CS365A : ARTIFICIAL INTELLIGENCE
Language Learning from Commentaries in Videos using Dynamic NLP
Guide: Prof. Amitabha Mukherjee

Anusha Chowdhury , Kaviti Sai Saurab

Indian Institute of Technology, Kanpur
March-April, 2015

Abstract. In this project we use different techniques of dynamic NLP to make a system learn a new language. There is already some literature on how little children learn a new language and it is not easy to simulate this for an agent. The system has to first learn new words in the language(it uses contrastive association), followed by the patterns in which the words occur in a sentence(it uses Adios), which basically comes under syntax learning. We demonstrate the process of learning a new language from scratch by an intelligent agent. The languages chosen are Bengali and Telegu.

Keywords: Dynamic NLP, Contrastive Association, ADIOS, Word learning, Syntax learning.

1 Introduction and Motivation

In dynamic NLP [1], the model starts with a set of sentences for which the related semantic concepts are available. At every step there is an expansion of the obtained language model. [1] applies dynamic NLP to learn English and Hindi from the video commentaries. Here we first use the techniques which were already present in D.Semwal's thesis, on Bengali and Telegu. We find better results for target compared to the ones found in the thesis. Also we apply the word2vec model which is an original piece of work. In addition, our way of morphosyntactic discovery is an improvement over the one present in the thesis. Here we give a brief outline of the steps we followed.
For a child, the first step of language learning is by contrastive association . Our intelligent agent(also referred to as system) uses the same principle. We collected a set of 18 commentaries in Bengali and 9 in Telugu for a given video. After that we typed out each of the commentaries in the respective language. This was quite a tedious job. For contrastive association we calculated all the required probabilities as outlined in section 4, to identify the high-confidence words for every concept in the video. Thus the system does word-learning and the results are quite satisfactory after this initial step.
Now a child cannot learn nicely if it is suddenly given a huge corpus. So we divide the corpus into family-lect and multi-lect. We create a histogram for the frequency of words in each commentary. Then we identify the closely related commentaries which come under family-lect, as described in 8. The rest of the commentaries which have a lot of variations come under multi-lect. After that we apply the ADIOS algorithm to learn the patterns in which the words occur in a language. Pattern learning is the first step towards syntax learning. In a language, syntax basically refers to the sequence of subject, object and verb. Through contrastive association the system has learned nouns. For verbs, we need to do morphosyntactic analysis. So, the last step is morphosyntactic discovery where we use the idea of Levenshtein distance along with stemming.

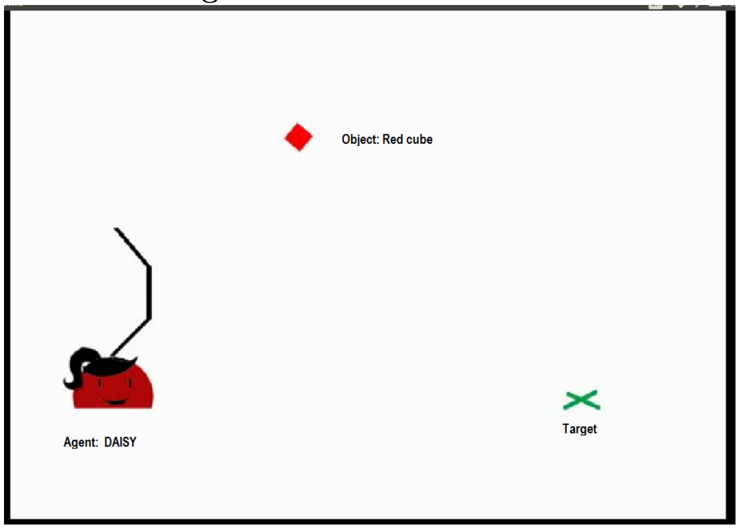
2 The Video

The video has 16 scenes where in each scene the agent performs some action on the object. A frame from the video where Daisy is throwing a red cube has been shown in figure 2. The following concepts are present in the video:

- Agent with subtypes (Dome,Daisy)
- Object with subtypes (Ball,Square)

– Colour with subtypes (Red,Blue)
– Action with subtypes (Throw, Roll)
– Path-goal with subtypes (Before Target, On Target, Beyond Target)

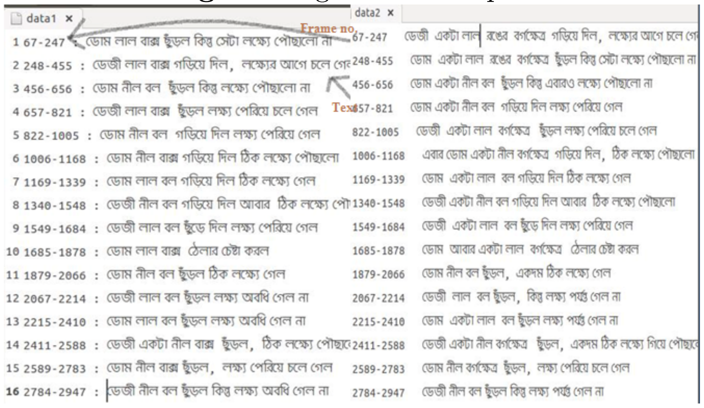
Fig. 1. A frame from the video



3 The Input Dataset

The input data consists of 18 typed(transcribed) Bengali commentaries and 9 Telugu commentaries for the 2D video with 16 scenes. Data was collected from people of different genders and different ages (to ensure enough variety) in the campus and also some of them were collected from outside the campus. An example commentary from the collected samples in Bengali is shown in figure. The audio was collected first using a web interface, via Wami collector, and then typed out in the respective language. Some samples of the dataset are shown in fig 2.

