Semantic Textual Similarity For Hindi

by

darshan.agarwal , vandan.mujadia , Radhika Mamidi, Dipti Misra Sharma

in

18th International Conference on Computational Linguistics and Intelligent Text Processing
(*CICLing-2017*)

Budapest, Hungary

Report No: IIIT/TR/2017/-1

Centre for Language Technologies Research Centre
International Institute of Information Technology
Hyderabad - 500 032, INDIA
April 2017

Semantic Textual Similarity For Hindi

Darshan Agarwal, Vandan Mujadia, Radhika Mamidi, and Dipti Misra Sharma

Kohli Center On Intelligent Systems (KCIS)
International Institute Of Information Technology (IIIT-H)
Hyderabad, Telangana, 500032
{agarwal.darshan95, vmujadia}@gmail.com,
{radhika.mamidi, dipti}@iiit.ac.in

Abstract Semantic textual similarity is the degree of equivalence between the two sentences semantically. We may also say, it is the ability to substitute one text for the other without changing its meaning. In this paper, we propose rule based and supervised systems which measure the semantic relatedness between two Hindi sentences on the scale of 0 (least similar) to 5 (most similar). Both systems make use of several syntactico-semantic features such as language specific linguistic characteristics, distributional semantics and dependency clusters. With several constraints on these features, our rule based system is able to achieve around 75.23% accuracy on Hindi news similarity corpus. In supervised approach, we use support vector machine (SVM) with above mentioned features and euclidean distance between dependency clusters to derive word level alignments. Later, we use these alignments to assign similarity score between two sentences. With this approach we are able to achieve considerable accuracy on a small set of our corpus.

1 Introduction

Semantic Textual Similarity (STS) is the measure of similarity between two sentences at the semantic level. The range of textual similarity can vary from 0 to 5, where 0 is completely unrelated, and 5 is exact semantic equivalence. The intermediate scores capture the minor aspects of differences in meaning to significant differences between the two sentences. Sentence level semantic similarity is a difficult task compared to word level (Wordnet) and phrase level similarity[1]; because at the sentence level, one needs to incorporate other factors such as word order, identification of the event and entities from the sentence, and the relation between them.

STS assumes bidirectional equivalence between the pair of texts. Also, instead of answering binary (yes/no) questions as in Textual Entailment and Paraphrase, STS introduces the notion of graded similarity. This comes in handy for a range of NLP tasks such as paraphrase recognition [2], automatic machine translation evaluation [3], machine translation [4], short answer grading [5], subtopic boundary identification [6] and schema matching. Due to non-availability of human annotated STS data, earlier approaches on computing semantic similarity [7,8,9,10] were unsupervised and had to be evaluated extrinsically on other

tasks. Development of supervised STS systems and intrinsic evaluation of STS systems was made possible by SemEval STS task series [11,12,13,14]. The task series has provided a large semantic similarity annotated data set for English. Currently, there are no STS systems available for Hindi language. This paper is an attempt at coming up with a STS system for Hindi.

Organisation of the Paper: The remaining sections of the paper are organised as follows: Section 2 describes the other STS systems and their limitations. Section 3 describes the system description. Section 4 describes our STS systems. Section 5 gives the evaluation and the section 6 gives error analysis of the systems presented, where our systems mainly failed. Finally, the section 7 gives the conclusion and the future work.

2 Related Work

Much work has been done in the field of Semantic Textual Similarity for English in the SEMEVAL tasks. With domain-specific traninig data for several test sets, supervised systems [15,16,17] were the best performing systems in SEMEVAL 2012. These systems typically generated the similarity score by using regression algorithm on different similarity features such as string similarity, syntactic similarity and word or phrase level semantic similarity. However, in 2013, 2014 and 2015, the best systems were unsupervised [18,19,20]. The main component of these unsupervised systems was alignment. Even in 2016, top systems were hybrid systems. These systems first aligned semantically related terms between the two sentences and finally computed score as a increasing function of the number of alignments.

3 System Description

In all our approaches, we extract a defined score based on the number of word alignments. We adopt a hypothesis that the degree of similarity between two sentences is dependent on number of similar semantic units the sentences contain and these similar units must occur in a similar context. Therefore, the word aligners play a central role in all the systems. The accuracy of the systems depends on how good the word aligner is. In order to extract the context, we make use of paninian based dependency trees [21]. We conducted all our experiments on news domain corpus.

