A fast and efficient semantic short text similarity metric

Conference Paper · September 2013

DOI: 10.1109/UKCI.2013.6651309

David Croft
Coventry University
9 PUBLICATIONS **19** CITATIONS

SEE PROFILE

Jethro Shell
De Montfort University
14 PUBLICATIONS **86** CITATIONS

SEE PROFILE

Stephen Christopher Brown
De Montfort University
76 PUBLICATIONS **365** CITATIONS

SEE PROFILE

A Fast and Efficient Semantic Short Text Similarity Metric

David Croft*, Simon Coupland†, Jethro Shell ‡, and Stephen Brown§
* Knowledge Media Design, De Montfort University, Leicester LE1 9BH, United Kingdom
Email: david.croft@email.dmu.ac.uk
† Centre for Computational Intelligence, De Montfort University, Leicester LE1 9BH, United Kingdom
Email: simonc@dmu.ac.uk
‡ Knowledge Media Design, De Montfort University, Leicester LE1 9BH, United Kingdom
Email: jethros@dmu.ac.uk
§ Knowledge Media Design, De Montfort University, Leicester LE1 9BH, United Kingdom
Email: sbrown@dmu.ac.uk

***Abstract*—**

The semantic comparison of short sections of text is an emerging aspect of Natural Language Processing (NLP). In this paper we present a novel Short Text Semantic Similarity (STSS) method, Lightweight Semantic Similarity (LSS), to address the issues that arise with sparse text representation. The proposed approach captures the semantic information contained when comparing text to process the similarity. The methodology combines semantic term similarities with a vector similarity method used within statistical analysis. A modification of the term vectors using synset similarity values addresses issues that are encountered with sparse text. LSS is shown to be comparable to current semantic similarity approaches, LSA and STASIS, whilst having a lower computational footprint.

I. Introduction

De Montfort University hosts a research database containing records of the Royal Photographic Society (RPS). This web accessible data contains the digitised contents of the exhibition catalogues produced by the RPS. The Exhibitions of the Royal Photographic Society (ERPS) catalogues are a contemporary account of photography during the period 1870 to 1915. They hold 34,197 records but only 1,040 associated images. Whilst being of significant interest to the photo-historical community, the catalogue can be enhanced by identifying possible missing images. A wider goal of the authors work is to populate the images by comparing meta-data from external digitised catalogue sources from associated Galleries, Libraries, Archives and Museums (GLAMs) with data within ERPS. A required element of this process, is the use of Natural Language Processing (NLP), more notably semantic similarity to help match meta-data across collections [1].

There is a large body of inter-disciplinary work looking at how human language can be processed by machines in such a way that word meaning is captured in a data structure or automated process. This is generally referred to as NLP. This is a complex and dynamic goal, considered to be a discipline within Artificial Intelligence (AI) as it strives to achieve human-like performance [2].

Although the overall goal of NLP is still elusive, there have been a number of steps made towards the understanding of language. The production of parsing software [3], Part-of-speech (POS) taggers [4], [5] and Decision Support Systems (DCS) [6] have all provided inroads into the problem. However, one of the most difficult aspects of NLP is understanding semantic similarity. Humans have little difficulty in understanding the intended meaning of different words, or associating the similarity. For example, it is easy to define a level of similarity between the words *eagle* and *crane*. This maybe high if both are viewed as birds. Changing the context of *crane* to a type of machine and the similarity reduces. This is a difficult task to replicate using computation. Areas of work within similar fields, such as document classification, face similar issues when identifying similarities in natural language texts. The predominant techniques, however, require significantly greater text than is on offer within the data available to this study.

Photographic description meta-data contains sparse text. The descriptions are typically brief in length, and often grammatically incorrect, sharing many attributes with the definition of *short text* proposed by [7]. The difficulty of semantic similarity is increased when there is a reduced quantity of text. Many approaches to Short Text Semantic Similarity (STSS) [7] have been based upon existing adaptations of long-text similarity methods [8]. These methods are less applicable to our problem domain. The impact of the sentence structure and word occurrence alters with the length of text. To address these issues, we propose a novel short text Lightweight Semantic Similarity (LSS) metric. This method is compared to current approaches, Latent Semantic Analysis (LSA) and Sentence Similarity (STASIS), using a gold standard corpus.