Fig. 2. Bengali data samples



```
1 67-247 : ডোম লাল বাক্স ছুঁড়ল কিন্তু সেটা লক্ষ্যে পৌছালো না
2 248-455 : ডেজী লাল বাক্স গড়িয়ে দিল, লক্ষ্যের আগে চলে গে
3 456-656 : ডোম নীল বল ছুঁড়ল কিন্তু লক্ষ্যে পৌছালো না
4 657-821 : ডেজী লাল বাক্স ছুঁড়ল লক্ষ্য পেরিয়ে চলে গেল
5 822-1005 : ডোম নীল বল গড়িয়ে দিল লক্ষ্য পেরিয়ে গেল
6 1006-1168 : ডোম নীল বাক্স গড়িয়ে দিল ঠিক লক্ষ্যে পৌছালো
7 1169-1339 : ডোম লাল বল গড়িয়ে দিল ঠিক লক্ষ্যে গেল
8 1340-1548 : ডেজী নীল বল গড়িয়ে দিল আবার ঠিক লক্ষ্যে পৌ
9 1549-1684 : ডেজী লাল বল ছুঁড়ে দিল লক্ষ্য পেরিয়ে গেল
10 1685-1878 : ডোম লাল বাক্স ঠেলার চেষ্টা করল
11 1879-2066 : ডোম নীল বল ছুঁড়ল ঠিক লক্ষ্যে গেল
12 2067-2214 : ডেজী লাল বল ছুঁড়ল লক্ষ্য অবধি গেল না
13 2215-2410 : ডোম লাল বল ছুঁড়ল লক্ষ্য অবধি গেল না
14 2411-2588 : ডেজী একটা নীল বাক্স ছুঁড়ল, ঠিক লক্ষ্যে পৌছাে
15 2589-2783 : ডোম নীল বাক্স ছুঁড়ল, লক্ষ্য পেরিয়ে চলে গেল
16 2784-2947 : ডেজী নীল বল ছুঁড়ল কিন্তু লক্ষ্য অবধি গেল না
```

```
67-247     ডেজী একটা লাল রঙের বর্গক্ষেত্র গড়িয়ে দিল, লক্ষ্যের আগে চলে গে
248-455    ডোম একটা লাল রঙের বর্গক্ষেত্র ছুঁড়ল কিন্তু সেটা লক্ষ্যে পৌছালো না
456-656    ডোম একটা নীল বল ছুঁড়ল কিন্তু এবারও লক্ষ্যে পৌছালো না
657-821    ডোম একটা নীল বল গড়িয়ে দিল লক্ষ্য পেরিয়ে গেল
822-1005   ডেজী একটা লাল বর্গক্ষেত্র ছুঁড়ল লক্ষ্য পেরিয়ে চলে গেল
1006-1168  এবার ডোম একটা নীল বর্গক্ষেত্র গড়িয়ে দিল, ঠিক লক্ষ্যে পৌছালো
1169-1339  ডোম একটা লাল বল গড়িয়ে দিল ঠিক লক্ষ্যে গেল
1340-1548  ডেজী একটা নীল বল গড়িয়ে দিল আবার ঠিক লক্ষ্যে পৌছালো
1549-1684  ডেজী একটা লাল বল ছুঁড়ে দিল লক্ষ্য পেরিয়ে গেল
1685-1878  ডোম আবার একটা লাল বর্গক্ষেত্র ঠেলার চেষ্টা করল
1879-2066  ডোম নীল বল ছুঁড়ল, একদম ঠিক লক্ষ্যে গেল
2067-2214  ডেজী লাল বল ছুঁড়ল, কিন্তু লক্ষ্য পর্যন্ত গেল না
2215-2410  ডোম একটা লাল বল ছুঁড়ল লক্ষ্য পর্যন্ত গেল না
2411-2588  ডেজী একটা নীল বর্গক্ষেত্র ছুঁড়ল, একদম ঠিক লক্ষ্যে গিয়ে পৌছাে
2589-2783  ডোম নীল বর্গক্ষেত্র ছুঁড়ল, লক্ষ্য পেরিয়ে চলে গেল
2784-2947  ডেজী নীল বল ছুঁড়ল কিন্তু লক্ষ্য পর্যন্ত গেল না
```

4 Contrastive association

It was found that little children learn new words in a language along with the related semantic concept, mainly by contrastive association as described in [4]. An experiment was done where a group of twelve children were shown a pair of metal tongs and every time the word 'pewter' was uttered. Now when this group was shown a pair of wooden tongs, they still agreed to attach the label 'pewter' to this semantic concept. However when they were shown a cup, they did not agree to attach the label 'pewter' to the cup. This behaviour was shown by eleven of the twelve children. From this experiment, one can infer that children basically learn by contrastive association. Given two concepts c_1, c_2 we first divide the corpus into subsets:

