A Neural Lemmatizer for Bengali

Abhisek Chakrabarty, Akshay Chaturvedi, Utpal Garain
CVPR Unit, Indian Statistical Institute
203 B.T. Road, Kolkata-700108, India
abhisek0842@gmail.com, thegphenomenon@gmail.com, utpal@isical.ac.in

Abstract

We propose a novel neural lemmatization model which is language independent and supervised in nature. To handle the words in a neural framework, word embedding technique is used to represent words as vectors. The proposed lemmatizer makes use of contextual information of the surface word to be lemmatized. Given a word along with its contextual neighbours as input, the model is designed to produce the lemma of the concerned word as output. We introduce a new network architecture that permits only dimension specific connections between the input and the output layer of the model. For the present work, Bengali is taken as the reference language. Two datasets are prepared for training and testing purpose consisting of $19,159$ and $2,126$ instances respectively. As Bengali is a resource scarce language, these datasets would be beneficial for the respective research community. Evaluation method shows that the neural lemmatizer achieves 69.57% accuracy on the test dataset and outperforms the simple cosine similarity based baseline strategy by a margin of 1.37%.

Keywords: Bengali, lemmatization, neural network

1. Introduction

In this article, we propose a novel neural network based lemmatization method and show its effectiveness for a highly inflected language namely, Bengali. Bengali belongs to the group of major Indic languages and currently finds its place among the top 10 most popular languages in the world. The very first hurdle in computational processing of major Indic languages is the presence of various morphological variants of root words in the text. Existence of morphological richness in those languages creates problems for developing several text processing tasks like Natural Language Processing (NLP), Information Extraction (IE), Information Retrieval (IR) etc. For IR, stemming serves the purpose well (Majumder et al., 2007; Paik and Parui, 2011; Paik et al., 2011a; Paik et al., 2011b; Paik et al., 2013) but where semantic processing is needed (e.g. Word Sense Disambiguation (WSD), Machine Translation (MT) etc.), there lemmatization stands as a mandatory pre-processing module. The role of a lemmatizer is to map a surface word in a context to its root form. In the lexical resources, the roots of the language are listed with their syntactic and semantic descriptions. Hence, determining the appropriate lemma is crucial for knowing the meaning or other linguistic properties of a word in a raw text.

1.1. State of the Art

So far, there has been a little research conducted on developing efficient lemmatization algorithms for Bengali (Faridee et al., 2009; Loponen et al., 2013; Bhattacharyya et al., 2014). The Bengali morphological analyzer built by Faridee et al. (2009) is a rule based one and capable of lemmatizing only colloquial words in the language. Also, the lemmas for verbs produced by their method are basically stems, and consequently they cannot be linked to standard dictionaries. The work by Loponen et al. (2013) was primarily targeted for improving IR performance. Bhattacharyya et al. (2014) proposed a lemmatization technique for Indic languages, that initially organizes all dictionary root words in a trie structure and for an input surface word, produces a set of prospective roots from the trie. If the obtained set contains more than one candidate, some heuristics are applied to select the appropriate lemma from them. But, this technique does not take the contextual information of the word to be lemmatized into account as well as its evaluation strategy is non-deterministic. Recently, we developed a lemmatizer for Bengali (Chakrabarty and Garain, 2016) that, given a surface word, selects a set of potential roots from dictionary using the string distance measure defined by Majumder et al. (2007) and then finds the lemma from them using manipulation of frequent suffixes found in the language. However, this approach does not consider the named entities for lemmatization and depends on a list of valid suffixes. Additionally, there are two prerequisites of the algorithm: 1. a part of speech tagger in the language 2. a sense inventory of the root words containing their semantic definitions. Apart from these, there has been no significant research on lemmatization for Bengali.

1.2. Our Contribution

Recent advances in NLP research are mostly governed by the statistical machine learning (ML) based algorithms. To tackle the lemmatization problem from ML paradigm, the following challenges appear. If we consider it as a classification task, then the number of classes equals to the number of roots present in the language. As there are around 50k roots in Bengali, so the very first challenge is to design an efficient classifier for this huge classification problem. Next, preparing the training data to train the classifier needs extensive manual labour. In this present work, we rather model lemmatization as a task of lemma transduction. The problem is formulated as follows: given an input surface word along with the context, whether it is possible to generate the appropriate lemma of the concerned word. We employ neural network to achieve the objective. For handling the words in a neural net based framework, trans-

forming them in numerical form is obvious. So, the concept of distributed representation of words as vectors (Mikolov et al., 2013) is used to build our model. We use the FIRE Bengali news corpus[1], Wikipedia and Tagore's short stories collection[2] together to obtain the vector embeddings. Given a word along with its context represented in an array of vectors as input, a novel network architecture is designed in such a way so that the output vector should correspond to the appropriate lemma of the input surface word. Initially the network is trained on a set of <surface word with context, lemma> mapped pairs to learn its parameters and then the trained model is tested to find the lemma of unknown surface words. Throughout the experiment, we set 3 as the context size i.e. combination of the previous word, the surface word and the next word together constitutes the context. However, the proposed model can be extended to support arbitrary context length.
The paper is organized as follows. In the next section, we describe the neural lemmatization model. Method of preparing the datasets and experimental results are provided in Section 3. Section 4 concludes the paper.

