How do different factors Impact the Inter-language Similarity? A Case Study on Indian languages

Sourav Kumar, Salil Aggarwal, Dipti Misra Sharma, Radhika Mamidi
LTRC, IIIT-Hyderabad
{sourav.kumar, salil.aggarwal}@research.iiit.ac.in
{dipti, radhika.mamidi}@iiit.ac.in

Abstract

India is one of the most linguistically diverse nations of the world and is culturally very rich. Most of these languages are somewhat similar to each other on account of sharing a common ancestry or being in contact for a long period of time (Bhattacharyya et al., 2016). Nowadays, researchers are constantly putting efforts in utilizing the language relatedness to improve the performance of various NLP systems such as cross lingual semantic search, machine translation (Kunchukuttan and Bhattacharyya, 2020), sentiment analysis systems, etc. So in this paper, we performed an extensive case study on similarity involving languages of the Indian subcontinent. Language similarity prediction is defined as the task of measuring how similar the two languages are on the basis of their lexical, morphological and syntactic features. In this study, we concentrate only on the approach to calculate lexical similarity between Indian languages by looking at various factors such as size and type of corpus, similarity algorithms, subword segmentation, etc. The main takeaways from our work are: (i) Relative order of the language similarities largely remain the same, regardless of the factors mentioned above, (ii) Similarity within the same language family is higher, (iii) Languages share more lexical features at the subword level.

1 Introduction

Recently, there has been an explosion in information (Wang et al., 2007) and a massive amount of natural language data is added daily on the Internet. Moreover, the human literature in different cultures is digitalized and became available in digital libraries (Farouk, 2019). A very large amount of this data is formatted in natural language. This makes NLP techniques crucial to make the use of this high amount of data. Since most of the NLP techniques either require linguistic knowledge that can only be developed by experts and native speakers of that language or they require a lot of labelled data which is again expensive to generate, NLP tasks become challenging for low resource languages like Indian languages. India is a multicultural country, a country with highly religious and ethnically diverse people. People of different races and classes live in different parts of the country, and they speak a variety of languages. Most of the Indian languages are divided into two main language families namely Indo-Aryan[1] and Dravidian[2]. Underlying the vast diversity in Indian languages are many commonalities. Because of contact over thousands of years, most of the Indian languages have undergone convergence to a large extent (Shridhar et al., 2020). Therefore, exploiting language relatedness becomes very crucial in NLP related tasks for Indian languages. Kunchukuttan and Bhattacharyya (2020) also presents an impressive case study for utilizing language relatedness for Machine translation but that study was more inclined toward exploring statistical approaches to MT. Prasanna (2018) in his work has explored efficient ways of exploiting relatedness in multilingualism and transfer learning for low resource machine translation.

But no such large scale study has been done on exploring different factors that may affect the process of calculating similarity among Indian languages. This could really help the future researchers in getting the clear picture while exploiting related languages in NLP related tasks. So, in this work, we performed an extensive case study on the language relatedness involving languages of the Indian subcontinent. This case study provides

[1]Indo Aryan languages - Hindi, Urdu, Punjabi, Gujarati, Marathi, Bangla, Oriya, Konkani

[2]Dravidian languages - Tamil, Telugu, Kannada, Malayalam

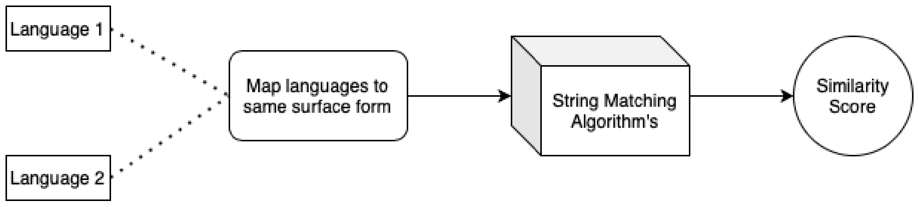

Language 1
Language 2
Map languages to same surface form
String Matching Algorithm's
Similarity Score

Figure 1: Pipeline for calculating similarity

the reader with valuable information about the different methodologies in measuring relatedness between Indian languages. In addition, it also compares some popular techniques for measuring sentence-to-sentence similarity. Moreover, datasets from different domains and sizes have also been used in comparing the similarity scores to enable the reader to build a complete background in this area. Also, these languages share a lot of cognates, that's why we have also compared the similarity among language pairs at both word and subword level.