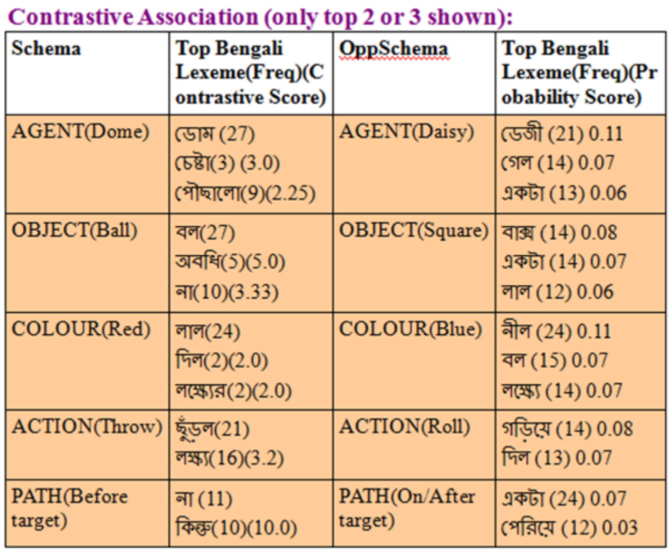
1. those that arise on commentaries for video involving concept c_1
2. those arising with concept c_2

Now a scoring function for association is defined as the ratio of the joint probabilities of word w occuring with concept c1 and that with c2:

$$S_{w_i,c_1} = \frac{P(w_i, c_1)}{P(w_i, c_2)}$$

The above equation for contrastive association score has been obtained from [2]. The contrastive association scores for the top two or three lexemes in Bengali and Telugu have been shown below in figure.

Fig. 3. Contrastive Association scores for Bengali



Contrastive Association (only top 2 or 3 shown):

Schema	Top Bengali Lexeme(Freq)(Contrastive Score)	OppSchema	Top Bengali Lexeme(Freq)(Probability Score)
AGENT(Dome)	ডোম (27) চেষ্টা(3) (3.0) পৌছালো(9)(2.25)	AGENT(Daisy)	ডেজী (21) 0.11 গেল (14) 0.07 একটা (13) 0.06
OBJECT(Ball)	বল(27) অবধি(5)(5.0) না(10)(3.33)	OBJECT(Square)	বাক্স (14) 0.08 একটা (14) 0.07 লাল (12) 0.06
COLOUR(Red)	লাল(24) দিল(2)(2.0) লক্ষ্যের(2)(2.0)	COLOUR(Blue)	নীল (24) 0.11 বল (15) 0.07 লক্ষ্যে (14) 0.07
ACTION(Throw)	ছুঁড়ল(21) লক্ষ্য(16)(3.2)	ACTION(Roll)	গড়িয়ে (14) 0.08 দিল (13) 0.07
PATH(Before target)	না (11) কিন্তু(10)(10.0)	PATH(On/After target)	একটা (24) 0.07 পেরিয়ে (12) 0.03

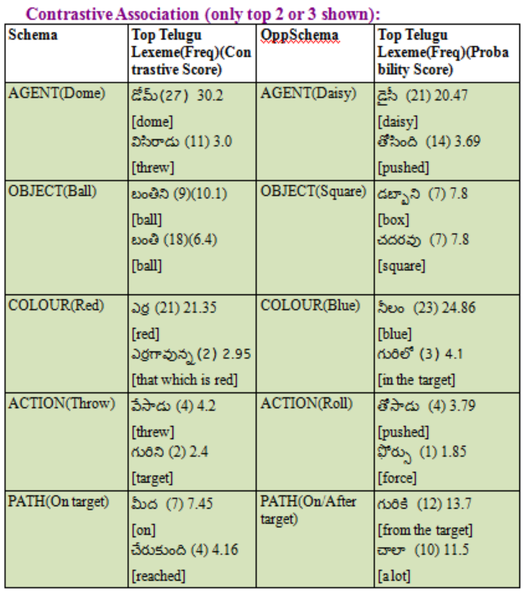
5 High Confidence words

For each of the concepts mentioned in 2, the system has learned some high confidence words after application of contrastive association. These are listed in the figure 5 below.

6 Bigram Analysis

Till now whatever scores we calculated was done only on mono-grams. One can also do bi-gram and in this manner n-gram analysis on the corpus. From bigram analysis we find that the frequency of occurence of pairs like red square ('laal baksho' in Bengali) is much more than say laal Dome. We collected all the bigrams occuring in the corpus. Such pattern analysis reveals some features about the syntax of the language. For actual syntax learning, one has to analyse the sequence of Subject,Object and Verb which is possible only after morphosyntactic discovery.