3.1 Corpus Details

A general domain news corpus of 750 pairs of sentences collected from Hindi newspapers and essays[1] was used. The average word length of a sentence in the corpus is 8. This corpus was annotated manually with rich linguistic information from shallow features like part of speech to deep features like dependency

[1] http://khabar.ndtv.com/, http://hindi.news18.com/, etc.

label. The dependency annotation in corpora were based on the Computational Paninian Grammar (CPG-henceforth) framework, as explained in [21]. Later, the same corpus was annotated with semantic textual similarity score on the range of 0 to 5 using a modified annotation scheme of [11] for Hindi. We modified the scheme by dividing a sentence into events and entities. We then, calculated the similarity between the events and entities individually. Thus, simplifying the scoring process and also reducing ambiguities. The annotation was done by 4 language annotators. The inter annotator agreement obtained for it was 0.80 Fleiss' Kappa measure [22]. In future, we will release the dataset.

3.2 Scoring Scheme

The scoring scheme of [19] gives scores depending on the degree of number of alignments, irrespective of which words are being aligned. Thus, giving equal weight to nouns, verbs, adjectives and adverbs. Using this scoring scheme, we observed that the alignment scores for the sentence pairs are greater than it actually should since words belonging to different categories contribute differently to the sentence meaning. Thus, we hypothesize that sentence meaning unfolds from the verb, therefore we assign highest weight to the verb followed by nouns, adjectives and adverbs.

Table 1: Weights

POS	Weight
Verb	0.4
Noun	0.25
Adjective	0.15
Adverb	0.1

w_i^s: i^{th} word $\in$ sentence s, Al: set of aligned words, $pos_m : m^{th}$ pos tag

$$Al^s_{pos_m} = |\{i : [\exists j : (i,j) \in Al] \, and \, w_i^s \in pos_m\}| \tag{1}$$

$$total^s_{pos_m} = |\{i : w_i^s \in pos_m\}| \tag{2}$$

$$k_m = \frac{max(Al^s_{pos_m}, Al^t_{pos_m})}{max(1, min(total^s_{pos_m}, total^t_{pos_m}))}, \tag{3}$$

$$Score = \sum_{m=1}^{4} weight_{pos_m} * k_m \tag{4}$$

As Table 1 denotes, weight to Verb is given the highest i.e. 0.4, followed by Noun, Adjective and Adverb. These values are decided after using the grid search, for news corpus. These values turned out to be the best. Equation(4) denotes how the score is computed for each Part of speech.

4 Experiments

In this section, we discuss our three different systems for measuring semantic similarity between two sentences.

4.1 Dependency based approach (Rule based)

We followed a ruled based approach in the first experiment that we conducted. We have used [19] word aligner with some modifications. We use only content word alignment module. Here, by the content words we mean the words with their part of speech as Verb, Noun, Adjective and Adverb. We align words in a set order (verb, noun, adjective, adverb). The words which are already aligned by the particular category are not checked again. The aligner takes two sentences S_1, S_2 as input and call alignment modules separately for each part of speech. The following steps are followed by the aligner :

- The aligner finds word pairs $w_i^1 \, \varepsilon \, S_1$ and $w_j^2 \, \varepsilon \, S_2$ which are similar at the word level. This process of checking if words are similar is explained in section 4.1.1. If w_i^1 and w_j^2 are not similar, then the aligner moves on to other set of word pairs.
- The semantic context of w_i^1, w_j^2 denoted by C_i^1 and C_j^2 respectively is extracted using either the dependency trees of S_1 and S_2 or the neighborhood of w_i^1 and w_j^2. The similarity between the semantic contexts C_i^1 and C_j^2 denoted by Sim_{ij} represents the contextual similarity score of w_i^1 and w_j^2. This process is explained in detail in section 4.1.2.
- In order to incorporate many to one alignment in our aligner, the contextual similarity for each possible word pair is calculated.
- Finally, the aligner aligns every word w_i^1 of sentence S_1 with a word w_j^2, which has maximum and non-zero contextual similarity Sim_{ij}. For many to one alignment, every word w_j^2 is also checked for alignment with w_i^1
- After obtaining the contextual similarity score Sim_{ij} for all similar word pairs, the word pairs are aligned according to Sim_{ij}.

Our aligner has mainly two modules which are called in the order as shown in figure 1. Since Hindi is a free word order language and to get similarity at the semantic level, we gave higher priority to context derived through dependencies than to textual neighborhood. So to avoid alignment of similar words with different roles in both the sentences, we first align those words which have some common context. For example, in the following two sentences, though the word *Ram* is identical in both the sentences, but their semantic roles are different, so semantic similarity between these two sentences should not be 5, though all the words are identical in these two sentences.

- राम ने श्याम को मारा ।
  Ram killed Shyam
- श्याम ने राम को मारा ।
  Shyam killed Ram

Figure 1: System Overview

In further sections, we explain how do we calculate word similarity and contextual similarity.

