Mikolov's Language Models:

Distributed Representations of Sentences and Documents
Recurrent Neural Language Model

Tomas Mikolov[1]

May 16, 2014

[1]Google Inc1

Table of contents

Motivation

Quoth Tomas Mikolov,
http://www.fit.vutbr.cz/ imikolov/rnnlm/google.pdf

- Statistical language models assign probabilities to word sequences
- Meaningful sentences should be more likely than ambiguous ones
- Language modeling is an artificial intelligence problem.

Classical Ngram Models

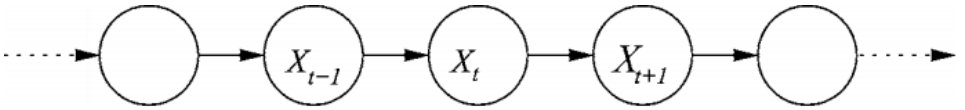


Figure: Text Modeling using Markov Chains, Claude Shannon (1984)

$$\max P(w_i | w_{i-1}, ...) \tag{1}$$

Where each w_i representation is a 1-N encoding.

Neural Representation of Words

Neural Language Model Bengio et al, 2006.

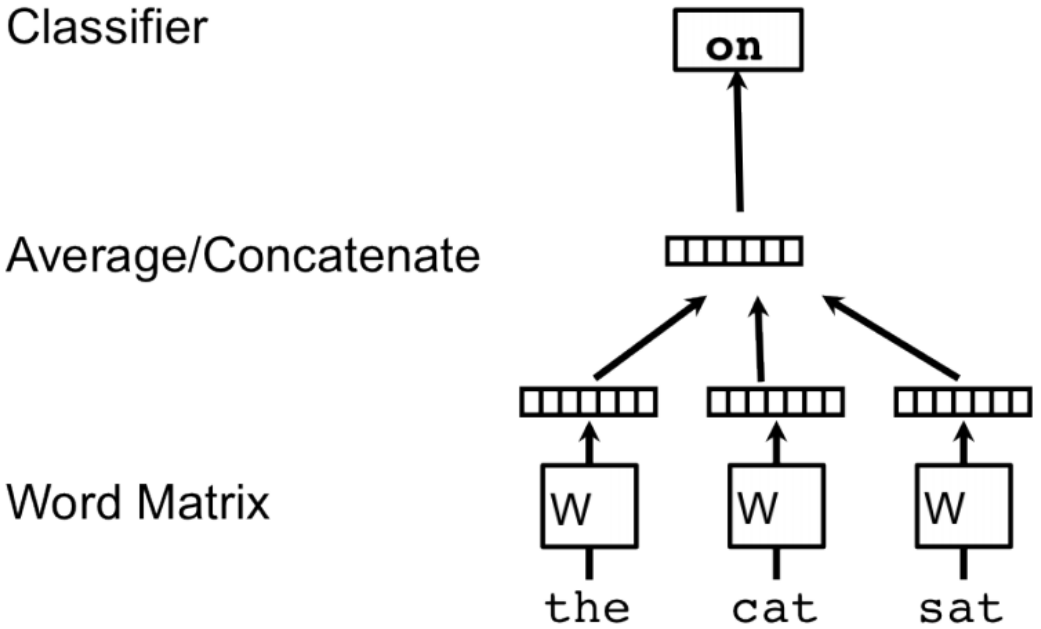


Figure: *Word2Vec*, Tomas Mikolov

Beyond Word Embeddings

Recursive Deep Tensor Models Socher et. al.