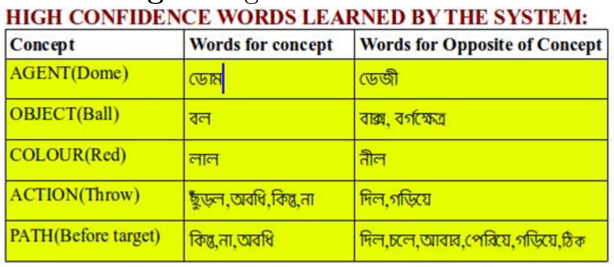
Fig. 4. Contrastive Association scores for Telugu



Contrastive Association (only top 2 or 3 shown):

Schema	Top Telugu Lexeme(Freq)(Contrastive Score)	OppSchema	Top Telugu Lexeme(Freq)(Probability Score)
AGENT(Dome)	డోమ్(27) 30.2 [dome] విసిరాడు (11) 3.0 [threw]	AGENT(Daisy)	డైసీ (21) 20.47 [daisy] తోసింది (14) 3.69 [pushed]
OBJECT(Ball)	బంతిని (9)(10.1) [ball] బంతి (18)(6.4) [ball]	OBJECT(Square)	డబ్బాని (7) 7.8 [box] చదరవు (7) 7.8 [square]
COLOUR(Red)	ఎర్ర (21) 21.35 [red] ఎర్రగావున్న (2) 2.95 [that which is red]	COLOUR(Blue)	నీలం (23) 24.86 [blue] గురిలో (3) 4.1 [in the target]
ACTION(Throw)	వేసాడు (4) 4.2 [threw] గురిని (2) 2.4 [target]	ACTION(Roll)	తోసాడు (4) 3.79 [pushed] ఫోర్సు (1) 1.85 [force]
PATH(On target)	మీద (7) 7.45 [on] చేరుకుంది (4) 4.16 [reached]	PATH(On/After target)	గురికి (12) 13.7 [from the target] చాలా (10) 11.5 [alot]

Fig. 5. High-confidence words



HIGH CONFIDENCE WORDS LEARNED BY THE SYSTEM:

Concept	Words for concept	Words for Opposite of Concept
AGENT(Dome)	ডোম	ডেজী
OBJECT(Ball)	বল	বাক্স, বর্গক্ষেত্র
COLOUR(Red)	লাল	নীল
ACTION(Throw)	ছুড়ল,অবধি,কিন্তু,না	দিল,গড়িয়ে
PATH(Before target)	কিন্তু,না,অবধি	দিল,চলে,আবার,পেরিয়ে,গড়িয়ে,ঠিক

7 Pruning the corpus

In order to narrow down the huge corpus, one good idea is to get rid of some prepositions and conjunctions like 'and'. To do this, we can either get rid of words which occur with high frequency in almost every dataset and at the same time have low contrastive score. Such words are generally the prepositions and conjunctions. We did pruning by this method. Another way is to use some existing corpus like TDIL corpus in Bengali, then train a model(like word2vec) to find out such unwanted words and then remove them from the corpus.

8 Dividing the corpus

The first step is to plot a histogram for each dataset. A sample from the 18 histograms we generated has been shown in fig 8 In order two find how much two given data samples are related, one can calculate the Bhattacharyya coefficient as:

$$Bhattcoef = \sum_{i=1}^{n} \sqrt{(\sum a_i)(\sum b_i)} \tag{1}$$

where the samples a and b have n partitions, and $\sum a_i$, $\sum b_i$ are the number of members in i^{th} partition of samples a and b. For discrete distribution $Bhattacharyyadistance = -ln(Bhattcoef)$. The higher the Bhattacharyya coefficient, the more closely related the two samples are. So, we can group the commentaries with higher coefficient into a cluster and call them as family-lect, whereas when we take the whole diversity in the data, it comes under multi-lect.