The following sections of this paper are set out as follows. In Section II-A and II-B, the comparative methods are introduced. In Section III, an outline of LSS is given. The following section outlines the performance comparison of each method to LSS. The final section concludes the paper, discussing the findings.

II. Short Text Similarity Metrics

There are a number of approaches to measuring short text similarity. In this section we discuss two of the most popular measures, Deerwester *et al's* Latent Semantic Analysis (LSA)

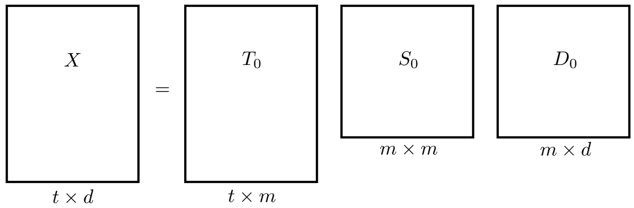

Fig. 1. Initial Matrices used in LSA.

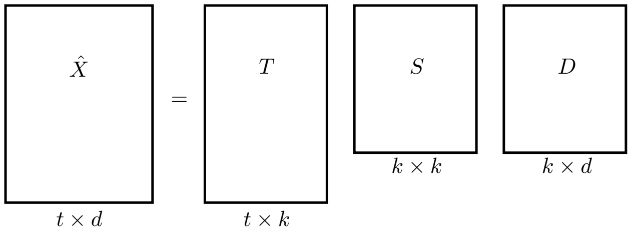

Fig. 2. Optimised Matrices used in LSA.

[9] and Li *et al's* Sentence Similarity approach (STATIS) [10]. These approaches will be compared to LSS in Section IV.

A. LSA

Deerwester *et al's* Latent Semantic Analysis (LSA) [9] is a widely used technique for comparing the similarity of short pieces of text, despite the fact that it was actually proposed for large scale data retrieval applications. LSA relates to the TF-IDF (Term Frequency - Inverse Document Frequency) approach but makes use of the singular value decomposition of TF matrices to calculate the similarity. Given d documents made up of t terms, the SVD matrices used in LSA are $X = T_0S_0D_0$ as depicted in Figure 1, where m is a value $\leq min(t,d)$. Redundant columns may then be removed giving a new matrix $\hat{X} = TSD \approx X$ as depicted in Figure 2, where k is number which is empirically chosen. Each row in $\hat{X}$ represents the occurrence of terms across the different pieces of text. The similarity of any two pieces of text is given by taking the dot product of two row vectors of $\hat{X}$. These can be held in a further matrix $\hat{X}\hat{X}' = TS^2T'$ where $\hat{X}\hat{X}_{i,j}$ is obtained from the cross product of row vectors $\hat{X}_i$ and $\hat{X}_j$. It is these similarity values which we are comparing the LSS method against in Section IV.

B. STASIS

1) Word semantic similarity: Similarity between individual words in STASIS is calculated as a property of relative word positions in a hierarchical knowledge base, WordNet was used in [10] but any could be used.

Terms in WordNet are represented by a set of synsets, each of which represents a differing meaning for that term. STASIS measures similarity between individual synsets using a combination of short path distance between the synsets across the WordNet's hierarchical structure and the depth of those synsets in the structure. Similarity between term pairs is calculated using equation 1.

$$s(w_1,w_2) = e^{\alpha l} \cdot f_2(h) \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \tag{1}$$

l represents the path length between two terms and h represents the depth of the subsumer (ancestor node) in the WordNet hierarchy. α and β are tuning values, both of which should have values $\in [0,1]$. [11] used values of $a = 0.2$ and $\beta = 0.45$.

2) Sentence semantic similarity: Overall semantic similarity is calculated as the cosine of two modified term vectors. The modifications to the original term vectors made by STASIS attempt to alter identify semantic similarities between terms and to modify each term's importance in order to reduce the emphasis placed on common terms.

