-Evaluation

This section reports on our evaluation of the refined THD algorithm against the baseline THD [1]. This evaluation is done following the workflow of OntoElect Requirements Elicitation Phase [31] and using the TIME document collection.

5.1 Set-up of the Experiment

The objective of our experiment was to find out if using the refined THD algorithm yields quicker and smoother terminological saturation compared to the use of the baseline THD algorithm. We were also looking at finding out which string similarity measure best fits for measuring terminological saturation.

For making the results comparable, the same datasets created from the TIME document collection – as described in Section 5.2 – has been fed into both the refined and baseline THD algorithms. We applied:

(i) The refined THD – sixteen times – one per individual string similarity measure *M* (Section 3) and per individual term similarity threshold *th* (Table 3); and
(ii) The baseline THD – one time

The values of: (i) the No of retained terns; (ii) absolute terminological difference (*thd*); and (iii) the time taken to perform term grouping by the STG algorithm (*sec*); were measured.

Finally, to verify if the refined THD is correct, we checked if it returns the same results as the baseline THD when the term similarity threshold is set to 1.0.

All the computations have been run on a Windows 10 64-bit PC with: Intel® Core™ 2 Duo CPU, E7400 @ 2.80 GHz; 4.0 Gb on-board memory.

5.2 Experimental Data

TIME document collection contains the full text papers of the proceedings of the TIME Symposia series[5]. The domain of the collection is Time Representation and Reasoning. The publisher of these papers is IEEE. It contains all the papers published in the TIME symposia proceedings between 1994 and 2013, which are 437 full text documents. These papers have been processed manually, including their conversion to plain texts and cleaning of these texts. So, the resulting datasets were not very noisy. We have chosen the increment for generating the datasets to be 20 papers. So, based on the available texts, we have generated 22 incrementally enlarged datasets $D1$, $D2$, …, $D22$[6] using our Dataset Generator[7]. The **chronological** order of adding documents has been used.

5.3 Results and Discussion

The results of our measurements of terminological saturation (*thd*) are pictured in a diagrammatic form in Fig. 4. The diagrams showing the time spent by the STG algorithm for detecting and grouping similar terms, based on the chosen term similarity thresholds are in Fig. 6. The diagrams in Fig. 4 and 6 have been built using the values

---

[5] http://time.di.unimi.it/TIME_Home.html

[6] The **TIME** collection in plain text and the datasets generated of these texts are available at: https://www.dropbox.com/sh/64pbodb2dmpndcy/AAAzVW7aEpgW-JrXHaCEqg2Sa/TIME?dl=0

[7] The dataset generator is available at: https://github.com/bwtgroup/SSRTDC-PDF2TXT

of the measurements from the four tables – one per term similarity threshold point (*Min*, *Ave*-1, *Ave*-2, and *Max*)[8].

Saturation (*thd*) measurements reveal that the refined THD algorithm detected terminological saturation faster than the baseline THD algorithm – no matter what was the chosen term similarity measure (*M*) or the similarity threshold (*th*). If the results for different measures are compared, then it may be noted that the respective saturation curves behave differently, depending on the similarity threshold point.