This paper is further divided into 4 sections. Section 2 elaborated the methodology behind the different techniques and experiments. Section 3 elaborates the experimental details including the dataset preparation and pre-processing. All the results and analysis have been discussed in Section 4. Section 5 talks about conclusion and possible future work.

2 Methodology

A universal characteristic of Indian languages is their complex morphology and their unique characteristics following default sentence structure as subject object verb **(SOV)**. Thus, we will be using the parallel corpora for calculating the similarity among them. But Indian languages are written in different scripts, so in order to calculate the similarity between two languages, one needs to first map every language to a common surface form i.e to a common script. To do so, we are using a very well-known technique of *'Unified Transliteration'*. It is a string homomorphism technique in which every character of the source is replaced with the target language script. Following are the steps for calculating the similarity between the two languages:

- Collect parallel data of languages of which we want to calculate similarity.
- Transliterate those languages to a common script
- Calculate similarity of each sentence in source with the corresponding transliterated sentence in the target language using some string similarity algorithm.
- Return the average score over all the sentences in the parallel corpora.

The pipeline of the above discussed procedure is shown in **Figure 1**. Also in computer science, string similarity is an important family of algorithms that try to find a place where one or several strings (also called patterns) are found within a larger string. Researchers have already put the efforts and showed that these algorithms effectively calculate the similarity between two strings (Levenshtein, 1965; Yujian and Bo, 2007; Masek and Paterson, 1980; Larsen, 1992; Kondrak, 2005). Some studies have also been done on calculating similarity particularly for Indian languages (Singh and Surana, 2007; Wagner and Fischer, 1974; Islam and Inkpen, 2008; Akhtar et al., 2017; Sengupta and Saha, 2015). In this work, we will consider sentences as a string and use some of the above algorithms for calculating the similarity between two languages.

2.1 Token Overlap

This is the most general approach that works by converting strings into sets of their tokens and then counting the number of tokens which are shared between the both sets. Similarity between two languages using token overlap is calculated as follows:

$$sim = \frac{\sum_1^n \frac{|Token_{s1} \cap Token_{s2}|}{max(|Token_{s1}|,|Token_{s2}|)}}{n} * 100$$

Here, n denotes the total number of sentences in the parallel corpora, and s1 & s2 represent sentences from language1 and language2 respectively. Major disadvantage of this technique could be identification of "false friends" i.e words that look identical in two different languages, but actually mean something completely different and don't have a common source.

2.2 Levenshtein Distance

The Levenshtein distance (LD) (Levenshtein, 1965) between two strings is the minimum number of single character edits (insertions, deletions, or substitutions) required to change one string into the another. The algorithm considers one character of the string at a time and it assigns cost to each of the edit operations. The algorithm weights the cost of each operation and chooses the operation with the lowest cost and then moves on to the next character. We can compute Levenshtein similarity between two languages as follows:

$$sim_{Levenshtein} = \frac{\sum_1^n 1 - \frac{LD(s1,s2)}{max(|s1|,|s2|)}}{n} * 100$$