Semantic similarity between terms are identified as follows. For each term in the common term vector T , if the term appears in the vector (T_1, T_2) then set the value in the semantic vector to be 1. If the term does not appear in the vector (T_1, T_2), then find the term in the vector with the highest term similarity (see section II-B1), if the similarity exceeds a threshold then set the value in the semantic vector to be the term similarity. If the highest similarity does not exceed the threshold then set the value to be 0.

Term importance is identified using the information content of the terms as provided the Brown corpus [12]. The information content and the value from the previous step are combined to produce a final value for the semantic vector using the Equation 2.

$$s_i = \tilde{s} \cdot I(w_i) \cdot I(\tilde{w}_i) \tag{2}$$

The overall sentence semantic similarity is then calculated as the cosine of the two semantic vectors.

3) Word order: In contrast to LSA (amongst others), STASIS takes word order into account. This is a major distinguishing feature of STASIS when compared to other approaches which treat text as a bag of words. For example the vectors [a b c] and [c b a] are equivalent under a bag of words approach as ordering differences are ignored. STASIS however includes a computational method for measuring word order similarity between texts and so will not consider the two equivalent.

Word order similarity under STASIS is assessed as follows. The first step is to convert T_1 and T_2 into word order vectors (r_1 and r_2), This is achieved by finding the position of each term in T with that terms position in T_1 and T_2. When a term does not appear in a term vector, the position of the term with the highest similarity to the missing term is used assuming it exceeds a pre-set threshold. Otherwise 0 is used to denote position.

$$\begin{aligned} T = T_1 \cup T_2 =& [\text{a b c}] & \\ T_1 =& [\text{a b c}] & \rightarrow r_1 = [1,2,3] \\ T_2 =& [\text{c b}] & \rightarrow r_2 = [0,2,1] \end{aligned}$$

A word order similarity value can then by generated by simply calculating the normalised difference of the word order vectors (see equation 3).

$$S_r = 1 - \frac{||r_1 - r_2||}{||r_1 + r_2||} \tag{3}$$

4) Overall similarity: The overall STASIS similarity for the pair of vectors being compared given by equation 4. Where $\delta \leq 1$ and controls the relative effect that the semantic similarity and word order values have on the overall text similarity value. [10] state that δ should be kept at a value > 0.5 as word order plays a lesser role in text processing[13], [10]

$$S(T_1, T_2) = \delta S_s + (1 - \delta) S_r \tag{4}$$

III. Lightweight Semantic Similarity Metric

In this Section we present a novel Lightweight Semantic Similarity (LSS) method which performs well when compared to existing approaches. This approach addresses issues when measuring textual similarity in small text sets.

The title field of a photograph is typically very short. It may also be an emotional or artistic description of the contents. The average number of title words within the description is small. The number of *useful* words is less, a mean of 5.4 words[1]. Therefore, given the briefness of the text per record, standard approaches for measuring textual similarity (such as Term Frequency (TF)) will be either unusable or will function poorly.

A secondary approach, the use of semantic meaning, additionally is problematic. The lack of sentence structure within the titles reduces the usability of the technique. The proposed methodology combines the two established approaches into a pseudo-semantic similarity with elements of the statistical techniques. The methodology combines semantic term similarities, the semantic similarity between individual terms, with a vector similarity method used within statistical analysis.

A. Text Pre-processing

The initial stage in our approach is to generate a term vector for each title field. This involves a three step process:

1) **Cleaning and tokenising each title:** The words in the title are separated and extraneous non-alphanumerical characters are removed.
2) **Removal of terms that have a high regularity:** Common terms, for example *and*, *a* and *on* commonly referred to as *stop words* within the title are extracted using the NLTK package [14]. This reduces the occurrence of high commonality words producing a high similarity measure. Many words appear frequently in searches, however, high frequency words such as *photograph*, which appears in 4% of the records collected [2], are specific to searches within the field of photographs. These are also removed.
3) **Identification of each word synset:** The synsets relating to each word are identified through the use of WordNet. WordNet is a lexical database of English words grouped into a structure of syntatic categories based on context [15]. Each word produces 0 to n synsets where n is the number of possible synsets within WordNet. Where zero synsets are identified, the raw form of the word is compared using a character based string matching. Words identified with zero synsets are maintained in the set as they can include relevant information, such as person and place names alongside technical terminology. This stage also has the effect of normalising multiple forms of the same word, for example plural, past, present and future tenses, into a single representation. This simplifies the comparisons at only a small cost to the degree of precision.

