Neural Morphological Disambiguation Using Surface and Contextual Morphological Awareness

Akhilesh Sudhakar
IIT (BHU), Varanasi, India
akhileshs.s4@gmail.com

Anil Kumar Singh
IIT (BHU), Varanasi, India
nlprnd@gmail.com

Abstract

Morphological disambiguation, particularly for morphologically rich languages, is a crucial step in many NLP tasks. Morphological analyzers provide multiple analyses of a word, only one of which is true in context. We present a language-agnostic deep neural system for morphological disambiguation, with experiments on Hindi. We achieve accuracies of around 95.22% without the use of any language-specific features or heuristics, which outperforms the existing state of the art. One contribution through this work is building the first morphological disambiguation system for Hindi. We also show that using phonological features can improve performance. On using phonological features and pre-trained word vectors, we report an accuracy of 97.02% for Hindi.

1 Introduction

Morphologically inflected words are derived from a root by modifying it (e.g., by applying prefixes, suffixes and infixes) based on linguistic features (manifested as the inflection tagset). Morphological analysis involves extracting this root word and the set of features that describe the inflected form. These analyses contain syntactic and semantic information about inflected words. Table 1 shows an example for the Hindi word 'पूरे' [pUre][1]. Existing morphological analyzers typically work in isolation, meaning that they generate multiple analyses of a word, purely based on surface structure. For many NLP tasks like machine translation and topic modeling, however, it is essential to know which morphological analysis is correct in the context of the sentence. Morphological disambiguation aims to solve this problem. The task of disambiguation is non-trivial and is complicated by the dependencies of the correct analysis on the surface structure of the inflected word, on the surface structures of the neighboring words, and on the analyses of neighboring words.

We present a deep neural morphological disambiguation system that leverages context information as well as surface structure. While we have experimented on Hindi in our work, we report accuracies without employing any language-specific features to show that our system can generalize across different languages. We also show performance boost by using phonological features and pre-training of word vectors. To the best of our knowledge, this is the first ever non-naive morphological disambiguation system to be built for Hindi.

Like other Indo-Aryan languages, Hindi is morphologically rich and a word form may have over 40 morphological analyses (Goyal and Lehal, 2008). Though the inflectional morphology of Hindi is not agglutinative, the derivational suffixes are. This leads to an explosion in the number of inflectional root forms (Singh et al., 2013). One of the reasons for our focus on Hindi is that it has a wide coverage of speaking population, with over 260 million speakers across 5 countries[2] and is the fifth most spoken language in the world[3]. We present four neural architectures for this task, each different from the others by the nature of context information used as disambiguating

[1]We use the Roman notation popularly known as WX for representing Hindi words

[2]https://www.ethnologue.com/statistics/size

[3]The exact rank may be a matter of debate due to the socio-linguistic scenario in South Asia, with some surveys claiming it to be even more popularly spoken.

Root	Category	Gender	Number	Person	Case	TAM	Suffix
pUrA	adj	m	sg	-	o	-	-
pUrA	adj	m	pl	-	d	-	-
pUrA	adj	m	pl	-	o	-	-
pUrA	n	m	pl	3	d	0	0
pUrA	n	m	sg	3	o	0	0
pUra	v	any	sg	2	-	ए	e
pUra	v	any	sg	3	-	ए	e
pUra	v	m	pl	any	-	या	yA

Table 1: Morphological analyses of the word ‘पूरे’ [pUre] (A ‘-’ indicates that the feature is not applicable and an ‘any’ indicates that it can take any value in the domain for that feature)

evidence. We assess our results by implementing an existing state-of-the-art system (Shen et al., 2016) on our Hindi dataset. Our system outperforms this state-of-the-art system.

2 Related Work

There is very little directly corresponding previous work on morphological disambiguation and it cannot be formulated in the same way as POS tagging. This is because the number of classes is fixed in POS tagging, whereas it is variable in our problem. Still, since part of speech (POS) tagging is a closely related task, the work on POS tagging can also provide useful insights. However, morphological disambiguation is a harder task to perform than POS tagging. The earliest approaches to POS tagging were rule-based (Karlsson et al. (1995), Brill (1992)) and required a set of hand-crafted rules learnt from a tagged corpus. More recently, Kessikbayeva and Cicekli (2016) present a morphological disambiguation system using rules based on disambiguations of context words.

