A Morphological Approach for Measuring Pair-Wise Semantic Similarity of Sanskrit Sentences

Vaishakh Keshava(✉), Mounica Sanapala, Akshay Chowdlu Dinesh, and Sowmya Kamath Shevgoor

Department of Information Technology,
National Institute of Technology Karnataka, Surathkal, India
kvaishakhnambiar@gmail.com, smounica9@gmail.com, akshaychowdlu10@gmail.com, sowmyakamath@nitk.edu.in

Abstract. Capturing explicit and implicit similarity between texts in natural language is a critical task in Computational Linguistics applications. Similarity can be multi-level (word, sentence, paragraph or document level), each of which can affect the similarity computation differently. Most existing techniques are ill-suited for classical languages like Sanskrit as it is significantly richer in morphology than English. In this paper, we present a morphological analysis based approach for computing semantic similarity between short Sanskrit texts. Our technique considers the constituent words' semantic properties and their role in individual sentences within the text, to compute similarity. As all words do not contribute equally to the semantics of a sentence, an adaptive scoring algorithm is used for ranking, which performed very well for Sanskrit sentence pairs of varied complexities.

Keywords: Morphological analysis · Semantic similarity · NLP

1 Introduction

Semantic measures are central elements of a large variety of Natural Language Processing applications and knowledge-based systems, and have therefore been subject to intensive and interdisciplinary research efforts during past decades. The process of semantic similarity measurement evaluates the depth of the semantic relationship between various constituent entities of the natural language under consideration, which is then examined at multiple levels to finally express the level of similarity numerically, for purposes such as ranking and scoring.

Sentence-level similarity measures focus on analyzing a word and its association with other words. The semantics of a sentence is determined by the agents, experiencers, instruments, location, verb etc., used in expressing the writers intent, which represent different thematic relations. The *tense* also contributes to the semantics, while *prefixes* tend to alter the meaning of a word, and

F. Frasincar et al. (Eds.): NLDB 2017, LNCS 10260, pp. 162–169, 2017.
DOI: 10.1007/978-3-319-59569-6_18

so should also be taken into account. Hence, each word has its own contribution to the semantics of the sentence, and each contribution is different.

Current methods for capturing semantic similarity are more geared towards the English language and focus towards Sanskrit is comparatively quite low. Existing semantic similarity computation methods can be categorized into two groups - those based on measuring path distance between concepts (dictionary/thesaurus based) and, those which use the information content of a concept (corpus-based). Hybrid methods have also been tried in certain contexts. Some approaches to determine similarity use a variety of computational models that implement distributional semantics based techniques like Latent Semantic Analysis [2] and semantic nets [7], syntax or dependency-based models [9,14], random indexing and grammar patterns [10].

Recent work in the field of Natural Language Processing for Indian Languages has attempted to deal with the problem of parsing. Huet et al. [7] applied computational linguistics techniques to Sanskrit Language. Goyal et al. [3] applied an analysis using semantic relations and parsing in Sanskrit text for understanding meaning. Grammar based approach towards the development of language tools for Sanskrit [1,4,8] are followed due to its well defined grammar. Hellwig et al. [6] used statistical approaches for building these tools with a small manually tagged corpus as a boot-strap. Kulkarni et al. [12] followed a combination of the grammar based approach with statistical evidences to push the most likely solution to the top. Sanskrit is a classical language with an ancient history. Sanskrit grammar (called *Vyakarana*) is culminated in Panini's monumental work 'Ashtadhyayi' [11]. To represent Sanskrit text in a machine processable form, a transliteration scheme called the WX notation [5] was used. Figure 1 illustrates the notations used in the WX scheme for representing vowels, phonetic sounds and consonants for the Devanagari script (used to write Sanskrit). The Sanskrit WordNet [13] is a linked lexical knowledge that captures the synonyms, provides *gloss* (an example of usage) which can be used for word sense disambiguation, and also indicates the *category* (noun, verb, adverb or adjective context) of the word. As most Dravidian languages are rooted in Sanskrit, any techniques developed for Sanskrit can easily be extended to other Indian languages. In this paper, a system that computes the semantic similarity of two Sanskrit texts by considering the core properties and surface structure of the Sanskrit language is presented. The rest of the paper is organized as follows. Section 2 presents the proposed methodology devised for semantic similarity computation. Section 3 presents the experimental results, followed by conclusion and references.