B. Similarity Metric

The pre-processing stage forms a series of term vectors that inform the similarity metric process. The term vectors represent the terms that appear in each piece of text and the number of times that each term appears in that text. Also calculated is a pair wise similarity matrix for all of the terms appearing across all pieces of text being compared. Term similarity is calculated as the maximum path similarity value (based on the shortest connecting path) between the synsets of the compared terms. This is determined by a pair wise comparison of all of the synsets corresponding to one term, with all of the synsets corresponding to the other.

The similarity metric uses a cosine similarity of the two term vectors to extract a similarity measure for the title fields being compared. Term vectors are a way to represent text and queries on the text, as vectors of identifiers. Each dimension in the vector represents a separate term. The corresponding value in the vector is non-zero, if the term appears within the text. The cosine similarities of term vectors is a common approach for identifying document similarity. Predominantly, the application of this method uses vectors contain high volume elements, hundreds or thousands. The briefness of the title fields within photographs means that is it unlikely that there will be any shared terms between pairs of titles even when they are semantically similar. Therefore the cosine similarity of the vectors will be zero.

To overcome this issue, we propose the use of a novel approach where the initial vectors are modified using the term similarity values taken from WordNet which are calculated using the method described previously. By calculating the cosine similarity on the modified, weighted term vectors, it is possible to compare according to a pseudo-semantic similarity of the terms mitigating issues caused by the shortage of text.

Cosine similarity measures the similarity of two n dimensional vectors through the use of the cosine of the angle between them. Using two elements, A and B, the similarity

[1] Combining the title and description fields for the 34,197 ERPS records gives a mean average of 8.1 words. Filtering out low values terms (for example *in* and *and*) produces a mean of 5.4 words per record.

[2] 65,491 of 1,783,280 records.

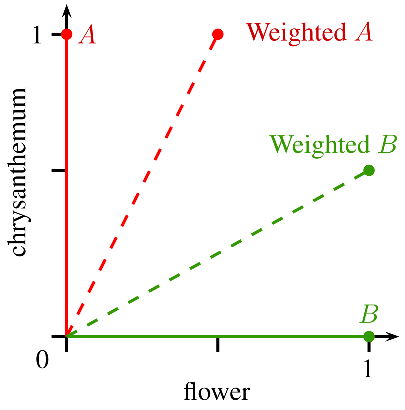

Fig. 3. Cosine of Weighted Vectors.

θ can be represented as

$$Similarity(A,B) = cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} \tag{5}$$

The results of cosine similarity produce a range of values between 0 and 1 where 0 indicates independence between the vectors and values > 0 show levels of similarity.

The limited nature of the title length can output cosine similarity values of 0, despite obvious semantic similarity. By weighting the original vectors, semantic information is incorporated. To weight the cosine similarity metric, a maximum path value is produced. The value is based on the shortest path distance needed to traverse between the two values within the WordNet tree structure. The impact of this is shown in a simple example in Fig 3.

The figure highlights two pieces of text, A = chrysanthemum, B = flower and a similarity where $\mathbf{sim}(\text{chrysanthemum}, \text{flower}) = 0.5$. Based on the cosine similarity, the original vectors show independence as they are perpendicular to one another. By adding the weighting extracted from the WordNet distance measure, the same pieces of text produce a similarity of 0.8.

In the following section, a worked example of the proposed method will be shown.

