# Learning Word Embeddings for Low Resource Languages: The Case of Buryat Language

V. Konovalov

Z. Tumunbayarova

Transbaikal State University, Russia

# How to learn word vector representation?

**Count-Based Methods**

- Pointwise mutual information (PMI)
- Singular-value decomposition (SVD)
- others

**Predict-Based Methods**

- Skip-Gram Model
- Continuous Bag-of-Words (CBOW)
- others

# Our Research

**Goals**

- Compare techniques to learn word vector representations of low resource languages.

# Our Research

**Goals**

- Compare techniques to learn word vector representations of low resource languages.
- Explore the influence of stemmer on the quality of the vectors.

# Our Research

Count-based methods vs Predict-based methods

# Our Research

Count-based methods VS Revised count-based methods

Smoothed PMI

Weighted SVD

# Techniques To Compare

- Pointwise mutual information (PMI)
- Smoothed PMI
- Singular Value Decomposition (SVD)
- Weighted SVD
- Continuous Bag-of-Words (CBOW)
- Skip-Gram Model
- Global Vectors (GloVe)

# Pointwise mutual information (PMI)

# Pointwise mutual information (PMI)

$$PMI(w, c) = \log \frac{P(w, c)}{P(w)P(c)}$$

# Pointwise mutual information (PMI)

$$PMI(w, c) = \log \frac{P(w, c)}{P(w)P(c)}$$

PMI is used to fill the cells of word-context matrix

# Pointwise mutual information (PMI)

$$PMI(w, c) = \log \frac{P(w, c)}{P(w)P(c)}$$

PMI is used to fill the cells of word-context matrix

To calculate $PMI(w, c)$ we need to count $\#(w, c), \#(w), \#(c)$

# Pointwise mutual information (PMI)

What to do when $\#(w, c) = 0$?

- $PMI(w, c) = 0$ when $\#(w, c) = 0$
- Positive PMI (PPMI), $PPMI(w, c) = max(0, PMI(w, c))$

# Smoothed PMI

$$PMI(w, c) = \log \frac{P(w, c)}{P(w)P(c)}$$

**Problem of PMI**

PMI suffers from its bias towards infrequent events [1]

[1] Turney, P., Pantel, P. (2010). From frequency to meaning: Vector space models of semantics

# Smoothed PMI

$$PMI(w, c) = \log \frac{P(w, c)}{P(w)P(c)}$$

**Problem of PMI**

PMI suffers from its bias towards infrequent events [1]

This leads top a situation when "distributional features" (contexts) of $w$ are often extremely rare words.

[1] Turney, P., Pantel, P. (2010). From frequency to meaning: Vector space models of semantics

# Smoothed PMI

## Solution - Smoothing PMI

$$PMI_\alpha(w, c) = \log \frac{P(w, c)}{P(w)P_\alpha(c)}$$

$$P_\alpha(c) = \frac{\#(c)^\alpha}{\sum_c \#(c)^\alpha}$$

$$P_\alpha(c) > P(c)$$

$\alpha = 0.75$ was found to be effective[2]

[2] Levy, O., Goldberg, Y., Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings

# Singular Value Decomposition (SVD)

# Singular Value Decomposition (SVD)

Any matrix $M$ can be represented as

$$M = U \times \Sigma \times V$$

where $U$ and $V$ are orthonormal and $\Sigma$ is a diagonal matrix of eigenvalues in decreasing order.