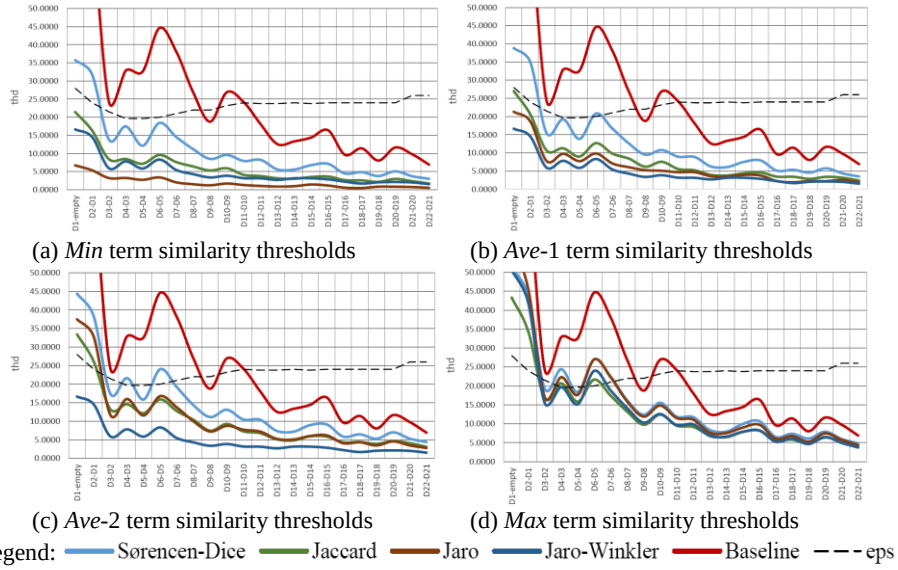


Fig. 4: Terminological saturation measurements grouped in four different term similarity threshold (*th*) points: (a) *Min*; (b) *Ave*-1; (c) *Ave*-2; and (d) *Max*. The legend shows the colors for different string similarity measures.

Overall, as it could be seen in Fig 4 (a) – (d), the use of the Sørensen-Dice measure demonstrated the least volatile behavior along the term similarity threshold points. This measure resulted in making the refined THD algorithm to detect saturation slower than the three other measures for *Min*, *Ave*-1, and *Ave*-2. For Max, it was as fast as Jaro and slightly slower than Jaccard and Jaro-Winker.

One more observation was that, integrally, all the implemented term similarity measures coped well with retaining important terms. These are indicated by terminology contribution peaks in the diagrams (a)-(d) of Fig. 4. It is well seen in Fig. 4(d), for the *Max* threshold point, that all the string similarity method curves follow the shape of the baseline THD curve quite closely. Hence, they have the peaks exactly in the same *thd* measurement points where the baseline has, pointing at more new significant terms.

---

[8] The tables are not presented in the paper due to the page limit, though are publicly available at: https://github.com/EvaZsu/OntoElect. File names are Results-Alltogether-{min, ave, ave2, max}-th.xlsx

At Min, Ave-1, and Ave-2, however, the method that have been most sensitive to terminology peaks, was Sørensen-Dice. This sensitivity is also confirmed by Fig. 5.

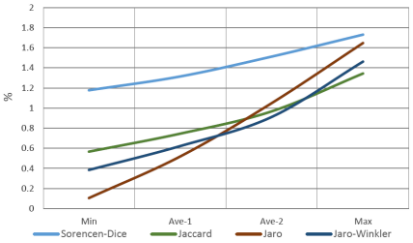


Fig. 5: Proportions of retained to all extracted terms for different term similarity measures

Fig. 5 pictures the proportions of the retained to all extracted terms computed at different term similarity threshold points. It is clear from Fig. 5 that Sørensen-Dice retains the biggest number of terms at all used term similarity thresholds.