C. Worked Example

In order to properly describe this metric a worked example of a single title pair is included. In this example the two title fields are defined as A and B, with the contents *the chrysanthemum lady* and *a woman selling flowers* respectively. Whilst the semantic similarity of A and B is obvious, there are no terms shared between the two. Therefore approaches such as TF-IDF would be ineffective. Following preprocessing of the raw fields, the original title strings produce the vectors A = [chrysanthemum, lady] and B = [flower, selling, woman]. The results of using the maximum synset similarity to generate the term similarity matrix are shown in Table I.

As the table shows, *chrysanthemum* and *flower* have a high similarity (0.50), the same applies to *lady* and *woman* (0.50), however unrelated terms such as *chrysanthemum* and *lady* have much lower values (0.09). The outcome of combining these weights with the values in the term vectors is shown in Table II.

TABLE I. EXAMPLE TERM SIMILARITY MATRIX

	chrysanthemum	flower	lady	selling	woman
chrysanthemum	1.00	0.50	0.09	0.06	0.10
flower		1.00	0.10	0.09	0.11
lady			1.00	0.07	0.50
selling				1.00	0.09
woman					1.00

TABLE II. EXAMPLE OF ORIGINAL AND CORRESPONDING WEIGHTED TERM VECTORS.

		chrysanthemum	flower	lady	selling	woman
Term	*A*	1	0	1	0	0
vectors	*B*	0	1	0	1	1
Sim matrix	chry...	1.00	0.50	0.09	0.06	0.10
values for *A*	lady	0.09	0.10	1.00	0.07	0.50
Sim matrix	flower	0.50	1.00	0.10	0.09	0.11
values for *B*	selling	0.06	0.09	0.07	1.00	0.09
	woman	0.10	0.11	0.50	0.09	1.00
Weighted	*A*	1.09	0.60	1.09	0.13	0.60
vectors	*B*	0.66	1.20	0.67	1.18	1.20

With the weighted term vectors calculated, it is possible to calculate the cosine similarity. Using the original term vectors a result of 0.00 would be achieved. However, if the similarity of the weighted vectors is calculated then a result of 0.76 is gained. We believe that this is in keeping with the semantic understanding that a human would place on the two title structures.

IV. PERFORMANCE COMPARISON

To investigate the performance of the LSS metric we ran an experiment looking at computation time and similarity compared to LSA and STASIS using a ground truth data set accepted in the literature. The performance of LSA and STASIS have already been compared by [7]. In O'Shea *et al* the results of the two approaches are compared to the averaged similarity scores from human testers using a subset of the STSS-65 dataset. In order to compare the quality of the results from the LSS metric against existing approaches, the metric was run against the same STSS-65 subset used by [7] (see table III). Figure 4 shows the LSA, STASIS and human produced similarity values plotted with the results from the LSS metric.

V. LSS METRIC TESTING

The testing data (STSS-65) consists of word pairs (see table III), whilst it more closely resembles the contents of the title fields from the ERPS collections than other data sets, it is not a perfect emulation. As such the results produced are only approximate representations of the relative performances of the tested techniques on the data we are most concerned with.

Throughput testing was conducted using Python implementations of both approaches running on an Intel Core2 Duo T5500 (1.66GHz). Alternative programming languages and/or hardware could produce faster implementations but as testing was intended to demonstrate the comparative performance the absolute performance was unimportant.

Five sets of results were produced, the first is the time taken for LSA to produce non-directional pair-wise similarity values for increasingly large record sets. The second is the time taken for the LSS to do the same using pre-calculated word similarity values. The third is the time taken for the LSS metric if the word similarity values are not cached. Since each word pair needs only be compared once and can then be stored in perpetuity, starting with no cached word similarity values is unlikely, these results are therefore included only for completeness. Forth is the time taken by STASIS using cached word similarity values. Fifth is the STASIS time without cached values, it should be noted that the LSS metric and STASIS have different methods for calculating word similarity values, the appropriate approach was used in both cases.

A. Similarity

Adopting the approach used by [7], the LSS metric values were compared to those of the human responses using Pearson's correlation coefficient. The results of the LSS metric produced a correlation value of 0.807 compared to 0.838 for LSA and 0.816 for STASIS. This means that the LSS metric represents a performance decrease of 3.1% compared to the best performing metric LSA and 0.9% decrease compared to STASIS.