2. The Proposed Model

Our proposed model is a feedforward neural network as shown in Figure 1. All words are handled by their corresponding vector embeddings and we use the word2vec tool (Mikolov et al., 2013) to obtain the 200-dimensional word vectors. For training of the recurrent neural model of word2vec, we use continuous bag of words (cbow) architecture and in the output layer of the model, negative sampling is used.

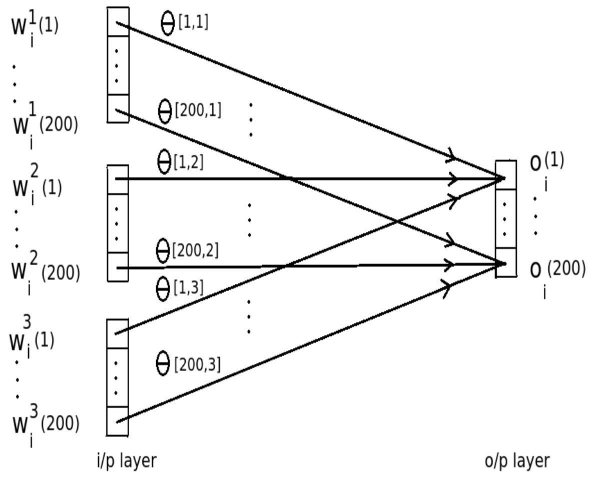

Figure 1: Architecture of the proposed neural lemmatization model.

Consider n training samples present in the training set and (w_i^1, w_i^2, w_i^3) be the i^{th} training sample representing the respective vectors of the preceding word, the target surface word and the succeeding word. Let o_i be the output for the i^{th} sample. For an arbitrary word vector w, its j^{th} dimension is denoted by $w(j)$. The input layer of the model accepts an array of three contextual word vectors and hence, consists of total 600 nodes and the output layer contains total 200 nodes corresponding to the transduced lemma vector. The network architecture is designed in a novel way. Instead of making all to all connections between the input and the output layer, we propose a dimension specific architecture of the network. Since every dimension of a word vector preserves a specific characteristic of the corresponding word, so we restrict that a particular dimension of the output would be attributed only from that same respective dimensions of the three input word vectors. That is, $\forall\, j = 1, \ldots, 200$, $o_i(j)$ is influenced by $w_i^1(j), w_i^2(j)$ and $w_i^3(j)$ only. So, there are total 600 (200 × 3) synaptic connections present in the network. Let $\theta \in \mathbb{R}^{200\times3}$ be the weight matrix where $\theta[j,k]$ is the weight of the connection connecting $o_i(j)$ and $w_i^k(j)$ $\forall\, j = 1, \ldots, 200$ and $\forall\, k = 1, 2, 3$. Here we use batch mode learning to obtain the optimum θ. To make the model work as a transducer, instead of choosing a bounded function like sigmoidal or hyperbolic tangent, we select the identity function for activation i.e.

$$o_i(j) = \sum_{k=1}^{3} w_i^k(j) \times \theta[j,k],\ \forall\, j = 1, \ldots, 200. \qquad (1)$$

Figure 1 shows the schematic diagram of the proposed network model. We train our model in the following way. Our main objective is to increase the cosine similarity between o_i and the desired word vector for i^{th} training example, say d_i. d_i corresponds to the vector of that particular dictionary root word which is the appropriate lemma for i^{th} training instance. We define the cost function for the i^{th} training example J_i as follows

$$J_i = 1 - cosine\_sim(d_i, o_i) \qquad (2)$$

where $cosine\_sim(d_i, o_i)$ denotes the cosine of the angle between d_i and o_i. The overall cost function J is the average of J_i over all n training examples. From equation (1) and (2) it can be derived that for the i^{th} training example,

$$\frac{\partial J_i}{\partial \theta[j,k]} = -\frac{w_i^k(j)}{\|d_i\|\|o_i\|}\left[d_i(j) - \frac{d_i \cdot o_i}{\|o_i\|^2}\,(o_i(j))\right] \qquad (3)$$
$$\forall\, j = 1, \ldots, 200\ ;\ \forall\, k = 1, 2, 3$$

where $d_i \cdot o_i$ denotes the dot product between d_i and o_i and $\|\bullet\|$ denotes the Euclidean norm operation. We minimise the cost function J using batch gradient descent with a learning rate of 0.1. $\theta[j,k]$ for the next iteration is updated by the following rule.