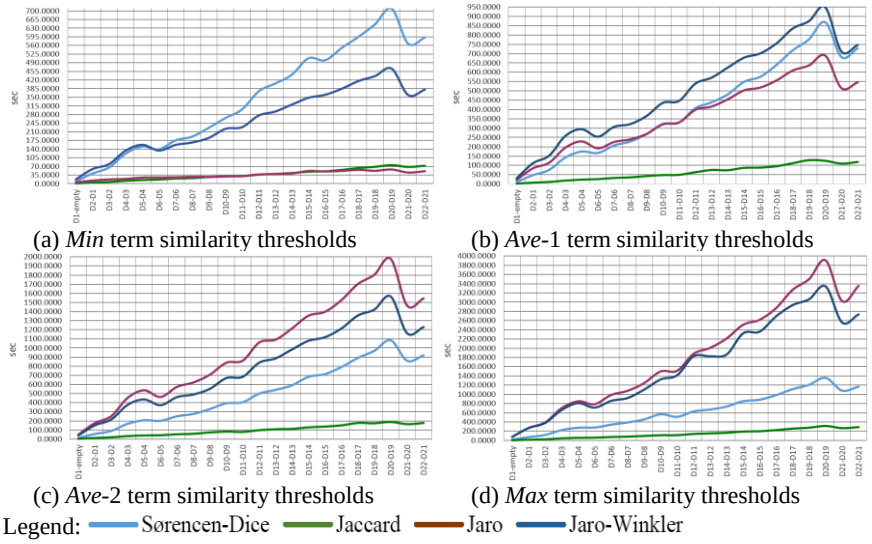


Fig. 6: Time (sec) spent for finding similar terms, grouped similarly to Fig. 4

Finally, it has to be noted that the introduction of string similarity measures in the computation of terminological difference (THD algorithm) increases the computational complexity of the algorithm quite substantially. Fig. 6 pictures the times (in seconds) taken by the pre-processor STG algorithm. As it could be noticed in Fig. 6(a)-(d), the times grow with the value of the term similarity threshold (*th*) and reach thousands of seconds for *Max* threshold values. It is interesting to notice that Sørensen-Dice and Jaccard are substantially more stable to the increase of *th* than Jaro and Jaro-Winkler. Sørensen-Dice takes, however, roughly an order of magnitude more time than Jaccard.

From the other hand, Jaccard was not very sensitive to terminological peaks and retained significantly less terms than Sørensen-Dice.

To sum up, the findings are put in Table 3 to rank the evaluated string similarity measures on a scale 1 (the best) to 5 (the worst).

Table 3: The ranking of the evaluated string similarity measures

Evaluation aspect	Rank (1-5)				
	Baseline THD	**Sørensen-Dice**	**Jaccard**	**Jaro**	**Jaro-Winkler**
Faster detection of terminological saturation	5	3	1	4	2
More significant terms retained	1	2	3	5	4
Less time taken	1	3	2	5	4
Total:	**7(2)**	**8(3)**	**6(1)**	**14(5)**	**10(4)**

Probably surprisingly, Jaccard, which is the most lightweight string similarity measure (Fig. 6), demonstrated the best performance among the rest, including the baseline THD. As it was well balanced on all evaluation aspects. This balance was also good in the case of Sørensen-Dice. However, Sørensen-Dice lost to Jaccard and baseline THD as it took too much time for term grouping. Jaro and Jaro-Winkler were clear negative outliers. Therefore, at an expense of a slightly higher execution time, the THD refined by Jaccard string similarity measure is the preferred choice for measuring terminological saturation in OntoElect.

6 Conclusions and Future Work

In this paper, we investigated if a simple string equivalence measure used in the baseline THD algorithm may be outperformed if a proper string similarity measure is used instead. For finding this out, we: (i) have chosen the four candidate measures from the broader variety of the available, based on the specifics of term comparison; (ii) developed the test set of specific term pairs to decide about term similarity thresholds for the chosen measures; (iii) implemented these measures, the algorithm for similar terms grouping (STG), and the refinement of the baseline THD algorithm; (iv) cross-evaluated the refined THD algorithm against the baseline, and also all individual measures against each other; (v) gave our recommendation about the use of the refined THD algorithm with Jaccard measure which demonstrated the most balanced performance in our experiments.

For the experiments we used the datasets generated, using our instrumental software suite, from the TIME document collection. This collection contains real scientific papers acquired from the proceedings series of the Time Representation and Reasoning Symposia.

Our future work is planned based on the results of the presented experiments and some additional observations we made. Firstly, we would like to explore the ways to improve the performance of the Sørensen-Dice measure implementation as its higher

computational complexity is the only flaw against the Jaccard measure implementation. Secondly, we are interested in finding out if a similar term grouping algorithm, using a sensitive similarity measure, like Sørensen-Dice, would be plausible for grouping features while building feature taxonomies. This task is on the agenda for the second (Conceptualization) phase of OntoElect [32]. Thirdly, we are keen to check if the evaluation results on the other document collections will be similar to that presented in this paper. To find this out we plan to repeat the same cross-evaluation experiments but on the datasets generated from DMKD and DAC collections [16].