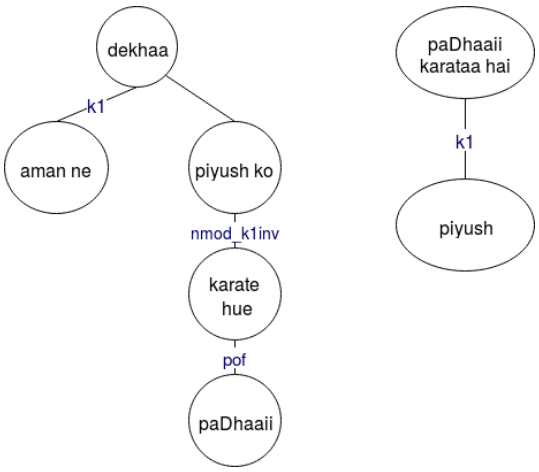
4.1.1 Word Similarity Word similarity is the similarity between two words at the semantic level, without taking their respective contexts into consideration. In order to derive similarity between words at the semantic level, there is no such database as PPDB [1] in Hindi, so for determining the word level similarity, we use word2vec [23]. We crawled a huge monolingual Hindi corpus, from many Hindi newspaper websites [1], consisting of 10 million words. Using this corpus, we trained a gensim model [2], choosing the dimension of each word as 400 (chosen by trial and error). To check if the words $w_i^1 \, \varepsilon \, sentence \, S_1$ and $w_j^2 \, \varepsilon \, sentence \, S_2$ should be considered as similar, we consider their lemmas and follow different approaches. Initially, we use knowledge resource Hindi Wordnet [24], and check if w_j^2 exists in the synsets, hypernyms or hyponyms of w_i^1 and vice versa. If it exists, we consider the words as similar. If the words are not found to be similar by Wordnet module, then we experimented with two different approaches.

1. **Top 20 :** The aligner extracts the top 20 similar words of w_i^1 and w_j^2 using the trained gensim model, and checks if w_j^2 exists in the top 20 similar words of w_i^1, similarly it is checked if w_i^1 exists in the top 20 similar words of w_j^2. If the either word exists in either of the lists, the words are considered similar.
2. **Similarity Score :** Here, similarity score is used as a measure. The aligner finds the similarity score between the two words, (using trained gensim model, which gives it on the scale of 0 to 1), and if the score is greater than the threshold chosen (0.8), we treat the words as similar.
   The approach of Top 20 turned out to be better in validating the similar words because there were such cases, where words which are similar at the semantic level didn't get a good similarity score, but both of them existed in the top 20 lists. For example, in the following sentences the similarity score between उड़ (fly) and उड़े (flew) was turning out to be less than the threshold, due to which these words could not align, but उड़ ε Top20(उड़े).
   – वे समुहों में घोंसले से बाहर उड़ गए ।
   ve samuhoN meN ghoNsle se baahar uD gaye.
   They flew out of the nest in groups.

[1] http://khabar.ndtv.com/, http://hindi.news18.com/, etc.
[2] https://radimrehurek.com/gensim/models/word2vec

– वे एक साथ घोंसले से उड़े ।
ve ek saath ghoNsle se uDe.
They flew from the nest.

4.1.2 Contextual Similarity The aligner extracts context for the first two modules of figure 1. The context for the word pair is the Cartesian product of the context of each of the words. The context of a word is a set of words. After the extraction of contexts, contextual similarity between two words $w_i^1 \in S_1$ and $w_j^2 \in S_2$ is the sum of word similarities for each word pair in the context of w_i^1 and w_j^2.



Alignment Using Dependencies As stated earlier, we are using dependency trees to extract the context. Given two input sentences S_1 and S_2, the contextual similarity between $w_i^1 \varepsilon S_1$ and $w_j^2 \varepsilon S_2$ is calculated by considering every word pair $w_k^1 \varepsilon S_1$, $w_l^2 \varepsilon S_2$, such that :

- w_k^1 is either a parent or child of w_i^1.
- w_l^2 is either a parent or child of w_j^2.
- The part of speech of w_i^1 is same as w_j^2.
- The part of speech of w_k^1 is same as w_l^2.
- The dependency relations of (w_i^1, w_k^1) and (w_j^2, w_l^2) must be equivalent. For dependency equivalence, we used the dependency equivalency table of [19]. This table gives equivalencies for Stanford dependency labels. We used a mapping from Stanford dependency labels to Paninian grammar dependency labels provided by [25] to use this table. To understand what we mean by equivalency between dependency types, consider the relationship between the words (*piyush* [Piyush], *paDhaaii* [studies]) in the dependency trees of the following two sentences in the figure 2:

- अमन ने पियूष को पढाई करते हुए देखा ।
  aman ne piyush ko paDhaaii karate hue dekhaa.
  Aman saw Piyush studying.
- पियूष पढाई करता है ।
  piyush paDhaaii karataa hai.
  Piyush studies.