2.3 Longest Common Subsequence

The Longest Common Subsequence (LCS) (Larsen, 1992) is a string similarity measurement that is based on the longest common substring in a given string pair. The rationale is that, parts of the string may be similar while their prefixes or suffixes differ. This algorithm finds the longest common character sequence, between a string pair. The characters in the LCS do not necessarily need to be contiguous in the original strings. We can compute similarity using LCS between two languages as follows:

$$sim_{LCS} = \frac{\sum_1^n \frac{LCS(s1,s2)}{max(|s1|,|s2|)}}{n} * 100$$

2.4 Shingle (qgram) Similarity

This works by converting strings into sets of qgrams (sequences of q characters, also sometimes called k-shingles) Kondrak (2005). The similarity or distance between the two strings is then the similarity or distance between the sets. Here we are using Jaccard index as our similarity technique which is a special case of shingle based algorithms. We can compute similarity using Jaccard between two languages as follows:

$$sim_{qgram} = \frac{\sum_1^n qgram(s1, s2)}{n} * 100$$

3 Experiments

For our case study, we are performing all the experiments using the ILCI (Indian Language Corpora Initiative) Jha (2010) and PMI (Prime Minister of India) Haddow and Kirefu (2020) multi parallel corpora for Indian languages. ILCI contains 50k sentences of health and tourism domain covering all the major languages of India like Hindi, Urdu, Punjabi, Gujarati, Marathi, Bangla, Konkani, Telugu, Tamil, Malayalam. PMI contains 30k sentences of news domain in every language mentioned above including Oriya and Kannada except Konkani.

3.1 Data Preprocessing

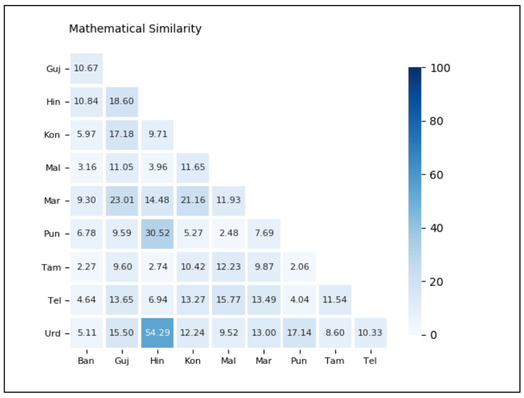
For transliterating the Urdu and Konkani to a common script, we used the Indic Trans library (Bhat et al., 2014), and for the others, we used Indic NLP library (Kunchukuttan, 2020) (as Urdu and Konkani not supported). In addition, there is an exception with Urdu because it follows a right to left writing system and all other Indian languages follow left to right writing order. Hence, in the processing step, we also changed the order of Urdu to maintain consistency among all languages, and doing this also made our string similarity algorithms work more efficiently.

3.2 Different scenarios

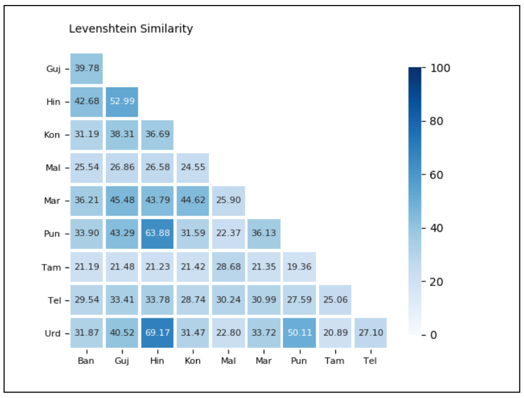
In the real world scenario there can be multiple possible cases that one can think of. But here, we are trying to cover the important cases according to our knowledge. Details of each use case is described below and for calculating the similarity among language pairs we are using the procedure mentioned in **section 2**.

Case 1: In this case, we are evaluating the effect of algorithm used for calculating sentence similarity on the similarity among the language pairs. We are computing the similarity for every language pair present in our ILCI corpora using each algorithm mentioned in **subsections 2**. Also, as per the requirement of our pipeline, we are also mapping each language to Devanagari script to share the same surface form.