Statistical approaches are also used for morphological disambiguation. Hakkani-Tür et al. (2000) propose a model based on joint conditional probabilities of the root and tags. Sak et al. (2007) use a perceptron model, while other statistical models use decision trees as by Görgün and Yildiz (2011). Hybrid approaches have also been tried, with Orosz and Novák (2013) using an approach combining rule-based and statistical approaches, to prune grammar-violating parses.

The use of deep learning for morphological disambiguation, has been explored. Straka and Straková (2017) build a neural system for tasks such as sentence segmentation, tokenization and POS tagging. Plank et al. (2016) build a multilingual neural POS tagger. While we draw insights from works on tasks such as POS tagging, we bear in mind that POS tagging and morphological disambiguation are significantly different. Morphological disambiguation is more complex because it works with multiple categories and not just part-of-speech. This introduces sparseness in the model, as well as considerations of whether the different categories are dependent on each other, on how to combine classifiers for each category, etc. The number of analyses for a word also varies.

Yildiz et al. (2016) propose a convolutional neural net architecture, which takes context disambiguation into account. Shen et al. (2016) use a deep neural model with character-level and well as tag-level LSTMs to embed analyses. Our work shares certain aspects in common with theirs but is different in many ways. We experiment on Hindi (which has significantly different morphological properties from the three languages they explore), use different neural structures, show the effect of language specific phonological features and study the impact of unsupervised pre-training of embeddings under different settings. Further, as mentioned earlier, we consider their results to be state-of-the-art because theirs is a language-agnostic system which gives state-of-the-art results on all 3 languages they have experimented on. We show that our model gives better performance than an implementation of their best model on Hindi.

3 Neural Models

We present four models for morphological disambiguation. Some aspects are common among them. They all use a deep neural network, which, given the current word in consideration and one of the candidate morphological analyses of the word, acts as a binary true/false classifier. A final softmax layer outputs probabilities for correct and incorrect, based on whether the candidate analysis is correct or not. An ideal classifier would predict the probability of correct as 1 and incorrect as 0 for the correct morphological analysis of the word. As is usual in word sense disambiguation, we make the 'one sense per collocation' assumption (a word in a particular context has only one correct morphological analysis), with which our dataset is in accordance. The choices of neural architectures used by us are influenced by the findings in the work of Heigold et al. (2016), in which the authors conclude that on morphological tagging tasks, different neural architectures (CNNs, RNNs etc.) give comparable results, and careful tuning of model structure and hyperparameters can give substantial gains. We also draw insights from their work on augmenting character and word-level embeddings.

3.1 Terminology Used

For each of 'category', 'gender', 'number', 'person', 'case', 'TAM' and 'suffix', we use the term ***'feature'***. We call each of the values of a feature for a particular word, a ***'tag'***. For instance the feature 'gender' can have tags 'M (male)', 'F (female)' and 'N (neuter)'. The root and the tagset together make up a morphological analysis for a word. We use the term ***'candidate analysis'*** to refer to each of the morphological analyses generated by the analyzer for a given word.

3.2 Broad Basis for the Architectures

We first establish an intuitive and statistical foundation to justify our decision choices in building the deep neural network. We extract dependencies between roots and features from the work by Hakkani-Tür et al. (2000), noting that the assumptions used for Turkish by the authors hold good for Hindi too. We also obtain surface-information-related dependencies from the work by Faruqui et al. (2016). The following is the full set of extended dependencies:

- **Dependency #1**: The root of a word depends on the roots as well as the features of all previous and following words in the window
- **Dependency #2**: Each feature of a word depends on the roots and the features of all previous and following words in the window
- **Dependency #3**: Each feature of a word depends on the root of the current word, as well as all other features of the current word
- **Dependency #4**: The root and each feature of a word depend on the surface form of the word
- **Dependency #5**: The root and each feature of a word depend on the surface forms of the word as well as those of all previous and following words in the window.

In all these four models, the following network components are also consistent.