B. Computational Performance

We now consider the computation time for the metrics, again with and without cache of values where possible. Figure 5 shows the time taken for the three approaches. As can be clearly seen, the LSS metric is significantly faster than LSA when using cached results. The performance without cached values is initially worse than that of LSA but quickly improves as the number of records to compare increases. This is because the number of word similarity values which need to be calculated is directly related to the number of unique words in the records being compared. However the number of unique words per record decreases as the number of records being compared increases. Therefore computation time for LSS and STASIS compared to LSA continuously improve as more records are compared. When compared to STASIS (using pre-cached term similarity values), LSS reduces computation time by an average of 9.8%.

VI. Conclusion

In this paper we have defined the LSS short text similarity metric. This metric works by looking at the distance between synsets in WordNet and to form a term vector and then calculates the cosine similarity of this term vector. This is a simple, lightweight approach which is ideal for the problem of comparing the titles of museum artifacts, our particular problem domain.

We compared the LSS metric to two established metrics, LSA and STASIS and we found that LSS gave the best computational performance, slightly above STASIS and vastly faster than LSA and gave similarity results very close to STASIS but not as good as LSA. We believe this metric is useful as it is computationally lightweight and works well on sentence fragments of the kind found in artifact titles.

Acknowledgment

The authors would like to acknowledge the funding provided by the Arts and Humanities Research Council (AHRC) for the Fuzzy Photo Project (AH/J004367/1).

References

[1] S. C. David Croft and S. Brown, "A hybrid approach to co-reference identification within museum collections," in *CIES 2013, IEEE Symposium on Computational Intelligence for Engineering Solutions*, 2013.

[2] G. G. Chowdhury, "Natural language processing," *Annual review of information science and technology*, vol. 37, no. 1, pp. 51–89, 2003.

[3] M.-C. De Marneffe, B. MacCartney, and C. D. Manning, "Generating typed dependency parses from phrase structure parses," in *Proceedings of LREC*, vol. 6, 2006, pp. 449–454.

[4] H. Schmid, "Probabilistic part-of-speech tagging using decision trees," in *Proceedings of international conference on new methods in language processing*, vol. 12. Manchester, UK, 1994, pp. 44–49.

[5] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith, "Part-of-speech tagging for twitter: annotation, features, and experiments," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, 2011, pp. 42–47.

[6] D. Demner-Fushman, W. W. Chapman, and C. J. McDonald, "Methodological review: What can natural language processing do for clinical decision support?" *Journal of biomedical informatics*, vol. 42, no. 5, pp. 760–772, 2009.

[7] J. O'Shea, Z. Bandar, K. Crockett, and D. McLean, "A comparative study of two short text semantic similarity measures," in *Proceedings of the 2nd KES International conference on Agent and multi-agent systems: technologies and applications*. Springer-Verlag, 2008, pp. 172–181.

[8] J. Oliva, J. I. Serrano, M. D. del Castillo, and Á. Iglesias, "Symss: A syntax-based measure for short-text semantic similarity," *Data & Knowledge Engineering*, vol. 70, no. 4, pp. 390–405, 2011.

[9] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Hasrhman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391 – 407, 1990.

[10] L. Yuhua, D. Mclean, Z. Bandar, J. O'Shea, and K. Crockett, "Sentence similarity based on semantic nets and corpus statistics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 1138 – 1150, 2006.

[11] Y. Li, Z. Bandar, and D. Mclean, "An approach for measuring semantic similarity between words using multiple information sources," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 15, no. 4, pp. 871–882, 2003.

[12] W. N. Francis and H. Kucera, "Brown corpus manual," Department of Linguistics, Brown University, Providence, Rhode Island, US, Tech. Rep., 1979. [Online]. Available: http://icame.uib.no/brown/bcm.html

[13] P. Wiemer-Hastings, "Adding syntactic information to lsa," in *PROCEEDINGS OF THE 22ND ANNUAL CONFERENCE OF THE COGNITIVE SCIENCE SOCIETY*. Morgan Kaufmann, 2000, pp. 989–993.

