Word Embeddings with Word2vec

ADVISOR: PROF. AMITABHA MUKERJEE

PRANJAL SINGH

10327511

Index

PART I

word2vec

- Two architectures

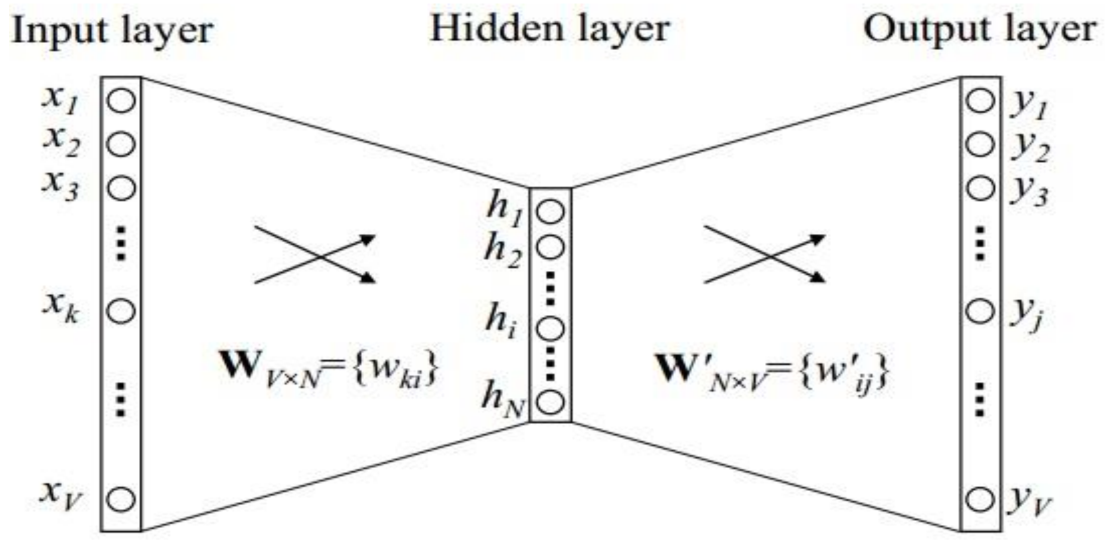
1. Continuous Bag-of-Word (CBOW)
2. Skip Gram

CBOW: Predict the word given the context

Skip-Gram: Predict the context given the word

One Word Context (CBOW)

- Vocabulary size is **V** and hidden layer size is **N**
- Input vector is one-hot encoded vector, i.e., only one node of {x_1,…..,x_v} is 1 and others 0
- Weights between the input layer and the output layer can be represented by a *V* × *N* matrix **W**



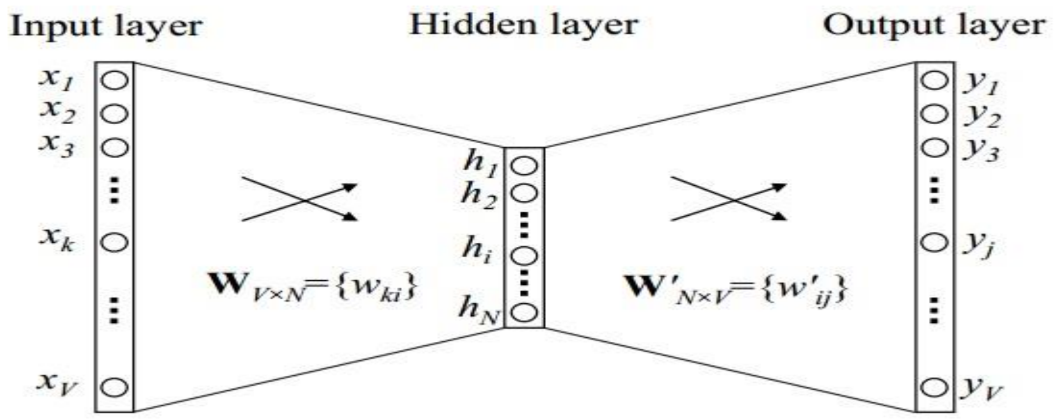
One Word Context (CBOW)

$$h = \mathbf{x}^T \mathbf{W} = \mathbf{v}_{\mathbf{Wi}}$$

$\mathbf{v}_{wI}$ is the vector representation of the input word w_I

$$u_j = {\mathbf{v}'_{w_j}}^T \cdot \mathbf{h}$$

u_j is score of each word in vocabulary and v'_{wj} is the *j-th* column of matrix **W'**