अ	आ	इ	ई	उ	ऊ	ऋ	ॠ	ऌ	ए	ऐ	ओ	औ	अं	अः	क्	ख्	ग्	घ्	ङ्	च्	छ्	ज्	झ्
a	A	i	I	u	U	q	Q	L	e	E	o	O	M	H	k	K	g	G	f	c	C	j	J
ञ्	ट्	ठ्	ड्	ढ्	ण्	त्	थ्	द्	ध्	न्	प्	फ्	ब्	भ्	म्	य्	र्	ल्	व्	श्	ष्	स्	ह्
F	t	T	d	D	N	w	W	x	X	n	p	P	b	B	m	y	r	l	v	S	R	s	h

Fig. 1. The scheme used in WX notation

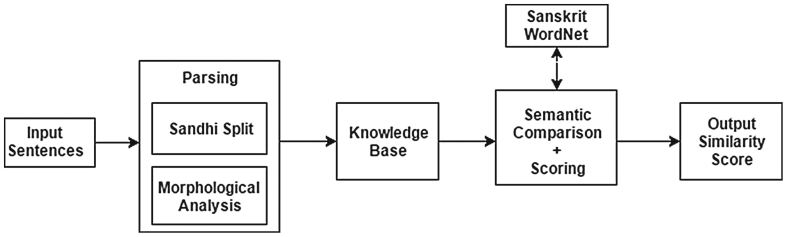

Fig. 2. Proposed Methodology

2 Proposed Methodology

The overall methodology designed for processing two Sanskrit sentences semantically for computing their similarity is depicted in Fig. 2. The system takes two Sanskrit sentences as input in WX notation. The input sentences are parsed to get the morphological properties. Parsing also involves performing Sandhi splitting and also helps in resolving real time ambiguities in the sentences. During morphological analysis phase, the morphology, i.e. the identification, analysis and description of the structure of a given languages (Sanskrit) morphemes and other linguistic units, such as root words, affixes, parts of speech, intonations and stresses, or implied context is determined. In the fusional language form of Sanskrit, compound words are likely to occur. During the sandhi splitting process, a compound word is split into its individual morphemes, which enables a clear analysis of morphemes in the sentences. The properties of each token obtained from the morphological analysis are used to create a knowledge base. This knowledge base stores the details about the agent, object, instrument, location, possession etc. of each sentence. Next, the semantics of both the sentences are compared using the extracted properties stored in the knowledge base. The Sanskrit WordNet is used at this stage to take care of synonyms. The karaka relationships in the sentences help in identifying the doer, action, place of action etc., which gives semantic information of the sentence. Separate scoring algorithms for nouns, verbs are other words are also incorporated, after which the percentage similarity between the two sentences is computed and displayed.

2.1 Implementation

Parsing: The Sanskrit tools available at *The Sanskrit Heritage Site*[1] were used for performing morphological analysis and parsing. These tools can resolve the real time ambiguities as well. The full parser [8] which takes sentences which contains Sandhis, i.e. compound words, was used for evaluation.

For example, consider two words *prawyupakAraH* and *prawi sahAyam*. Without a sandhi split, *prawyupakAraH*, there will not be any match *prawi sahAyam* in WordNet. Upon sandhi split, we get *prawyupakAraH = prawi + upakAraH*.

[1] Available at http://sanskrit.inria.fr/.

Table 1. Word attributes

Property	Description
Statement	Statement number
Word	Morpheme from sentence
Form	Noun/Verb/Other word
Gender	Male/Female/Neutral
Tense	One of the 10 tenses
Case	One of the 7 cases
Person	First/Second/Third
Number	Singular/Dual/Plural
Base form	Root/First form

Table 2. Word relations

Property/case	Description
Nominative	Subject of a verb
Accusative	Direct object of a verb
Instrumental	Object used to perform action
Dative	Indirect object of a verb
Ablative	Movement from something
Genitive	Possession
Locative	Location
Verb	Action or Event
Other words	Other than nouns/verbs

sahAyam is a synonym of *upakAraH*. So now we get *prawyupakAraH = prawi + sahAyam*. Now, the system understands that both mean the same and is able to return a better similarity score. This is why the process of Sandhi split is crucial.