In the two sentences, the relationship k1 (doer) is equivalent to k2 (theme), since in both the sentences, *Piyush* is the *doer* of the action *studying*.

– If the relation is equivalent, word similarity score of (w_k^1, w_l^2) is added to the contextual similarity Sim_{ij} of w_i^1 and w_j^2. Thus Sim_{ij} is obtained for each word pair.

Alignments Using Textual Neighborhood In this module, a fixed window of three content words to the left and right of the word is used as a context. As stated earlier, the context of the word pair (w_i^1, w_j^2) is the cartesian product of the context of w_i^1 and the context of w_j^2. Here, w_i^1 and w_j^2 can have different parts of speech. The window size is chosen, to avoid the unrelated words and make the sufficient contextual information available. Since a content word can be surrounded by stop words providing no contextual information.
Once the words are aligned using the two modules as shown in figure 1. The similarity score between the two sentences are obtained following the scoring scheme given in section 3.2. The scores obtained for the rule based system are very poor. On analyzing the errors, we observed there were two limitations :

– The equivalency table constructed for karakas. The reason for this could be the Stanford dependencies which are syntactic in nature, while the dependency labels of Panini are syntactic-o-semantic. Since the labels in the two schemes do not map exactly, the equivalency between dependency labels may not be correct.
– Certain linguistic phenomena in Hindi and specific alignment issues which we failed to capture. Thus, we decided to change our approach with a modified equivalence method.

4.2 Modified Rule Based Approach

In this section, we modify the existing system to handle it's limitations.

Word Embedding based Dependency Clustering To get the equivalency between dependency types a statistical approach is used. We transformed every word's 'karaka' or dependency label into a vector representation. For this transformation, the knowledge of placing all words in a semantic space, and treating each word as a vector in that space is used.

We have used the same gensim model as discussed in Word Similarity section to get a 400 dimensional vector representation for every word. Every dependency label is transformed into a vector representation by creating clusters.

Algorithm 1 Cluster Creation(D)

```
INPUT:
    D = [S1, S2, S3, ...]
OUTPUT:
    ChildCluster[wi][reli] : Child Cluster of reli for wi
    ParCluster[wi][reli] : Parent Cluster of reli for wi
 1: procedure CLUSTER CREATION(D)
 2:     for each sentence Si in D do
 3:         for each word wi in Si do
 4:             for each word wj in Children of wi do
 5:                 R = Relation between wi and wj
 6:                 Add wj to Childwrd[wi][R]
 7:                 Add wi to Parentwrd[wj][R]
 8:             end for
 9:         end for
10:     end for
11:     for each word wi in Childwrd do
12:         for each relation reli in Childwrd[wi] do
13:             Sum = [400 dim zero vector]
14:             Len = 0
15:             for each word wj in Childwrd[w[i]][reli] do
16:                 Add vector representation of wj to Sum
17:                 Len = Len + 1
18:             end for
19:             Mean = Sum / Len
20:             ChildCluster[wi][reli] = Mean
21:         end for
22:     end for
23: end procedure
```

To create clusters, we used the Hindi Dependency Treebank Data [26] which has 10,000 manually parsed sentences. As explained in Algorithm1, two types of clusters, Parent cluster (*ParCluster*) and *ChildCluster* are created. Initially, for each word w_i, $Childwrds[w_i]$ is created, which has clusters of dependency labels rel_1, rel_2, rel_3,... . $Childwrds[w_i][rel_j]$ represents the cluster consisting of all the words which appear as a child of w_i with dependency label as rel_j. Finally, $ChildCluster[w_i][rel_j]$ is derived which is a single 400 dimensional vector representing the dependency label rel_j of w_i. The vector is derived by taking the mean of all vector representations of words in $Childwrds[w_i][rel_j]$. Similarly, a Parent Cluster is calculated where $ParCluster[w_i][rel_j]$ represents the mean of all words which have existed as parent of w_i with dependency label as rel_j.

Since a word w_i, can act as a child of w_j in different roles (rel_k, rel_l), many clusters overlap. We observed that each of these clusters are different from the other as there are some words which only act as rel_l but not as rel_k. This separates the cluster of rel_l from rel_k, thereby making the mean of those clusters also different. To find equivalency between relation rel_2 of word w_i and relation rel_1 of word w_j :

1. The distance $dist_{i2j1}$ between means of rel_2 cluster of w_i (rel_{i2}) and rel_1 cluster of w_j (rel_{j1}) is calculated. As shown in figure 3.
2. If $dist_{i2j1}$ is lesser than threshold, then the dependencies rel_{i2} and rel_{j1} of words w_i and w_j are considered to be equivalent. The threshold mentioned is identified by trial and error method. Four different thresholds are calculated, depending on 4 different permutations of parent and child.