One Word Context (CBOW)

We then use soft-max, a log-linear classification model, to obtain the posterior distribution of words, which is a multinomial distribution

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})}$$

$$p(w_j|w_I) = \frac{\exp\left({\mathbf{v}'_{w_O}}^T \mathbf{v}_{w_I}\right)}{\sum_{j'=1}^{V} \exp\left({\mathbf{v}'_{w'_j}}^T \mathbf{v}_{w_I}\right)}$$

One Word Context (CBOW)

Update Equation for hidden->output weights

The training objective is to maximize the conditional probability of observing the actual output word *w_O* (denote its index in the output layer as j*) given the input context word *w_I* with regard to the weights

$$\begin{aligned} \max p(w_O|w_I) &= \max y_{j^*} \\ &= \max \log y_{j^*} \\ &= u_{j^*} - \log \sum_{j'=1}^{V} \exp(u_{j'}) := -E \end{aligned}$$

Using Stochastic Gradient Descent, update equation becomes,

$$\mathbf{v}'_{w_j}{}^{(\text{new})} = \mathbf{v}'_{w_j}{}^{(\text{old})} - \eta \cdot e_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V.$$

One Word Context (CBOW)

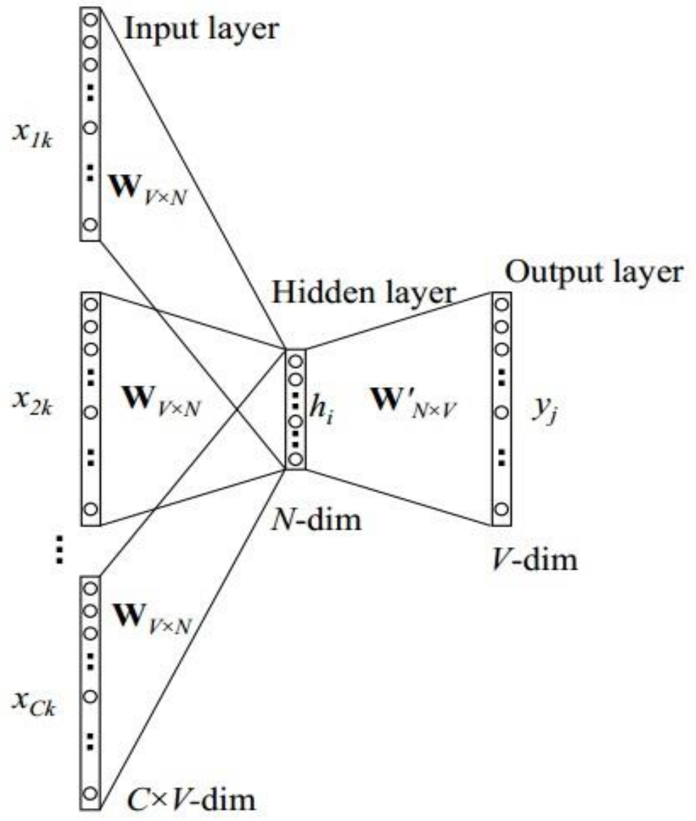
Update Equation for input->hidden weights

$$\mathbf{v}_{w_I}^{(\mathrm{new})} = \mathbf{v}_{w_I}^{(\mathrm{old})} - \eta \cdot \mathrm{EH}$$