$$\theta_{next}[j,k] = \theta_{previous}[j,k] - \frac{0.1}{n}\sum_{i=1}^{n}\frac{\partial J_i}{\partial \theta[j,k]} \qquad (4)$$

The process to obtain the lemma of a test sample is as follows. If there are m root words $r_1, r_2, \ldots, r_m$ present in the dictionary and o is the output vector of the neural model for the input test sample, then choose r_i as the lemma where $r_i \leftarrow \underset{r_j}{\operatorname{argmax}}\, cosine\_sim(vec(r_j), o)\, \forall\, j \in \{1, \ldots, m\}$. The function vec is defined as follows. Taking a word as an argument, vec returns the corresponding embedded vector.

[1]http://fire.irsi.res.in/fire/data
[2]http://www.rabindra-rachanabali.nltr.org

Word	Context	Lemma
‘তার’/taar	(i) ইলেকট্রিক/electric ‘তার’/taar সুপরিবাহী/suparibaahii	‘তার’/taar
	(ii) এটা/eTaa ‘তার’/taar জিনিস/jinis	‘তাহার’/taahaar
‘কর’/kara	(i) কাজ/kaaj ‘কর’/kara তুমি/tumi	‘করা’/karaa
	(ii) তব/taba ‘কর’/kara পল্লব/pallaba	‘কর’/kara
‘কমল’/kamal	(i) সুগন্ধী/sugandhii ‘কমল’/kamal পুষ্প/puShpa	‘কমল’/kamal
	(ii) দাম/daam ‘কমল’/kamal জিনিসের/jiniser	‘কমা’/kamaa

Table 1: Examples showing the impact of context for lemmatization.

3. Experimentation

3.1. Datasets Preparation

For training purpose, a Bengali dataset is developed containing $19,159$ samples involving minimal human intervention. To evaluate the lemmatizer, a test dataset is also manually built having $2,126$ test instances, each of which consists of a surface word accompanied with its contextual neighbours. In Bengali, considering the context of a surface word is an important factor for lemmatization. Because, on varying contexts, a particular word may originate from different roots. Table 1 shows examples of three surface words (‘তার’/taar, ‘কর’/kara and ‘কমল’/kamal), two different contexts for each of them and the appropriate lemmas of the surface words for the respective contexts. Depending on the contexts, the lemmas get changed reflecting the impact of contextual information on lemmatization.
The process of creating the training dataset consists of two parts: 1. generating the training samples for which transformation rules from roots to inflections follow regular patterns (e.g. where simple addition/subtraction of a frequent suffix to/from a root produces a surface word) 2. generating the irregular samples for which no well-formed transformation rule works. In Bengali, usually a root word and its morphological variants are both orthographically and semantically similar (e.g. root; inflections: ‘করা’/karaa; ‘করার’/karaar, ‘করলাম’/karlaam). In cases of irregular samples, only semantic similarity exists between a lemma and its variants (e.g. root; inflections: ‘থাকা’/thaakaa; ‘ছিলে’/chhile, ‘ছিল’/chhila). To generate the regular samples, we follow the rationale that for suffixing languages like Bengali, “morphologically-related words typically share a long common prefix” (Paik et al., 2011a). Initially, all the roots having length[3] greater than or equal to 6, are mined from the Bengali digital dictionary available with the Chicago university[4]. Next, for every root word, its 10 nearest words in the vector space are considered (cosine similarity is taken as the distance measure) and among them, only those are selected for which the overlapping prefix length with the root is atleast 6. Following the process, we are able to generate a set of $<$token; lemma$>$ mapped pairs which mutually have very high syntactic and semantic association and thus, these mappings are mostly context invariant. To generate the contexts of the inflections in the mapped set, we search for their instances in the FIRE corpus and when an instance is found, the surrounding context is extracted. There are $9,536$ regular samples in the training dataset. To generate the irregular samples, at first 120 different irregular word forms are crafted and out of them, those ones which have association with fixed lemmas irrespective of context, are spotted and their contexts are randomly picked up from the FIRE corpus. The particular word forms that have varying lemmas depending on the context, are handled specially and their combinations with different context-lemma pairs are manually arranged. In this way, we get 9,623 irregular samples. Both the regular and the irregular samples are put in the training data with almost equal proportion to make the dataset balanced. We prepare the test dataset by taking samples randomly from the FIRE corpus. Each sample in the test data consists of 3 adjacent words appearing in the text where the middle one is the target word for lemmatization. For all the samples, the respective gold lemmas of the target words are also provided.