Fig. 6. Histogram samples for Bengali dataset

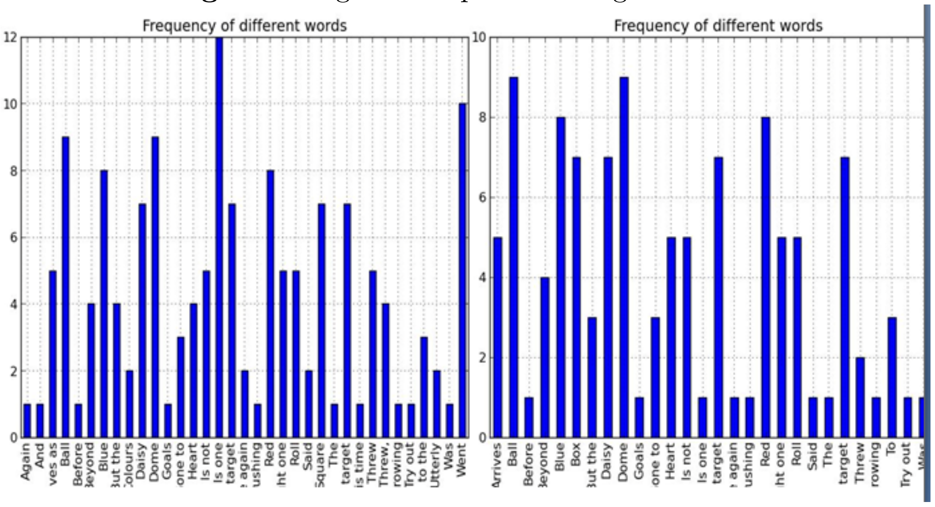
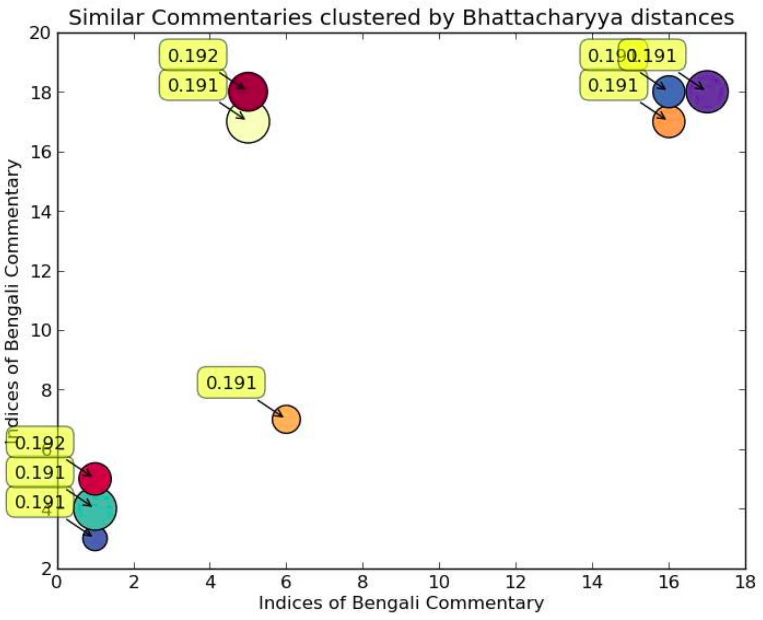


Fig. 7. Clustering by Bhattacharyya distance



9 ADIOS(Automatic Distillation Of Structure)

The ADIOS algorithm has been outlined in [3] and is now widely used for pattern learning. This unsupervised algorithm recursively distills hierarchically structured patterns from a given corpus. The ADIOS algorithm makes use of the MEX criterion (motif extraction criterion) *'where the sentences correspond to the different paths and vertices are the unique lexicon entries, augmented by two special symbols begin and end* as stated in [7]. The main purpose of using this algorithm is to learn patterns in which the words occur in a sentence. The algorithm has 3 steps as described by Rishav et.al. in [7]:
'1. INITIALIZATION: sentence loading
2. PATTERN DISTILLATION: iterative search for significant patterns which are added to lexicons as new units
3. GENERALIZATION: it generates more and more candidate pattern to be considered by pattern distillation'
Actually the ADIOS algorithm works using two principal components:

- Representational Data Structure(RDS) graph
- MEX pattern extraction algorithm

The algorithm goes on searching for new patterns at every iteration and it terminates when no more new patterns can be added. The equations for the probability of taking a left or right search path are provided in the Zach Solan thesis. So, basically it is an unsupervised algorithm for learning grammar from a corpus of sentences.
The algorithm bootstraps by loading all sentences of the corpus into the RDS graph which allows for both loops and multiple edges. The lexicon of the corpus(set of words) are loaded as the nodes of the graph. Two additional nodes are added to the graph [begin] and [end]. If two words consecutively appear in a sentence of the corpus a corresponding edge is added in the graph. So every sentence in the corpus becomes a path in the graph starting at [begin] and ending at [end]. Here we have given a brief overview. For more details refer to the thesis.

9.1 MEX pattern acquisition algorithm

Significance of patterns is determined using some probability functions which tap into the context sensitivity of natural languages. $(e_1; e_k)$ represents a search path in the RDS $e_1 \rightarrow e_2 \rightarrow \ldots \rightarrow e_k$.

$$P_R(e_i; e_j) = p(e_j | e_i e_{i+1} \ldots e_{j-1}) = \frac{l(e_i; e_j)}{l(e_i; e_{j-1})} \tag{2}$$

$$P_R(e_j; e_i) = p(e_i | e_{i+1} e_{i+2} \ldots e_j) = \frac{l(e_j; e_i)}{l(e_j; e_{i+1})} \tag{3}$$

The above equations have been taken from Zach Solan's thesis [3]. $l(e_i; e_j)$ represents the number of subpaths from e_i to e_j in the graph.
Intuitively, a higher values of P_R and P_L suggest point to the presence of patterns at $e_{j-1} \rightarrow e_j$. And using decrease ratios, which measure how much P_R or P_L falls from one node to the next, we can demarcate the beginning and end of patterns.
When a significant pattern is identified, it is added as a new node in the graph. And iteratively, again the search begins for newer significant patterns. The algorithm stops when no more significant patterns are discovered in an iteration. Hence, ADIOS, by this process is able to capture both the aspects of hierarchy and context sensitivity of language. The following *diagram taken from Zach Solan Thesis* explains the entire process of ADIOS very well.