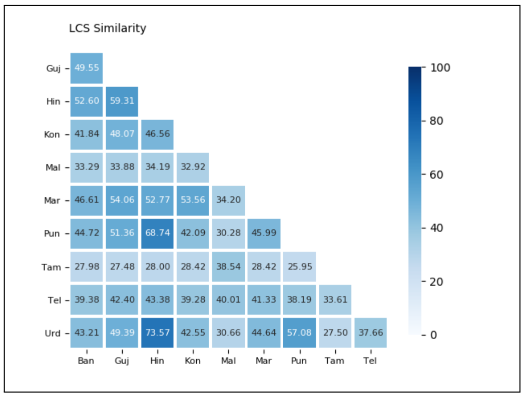
Case 2: Here, we are performing the experiments to confirm whether the choice of script selection matters in transliteration step of our pipeline for calculating similarity. To do so, we are mapping every language to Abugida instead Devanagari script and then compared results of both the cases. For this, we are only performing experiments using LCS and K-shingle algorithm on ILCI dataset.



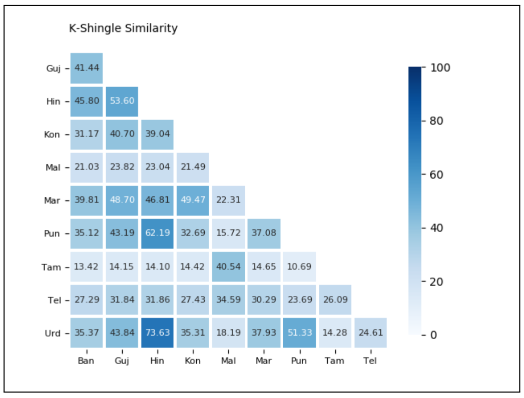
(a) Token Overlap



(b) Levenshtein

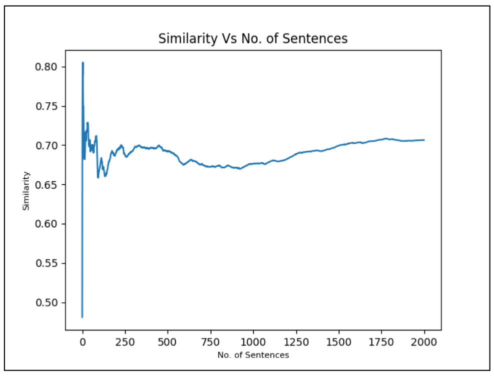


(c) Longest Common Subsequence

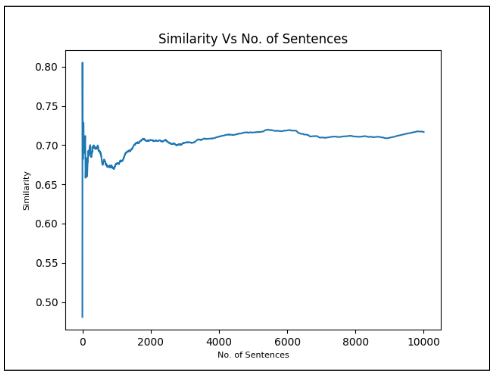


(d) Shingle Q-gram

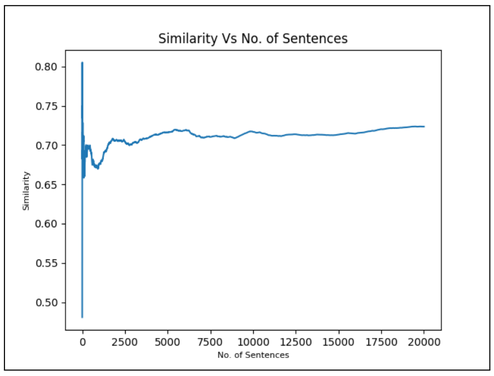
Figure 2: Similarity Matrix calculated using different algorithms