Multi Word Context (CBOW)

$$\begin{aligned}\mathbf{h} &= \frac{1}{C}\mathbf{W}\cdot(\mathbf{x}_1+\mathbf{x}_2+\cdots+\mathbf{x}_C)\\ &= \frac{1}{C}\cdot(\mathbf{v}_{w_1}+\mathbf{v}_{w_2}+\cdots+\mathbf{v}_{w_C})\end{aligned}$$

$$\begin{aligned}E &= = -\log p(w_O|w_{I,1},\cdots,w_{I,C})\\ &= -u_{j^*}+\log\sum_{j'=1}^{V}\exp(u_{j'})\\ &= -\mathbf{v}'_{w_O}{}^{T}\cdot\mathbf{h}+\log\sum_{j'=1}^{V}\exp(\mathbf{v}'_{w_j}{}^{T}\cdot\mathbf{h})\end{aligned}$$



Multi Word Context (CBOW)

Update Equations

1) $$\mathbf{v}'_{w_j}{}^{(\text{new})} = \mathbf{v}'_{w_j}{}^{(\text{old})} - \eta \cdot e_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V.$$

2) $$\mathbf{v}_{w_{I,c}}^{(\text{new})} = \mathbf{v}_{w_{I,c}}^{(\text{old})} - \frac{1}{C} \cdot \eta \cdot \text{EH} \qquad \text{for } c = 1, 2, \cdots, C.$$

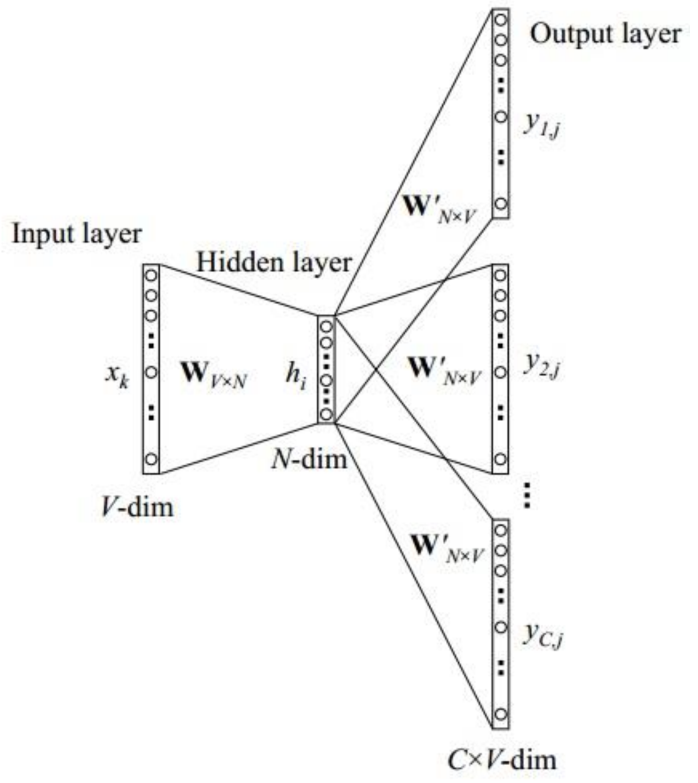
Skip-Gram Model

- It is the opposite of CBOW Model

- On the output layer, instead of outputing one multinomial distribution, we are outputing *C* multinomial distributions.