[14] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python*. O'Reilly Media, 2009.

[15] G. A. Miller, "Wordnet: a lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995.

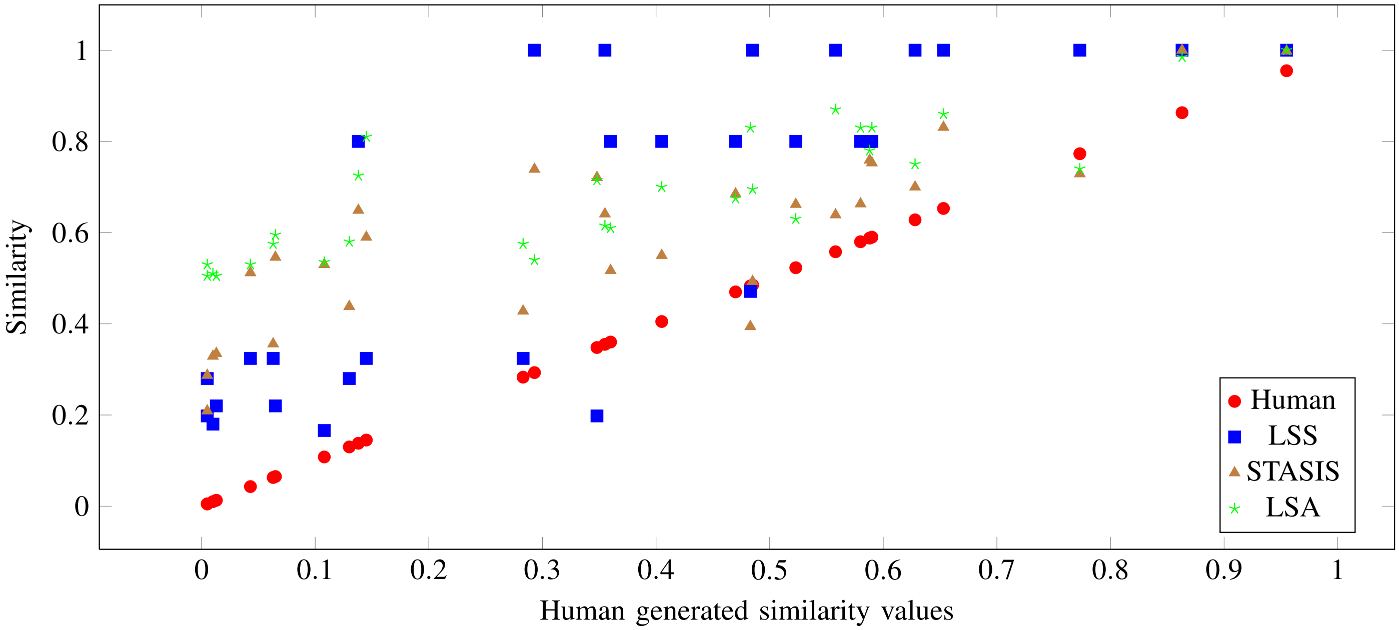

Fig. 4. Human, LSA, STASIS and LSS metric generated similarity values for STSS-65 subset.