[3]Length of a word is measured in terms of the number of unicode character/s present in that word.

[4]http://dsal.uchicago.edu/dictionaries/list.html

	Accuracy
Proposed Lemmatizer	69.57%
Baseline Method	68.20%

Table 2: Lemmatization accuracy.

3.2. Results

We use 10-fold cross validation on the training data to obtain the optimum θ. The proposed lemmatization method is evaluated on the test dataset by computing the direct accuracy i.e. the fraction of the total number of surface words which are correctly lemmatized. To choose a suitable baseline, we calculate the respective cosine similarities of the embedded vector of every surface word in the test dataset with the vectors of all dictionary root words and select that root as the lemma for which the corresponding vector possesses maximum similarity with the surface word vector. Table 2 presents the experimental results. Our proposed neural lemmatizer beats the simple cosine similarity based baseline by a 1.37% margin out of $2,126$ instances i.e. 29 more instances are correctly lemmatized by our method.

4. Conclusion

In this article, a novel neural model for lemmatization is presented and it is evaluated on Bengali. Due to the supervised nature of the proposed method, two Bengali datasets have been prepared for training and testing purpose. As

Bengali is a resource poor language, these two datasets will help to advance the research in Bengali NLP. Although we agree that the present work needs a token-lemma tagged training corpus which is also the requirement of the state of the art lemmatization algorithms (Gesmundo and Samardžić, 2012; Müller et al., 2015) for languages like English, Spanish, German, Hungarian, Czech etc. but porting them for the inflected Indic languages is a major challenge. The modular log linear model proposed by Müller et al. (2015) sets the new state of the art in token-based statistical lemmatization but it needs sufficiently large training corpora. Also, exploring the suitable features to be integrated into their model for lemmatizing Indic languages is another research issue. Regarding the architecture of our proposed model, we could have used a more complicated network topology exploiting the inter-relationships among neighbouring dimensions of the word vectors, but that would require larger training data. In addition, it will need considerably more training time. The future direction of this work would drive us to compare the performance of the proposed lemmatization method with the state of the art lemmatization algorithms as well as investigate the impact of different network architectures on lemmatization accuracy.

5. Acknowledgement

We sincerely acknowledge Arjun Das for his support in preparing the Bengali resources.

6. Bibliographical References

Bhattacharyya, P., Bahuguna, A., Talukdar, L., and Phukan, B. (2014). Facilitating multi-lingual sense annotation: Human mediated lemmatizer. In *Global WordNet Conference*.

Chakrabarty, A. and Garain, U. (2016). Benlem (a bengali lemmatizer) and its role in wsd. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 15(3):12:1–12:18, February.

Faridee, A. Z. M., Tyers, F. M., et al. (2009). Development of a morphological analyser for bengali. In *Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation*, pages 43–50. Universidad de Alicante. Departamento de Lenguajes y Sistemas Informáticos.

Gesmundo, A. and Samardžić, T. (2012). Lemmatisation as a tagging task. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 368–372. Association for Computational Linguistics.

Loponen, A., Paik, J. H., and Järvelin, K. (2013). Uta stemming and lemmatization experiments in the fire bengali ad hoc task. In *Multilingual Information Access in South Asian Languages*, pages 258–268. Springer.

Majumder, P., Mitra, M., Parui, S. K., Kole, G., Mitra, P., and Datta, K. (2007). Yass: Yet another suffix stripper. *ACM Trans. Inf. Syst.*, 25(4):18:1–18:20, October.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Müller, T., Cotterell, R., Fraser, A., and Schütze, H. (2015). Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274, Lisbon, Portugal, September. Association for Computational Linguistics.

Paik, J. H. and Parui, S. K. (2011). A fast corpus-based stemmer. *ACM Transactions on Asian Language Information Processing (TALIP)*, 10(2):8:1–8:16, June.

Paik, J. H., Mitra, M., Parui, S. K., and Järvelin, K. (2011a). Gras: An effective and efficient stemming algorithm for information retrieval. *ACM Trans. Inf. Syst.*, 29(4):19:1–19:24, December.

Paik, J. H., Pal, D., and Parui, S. K. (2011b). A novel corpus-based stemming algorithm using co-occurrence statistics. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 863–872. ACM.

Paik, J. H., Parui, S. K., Pal, D., and Robertson, S. E. (2013). Effective and robust query-based stemming. *ACM Trans. Inf. Syst.*, 31(4):18:1–18:29, November.