- Each output is computed using the same hidden →output matrix

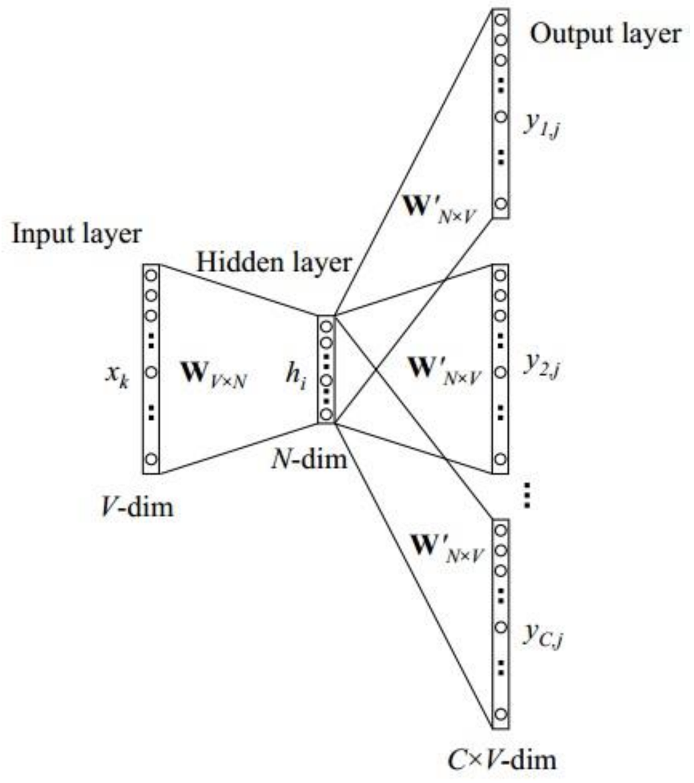
$$p(w_{c,j} = w_{O,c} | w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V} \exp(u_{j'})}$$



Skip-Gram Model

Loss Function

$$\begin{aligned} E &= -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C} | w_I) \\ &= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j_c^*})}{\sum_{j'=1}^{V} \exp(u_{j'})} \\ &= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp(u_{j'}) \end{aligned}$$



Optimization

1. Hierarchical Softmax (Morin & Bengio, 2005)

2. Negative Sampling (Gutmann & Hyvarinen, 2012)

Dataset

- Hindi :Wikipedia text dump (290 MB) ~4 mins of training
  - 23848940 words
  - 723737 words in vocabulary
  - 106876 words after removing words with *freq<5*

- English: Wikipedia text dump (9.3 GB) ~3.5 hrs of training
  - 1,703,849,452 words
  - 13,027,758 words in vocabulary
  - 1,778,685 words after removing words with *freq<5*

Result (English)

Odd One out

1) breakfast cereal dinner lunch

Ans: Cereal

2) eight six seven five three owe nine

Ans: owe

3) math shopping reading science

Ans: shopping

Result (English)

Top Similar Words

Father	FRANCE	XBOX	scratched	megabits
1) grandfather	1) GERMANY	1) XBLA	1) scraped	1) gigabits
2) uncle	2) FRENCH	2) Xbox360	2) rubbed	2) kilobits
3) mother	3) GREECE	3) SmartGlass	3) bruised	3) megabit
4) father-in-law	4) NETHERLANDS	4) 360/PS3	4) cracked	4) terabits
5) brother	5) SCOTLAND	5) Playstation	5) discarded	5) MB/s
		6) Qubed	6) shoved	6) Tbit/s
		7) Kinect	7) tripped	
			8) scratches	
			9) ripped	
			10) pasted	

Result (Hindi)

Odd One out

1) भारत   रूस   मुम्बई   चीन

Ans: मुम्बई

2) लड़की   बेहन   महिला   मर्द

Ans: मर्द

3) नेता मंत्री सरकार उद्योग

Ans: उद्योग

Result (Hindi)

Top Similar Words

भारत	व्यापार	ओबामा
1) प्रदेश	1) व्यवसाय	1) क्लिंटन
2) तिब्बत	2) पुनर्बीमा	2) बराक
3) देश	3) वाणिज्य	3) सीनेटर
4) आँध्रप्रदेश	4) बैंकिंग	4) राष्ट्रपति
5) लद्दाख	5) उद्योग	5) उम्मीदवार

PART II

Sentiment Analysis

- Sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document.

- A basic task in sentiment analysis is classifying the *polarity* of a given text at the document, sentence, or feature/aspect level — whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral.

(Source: Wikipedia)

tf-idf(term frequency–inverse document frequency)

- It is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus

- The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus

$$tfidf(d,t) = tf(d,t) \times \log(\frac{|D|}{df(t)})$$

Here $df(t)$ is the number of documents in which term t appears.