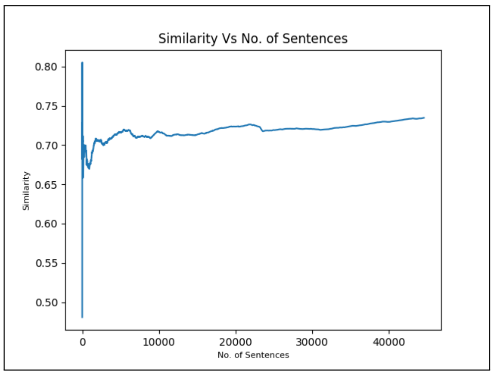
(a) Similarity using 2k Sentences



(b) Similarity using 10k Sentences

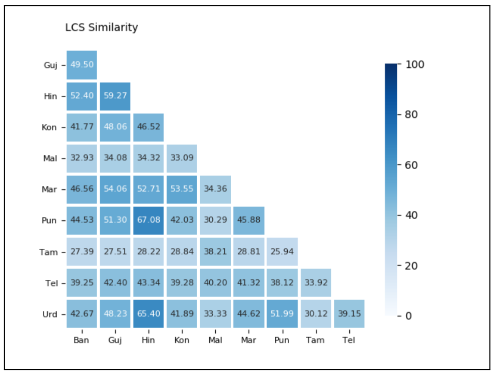


(c) Similarity using 20k Sentences

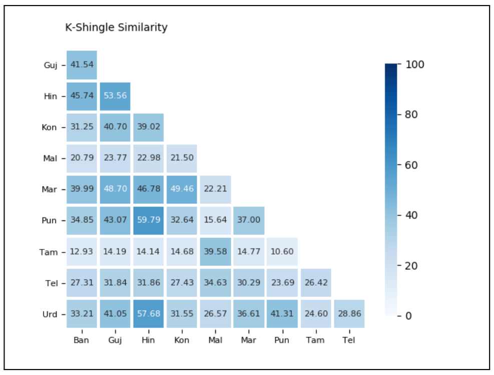


(d) Similarity using 50k Sentences

Figure 3: Similarity V/s No. of Parallel Sentences

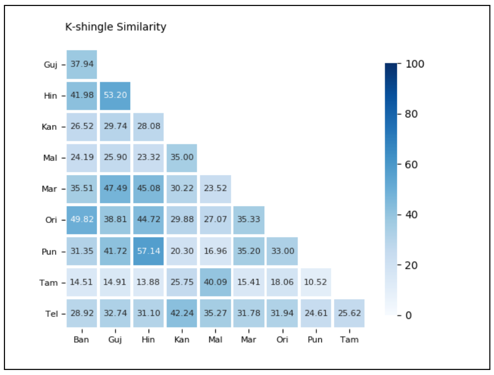


(a) LCS similarity

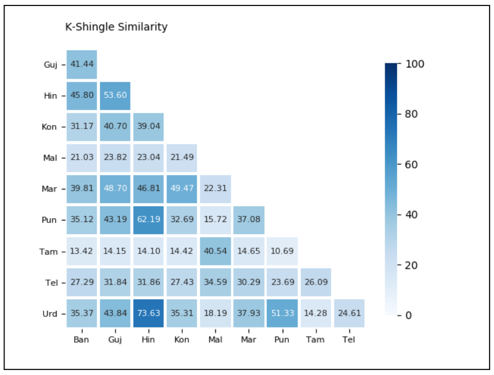


(b) Shingle Q-gram similarity

Figure 4: Effect of Script on Similarity; In this case we are converting every language to Abugida