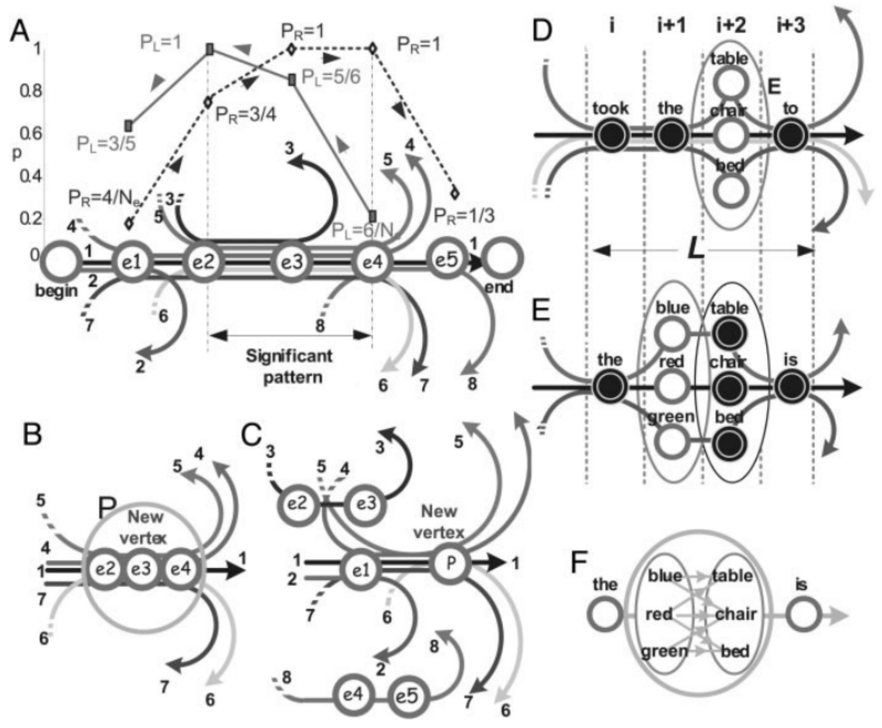
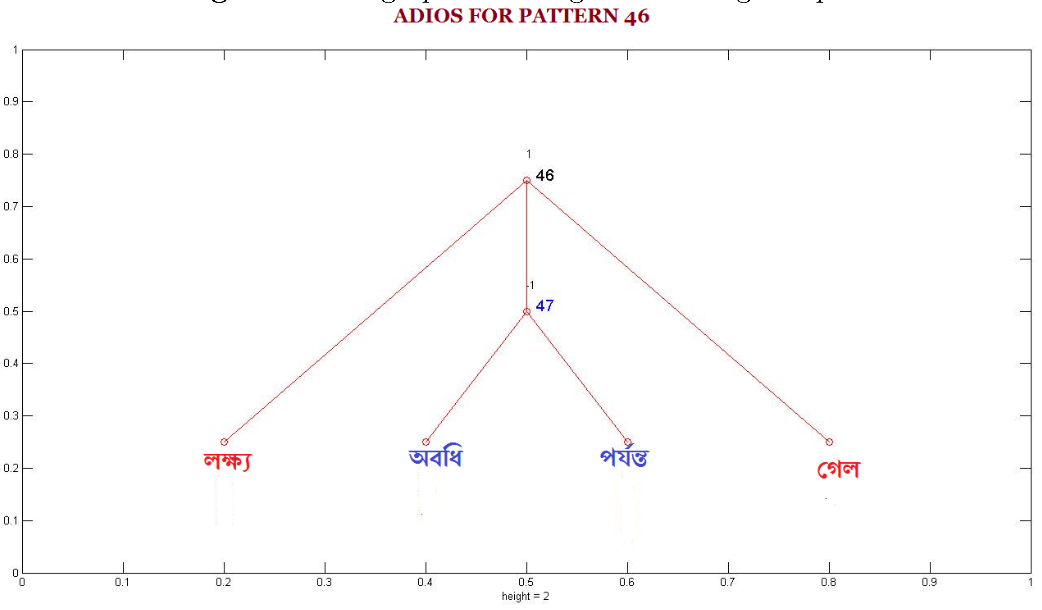
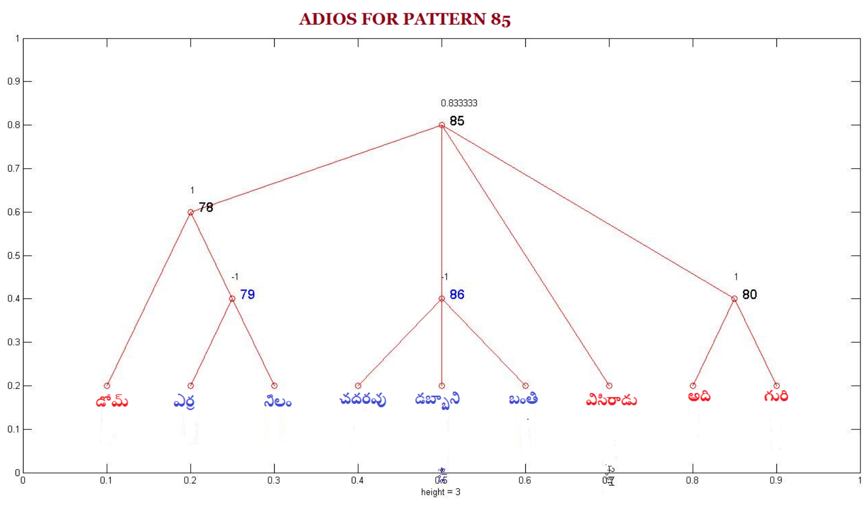