D is the total number of documents

tf(d,t) is the term-frequency

Dataset

1. IMDB 50,000 Movie Review Dataset(English)
   I. Contains 25,000 Positive and 25,000 Negative reviews
   II. 25,000 training examples and 25,000 testing examples

2. Hindi Product Review Dataset(IIIT)
   I. Contains 350 Positive and 350 Negative reviews

3. Hindi Movie Review Dataset(IITB)
   I. Contains 127 Positive and 125 Negative reviews

Methodology

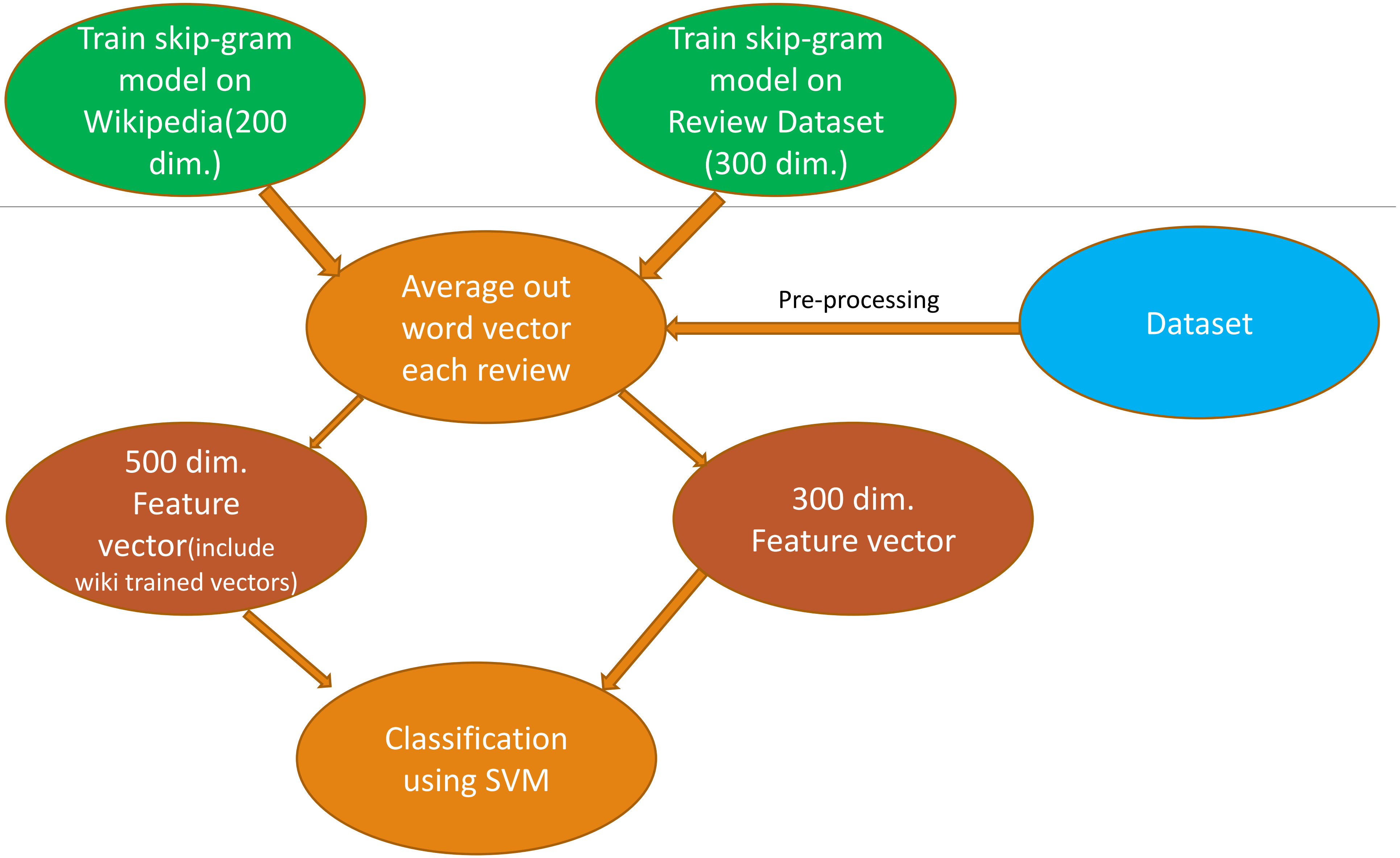
- Trained word2vec on each dataset for respective evaluation