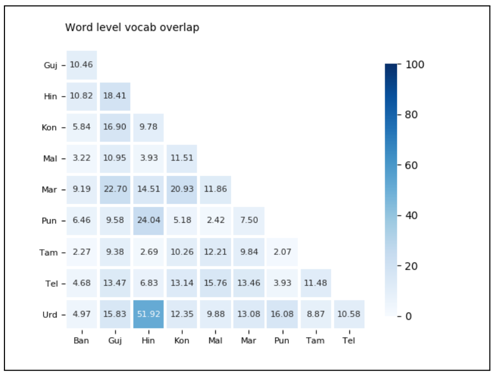


(a) PMI imilarity

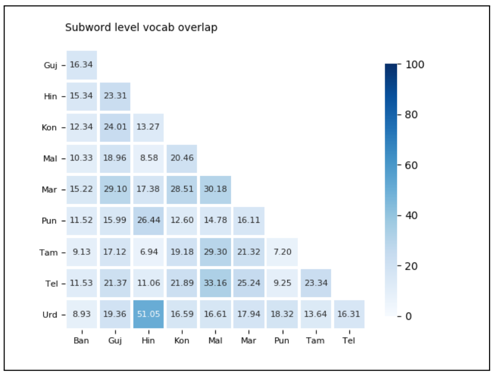


(b) ILCI similarity

Figure 5: Effect of Dataset on Similarity



(a) Word level similarity



(b) Sub-word level similarity

Figure 6: Effect of word segmentation on Similarity

Case3: In this scenario, we trying to figure out that after how many parallel sentences, the similarity score curve stabilizes itself. This will give us a rough idea of required size of parallel corpora for calculating similarity. To minimize our efforts, we are only performing experiments for Hindi-Urdu of ILCI corpora using the K-shingle algorithm.

Case 4: Here, we will be evaluating the important factor whether the type/domain of the dataset chosen effects the similarity among the different language pairs. For this case, we are calculating and comparing the results of similarity for every language pair on both ILCI and PMI dataset using the K-shingle algorithm.

Case 5: As Indian languages are morphological rich and share more common words at root level due to same ancestry. So in this case, we are evaluating the effect of using root word instead of words while calculating similarity using the Token overlap algorithm discussed in **section 2.1** among Indian languages on ILCI dataset.

4 Results & Analysis

In **Case 1**, we are evaluating the effect of different algorithms on the similarity. **Figure 2a** shows the results corresponding to every language pair using the algorithm mentioned in section 2.1 (Token Overlap). Similarly **Figure 2b, 2c, 2d** represent the results corresponding to other algorithms discussed in **section 2.2, 2.3 and 2.4** respectively. Here, we observed a small amount of variation in similarity values with the different algorithms. However, more importantly, relative values within the similarity matrix remain almost constant even in the different algorithms, e.g., Hindi-Urdu has shown the most similarity in all algorithms. Our results confirm that this also holds for other language pairs.

For **Case2**, we evaluated whether common script selection matters in calculating language similarity or not. To do so, we also performed experiments by calculating similarity for all languages using the Abugida script. In these experiments, it can be seen that the scripts do not matter in calculating the similarity. We got similar results with both the Devanagari and Abugida scripts with a variation of 0.25%. This can be seen by comparing **Figure 4a** with **Figure 2c** and **Figure 4b** with **Figure 2d**.

Case 3 shows variation in similarity to the number of sentences we are using for calculating it. **Figure 3a** shows variation plots of similarity v/s No. of sentences used. We used overall 2000 sentences and observed that the similarity value gets stable by the end of the curve; in addition to observing that, we also see that the value does not vary much with larger sentences. We also performed experiments with 10k, 20k, 50k sentences; **Figure 3b, 3c and 3d** shows the plot corresponding to each case respectively. It can be seen there is not much fluctuation in the curve, even with the introduction of more sentences after 2k. Thus, we can say for calculating similarity, a small parallel data-set of 2k sentences is enough.

We can further see in **Case 4** that the similarity is not dependent on the nature of data, and is thus independent of external factors such as domain. **Figure 5a** and **Figure 5b** show the results corresponding to ILCI and PMI corpus, respectively. We observed that the similarity values might vary with the change in data-set, but the overall relative similarity matrix will remain constant. More clearly, if the similarity value of L1-L2 varies by a magnitude of k, then there will be a approximate change of k in the magnitude for the other language pairs. This can be confirmed by observing the results from the above experiments.