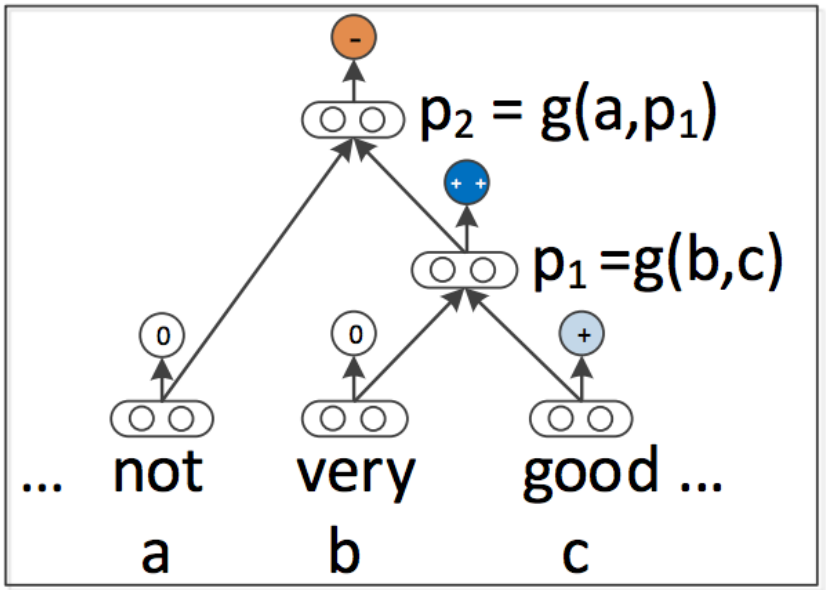


Figure: *Recursive Tree Structure*, Richard Socher 2013

Beyond Word Embeddings

- *Recurrent Neural Network Language Model* Mikolov et. al.