# Singular Value Decomposition (SVD)

Any matrix $M$ can be represented as

$$M = U \times \Sigma \times V$$

where $U$ and $V$ are orthonormal and $\Sigma$ is a diagonal matrix of eigenvalues in decreasing order.
By keeping only top $d$ elements of $\Sigma$ we obtain
$M_d = U_d \times \Sigma_d \times V_d$

# Singular Value Decomposition (SVD)

Any matrix $M$ can be represented as

$$M = U \times \Sigma \times V$$

where $U$ and $V$ are orthonormal and $\Sigma$ is a diagonal matrix of eigenvalues in decreasing order.
By keeping only top $d$ elements of $\Sigma$ we obtain
$M_d = U_d \times \Sigma_d \times V_d$
A common approach is to approximate $M$ by $W_{SVD} = U_d \times \Sigma_d$

# Weighted SVD

It was shown that weighting the eigenvalues matrix $\Sigma_d$ (p=0.5) can have a significant effect on the performance on the semantic similarity tasks

$$W_{SVD}^{p} = U_d \times \Sigma_d^{p}$$

# Word Normalization

## Word Normalization

- Lemmatizer
- Stemmer

There are normalizers for popular languages

- Porter stemmer for English
- Yandex Mystem for Russian
- others

**Word Normalization**

- Lemmatizer
- Stemmer

There are normalizers for popular languages

- Porter stemmer for English
- Yandex Mystem for Russian
- others

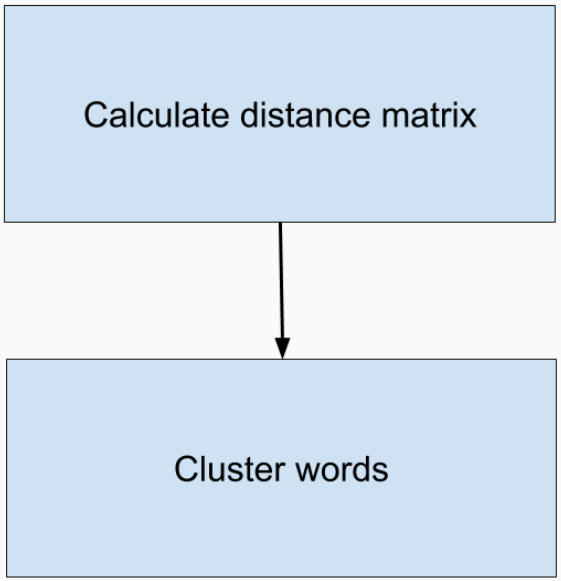
However, usually low resource languages lack such normalizers.

# Language-independent Stemmer

- **N-Gram Stemmer**
- **HMM Stemmer** (Melucci Massimo and Orio Nicola. "A novel method for stemmer generation based on hidden Markov models")
- **YASS Stemmer** (Prasenjit Majumder, Mandar Mitra, Swapan K. Parui, Gobinda Kole, Pabitra Mitra and Kalyankumar Datta. "YASS: Yet another suffix stripper")

# Yet Another Suffix Striper (YASS)[3]

## General Scheme




[3]Majumder, P., Mitra, M., Parui, S., Kole, G., Mitra, P., Datta, K. .., et al. (2007). YASS: Yet another suffix stripper

# Yet Another Suffix Striper (YASS) - Calculate Distances

Given two strings $X = x_0x_1...x_n$ and $Y = y_0y_1...y_m$.

**Calculate Distances. Step 1**

If the strings are of unequal lengths we pad the shorter string with null characters to make the strings $n$ lengths equal.

# Yet Another Suffix Striper (YASS) - Calculate Distances

**Calculate Distances. Step 2**

Calculate $p_i$ for every position $i$, $p_i$ equals 1 if there is a mismatch in the $i$-th position of $X$ and $Y$

**Calculate Distances. Step 3**

$$D(X,Y) = \begin{cases} \frac{n-m+1}{m} \sum_{i=m}^{n} \frac{1}{2^{i-m}} & \textit{if} \quad m > 0 \\ \infty \quad \textit{otherwise} \end{cases} \tag{1}$$

where $m$ denotes the position of the first mismatch between $X$ and $Y$

# Yet Another Suffix Striper (YASS)

## Example 1

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | s | t | r | o | n | o | m | e | r | x | x | x | x |
| a | s | t | r | o | n | o | m | i | c | a | l | l | y |

$$D = \frac{6}{8} \times \left( \frac{1}{2^0} + ... + \frac{1}{2^{13-8}} \right) = 1.4766$$