Fig. 1. The graph structures used by the MEX and ADIOS algorithms. (*A*) The search path no. 1 (*begin* → *e*1 → . . . → *e*5 → *end*) is rendered as a solid black line connecting the special *begin* and *end* vertices. Four other paths (nos. 4, 5, 6, and 7) join it along the vertices *e*2, *e*3, *e*4, thus forming a bundle that may constitute a significant pattern subject to the MEX criterion described in *BOX 1: The MEX Procedure*. Values of P_R and P_L, originating at e1 and e4, respectively, are displayed for the example shown here. (*B*) A significant pattern (*P* = *e*2 → *e*3 → *e*4) has been identified. (*C*) A new vertex is added to the graph, replacing the elements subsumed by *P*. Paths that belong to sequences not subsumed by it, such as no. 3 here, are left untouched. (*D*) The path is generalized: the algorithm picks among the set of path segments encountered in a window of size *L* = 4 those that differ in a single slot and are embedded in a common context (the relevant vertices are marked by open circles). The vertices in this slot form an equivalence class E to be treated as a single unit, resulting in the first generalization step (see *BOX 2: The* ADIOS *Algorithm*). (*E*) The just-detected E(*i* + 2) is used to find an additional equivalence class; it is specific to the current common context, thus enhancing the safety of generalization. (*F*) Stacking two equivalence classes leads to further generalization (see *BOX 2: The* ADIOS *Algorithm* for details).

9.2 Results from ADIOS

Fig. 8. ADIOS graphs for Bengali and Telugu corpus





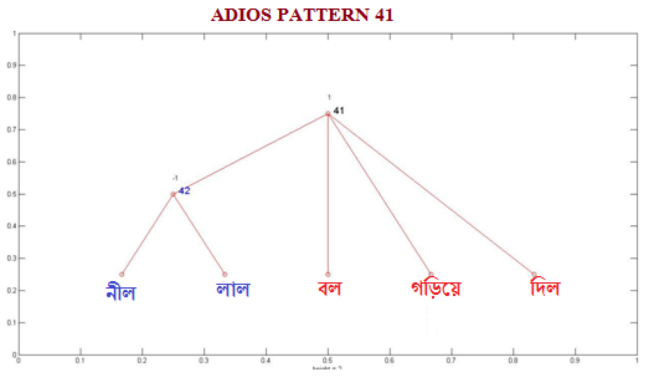
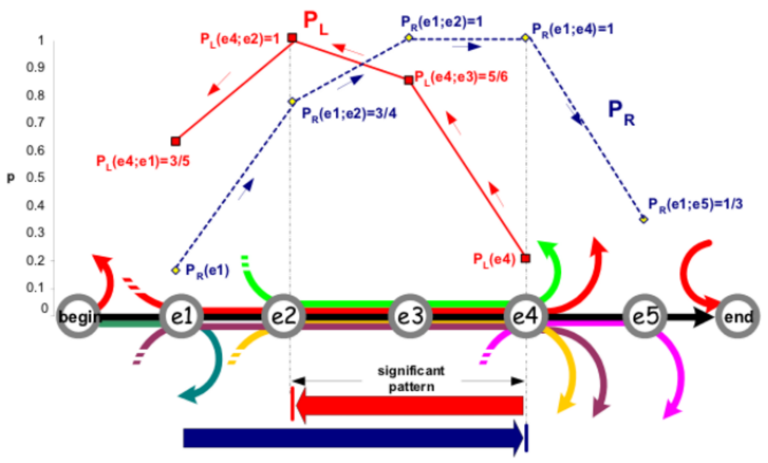


Fig. 9. Search path in ADIOS(taken from Zach Solan Thesis)