Knowledge Base Creation: During parsing, the morphological properties of the words are obtained, which are stored as attributes as shown in Table 1. The root forms or base forms of each word are stored as per their category. If the word is a noun, it is further categorized into 7 cases (Vibhakti). Verbs can be found under 'Verb' column. If it is neither a noun nor a verb, it is stored in 'Other Words' column (Table 2).

Similarity Scoring: *Scoring for verbs:* Let X = $\{x_1, x_2, ...x_n\}$ be the set of verb roots in sentence 1 and Y = $\{y_1, y_2, ...y_m\}$ be the set of verb roots in sentence 2. Let $|X|$ and $|Y|$ be the size of set X and Y respectively. Let the synsets of a word be stored in a set S. Let SV[m] be an array to store the score initialized to 0. The proposed scoring mechanism gives more weight to the root (which determines the action) followed by the tense. Since the *Person* and *Number* is also determined by the noun, it is given lesser weight in verb scoring part. It is observed that when multiple attributes match, the similarity is more due to the dependency between the word attributes. So, a component $floor(count/2)$ is added to the score at the end. Algorithm 1 illustrates the process of scoring verbs.

Algorithm 1. Scoring Verbs

```
for each x_i , i from 1 to n do
    S = Synonyms of x_i;
    for each y_j, j from 1 to m do
        score = count = 0;
        if (y_j in S) then
            score = score + 4;
            count + +;
            if (Tense matches) then
                score = score + 2;
                count + +;
            end
            if (Number matches) then
                score = score + 1;
                count + +;
            end
            if (Person matches) then
                score = score + 1;
                count + +;
            end
            score = score + floor(count/2);
            if (score > SV[j]) then
                SV[j] = score;
            end
        end
    end
end
V_score = sum_{i=1}^{m} (SV[i]);
V_score_max = (10 * max(|X|, |Y|));
```

Scoring Nouns of a Particular Vibhakti: Like in the case of verbs let X and Y be the set of nouns (of a particular case in its base form) of sentence 1 and sentence 2 respectively. $SN[m]$ be an array to store the score. A score of 5 is given in case the root matches and a score of 2 in case the number *matches*. Like in the case of verb a component $floor(count/2)$ is added to the score, where $N_{score} = \sum_{i=1}^{m} (SN[i])$ and $N_{score_{max}} = (8 * \max(|X|, |Y|))$.

Scoring of Other Words: Let X and Y be the set of other words in sentence 1 and sentence 2 respectively. $SO[m]$ be an array to store the score. For other words, we give a score of 5 if the exact same word (or its synonym) is present in both the sentences. Accordingly, we get $O_{score} = \sum_{i=1}^{m} (SO[i])$ and $O_{score_{max}} = (5 * \max(|X|, |Y|))$. Overall similarity score between the two sentences is given by,

$$\frac{(V_{score} + N_{score} + O_{score})}{(V_{score_{max}} + N_{score_{max}} + O_{score_{max}})} * 100 \tag{1}$$

where, V_{score}, N_{score} and O_{score} is the total verb, noun and other word score respectively. $V_{score_{max}}$, $N_{score_{max}}$ and $O_{score_{max}}$ are the maximum verb, noun and other-word scores respectively.

3 Experimental Results

To evaluate the effectiveness of the proposed approach for similarity measurement, sentence pairs of various complexity levels were used. These were considered as different testcases, categorized into classes of successively increasing

complexity. Six such levels were considered and for each testcase, sentence pairs which were *perfectly similar*, *partially similar* and *perfectly dissimilar* were used to assess the similarity score calculation performance and accuracy. A total of 180 sentence pairs (30 of each level) were used for the evaluation. Table 3 shows the accuracy of the proposed system for all class of sentences.

Table 3. Experimental results for all the levels of sentences

Level	Class	Sentence type	Correct	Incorrect	Avg. accuracy
Level 1	Noun+verb	Perfectly similar	10	0	96.66%
		Partially similar	9	1	
		Dissimilar	10	0	
Level 2	Actor+object+verb	Perfectly similar	10	0	93.33%
		Partially similar	9	1	
		Dissimilar	9	1	
Level 3	Prefixes	Perfectly similar	9	1	90%
		Partially similar	9	1	
		Dissimilar	9	1	
Level 4	Active and passive voice	Perfectly similar	8	2	83.33%
		Partially similar	9	1	
		Dissimilar	8	2	
Level 5	Complex - 1	Perfectly similar	7	3	80%
		Partially similar	9	1	
		Dissimilar	8	2	
Level 6	Complex - 2	Perfectly similar	7	3	80%
		Partially similar	9	1	
		Dissimilar	8	2	
Overall Accuracy					**87.22%**

As an example, consider a sentence pair which has more than one noun or verb. These can categorized as level 5 based on its complexity.