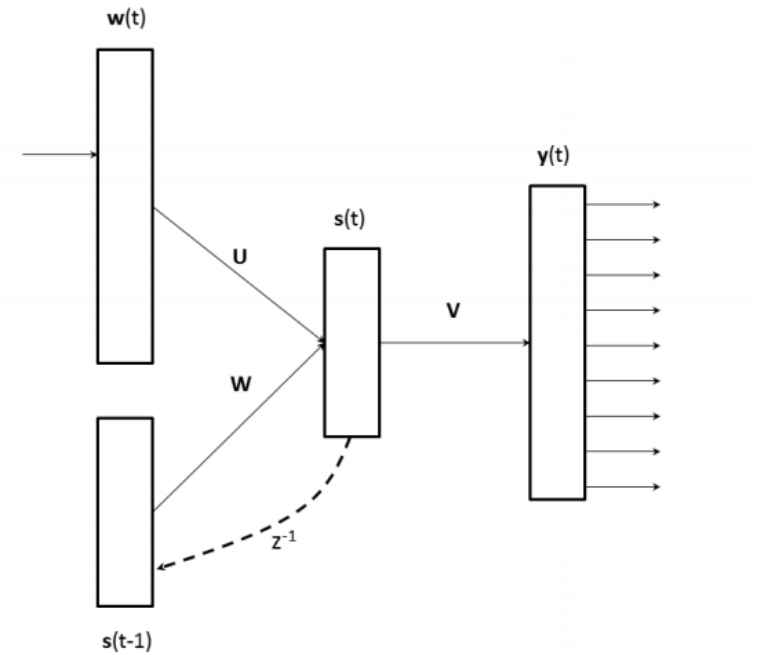


Figure: *Recurrent NN*, Tomas Mikolov 2010

Beyond Word Embeddings

- *Character-Level Recognition*

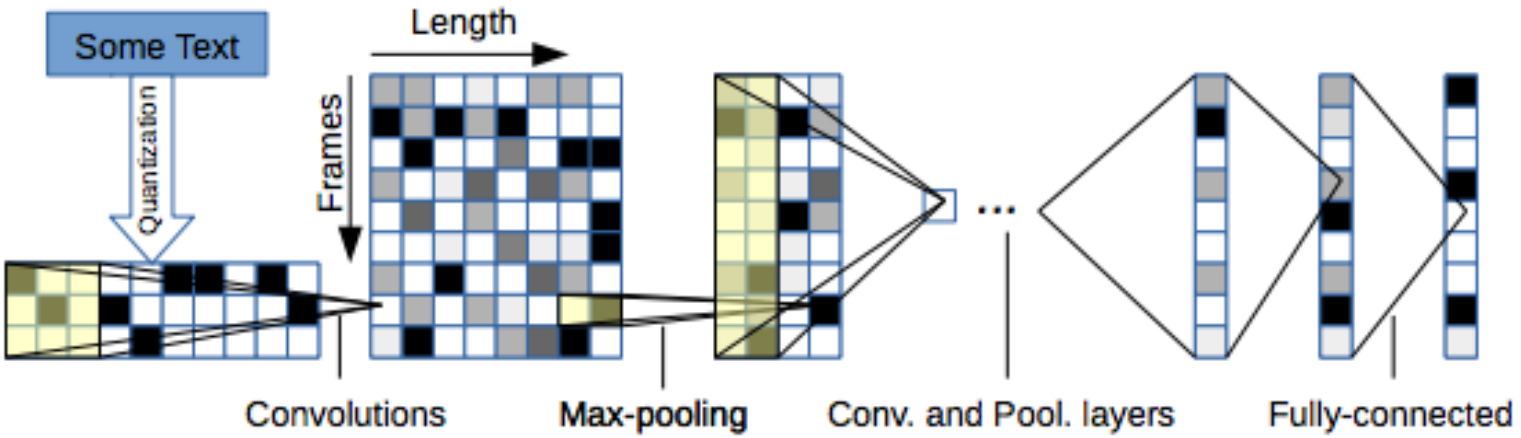


Figure: *Text Understanding from Scratch*, Zhang, LeCun 2015

Algorithm Overview

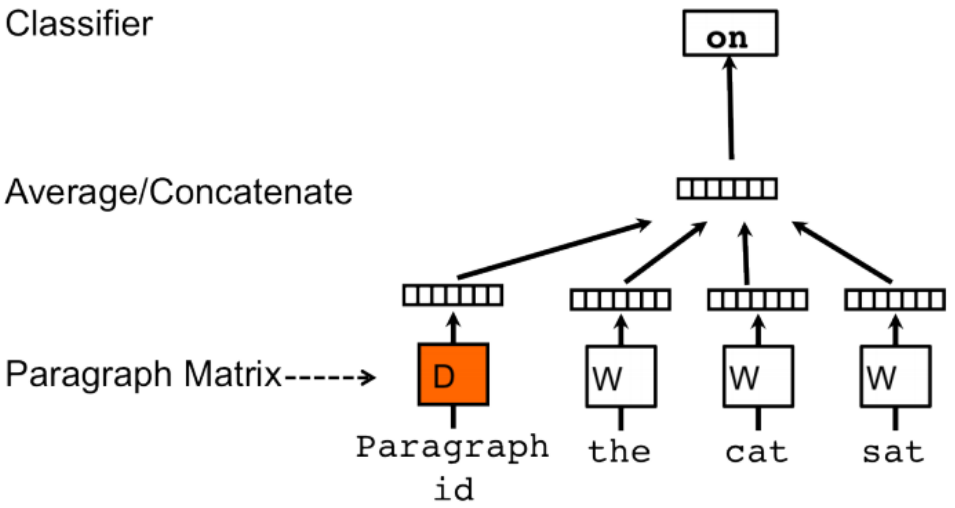


Figure: *Paragraph Embedding, Learning Model*, Tomas Mikolov 2013

Algorithmic Overview

Part 1. Word embeddings.

Given sentence $w_1, w_2, w_3 ...:$

$$\max \frac{1}{T} \sum_{t=k}^{T-k} \log p(w_t | w_{t-k}, ..., w_{t+k}) \tag{2}$$

where

$$p(w_t | w_{t-k}, ..., w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \tag{3}$$