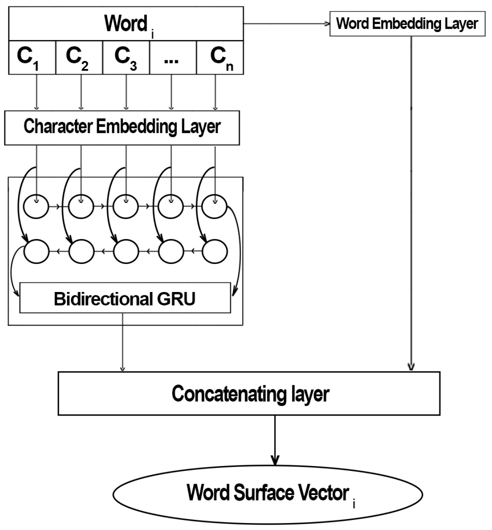

Figure 1: Architecture for the word surface vector. 'i' indicates the i^{th} input word.

3.3 Word Input

Word inputs to the network are embedded at two levels. A word embedding vector is generated using the word as a whole. Each character in the word is also embedded in a character embedding vector and these character embeddings are fed, in sequence, to a bidirectional GRU. The output vector of the GRU and the word embedding vector are concatenated together to form the ***'word surface vector'*** that takes into account surface features of the word. The part of the network that generates the word surface vector is shown in Figure 1.

The character-level GRU allows for capturing of surface properties of a word, and takes into account Dependency #4 (section 3.2). We obtain marginal accuracy gains (of around 0.1%) by using a GRU instead of an LSTM at the character-level.

3.4 Candidate Analysis Input

Candidate analysis inputs to the network are treated as two inputs: the root word and rest of the tags. The root is treated in the same exact fashion as the word inputs (for the same reasons mentioned in the above section) with the only difference being that all words share a common embedding layer, while all roots share a separate common embedding layer. Consequently, a corresponding ***'root surface vector'*** will be generated as described for each root input. Tagsets are represented as binary vectors, with positional encoding. The root surface vector and all the tag encodings are treated as a sequence and fed as input to a bidirectional LSTM. This design choice, including the bidirectionality, has been made to address Dependency #3 (section 3.2). We call the output vector of this LSTM, the ***'root features sequence vector'***. The part of the network architecture that generates the root features sequence vector is shown in Figure 2.

3.5 Hyperparameters and Training

All GRUs and LSTMs have a hidden layer size of 256, and deep GRUs and LSTMs have a depth of 2 layers. Deep convolutional networks have a filter width of 3, hidden layer size of 64 and depth of 3 layers. The model can run for 10,000 epochs but we make use of early stopping with a patience of 10 epochs in order to keep the generalization error in check. This is a validation-based early stopping on the development set. The word and root embedding layers have a dimension of 100, while the character embedding layer has a dimension of 64. All models use the categorical cross-entropy loss function and the Adam optimization method as proposed by Kingma and Ba (2014). The sequence of words in each

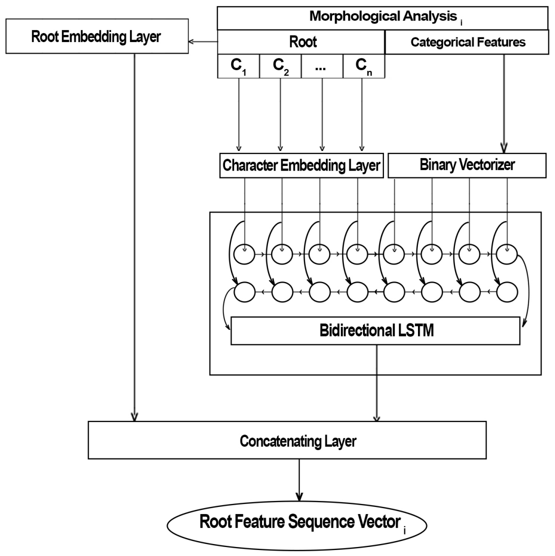

Figure 2: Network architecture for generating the root feature sequence vector. The subscript i indicates the morphological features of the i^{th} input word. These might be candidate analyses or correct analyses, depending on where (training or testing) they are used in other figures.

sentence act as a mini-batch during training. The best model is saved for predictions on the test data.