# Yet Another Suffix Striper (YASS)

## Example 2

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| a | s | t | r | o | n | o | m | e | r |
| a | s | t | o | n | i | s | h | x | x |

$$D = \frac{7}{3} \times \left( \frac{1}{2^0} + ... + \frac{1}{2^{9-3}} \right) = 4.6302$$

# Yet Another Suffix Striper (YASS)

**Intuition**

The main intuition behind defining these distances was to reward long matching prefixes, and to penalize an early mismatch.

# Yet Another Suffix Striper (YASS)

**Stem words by using Hierarchical clustering**

- Single-linkage: $min\{d(a,b), a \in A, b \in B\}$
- Average-linkage: $\frac{1}{|A||B|}\sum_{a\in A}\sum_{b\in B} d(a,b)$
- Complete-linkage: $max\{d(a,b), a \in A, b \in B\}$

where $d$ - Euclidean distance

# Yet Another Suffix Striper (YASS)

**Single-linkage**

Single linkage clusters can be long and branched in high-dimensional space and the merging criterion often causes chaining, where a single element is continually added to the tail of the biggest cluster

**Complete-linkage**

Complete linkage clustering avoids a drawback of the alternative single linkage method - the so-called chaining phenomenon, where clusters formed via single linkage clustering may be forced together due to single elements being close to each other, even though many of the elements in each cluster may be very distant to each other. Complete linkage tends to build very compact clusters.

# Yet Another Suffix Striper (YASS)

**Average-linkage**

Average-linkage is a balance between single-linkage and complete-linkage approaches.

# Experimental setup

# Experimental setup

- Buryat Wikipedia consists of 1381 articles

# Experimental setup

- Buryat Wikipedia consists of 1381 articles
- articles were lower-cased

# Experimental setup

- Buryat Wikipedia consists of 1381 articles
- articles were lower-cased
- non-textual elements were removed

# Experimental setup

- Buryat Wikipedia consists of 1381 articles
- articles were lower-cased
- non-textual elements were removed
- context windows: 2, 5, 10 words from each side of the target word

# Experimental setup

- Buryat Wikipedia consists of 1381 articles
- articles were lower-cased
- non-textual elements were removed
- context windows: 2, 5, 10 words from each side of the target word
- we learned 50, 100, 500-dimensional word-vector representation

# Evaluation

# Evaluation

**Semantic Similarity Datasets (English)**

- RG (Rubenstein and Goodenough, 1965)
- WordSim-353 (Finkelstein et al., 2001)
- WS-Sim (Agirre et al., 2009)
- MEN (Bruni et al., 2012)

# Evaluation

**Semantic Similarity Datasets (English)**

- RG (Rubenstein and Goodenough, 1965)
- WordSim-353 (Finkelstein et al., 2001)
- WS-Sim (Agirre et al., 2009)
- MEN (Bruni et al., 2012)

Each of these datasets is a collection of word pairs together with their similarity scores as assigned by human annotators.

# Evaluation

A model is evaluated by assigning a similarity score to each pair, sorting the pairs according to their similarity, and calculating the correlation (Spearmans $\rho$) with the human ranking.