Algorithmic Overview

Parameters for Step 1: U, b

$$y = b + Uh(w_{t-k}, ..., w_{t+k}; W) \tag{4}$$

Algorithmic Overview

Part II. Joint Word and Paragraph

$$y = b + Uh(w_{t-k}, ..., w_{t+k}; W, D) \tag{5}$$

$$W \in R^{p \times N}$$

$$D \in R^{p \times M}$$

$$p \times (M + N)$$

Algorithm Overview

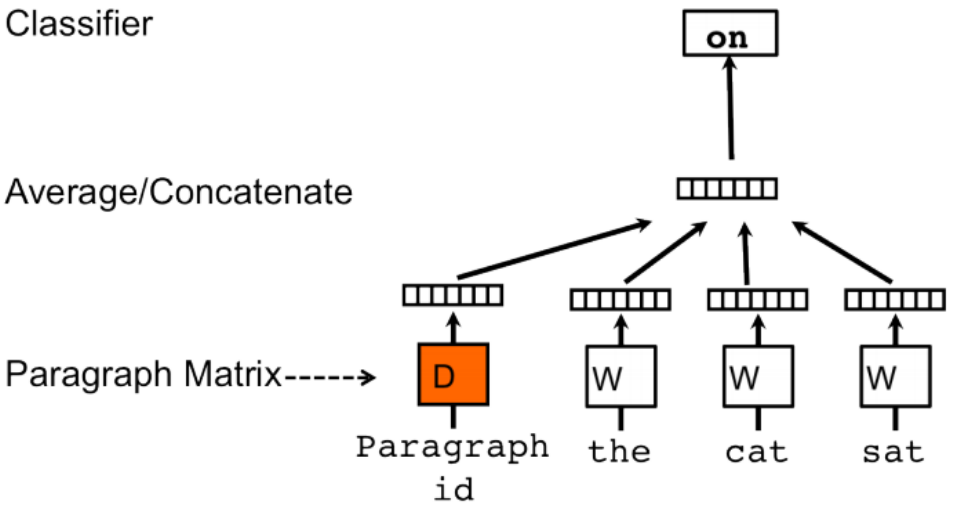


Figure: *Distributed Memory Model*

Algorithm Overview

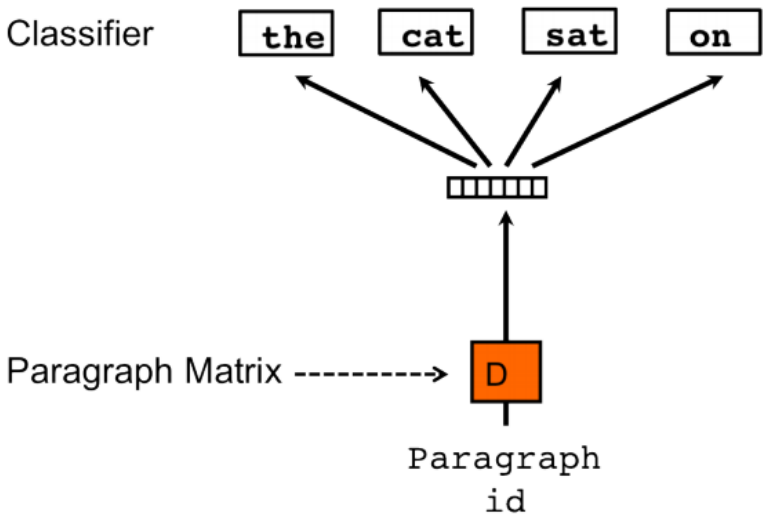


Figure: *Distributed Bag of Words Model Model*

Sentiment Analysis

Model	Error rate (Positive/ Negative)	Error rate (Fine-grained)
Naïve Bayes (Socher et al., 2013b)	18.2 %	59.0%
SVMs (Socher et al., 2013b)	20.6%	59.3%
Bigram Naïve Bayes (Socher et al., 2013b)	16.9%	58.1%
Word Vector Averaging (Socher et al., 2013b)	19.9%	67.3%
Recursive Neural Network (Socher et al., 2013b)	17.6%	56.8%
Matrix Vector-RNN (Socher et al., 2013b)	17.1%	55.6%
Recursive Neural Tensor Network (Socher et al., 2013b)	14.6%	54.3%
Paragraph Vector	**12.2%**	**51.3%**

Figure: *Stanford Sentiment Treebank Dataset*

Sentiment Analysis

Model	Error rate
BoW (bnc) (Maas et al., 2011)	12.20 %
BoW (bΔt'c) (Maas et al., 2011)	11.77%
LDA (Maas et al., 2011)	32.58%
Full+BoW (Maas et al., 2011)	11.67%
Full+Unlabeled+BoW (Maas et al., 2011)	11.11%
WRRBM (Dahl et al., 2012)	12.58%
WRRBM + BoW (bnc) (Dahl et al., 2012)	10.77%
MNB-uni (Wang & Manning, 2012)	16.45%
MNB-bi (Wang & Manning, 2012)	13.41%
SVM-uni (Wang & Manning, 2012)	13.05%
SVM-bi (Wang & Manning, 2012)	10.84%
NBSVM-uni (Wang & Manning, 2012)	11.71%
NBSVM-bi (Wang & Manning, 2012)	8.78%
Paragraph Vector	**7.42%**