- Features are constructed by taking average of word vectors for each review(not weighted average)

- Ignored stop-words(the, a, there, etc.) in English but not Hindi(मैं, वह, ने, etc.)

- Also constructed feature set by merging word vectors of Wikipedia trained word2vec(Hindi & English Wikipedia)
    - E.g. 300 dim.(from first training) + 200 dim.(from second training)= 500 dim. Feature vector

- Used SVM implementation of libSVM in scikit


Train skip-gram model on Wikipedia(200 dim.)
Train skip-gram model on Review Dataset (300 dim.)
Average out word vector each review
Pre-processing
Dataset
500 dim. Feature vector(include wiki trained vectors)
300 dim. Feature vector
Classification using SVM

Result(English-IMDB)

Classifier	Accuracy(%)
SVM(linear)	87.05
Random Forest(280 Trees)	84.14
Naïve Bayes	75.95
Logistic Regression(l1 penalty)	86.90
k-NN(20)	76.76
SVM(linear) with Wiki trained word vectors merged(300+200 dim)	87.56

*** Skip-gram model was trained with vector dimension as 300 and min word-count as 20**

Result(Hindi-Product Review: IIIT)

Feature Set	Accuracy(%)
500 dimensional word vector	78.0
500 dimensional word vector + tf-idf	90.73
Using subjective Lexicon construction(Thesis by Piyush Arora, IIIT’13)	79.62

- **Skip-gram model was trained with vector dimension as 300 and min word-count as 20**
- **Used Linear SVM as classifier**

Result(Hindi-Movie Review: IITB)

Feature Set	**Accuracy(%)**
500 dimensional word vector	79.62
500 dimensional word vector + tf-idf	88.13
Tf-idf & n-gram [6]	78.14

- **Skip-gram model was trained with vector dimension as 300 and min word-count as 20**
- **Used Linear SVM as classifier**

References

1. Manning, W. Y. (2013). Bilingual Word Embeddings for Phrase-Based Machine Translation. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, (p. 2013).

2. Weston, R. C. (2011). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research, 12*, 2493--2537.

3. Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781.

4. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani & K. Weinberger (ed.), *Advances in Neural Information Processing Systems 26* (pp. 3111--3119) .

5. Aditya Joshi, Balamurali A. R. and Pushpak Bhattacharyya. A Fall-back Strategy for Sentiment Analysis in Hindi: a Case Study. International Conference on Natural language Processing (ICON), Karagpur.-2010

6. Sarath Chandar A P, Mitesh M Khapra, Ravindran B, Vikas Raykar, Amrita Saha, "Multilingual Deep Learning". In Deep Learning Workshop at NIPS 2013.

Thank You!!!

NLP from Scratch

- Built a unified architecture for tasks such as POS tagging, Chunking, NER
- Compared against classical NLP benchmarks
- Avoided task specific engineering
- Generalize a system to handle multiple tasks

NLP from Scratch

- Learn lookup table by back propagation
- Words are mapped to *d-dimensional* vector using lookup table operation
- Lookup table returns a matrix for a given sentence

NLP from Scratch

Used entire English Wikipedia to learn word embeddings (631 million words)

Tokenized using Penn Treebank Tokenizer

The total training time was about four weeks

Window size: 11 and a Hidden layer with 100 units

Objective: Seek a network that computes a higher score when given a legal phrase than when given an incorrect phrase

$$\theta \mapsto \sum_{x \in \mathcal{X}} \sum_{w \in \mathcal{D}} \max\left\{0, 1 - f_\theta(x) + f_\theta(x^{(w)})\right\}$$