# Evaluation

**Drawbacks were found** [4]

- high rank of associated but dissimilar words, e.g. *(singer, microphone)*
- low interrater agreement over the human assigned similarity scores
- unclear definition what are loosely related words, e.g. *sim(smart, dumb)=0.55 and sim(winter,summer) = 2.38*
- rating different target words on the same scale. *(cat, pet) vs. (winter, season)*
- the evaluation measure does not consider annotation decisions reliability

[4] Oded Avraham, Yoav Goldberg. Improving Reliability of Word Similarity Evaluation by Redesigning Annotation Task and Performance Measure

# Evaluation

**Hypernym-Hyponyms**

- animal - cat, dog
- color - white, black
- fruit - apple, banana

# Evaluation

## Our Solution

- the pairs of hypernyms-hyponyms are used for evaluation (we took  30 hypernyms with 3-5 corresponding hyponyms)

# Evaluation

## Our Solution

- the pairs of hypernyms-hyponyms are used for evaluation (we took  30 hypernyms with 3-5 corresponding hyponyms)
- the pair of hypernym with corresponding hyponym (e.g animal-cat) were used as positive pairs with score 1

# Evaluation

## Our Solution

- the pairs of hypernyms-hyponyms are used for evaluation (we took  30 hypernyms with 3-5 corresponding hyponyms)
- the pair of hypernym with corresponding hyponym (e.g animal-cat) were used as positive pairs with score 1
- the pair of hypernym with hyponym of different hypernym (e.g animal-summer) were used as negative pairs with score 0

# Evaluation

## Our Solution

- the pairs of hypernyms-hyponyms are used for evaluation (we took  30 hypernyms with 3-5 corresponding hyponyms)
- the pair of hypernym with corresponding hyponym (e.g animal-cat) were used as positive pairs with score 1
- the pair of hypernym with hyponym of different hypernym (e.g animal-summer) were used as negative pairs with score 0
- then we calculated Spearman correlation between gold standard 0-1 vector and vector of cosine similarities

# Evaluation

## Stemming Results

- As a baseline approach for comparison we used PMI

# Evaluation

## Stemming Results

- As a baseline approach for comparison we used PMI
- The PMI performance score without the stemming was 0.515