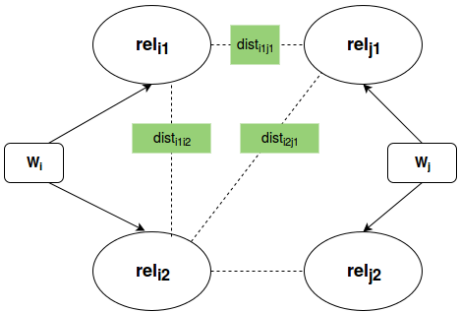


Figure 2: $dist_{i1j1}$: distance between rel_{i1} and rel_{j1}.

Specific Alignment Issues After analyzing the results of our initial rule based experiment, there were changes made to the aligner specifically for Hindi and for Paninian dependency labels.

Complex Predicates (POF) Indian languages have extensive use of conjunct verbs. A conjunct verb is composed of a noun or an adjective followed by a verbalizer. In the Paninian dependency scheme the two are treated as separate tokens and the noun/adjective is attached to the verb with a label *pof*.

Example 41
बीजेपी सांसद शत्रुगन सिन्हा ने शनिवार रात करीब दस बजे सीएम नितीश कुमार से उनके सरकारी आवास पर मुलाक़ात की ।
biijepii saaMsad shatrughan sinhaa ne shanivaar raat kariib 10 baje CM nitish kumAr se unake sarkaarii aavaas par mulaakaat kii.
BJP MP Shatrugan Sinha met CM Nitish Kumar on saturday night around 10 o' clock at his official residence.

सिन्हा और कुमार के बीच मुलाक़ात मुख्यमंत्री आवास में हुई ।
sinhaa aur kumaar ke beecha mulaakaat mukhyamantrii aavaas meN huii.
The meeting between Sinha and Kumar took place at the Chief Ministers' residence.

Here, the event is actually represented by the noun, but not by the verb. So, we treat the noun of the conjunct verb as a verb, while assigning the similarity score between the sentences. If *pof* is the dependency label between the verb w_i^1 and its child w_k^1, then w_k^1 is checked for alignment with every content word in sentence s_2, irrespective of its part of speech. This is done, since the verb can be nominalized in the first sentence, it can be done in the second one too, so we check alignment of w_k^1 with every content word in S_2. If w_k^1 is aligned to any word $w_j^2 \, \varepsilon \, S_2$, then w_i^1 is also aligned to w_j^2. In example 4.1, the conjunct verb मुलाक़ात की *(met)* is used in the first sentence, so the noun मुलाक़ात *(meeting)* is treated as verb and is checked for alignment with every word in s_2. It get's aligned to the identical word मुलाक़ात *(meeting)* in s_2, which is also used as a nominalized verb.

Coordination (*CCOF*) and Noun Modifier (*nmod*) The *ccof* dependency label represents coordination. If *ccof* is the parent relation of the word, then we treat its grand parent in the dependency tree (i.e. the parent of its existing parent) as the actual parent. Since the coordination word does not give any contextual information of its child. In the example 4.1, the parent of the words सिन्हा *(Sinha)* and कुमार *(kumAr)* is a conjunction और *(and)* with the dependency label as *ccof*, which does not give any context information of कुमार *(kumAr)* or सिन्हा *(Sinha)*. The parent of और *(and)* is the verb हुई *(happened)* with relation k2, so when ever the parent of *Sinha* or *kumar* is required then हुई *(happened)* with relation *k2* is returned instead of और *(and)* with relation *ccof*. Similarly, in the case of noun modifier *nmod* as the dependency relation with the parent word, it signifies that the current word is only modifying its parent, so the current word is aligned to the words to which its parent is also aligned. In the example 4.1, the relation between the word शत्रुगन *(Shatrugan)* and its parent सिन्हा *(Sinha)* is *nmod*, so *Shatrugan* is also treated as *Sinha* for alignment.

Chunk Property If $(w_{11}, w_{12}, w_{13}) \varepsilon S_1$ are part of the same chunk C_1, w_{13} is aligned to $w_{21} \varepsilon S_2$ and w_{21} is the only content word in its chunk C_2, then w_{11}, w_{12} is also aligned to w_{21}. In the example 4.2, the chunk C_1 : भारतीय सेना *(Indian army)* εS_1, is aligned to the chunk C_2 : सेना *(army)*, since the word सेना (army) εC_1 and C_2 are aligned, भारतीय *(Indian)* is also aligned to सेना *(army)* εS_2, since C_2 does not contain any other word. We align *Indian army* and *army* assuming that in S_2, it is implicitly assumed that the topic of discussion is on *army* which is *Indian*. But if, the chunk C_2 was (*Chinese Army*), the aligner would not align C_1 to C_2.