Figure: *iMDB Dataset*

Model

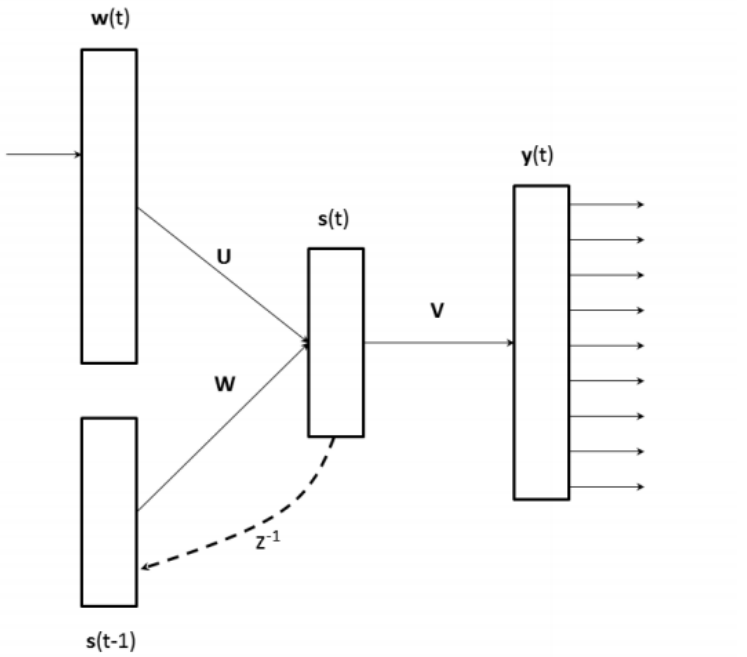


Figure: *Recurrent NN*, Tomas Mikolov 2010

Components:

$$input : x(t) = w(t) + s(t-1)$$

$$hidden : s_j(t) = f\left(\sum_i x_i(t) * u_{ji}\right)$$

$$output : y_k(t) = g\left(\sum_j s_j(t) * v_{kj}\right)$$

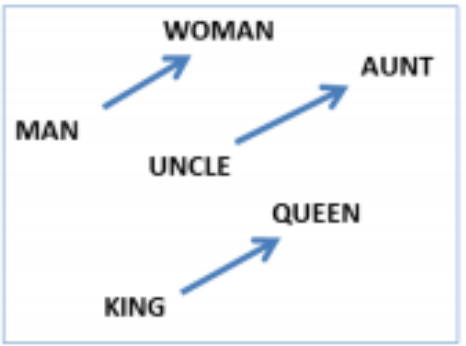
where f is sigmoid and g is softmax.

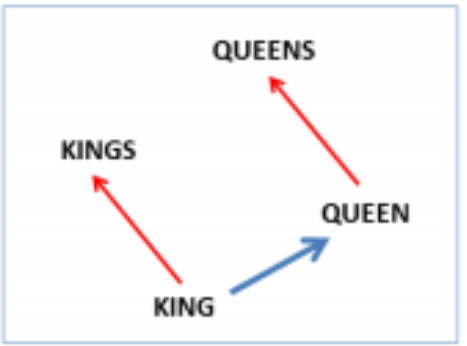
Spatial Meaning:

Vector Offset Method for Running Linguistic Analogy Questions:

$$y = x_b - x_a + x_c$$

$$w^* = \arg\max_w \frac{x_w y}{||x_w||||y||}$$





Results

Method	Adjectives	Nouns	Verbs	All
LSA-80	9.2	11.1	17.4	12.8
LSA-320	11.3	18.1	20.7	16.5
LSA-640	9.6	10.1	13.8	11.3
RNN-80	9.3	5.2	30.4	16.2
RNN-320	18.2	19.0	45.0	28.5
RNN-640	21.0	25.2	54.8	34.7
RNN-1600	**23.9**	**29.2**	**62.2**	**39.6**