In the last **Case 5**, we observe from **Figure 6a** and **Figure 6b** that similarity increases drastically for the lexemes of all languages pairs. That is, if we ignore affixes and consider the root form of a word, we can notice that the similarity increases.

Also from the above experiments, we can conclude some general results as:

- Hindi and Urdu being the most similar and Tamil and Punjabi being least similar among Indian languages.
- Language similarity increases within the family, and it grows even more, when geographical distance is less. For example, Urdu-Punjabi's similarity is more than Urdu-Gujarati.
- Different Families also show some reasonable amount of similarity due to contact between them over a long time. For example, Telugu belongs to Dravidian family but it shows considerable similarity with Indo-aryan languages like Hindi and Marathi.
- Telugu from Dravidian and Marathi from the Indo-Aryan language family have more cross-family similarity than the others because they have geographical proximity thus exhibit greater lexical convergence.

5 Conclusion & Future Work

In this paper, we did an extensive study on similarity involving the languages of Indian subcontinent. We explored different factors that may affect the process of calculating similarity. Our results led to some interesting conclusions, such as how the relative order of similarity among languages remains same irrespective of the factors we considered, and how the maximum similarity is observed within pairs of the same language family and it increases more with geographic proximity. Thus, this study will help future research which focuses on exploiting the language relatedness for NLP tasks. Future work along these lines can focus on using semantic similarity alongside lexical similarity to increase accuracy.

References

Syed Sarfaraz Akhtar, Arihant Gupta, Avijit Vajpayee, Arjit Srivastava, and Manish Shrivastava. 2017. Word similarity datasets for indian languages: Annotation and baseline systems. In *Proceedings of the 11th Linguistic Annotation Workshop*, pages 91–94.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2014. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, pages 48–53.

Pushpak Bhattacharyya, Mitesh M Khapra, and Anoop Kunchukuttan. 2016. Statistical machine translation between related languages. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, pages 17–20.

Mamdouh Farouk. 2019. Measuring sentences similarity: a survey. *arXiv preprint arXiv:1910.03940*.

Barry Haddow and Faheem Kirefu. 2020. PMIndia – A Collection of Parallel Corpora of Languages of India. *arXiv e-prints*, page arXiv:2001.09907.

Aminul Islam and Diana Inkpen. 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):1–25.

Girish Nath Jha. 2010. The tdil program and the indian langauge corpora intitiative (ilci). In *LREC*.

Grzegorz Kondrak. 2005. N-gram similarity and distance. In *International symposium on string processing and information retrieval*, pages 115–126. Springer.

Anoop Kunchukuttan. 2020. The IndicNLP Library. `https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf`.

Anoop Kunchukuttan and Pushpak Bhattacharyya. 2020. Utilizing language relatedness to improve machine translation: A case study on languages of the indian subcontinent. *arXiv preprint arXiv:2003.08925*.

Kim S Larsen. 1992. *Length of maximal common subsequences*. Aarhus Universitet. Department of Computer Science.

V Levenshtein. 1965. Levenshtein distance.

William J Masek and Michael S Paterson. 1980. A faster algorithm computing string edit distances. *Journal of Computer and System sciences*, 20(1):18–31.

Raj Noel Dabre Prasanna. 2018. Exploiting multilingualism and transfer learning for low resource machine translation.

Debapriya Sengupta and Goutam Saha. 2015. Study on similarity among indian languages using language verification framework. *Advances in Artificial Intelligence*, 2015.

Kumar Shridhar, Harshil Jain, Akshat Agarwal, and Denis Kleyko. 2020. End to end binarized neural networks for text classification. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 29–34, Online. Association for Computational Linguistics.

Anil Kumar Singh and Harshit Surana. 2007. Using a single framework for computational modeling of linguistic similarity for solving many nlp problems. *EUROLAN 2007 Summer School Alexandru Ioan Cuza University of Iaşi*, page 46.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32.

Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095.