Example 42
S_1 :बलवीर सिंह ने भारतीय सेना प्रमुख को चुनौती दी थी ।
balveer siNha ne bhaaratiya senaa pramukh ko chunautii dii thii.
Balvir singh had challenged [***Indian Army Chief***]$_{C_1}$.

S_2 :बलवीर सिंह ने सेना प्रमुख को चुनौती दी थी ।
balveer siNha ne senaa pramukh ko chunautii dii thii.
Balvir singh had challenged the [***Army Chief***]$_{C_2}$.

Children Alignment As we have discussed previously, to get the contextual similarity (Sim_{ij}) of $w_i^1 \varepsilon S_1$ and $w_j^2 \varepsilon S_2$, we sum up the similarities of their parents and their children. Here, we would just check the similarity at one level above and below w_i^1 in its dependency tree. This does not give the complete context for the two words, so in order to get the complete context, we traverse down the tree till we reach its lowest level. As explained in Algorithm 2 (in appendix), if the similarity between w_k^1 (the child of w_i^1) and w_l^2 (child of w_j^2) is greater than a threshold or if the words are already aligned, then other than adding the similarity between (w_k^1, w_l^2) to $CSim_{ij}$, the aligner sums up even the

Algorithm 2 Align Children

INPUT:
$w_i^1 \varepsilon S_1$, $w_j^2 \varepsilon S_2$, $Sim(w_i^1, w_j^2)$: similarity score between w_i^1 and w_j^2
OUTPUT:
$CSim_{ij}$: Contextual Similarity of w_i^1, w_j^2 due to children

```
procedure AlignChildren()
    Children_i^1 = FindChildren of w_i^1 with their relations
    Children_j^2 = FindChildren of w_j^2 with their relations
    for word w_k^1, relation rel_ki^1 in Children_i^1 do
        for word w_l^2, relation rel_lj^2 in Children_j^2 do
            if (Sim(w_k^1, w_l^2) > ChildTh) and (rel_ki^1 ~ rel_lj^2) then
                Add Sim(w_k^1, w_l^2) to CSim_ij
                CSim_kl = AlignChildren(w_k^1, w_l^2)
                Add CSim_kl to CSim_ij
            end if
        end for
    end for
    return CSim_ij
end procedure
```

contextual similarity ($CSim_{kl}$) of w_k^1, w_l^2 to $CSim_{ij}$ by recursively calling the same function.
Thus, using the discussed method for dependency equivalency and the certain modifications in the aligner, we ran our semantic similarity system on the corpus. The results of the system are discussed in later sections.

Table 2: Results : Supervised Approach

Features	Size of Input	Accuracy
$Vec(W_1)[W_1]+Vec(W_2)[W_2]$	800	74.23
$W_1+W_2+Vec(Par[W_1])[Par_1]+Vec(Par[W_2])][Par_2]$	1600	68.0
$W_1+W_2+Par_1+Par_2+rel_{11}+rel_{22}$	1602	71.71
$W_1+W_2+Par_1+Par_2+rel_{11}+rel_{22}+dist(rel_{11},rel_{22})$	1603	79.98
NlW_1(Not lemma)$+NlW_2+NlPar_1+NlPar_2+rel_{11}+rel_{22}+dist(rel_{11},rel_{22})$	1603	71.23

4.3 Supervised Approach

In our final approach, we experimented by running a basic linear SVM classifier [27] using the data of our previous system. We developed a supervised semantic similarity system, and performed experiments by incrementally adding more features with optimal combination, thereby increasing context information. The output of the system is a boolean whether the two input words should be aligned or not. It uses the output of the second experiment discussed in section 4.2. So, word alignments of those sentence pairs for which the scores did not vary from Gold scores by 0-0.9 were included in the data for our Supervised approach. We divided these alignments into two parts training and test data. 1000 word alignments constituted our training data and the remaining 1000 alignments were treated as test data.

In the first experiment, only the vectors of both the words were used as features, to check if the two words should be aligned. In the second experiment, the parents of both the words are also added as features. But as we see from Table 2, the accuracy is reduced, suggesting that SVM has got more confused with these features. In the next experiment, we even add dependency relation information as features. In this case, the accuracy has increased from the previous experiment, but still the results are not better. In the next experiment, upon adding the distance between the parent clusters as a feature, the accuracy rises to 79%. Thus, proving that the distance measure helped the SVM to resolve many confusions. In the final experiment, instead of considering lemmas of the words, we considered the word originally, but the results were not observed to be good. So, to get the word alignments between two sentences s_1 and s_2, we run the system for every word pair $(w_i^1 \varepsilon s_1, w_j^2 \varepsilon s_2)$ and check if the words can be aligned. Finally, return the alignments for which system returned true. Using these alignments we computed the similarity score using equation 4.