3.6 Baseline Model

As mentioned earlier, since there does not exist a non-naive state of the art system for Hindi, we use a low baseline model that picks one candidate analysis at random and predicts this to be the correct morphological analysis for the given word. This[4] is the default for building several machine translation (MT) systems, such as the Sampark[5] system, for Indian languages. The most that is currently done for these MT systems is to apply some agreement based rules[6]. Proper evaluation of these modules may be needed, but it is beyond the scope of this paper. It must be mentioned here that the baseline we have picked is a weak baseline. However, we have done so for a couple of reasons. Firstly, we compare our results to the state-of-the-art system mentioned earlier and show performance gain. Therefore it is not a case of inflation of results using a weak

[4]The so called 'pick one morph' module.

[5]`http://sampark.org.in`

[6]The 'guess morph' module.

baseline. Secondly, we have presented 4 models which show a gradation of performance on this task (as shown in Table 5). Thirdly, the baseline presents the case when absolutely no character, word or context-level information is available to the model. We believe that since we study the impact of each of these kinds of knowledge on the models' performance, we must also study the case when none of this knowledge is available.

3.7 Model 1

This model solely relies on the surface information of the current word to make predictions about its morphological analysis. Given a current word and the root and tags of the candidate analysis, the model uses only the word surface vector (section 4.2) with the root features sequence vector (section 4.3) of the candidate to make predictions. Figure 3 shows the exact structure of the network used.

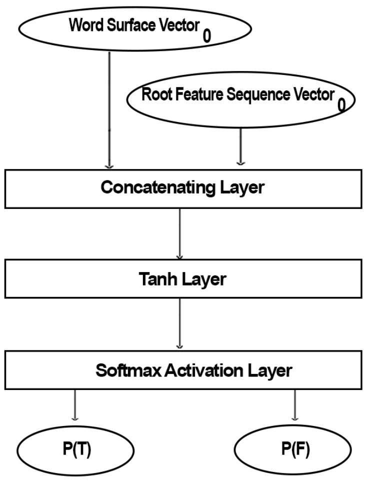

Figure 3: Structure of Model 1. The subscript '0' indicates the current word. P(T) and P(F) indicate probability of True and False respectively.

3.8 Model 2

This model makes use of not only the current word's surface information but also the surface information of all words in a window which has 4 words to the left and 4 words to the right of the current word in the sentence. (We experiment with values from 2 to 6). Building the model this way accounts for Dependency #5 (section 3.2). This model uses out-of-sentence tokens too, to ensure that words towards the beginning and end of the sentence also have a full window. The word surface vectors (section 4.2) of all the words in this window, along with the root features sequence vector (section 4.3) of the candidate are fed into a bidirectional LSTM in this model. Figure 4 shows the exact network structure used.

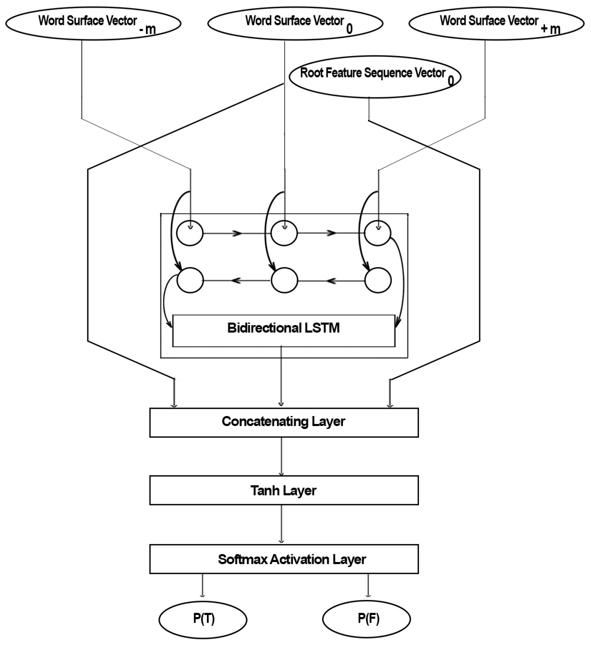

Figure 4: Model 2. The subscript '0' indicates the current word, the subscript '-m' represents any word in the left context of the current word and the subscript '+m' represents any word in the right context of the current word. P(T) and P(F) indicate probability of True and False respectively.