1. *rAmaH sIwayA saha vipinam acalaw* (Rama went to forest with Sita)
2. *rAmaH vanam calawi* (Rama goes to forest)

In these sentences, the actor is 'Rama', the act is 'to go' and the destination is 'forest' (vipinam, vanam). Sentence 1 is in past tense while sentence 2 is in present tense. Also in sentence 1, the actor 'Rama' is accompanied by 'Sita' to forest. Due to this, the system computed similarity score can be considered appropriate at 66%.

It can be seen that the proposed approach was quite effective and was able to handle increasing levels of complexity in Sanskrit sentences very well. The overall accuracy of the system considering all the six levels was about 87.22%.

4 Conclusion and Future Work

A morphological analysis based technique for measuring semantic similarity between two Sanskrit sentences of varied morphological characteristics was discussed in this paper. The latent properties of words in a sentence are captured and used by the scoring algorithm, that associates different weights to various components of the sentence. Cases like active/passive voice, prefixes and words other than noun/verb that contribute to semantics to some extent were also addressed. Experimental evaluation showed that our approach works very well for sentence pairs of varying complexity levels. For improving our model further, we intend to apply the Word2Vec model to capture other related words in addition to synsets obtained from WordNet. The model could be further enhanced by addressing compound sentence similarity and also by extending support to other Dravidian languages.

References

1. Ambati, B., Gadde, P., Jindal, K.: Experiments in Indian language dependency parsing. In: ICON09 NLP Tools Contest: Indian Language Dependency Parsing (2009)
2. Deerwester, S., Dumais, S., et al.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. **41**(6), 391 (1990)
3. Goyal, P., Arora, V., Behera, L.: Analysis of Sanskrit text: parsing and semantic relations. In: Huet, G., Kulkarni, A., Scharf, P. (eds.) ISCLS 2007-2008. LNCS, vol. 5402, pp. 200–218. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00155-0_7
4. Goyal, P., Huet, G.: Completeness analysis of a Sanskrit reader. In: 5th International Symposium on Sanskrit Computational Linguistics, pp. 130–171 (2013)
5. Gupta, R., Goyal, P., Diwakar, S.: Transliteration among Indian languages using WX notation. In: KONVENS, pp. 147–150 (2010)
6. Hellwig, O.: Extracting dependency trees from Sanskrit texts. In: Kulkarni, A., Huet, G. (eds.) ISCLS 2009. LNCS, vol. 5406, pp. 106–115. Springer, Heidelberg (2008). doi:10.1007/978-3-540-93885-9_9
7. Huet, G.: Shallow syntax analysis in Sanskrit guided by semantic nets constraints. In: International Workshop on Research Issues in Digital Libraries, p. 6. ACM (2006)
8. Huet, G.: Formal structure of Sanskrit text: requirements analysis for a mechanical Sanskrit processor. In: Huet, G., Kulkarni, A., Scharf, P. (eds.) ISCLS 2007-2008. LNCS, vol. 5402, pp. 162–199. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00155-0_6
9. Husain, S.: Dependency parsers for Indian languages. In: ICON09 NLP Tools Contest: Indian Language Dependency Parsing (2009)
10. Chang, J., et al.: Using grammar patterns to evaluate semantic similarity for short texts. In: Computing Technology & Information Management (ICCM) (2012)
11. Katre, S.M., et al.: Astādhyāyī of Pāini. Motilal Banarsidass Publisher, Delhi (1989)

12. Kulkarni, A., Pokar, S., Shukl, D.: Designing a constraint based parser for Sanskrit. In: Jha, G.N. (ed.) ISCLS 2010. LNCS, vol. 6465, pp. 70–90. Springer, Heidelberg (2010). doi:10.1007/978-3-642-17528-2_6
13. Kulkarni, M., Dangarikar, C., et al.: Introducing Sanskrit Wordnet. In: 5th Global Wordnet Conference (2010)
14. Padó, S., Lapata, M.: Dependency-based construction of semantic space models. Comput. Linguist. **33**(2), 161–199 (2007)