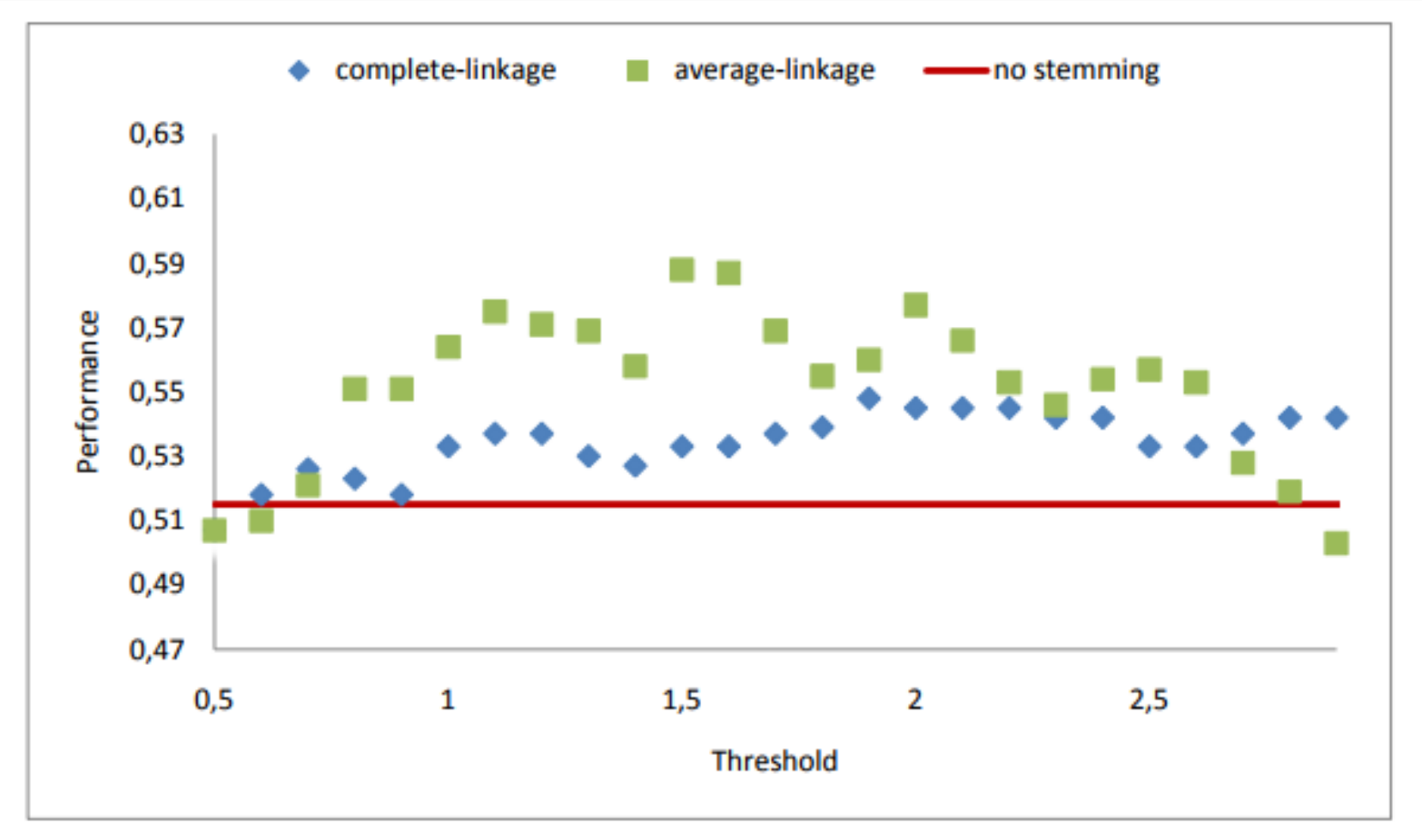
# Evaluation

## Stemming Results

- As a baseline approach for comparison we used PMI
- The PMI performance score without the stemming was 0.515
- The preliminary results showed that complete-linkage and average-linkage approaches highly outperformed the single-linkage clustering

# Evaluation

## Stemming Results



# Evaluation

**Stemming Results**

- The average-linkage clustering outperformed the complete-linkage clustering
- The average-linkage clustering achieved its best results at 1.5

# Evaluation

| | 2 | 5 | 10 |
|---|---|---|---|
| PMI | .517 | .515 | .528 |
| PMI with stemming | .585 | .588 | .611 |
| Smoothed PMI ($\alpha = 0.75$) | .555 | .571 | .599 |

- Stemming substantially improve results
- Smoothed PMI was shown to outperform traditional PMI on English Wikipedia corpus [5], however it lost to traditional PMI when small corpus was used.

[5] Levy, O., Goldberg, Y., Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings.

# Evaluation

| | 50 | 100 | 500 |
|---|---|---|---|
| CBOW | .042 | .038 | .011 |
| SGNS | .293 | .290 | .280 |

- The results justified that Skip-Grams approach works much better on the semantic tasks [6]
- Our findings confirmed that SGNS outperforms CBOW on small datasets [7]

[6] Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient estimation of word representations in vector space

[7] Mikolov, T., Le, Q., Sutskever, I. (2013). Exploiting similarities among languages for machine translation

# Evaluation

| | 50 | 100 | 500 |
|---|---|---|---|
| CBOW | .042 | .038 | .011 |
| SGNS | .293 | .290 | .280 |
| GloVe | .363 | .363 | .390 |

- The results confirmed that GloVe outperforms word2vec on similarity tasks when trained on small corpus[8]

[8] Pennington J, Socher R, Manning CD. Glove: Global Vectors for Word Representation

# Evaluation

| | 50 | 100 | 500 |
|---|---|---|---|
| GloVe | .363 | .363 | .390 |
| SVD | .690 | .722 | .691 |
| Weighted SVD ($p = 0.5$) | .696 | .714 | .680 |

- SVD outperformed word2vec algorithms, GloVe and PMI
- Weighted SVD did not change the performance significantly

Thank you!