3.9 Model 3

This model makes use of not only the current word's surface information and the surface information of all words in a window which has 5 words to the left of the current word in the sentence, but also the correct morphological annotations of the words in this left-context. This model partially accounts for Dependency #1 and Dependency #2 (section 3.2). The word surface vector (section 4.2) of each word is concatenated with the root features sequence vector (section 4.3) of the word to give a 'complete vector'. Complete vectors, each concatenated with their own convolutions are fed as inputs to a deep LSTM. Model 3 uses the network structure shown in Figure 5, except that Figure 5 shows the model using the current word's left and right context, while Model 3 uses only its left context.

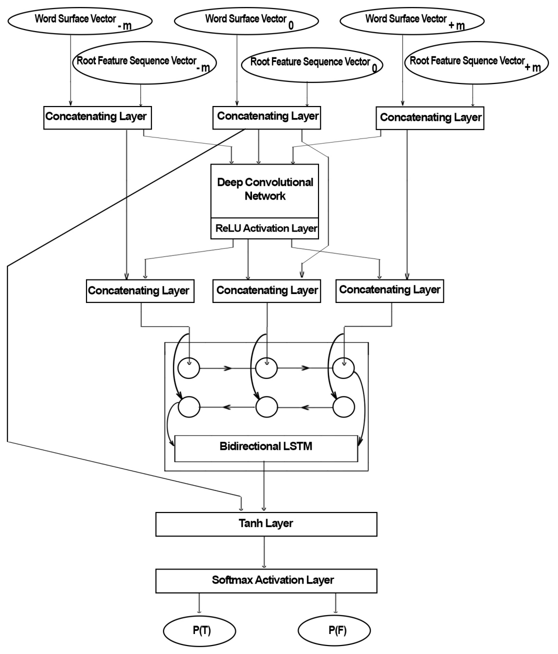

Figure 5: Model 4. Model 3 also has a similar configuration except for the context of the current word. The subscript '0' indicates the current word, the subscript '-m' represents any word in the left context of the current word and the subscript '+m' represents any word in the right context of the current word. P(T) and P(F) indicate probability of True and False respectively.

3.10 Model 4

Model 4 is similar to Model 3, except that this model makes use of surface information and correct morphological annotations of not only words to the left but also to the right of the current word. The complete window for the current word has 4 words to the left and 4 words to the right of it. (We experiment with values from 2 to 6). This model takes into account all the dependencies mentioned in section 3.2. Figure 5 shows the network structure for Model 4.

It must be mentioned here that Model 3 and Model 4 are slightly complex due to the presence of a CNN and further concatenation of the convolved inputs with the original inputs. We have conducted experiments on simpler versions of Model 3 and Model 4 but with poorer results (an average drop in accuracy of 1.2%). An elaborate discussion of these results is not possible due to space constraints. Specifically, we used the same context as used in these two models but did not use the intermediary CNN in these simpler experiments. The reasons for the performance improvement upon using a CNN could be that CNNs have proved to be particularly useful for classification tasks on data that has the property of local consistency. This is evident from previous work on using CNNs for similar classification tasks such as those by Collobert et al. (2011), Yildiz et al. (2016) and Heigold et al. (2016). Well-formed sentences of any language (Hindi, in our case) display local consistency because they have a natural order, with context words forming abstractive concepts/features. Hence, a CNN was an intuitive choice for our task.

4 Experimental setup

4.1 Dataset

We use a manually annotated Hindi Dependency TreeBank[7], which is part of the Hindi-Urdu Dependency TreeBank (HUTB)[8] as the source of the correct morphological analysis of words in the context of their sentences. The treebank annotates words from sentences taken from news articles and textual conversations. Each word in every sentence of the treebank is annotated with the correct morphological analysis. We use a morphological analyzer for Hindi[9], which was developed earlier, but is now used for the Sampark MT system and other purposes. We use it to generate the different possible morphological analyses for each word (in isolation) in the treebank. For each word in the treebank, the candidate analysis matching the word's treebank-annotated morphological analysis is labeled as true while all other candidate analyses are labeled as false. In the entire dataset, we ensure that out of all the candidate analyses of a word, there is one that matches the treebank annotation for that word. Table 2 provides specific statistics about the dataset used. Table 3 describes the features of a morphological analysis as well as provides the domain of possible tags (values) for each feature.