5 Evaluation

We ran our second experiment on entire Hindi news similarity corpus consisting of 750 pairs of sentences and derived the scores using the scoring scheme discussed in section 3.2. The output scores of the system is evaluated by comparing it with the human-annotated similarity scores and measured using the Pearson Correlation Coefficient. We obtained a mean correlation of 75.12%, though we tested on a smaller dataset. We also tested our supervised approach on a small set of data of 300 pair of sentences, which have not been used as the training data. We obtained a correlation of 79.98%.

6 Error analysis

The cases where our proposed system were not showing good results are as follows:

– The system fails to capture similarity in cases where world knowledge is used. For example, in the following sentences, our system could not give good results, since it could not match the entities अयोध्या के राजा राम(Ram, king of Ayodhya) and भगवान श्री रामचंद्रजी(God Ram), though by world knowledge one would know that God Ram in Hindu mythology was the king of Ayodhya.

Example 61
S_1 :राम भक्तों के अनुसार दीवाली वाले दिन अयोध्या के राजा राम लंका के अत्याचारी राजा रावण का वध करके घर लौटे थे ।
raam bhaktoN ke anusaar diivaalii vaale din ayodhyaa ke raajaa raam laNkaa ke atyaachaarii raajaa raavaN kaa vadh karke ghar lauTe the.
According to his devotees, Lord Rama, the king of Ayodhya returned home on the day of Diwali after killing the tyrannical king of Lanka Ravana.

S_2 :हिंदू मान्यताओं में राम भक्तों के अनुसार भगवान श्री रामचंद्रजी असुरी वृत्तियों के प्रतीक रावणादि का संहार करके अयोध्या लौटे थे ।
hinduu maanyataaoN meN raam bhaktoN ke anusaar bhagvaan shrii ramchandrajii asurii vrattiyoN ke pratiik ravaNaadi kaa saNhaar karke ayodhyaa lauTe the.
In Hindu beliefs, according to the devotees of Ram, Lord Shri Ramchandra returned to Ayodhya after killing the symbol of daemonic instincts like Ravana.

– For events, composed of multiple terms, our system failed to identify similarity among them. For example, in the below two sentences, the events स्नान कर रहा है (*bathing*) and अपने आप को धो रहा है (*washing itself*) could not be aligned by our system, since in the first sentence the event of bathing is explained using a single term स्नान (*bathing*), while in the other, the same event is explained using multiple terms.
  - पंछी नदी में स्नान कर रहा है ।
    paNchii nadii meN snaan kar rahaa hai.
    The bird is bathing in the river.
  - पंछी अपने आप को तालाब में धो रहा है ।
    paNchii apane aap ko talaab meN dho rahaa hai.
    Bird is washing itself in the lake.
– It was also observed that system could not give good results when metaphors like हरी झंडी दिखाई (*show green signal*) are used in sentences instead of उद्घाटन किया (*inaugurated*). Currently, our system cannot relate the two events.
  - पीएम मोदी ने धनियावा – बिहार शरीफ रेल लाइन को भी हरी झंडी दिखाई ।
    PM modi ne dhaniyaavaa - bihaar shariif rael laain ko bhi harii jhanDii dikhaaii.
    *PM Modi also **gave a green signal** to Dhaniyava-Bihar sharif rail line.*
  - प्रधानमंत्री नरेन्द्र मोदी ने धनियावा – बिहार शरीफ रेल लाइन का उद्घाटन किया ।
    pradhaanmantrii narendar modii ne dhaniyaava-bihaar shariif rael laain kaa udghaaTan kiyaa.
    *Prime minister Narendra Modi **inaugurated** Dhaniyava-Bihar sharif rail line.*

7 Conclusion & Future Work

In this paper, we have shown that alignment of related words by using Paninian dependency grammar can be used to compute semantic similarity between two sentences for Hindi. Our system has the quality of being fast and accurate. It is the first system developed for computing semantic textual similarity specifically for Hindi. Use of vector similarity feature for relevancy between dependency roles has produced good results. We have also proposed a supervised approach, using features as vector representations of the words and their contexts, also the dependency roles and their equivalency. We could not test this system on a large data set, since it requires a considerable amount of training data. In the future, we would like to use some of the resources which can give world knowledge, that

can help in aligning entities such as metaphors. Further, in future we would also like to incorporate word sense disambiguation and more robust strategies for deriving word similarity.