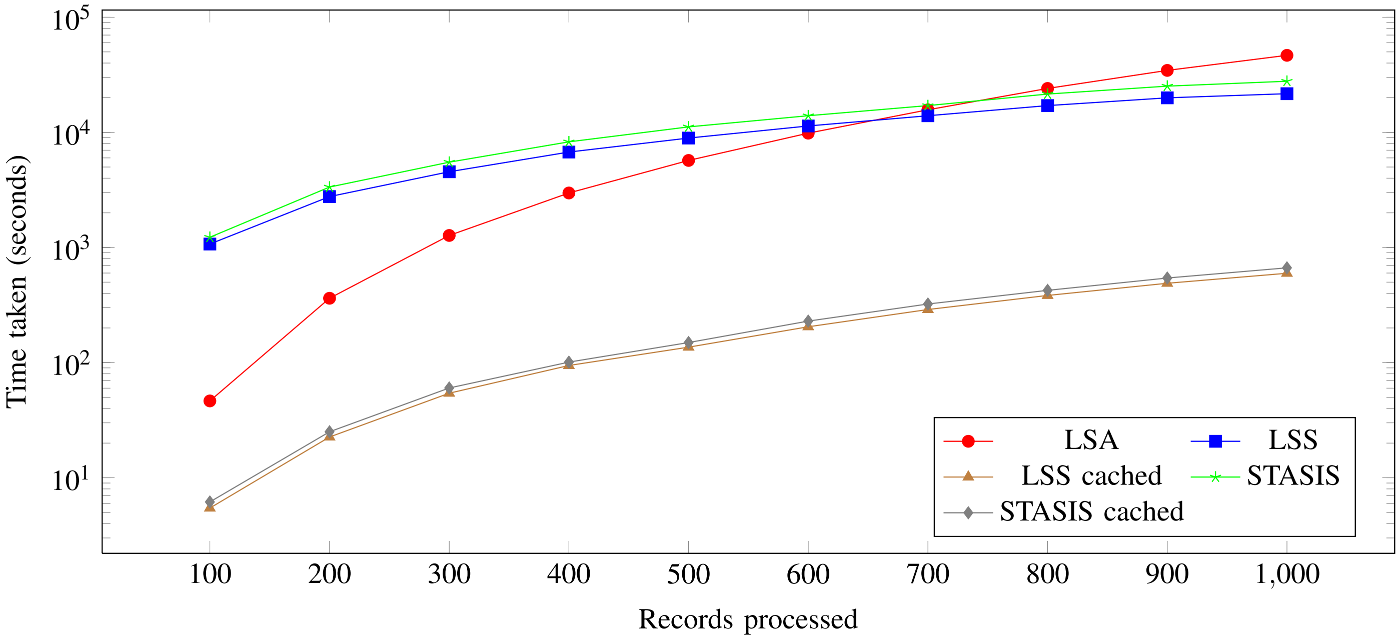

Fig. 5. Comparison of processing time requirements for LSA, LSS and STASIS.

TABLE III. Raw results for LSS metric testing using STSS-65.

Sentence pair			Semantic similarity measure			
Id	A	B	Human	LSS	STASIS	LSA
1	cord	smile	0.010	0.180	0.329	0.510
5	autograph	shore	0.005	0.198	0.287	0.530
9	asylum	fruit	0.005	0.280	0.209	0.505
13	boy	rooster	0.108	0.166	0.530	0.535
17	coast	forest	0.063	0.324	0.356	0.575
21	boy	sage	0.043	0.324	0.512	0.530
25	forest	graveyard	0.065	0.220	0.546	0.595
29	bird	woodland	0.013	0.220	0.335	0.505
33	hill	woodland	0.145	0.324	0.590	0.810
37	magician	oracle	0.130	0.280	0.438	0.580
41	oracle	sage	0.283	0.324	0.428	0.575
47	furnace	stove	0.348	0.198	0.721	0.715
48	magician	wizard	0.355	1.000	0.641	0.615
49	hill	mound	0.293	1.000	0.739	0.540
50	cord	string	0.470	0.800	0.685	0.675
51	glass	tumbler	0.138	0.800	0.649	0.725
52	grin	smile	0.485	1.000	0.493	0.695
53	serf	slave	0.483	0.471	0.394	0.830
54	journey	voyage	0.360	0.800	0.517	0.610
55	autograph	signature	0.405	0.800	0.550	0.700
56	coast	shore	0.588	0.800	0.759	0.780
57	forest	woodland	0.628	1.000	0.700	0.750
58	implement	tool	0.590	0.800	0.753	0.830
59	cock	rooster	0.863	1.000	1.000	0.985
60	boy	lad	0.580	0.800	0.663	0.830
61	cushion	pillow	0.523	0.800	0.662	0.630
62	cemetery	graveyard	0.773	1.000	0.729	0.740
63	automobile	car	0.558	1.000	0.639	0.870
64	midday	noon	0.955	1.000	0.998	1.000
65	gem	jewel	0.653	1.000	0.831	0.860