[7] http://ltrc.iiit.ac.in/treebank_H2014
[8] http://verbs.colorado.edu/hindiurdu
[9] http://sampark.iiit.ac.in/hindimorph

Attribute	Count
Total words	298,285
Unique words	17,315
Manual additions of treebank annotation	115432
Ambiguous words	179,453
Unambiguous words	118,742
Sentences in treebank	13,933
Mean sentence length	21.40
Mean morphological analyses per word	2.534
Mean morphological analyses per ambiguous word	3.550
Standard deviation of morpho--logical analyses per word	1.620
Maximum morphological analyses for a word	10

Table 2: Dataset statistics

Feature name	List of possible tags
Root	Not fixed
Category	Noun(n), Pronoun(pn), Adjective(adj) verb(v), adverb(adv) post-position(psp), avvya(avy)
Gender	Masculine(m), Feminine(f), Neuter(n)
Number	Singular(sg), Plural(pl), Dual(d)
Person	1st Person(1), 2nd Person(2), 3rd Person(3)
Case	Direct(d), Oblique(o)
TAM	है,का,ना,में,या,या1,से,ए, कर,ता,0,था,को,गा,ने
Suffix	kA,e,wA,WA,yA,nA,ko,ne, 0,kara,gA,yA1,meM,se,hE

Table 3: Domain of tags (values) for each feature. The tag 'TAM' denotes the tense, aspect and modality marker.

We would like to mention here that since our system is built to pick the correct analysis from the morphological analyses generated by the analyzer, it assumes that every word has a set of candidate analyses. In the context of our task, it is not relevant to discuss the case when the morphological analyzer itself fails to provide candidate analyses.

Table 4 shows the number of ambiguous words and the total number of words used for training, development and testing. The test set is held-out and is used solely for reporting final results. The development set is used to validate and tune model hyperparameters. During testing, the model is provided with only the different candidate morphological analysis outputs from the morphological analyzer, for each word. The correct analyses (also referred to as annotations in the following section) provided by the treebank for each word are used to calculate the reported accuracies.

Phase	Ambiguous Words	Total Words
Training	149,540	248,572
Development	11,963	19,885
Testing	17,950	29,828

Table 4: Word counts in training, development and test splits

4.2 Methods of Testing

Models that use morphological analyses of the context words (Models 3 and 4) have access to correct annotations of these contexts during training and validation but not during testing. During testing, in order to provide 'correct annotations' of words in the context of a word, to these models, we use Model 2 or Model 3. This is because Model 2 does not itself use context annotations and Model 3 itself uses only left context annotations (it can hence, predict the correct morphological analysis for each word in the test data from left to right, treating the predictions of the previous words as the correct annotations of the left-context of the words that occur next). However, in the case where Model 3 is being used as the test set annotator, a strategic choice has to be made for annotation. A greedy strategy would pick the morphological analysis with the highest softmax probability of being the correct annotation and annotate the word with this annotation. However, the greedy strategy fails if the model makes mistakes towards the start of a sentence or performs poorly on only some types of words, because these errors propagate to every consecutive word and get compounded. In order to avoid these kinds of errors, we use a beam search with width 10 for pre-annotating the test set in the case of context-based models.

5 Results and Analysis

Table 5 presents performance accuracies of different models and with different methods used to annotate test data in the cases where an initial pre-annotation of test data is needed, as discussed in the previous section. As mentioned before, model accuracy is calculated by comparing each trained model's predictions on

Model	Pre-testing Test Data Annotation Model	Accuracy on Ambiguous Words	Accuracy on All Words
Baseline	NA	29.40	38.43
S-O-T-A	S-O-T-A	90.13	92.06
Model 1	NA	80.21	83.96
Model 2	NA	87.35	89.18
Model 3	Treebank	93.82	96.17
Model 3	Model 2	89.40	93.35
Model 3	Model 3	91.23	94.90
Model 4	Treebank	**94.77**	**97.59**
Model 4	Model 2	90.41	94.74
Model 4	Model 3	92.65	95.22