Acknowledgements

The research leading to this publication has been done in part in cooperation with the Ontology Engineering Group of the Universidad Politécnica de Madrid in frame of FP7 Marie Curie IRSES SemData project (http://www.semdata-project.eu/), grant agreement No PIRSES-GA-2013-612551. While performing this research, the first author has been a master student on the program on Computer Science and Information Technologies at Zaporizhzhia National University. The second author is funded by a PhD grant provided by Zaporizhzhia National University and the Ministry of Education and Science of Ukraine.

References

1. Tatarintseva, O., Ermolayev, V., Keller, B., Matzke, W.-E.: Quantifying ontology fitness in OntoElect using saturation- and vote-based metrics. In: Ermolayev, V., et al. (eds.) Revised Selected Papers of ICTERI 2013, CCIS, vol. 412, pp. 136--162 (2013)
2. Fahmi, I., Bouma, G., van der Plas, L.: Improving statistical method using known terms for automatic term extraction. In: Computational Linguistics in the Netherlands, CLIN 17 (2007)
3. Wermter, J., Hahn, U.: Finding new terminology in very large corpora. In: Clark, P., Schreiber, G. (eds.) Proc. 3rd Int Conf on Knowledge Capture, K-CAP 2005, pp. 137--144, Banff, Alberta, Canada, ACM (2005) DOI: 10.1145/1088622.1088648
4. Zhang, Z., Iria, J., Brewster, C., Ciravegna, F.: A comparative evaluation of term recognition algorithms. In: Proc. 6th Int Conf on Language Resources and Evaluation, LREC 2008, Marrakech, Morocco (2008)
5. Daille, B.: Study and implementation of combined techniques for automatic extraction of terminology. In: Klavans, J., Resnik, P. (eds.) The Balancing Act: Combining Symbolic and Statistical Approaches to Language, pp. 49--66. The MIT Press. Cambridge, Massachusetts (1996)
6. Cohen, J. D.: Highlights: Language- and domain-independent automatic indexing terms for abstracting. J. Am. Soc. Inf. Sci. 46(3), 162--174 (1995) DOI: 10.1002/(SICI)1097-4571(199504)46:3<162::AID-ASI2>3.0.CO;2-6
7. Caraballo, S. A., Charniak, E.: Determining the specificity of nouns from text. In: Proc. 1999 Joint SIGDAT Conf on Empirical Methods in Natural Language Processing and Very Large Corpora, pp. 63--70 (1999)
8. Astrakhantsev, N.: ATR4S: toolkit with state-of-the-art automatic terms recognition methods in scala. arXiv preprint arXiv:1611.07804 (2016)