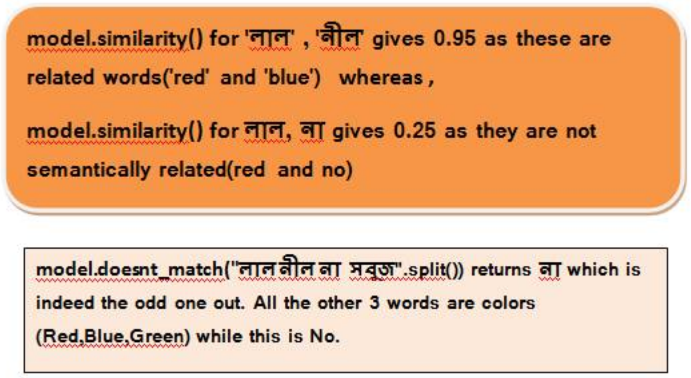
10 Application of word2vec

The word2vec tool is one of those areas of NLP which has experienced a lot of growth in the last few years. and is a very dynamic area now. Word2vec takes a corpus in some language and then gives the word vectors as output. First a vocabulary is constructed here from the training data and then a vector representation of the words is learnt. Around eight years ago (2007), Collobert and Weston who were working on neural nets, started working on an idea: Can we map each word to a vector in R^n? They decided to pass a lot of k-grams and then find whether it is correct or incorrect as feedback. The positive (correct) sentence gives a feedback of 1 and incorrect sentence gives a feedback of 0. They had around fifteen hidden layers. This set of vectors became very popular. In 2011 Mikolov et. al. simplified the process with a one layer neural network.
There are two available models: the continuous bag-of-words model and the skip-gram model. In the continuous bag-of-words model, the training is amazingly fast and generates word vector. We use gensim.models.word2vec.Word2Vec and train the model on our corpus. This class has a parameter named workers. We can increase the no. of workers to get the results faster. The results have been taken from the Bengali corpus as 18 commentaries were available. model.most_similar: Here we can give positive and negative fields. For example, model.most_similar(positive=['woman', 'king'], negative=['man']) returns queen
model.similarity(w1,w2): This returns a similarity measure between the 2 given words w1 and w2.
model.doesnt_match(wordlist): This finds the odd word out from the given wordlist.

10.1 Results from word2vec

In general word2vec model performs well when there is a large corpus. However, we found that it is performing really well after we trained it on the Bengali data. It has learnt synonyms and can find the odd word out from a given list of words. Some of the results are shown in figure 10.1.

Fig. 10. word2vec results



model.similarity() for 'लाल' , 'नील' gives 0.95 as these are related words('red' and 'blue') whereas ,

model.similarity() for लाल, ना gives 0.25 as they are not semantically related(red and no)

model.doesnt_match("लाल नील ना सबुज".split()) returns ना which is indeed the odd one out. All the other 3 words are colors (Red,Blue,Green) while this is No.

11 Morphosyntactic discovery

We use the idea of Levenshtein distance, but only this way of calculating edit distance between two strings is not sufficiently good because two semantically unrelated strings may have less Levenshtein distance than two related strings. So, we can also use the idea of stemming. Wikipedia defines stemming as *'the term used in linguistic morphology and information retrieval to describe the process for reducing inflected (or sometimes derived) words to their word stem, base or root form'*. To say in brief, we basically look at the structural similarity between two strings and also ensure that their starting is same. Only then these are considered to be sufficiently morphologically similar.
Wikipedia defines Levenshtein distance between two words as *'the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. It is named*

after Vladimir Levenshtein, who considered this distance in 1965.' The closer the two words, the lesser will be the Levenshtein distance(also referred to as edit distance). Now when the Levenshtein distance is below a threshold, there is a lot of chance that the two words refer to the same semantic concept. However in order to be more sure, we also introduce the idea of keeping the prefix same. This morphological analysis is mostly done for verbs. The difference in the structure is often due to tense (for example, throw(goralo in Bengali) and threw(goriye dilo in Bengali). Morphological variance can give a lot of challenge during language learning. An easy solution(though a bit time-consuming) is to have a handwritten grammar of morphology, in order to identify the related words.

12 Conclusion

We shall end with a brief overview of the strengths and weaknesses. First let us browse through the strengths. Compared to D.Semwal's thesis, we got better results for the target. We got some good high-confidence lexemes in both Bengali and Telugu for on,before or after target. Moreover, we applied word2vec and got interesting results from the corpus. This was a new idea which we introduced here. We included the concept of stemming to improve the morphosyntactic discovery in already-existing literature. Now, coming to the weaknesses, we got some extra lexemes after contrastive association, all of which were not useful. Also, the method of morphosyntactic discovery is still not full-proof.

13 Acknowledgement

We would like to thank our guide Prof.Amitabha Mukherjee for helping us all throughout the project. We would also like to thank Shashwat Chandra, Triya Bhattacharya, Nimisha Agarwal and Rajesh Shubhankar for helping us a lot during the project.

References

1. Deepali Semwal. Dynamic NLP : A model for language bootstrapping and lifelong learning M.Tech. Thesis under Dr. Amitabha Mukerjee at IIT Kanpur, 2014. Available at `http://www.cse.iitk.ac.in/users/deepalis/thesis/report.pdf`.
2. Deepali Semwal, Sunakshi Gupta, Amitabha Mukerjee. From visuo-motor to language.(2014) Available at `http://www.cse.iitk.ac.in/users/amit/pub/aaai-fs_14.pdf`.
3. Zach Solan, David Horn, Eytan Ruppin, Shimon Edelman. Unsupervised learning of natural languages. Proceedings of the National Academy of Sciences of the United States of America, 102(33):11629-11634, 2005.
4. Ellen M Markman. Constraints children place on word meanings. Cognitive Science, 14(1):5777, 1990. Available at `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.9253&rep=rep1&type=pdf`.
5. Susan P Thompson, Elissa L Newport. Statistical learning of syntax: The role of transitional probability. Language Learning and Devvelopment, 3(1):142, 2007. Available at `http://www.ehu.eus/HEB/KEPA/teaching/NeuroAdvance2011/Thompson.Newport.statistical.learning.of.syntax.2007.pdf`.
6. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space(2013). Available at `http://arxiv.org/pdf/1301.3781.pdf`
7. Rishabh Nigam, Shubhdeep Kochhar Computational Modeling of Language Acquisition under Dr.A.Mukherjee(2012).