Table 5: Performance of different models (all accuracies are percentages). S-O-T-A stands for state-of-the-art, which is the full-context model of Shen et al. (2016). The accuracy gain we have achieved over S-O-T-A is 2.8% on ambiguous words and 3.43% on all words.

the test data with the correct analyses from the treebank data, regardless of which model was used for the initial test annotation (if any). The standard measure for accuracy is used:

$$\frac{\#\ of\ correct\ disambiguations}{total\ \#\ of\ words\ in\ test\ set}$$

For practical purposes, the best performing system is the last row in Table 5, i.e., in which Model 4 uses Model 3 for pre-annotating the test set (though the 7th row has the highest accuracy, we cannot assume treebank annotations on the test data as well). Table 5 also presents the results of using the existing state of the art (S-O-T-A) model on our Hindi dataset. We have used the best performing model (on our Hindi dataset) proposed by Shen et al. (2016), the full-context model, as the state of the art. The accuracy gain we have achieved over state of the art is 2.8% on ambiguous words and 3.43% on all words.

We suggest possible reasons for the observed performance behavior in table 5. Typologically, Hindi is a Subject-Object-Verb, head-final language and uses post-positional case marking. This means that on an average, words show disambiguation dependencies on the words following them. However, there is also disambiguation evidence for a word to be gained from its left context. For instance, adverbs usually occur before (to the left of) the verb or object they refer to. Similarly, relative clauses, adjectives and articles are written before the noun they refer to. Model 4 uses the morphological analyses of the right-context of a word as well as the left context and hence is able to leverage information from both preceding and following words. Hence it is able to achieve better performance than Model 3. Models 2 and 1 do not leverage evidence about the morphological analysis of the words in the window and perform worse than the other two models. This shows (as is also quite intuitive) that the morphological analysis of the context is far stronger evidence in disambiguating a word, than just the surface forms of the words and its context. Model 2 performs better than Model 1 as it has access to the surface forms of the surrounding words, which in turn provide some level of knowledge about their inflected properties.

From control experiments, we conclude that our gain over the state of the art is due to factors that include careful tuning of hyper-parameters, increasing model complexity and leveraging the strength of combining models. At the end of section 3.10, we have already described the advantage of using a CNN. The existing state of the art does not leverage this advantage. Further, in allowing Model 3 to pre-annotate the test data, we have allowed our full-context model to take advantage of the strengths of a left-to-right model, which is also something that the existing state of the art does not explore.

5.1 Language-specific Enhancements

While the reported results in Table 5 are obtained without using pre-training of word vectors or phonological features, we also experimented with using these enhancements. We present results on the experimental setup where we train using Model 4 and pre-annotate the test set using Model 3. All performance improvements are reported as those obtained over and above the performance of this particular experiment setup.

5.1.1 Pre-training of Word Vectors

We pre-trained word embeddings using the word vector representation methods proposed by Bojanowski et al. (2016). This method makes use of an unsupervised skip-gram model to generate word vectors of dimension 100. We used an augmented corpus comprising of Wikipedia text dump for Hindi, as well as

Model	Accuracy	Accuracy gain over Baseline (%)
Baseline	38.43	0
Model 4	95.22	147.74
Model 4 + Pre-training	96.64	151.47
Model 4 + Phonological Features	96.04	149.91
Model 4 + Pre-training + Phonological Features	**97.02**	**152.46**

Table 6: Performance with language-specific enhancements

the collection of news articles and conversations that the Hindi treebank annotated words come from. Using vector pre-training gave us an accuracy improvement of 1.42%. One of the main reasons for the performance boost obtained during pre-training could perhaps be that the pre-trained word vectors capture syntactic and morphological information from short neighbouring windows.

5.1.2 Use of Phonological Features

Morphology interacts closely with phonology and there is ample work on the phonology-morphology interface (Booij, 2007). It is quite intuitive, therefore, to use phonological features (Chomsky and Halle, 1968) for a morphological problem. Besides, Hindi is written in the Devanagari script, in which the mapping from letters to phonemes is almost one to one. Each letter can therefore be represented as a set of feature-value pairs, where the features are phonological features such as type (whether consonant or vowel), place, manner etc. (Singh, 2006). This is true for almost all languages that use Brahmi-derived scripts. Phonological features are incorporated into the model by concatenating them with the character-level embeddings for words. We observe a performance enhancement of 0.82% upon using these phonological features.