9. Medelyan, O., Witten, I. H.: Thesaurus based automatic keyphrase indexing. In: Marchionini, G., Nelson, M. L., Marshall, C. C. (eds.) Proc. ACM/IEEE Joint Conf on Digital Libraries, JCDL 2006, pp. 296--297, Chapel Hill, NC, USA, ACM (2006) DOI: 10.1145/1141753.1141819
10. Ahmad, K., Gillam, L., Tostevin, L.: University of surrey participation in trec8: Weirdness indexing for logical document extrapolation and retrieval (wilder). In: Proc. 8th Text REtrieval Conf, TREC-8 (1999)
11. Sclano, F., Velardi, P.: TermExtractor: A Web application to learn the common terminology of interest groups and research communities. In: Proc. 9th Conf on Terminology and Artificial Intelligence, TIA 2007, Sophia Antipolis, France (2007)
12. Frantzi, K. T., Ananiadou, S.: The c/nc value domain independent method for multi-word term extraction. J. Nat. Lang. Proc. 6(3), 145--180 (1999) DOI: 10.5715/jnlp.6.3_145
13. Kozakov, L., Park, Y., Fin, T., Drissi, Y., Doganata, Y., Cofino, T.: Glossary extraction and utilization in the information search and delivery system for IBM Technical Support. IBM System Journal 43(3), 546--563 (2004) DOI: 10.1147/sj.433.0546
14. Astrakhantsev, N.: Methods and software for terminology extraction from domain-specific text collection. PhD thesis, Institute for System Programming of Russian Academy of Sciences (2015)
15. Bordea, G., Buitelaar, P., Polajnar, T.: Domain-independent term extraction through domain modelling. In: Proc. 10th Int Conf on Terminology and Artificial Intelligence, TIA 2013, Paris, France (2013)
16. Kosa, V., Chaves-Fraga, D., Naumenko, D., Yuschenko, E., Badenes-Olmedo, C., Ermolayev, V., Birukou, A.: Cross-evaluation of automated term extraction tools by measuring terminological saturation. In: Bassiliades, N., et al. (eds.) ICTERI 2017. Revised Selected Papers. CCIS, vol. 826, pp. 135--163 (2018)
17. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: Proc. 2003 Int. Conf. on Information Integration on the Web, pp 73--78, AAAI Press (2003)
18. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10 (8), 707--710 (1966)
19. Monge, A., Elkan, C.: The field-matching problem: algorithm and applications. In: Proc. 2nd Int Conf on Knowledge Discovery and Data Mining, pp. 267--270, AAAI Press (1996)
20. Jaro, M. A.: Probabilistic linkage of large public health data files (disc: P687-689). Statistics in Medicine 14, 491--498 (1995)
21. Winkler, W. E.: String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In: Proc. Section on Survey Research Methods. ASA, pp. 354--359 (1990)
22. Jaccard, P.: The distribution of the flora in the alpine zone. New Phytologist 11, 37--50 (1912) DOI:10.1111/j.1469-8137.1912.tb05611.x
23. Lu, J., Lin, C., Wang, W., Li, C., Wang, H.: String similarity measures and joins with synonyms. In: Proc. 2013 ACM SIGMOD Int Conf on the Management of Data, pp. 373--384 (2013)
24. Lee, H., Ng, R. T., Shim, K.: Power-law based estimation of set similarity join size. Proc. of the VLDB Endowment 2(1), 658--669 (2009)
25. Tsuruoka, Y., McNaught, J., Tsujii, J., Ananiadou, S.: Learning string similarity measures for gene/protein name dictionary look-up using logistic regression. Bioinformatics 23(20), 2768--2774 (2007)

26. Qin, J., Wang, W., Lu, Y., Xiao, C., Lin, X.: Efficient exact edit similarity query processing with the asymmetric signature scheme. In: Proc. of the 2011 ACM SIGMOD Int Conf on Management of data, pp. 1033--1044. ACM New York, USA (2011)
27. Hamming, R. W.: Error detecting and error correcting codes. Bell System Technical Journal 29 (2), 147--160 (1950), DOI:10.1002/j.1538-7305.1950.tb00463.x.
28. Dice, Lee R.: Measures of the amount of ecologic association between species. Ecology 26 (3), 297--302 (1945), DOI:10.2307/1932409
29. Badenes-Olmedo, C., Redondo-García, J. L., Corcho, O.: Efficient clustering from distributions over topics. In: Proc. K-CAP 2017, ACM, New York, NY, USA, Article 17, 8 p. (2017), DOI: 10.1145/3148011.3148019
30. Corcho, O., Gonzalez, R., Badenes, C., Dong, F.: Repository of indexed ROs. Deliverable No. 5.4. Dr Inventor project (2015)
31. Ermolayev, V.: OntoElecting requirements for domain ontologies. The case of time domain. EMISA Int J of Conceptual Modeling 13(Sp. Issue), 86--109 (2018) DOI: 10.18417/emisa.si.hcm.9
32. Moiseenko, S., Ermolayev, V.: Conceptualizing and formalizing requirements for ontology engineering. In: Antoniou, G., Zholtkevych, G. (eds.) Proc. ICTERI 2018 PhD Symposium, Kyiv, Ukraine, May 14-17, CEUR-WS (2018) online – to appear