References

1. Ganitkevitch, J., Van Durme, B., Callison-Burch, C.: Ppdb: The paraphrase database. In: HLT-NAACL. (2013) 758–764
2. Dolan, B., Quirk, C., Brockett, C.: Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In: Proceedings of the 20th international conference on Computational Linguistics, Association for Computational Linguistics (2004) 350
3. Kauchak, D., Barzilay, R.: Paraphrasing for automatic evaluation. In: Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Association for Computational Linguistics (2006) 455–462
4. Sachdeva, K., Sharma, D.M.: Exploring the effect of semantic similarity for phrase-based machine translation. ACL-IJCNLP 2015 (2015) 41
5. Mohler, M., Mihalcea, R.: Text-to-text semantic similarity for automatic short answer grading. In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics (2009) 567–575
6. Agarwal, D., Mujadia, V., Mamidi, R.: Subtopic boundary identification in hindi dialogue. In: Asian Language Processing (IALP), 2015 International Conference on, IEEE (2015) 156–159
7. Corley, C., Mihalcea, R.: Measuring the semantic similarity of texts. In: Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment, Association for Computational Linguistics (2005) 13–18
8. Mihalcea, R., Corley, C., Strapparava, C., et al.: Corpus-based and knowledge-based measures of text semantic similarity. In: AAAI. Volume 6. (2006) 775–780
9. Li, Y., McLean, D., Bandar, Z.A., O'shea, J.D., Crockett, K.: Sentence similarity based on semantic nets and corpus statistics. IEEE transactions on knowledge and data engineering **18** (2006) 1138–1150
10. Islam, A., Inkpen, D.: Semantic text similarity using corpus-based word similarity and string similarity. ACM Transactions on Knowledge Discovery from Data (TKDD) **2** (2008) 10
11. Agirre, E., Diab, M., Cer, D., Gonzalez-Agirre, A.: Semeval-2012 task 6: A pilot on semantic textual similarity. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation, Association for Computational Linguistics (2012) 385–393
12. Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W.: sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In: In* SEM 2013: The Second Joint Conference on Lexical and Computational Semantics. Association for Computational Linguistics, Citeseer (2013)
13. Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W., Mihalcea, R., Rigau, G., Wiebe, J.: Semeval-2014 task 10: Multilingual semantic textual similarity. In: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014). (2014) 81–91

14. Agirre, E., Banea, C., et al.: Semeval-2015 task 2: Semantic textual similarity, english, s-panish and pilot on interpretability. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), June. (2015)
15. Bär, D., Biemann, C., Gurevych, I., Zesch, T.: Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation, Association for Computational Linguistics (2012) 435–440
16. Šarić, F., Glavaš, G., Karan, M., Šnajder, J., Bašić, B.D.: Takelab: Systems for measuring semantic text similarity. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation, Association for Computational Linguistics (2012) 441–448
17. Jimenez, S., Becerra, C., Gelbukh, A.: Soft cardinality: A parameterized similarity function for text comparison. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation, Association for Computational Linguistics (2012) 449–453
18. Han, L., Kashyap, A., Finin, T., Mayfield, J., Weese, J.: Umbc ebiquity-core: Semantic textual similarity systems. In: Proceedings of the Second Joint Conference on Lexical and Computational Semantics. Volume 1. (2013) 44–52
19. Sultan, M.A., Bethard, S., Sumner, T.: Dls@ cu: Sentence similarity from word alignment. In: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014). (2014) 241–246
20. Sultan, M.A., Bethard, S., Sumner, T.: Dls@ cu: Sentence similarity from word alignment and semantic vector composition. In: Proceedings of the 9th International Workshop on Semantic Evaluation. (2015) 148–153
21. Begum, R., Husain, S., Dhwaj, A., Sharma, D.M., Bai, L., Sangal, R.: Dependency annotation scheme for indian languages. In: IJCNLP. (2008) 721–726
22. Randolph, J.J.: Free-marginal multirater kappa (multirater k [free]): An alternative to fleiss' fixed-marginal multirater kappa. Online submission (2005)
23. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. (2013) 3111–3119
24. Narayan, D., Chakrabarti, D., Pande, P., Bhattacharyya, P.: An experience in building the indo wordnet-a wordnet for hindi. In: First International Conference on Global WordNet, Mysore, India. (2002)
25. Kumar, M., Dua, M.: Adapting stanford parser's dependencies to paninian grammar's karaka relations using verbnet. Procedia Computer Science **58** (2015) 363–370
26. Palmer, M., Bhatt, R., Narasimhan, B., Rambow, O., Sharma, D.M., Xia, F.: Hindi syntax: Annotating dependency, lexical predicate-argument structure, and phrase structure. In: The 7th International Conference on Natural Language Processing. (2009) 14–17
27. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20** (1995) 273–297