Employing pre-training as well as phonological features boosted our model's performance from 95.22 % to 97.02%. These enhanced results are summarized in Table 6.

6 Future Work

We plan to test all our models on different languages and analyze which models perform best on each language and hope to be able to correlate these results with the linguistic phono-morphological properties of the languages. We will also try out this model in the Sampark[10] machine translation system to evaluate the effect it has on translation.

Recently, an attention-based machine translation model was proposed by Bahdanau et al. (2014) that defines a selective context around a word rather than a fixed window for all words. Models 3 and 4 can be modified to use an attentional mechanism based on the context words' positional and morphological properties. This would allow these models to increase their range of information-capturing across words in the sentence, without losing information due to propagation in a recurrent unit running across a large window. Experiments have been done in the past for morphological disambiguation using Conditional Random Fields (CRFs). It might be interesting to see the hybrid use of CRF models with the models we propose.

7 Conclusion

We propose multiple deep learning models for morphological disambiguation. We show that the model that makes use of morphological information in both the left and right context of a word performs best on this task, at least in the case of Hindi. We also study the effect of different context settings on model performance. The differences in performance obtained using these different context settings, we believe, follows from the typological and morphological properties of the language. Hence, we also believe that different languages may work better with different models that we propose. The use of phonological features enhances the quality of predictions by these models, at least in the case of Hindi.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473.*

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vec-

[10]`https://sampark.iiit.ac.in/sampark/web/index.php`

tors with subword information. *arXiv preprint arXiv:1607.04606*.

Geert Booij. 2007. *The interface between morphology and phonology*. Oxford University Press.

Eric Brill. 1992. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, ANLC '92, pages 152–155, Stroudsburg, PA, USA. Association for Computational Linguistics.

Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English*. Harper & Row, New York.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proc. of NAACL*.

Onur Görgün and Olcay Taner Yildiz. 2011. A novel approach to morphological disambiguation for turkish. In *Computer and Information Sciences II*, pages 77–83. Springer.

Vishal Goyal and Gurpreet Singh Lehal. 2008. Hindi morphological analyzer and generator. In *Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference on*, pages 1156–1159. IEEE.

Dilek Z Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2000. Statistical morphological disambiguation for agglutinative languages. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 285–291. Association for Computational Linguistics.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2016. Neural morphological tagging from characters for morphologically rich languages. *arXiv preprint arXiv:1606.06640*.

Fred Karlsson, Atro Voutilainen, Juha Heikkila, and Arto Anttila, editors. 1995. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Walter de Gruyter & Co., Hawthorne, NJ, USA.

Gulshat Kessikbayeva and Ilyas Cicekli. 2016. A rule based morphological analyzer and a morphological disambiguator for kazakh language. *Linguistics and Literature Studies*, 4(1):96–104.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

György Orosz and Attila Novák. 2013. Purepos 2.0: a hybrid tool for morphological disambiguation. In *RANLP*, volume 13, pages 539–545.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529*.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2007. Morphological disambiguation of turkish text with perceptron algorithm. *Computational Linguistics and Intelligent Text Processing*, pages 107–118.

Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell, and Chris Dyer. 2016. The role of context in neural morphological disambiguation. In *COLING*, pages 181–191.

Pawan Deep Singh, Archana Kore, Rekha Sugandhi, Gaurav Arya, and Sneha Jadhav. 2013. Hindi morphological analysis and inflection generator for english to hindi translation. *International Journal of Engineering and Innovative Technology (IJEIT)*, pages 256–259.

Anil Kumar Singh. 2006. A computational phonetic model for indian language scripts. In *Constraints on Spelling Changes: Fifth International Workshop on Writing Systems. Nijmegen*, Nijmegen, The Netherlands.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99.

Eray Yildiz, Caglar Tirkaz, H. Bahadir Sahin, Mustafa Tolga Eren, and Ozan Sonmez. 2016. A morphology-aware network for morphological disambiguation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2863–2869. AAAI Press.