Text Mining Seminar

GloVe: Global Vectors for Word Representation

Munich, March 25, 2018

Department of Statistics
Ludwig Maximilian University of Munich

Degree course: M.Sc. Statistics

Student:

Daniel Schalk
(11470019)

Advisor:

Prof. Dr. Christian Heumann

Contents

1 Introduction

An important thing when it comes to text mining is to create word embeddings. An important technique within the context of text mining is to map words to vectors. Those so called word embeddings are used for further analyses or more general as features in statistical models. GloVe is a technique to create word embeddings out of a given corpus. A very interesting thing is that we want to learn from text without having labels. Hence, we have an unsupervised task.

Word vectors should be able to represent the human understanding of text in a very basic way. That means that word vectors should be able to display the analogy of different words as well as general semantic questions. How this is achieved shows image .
- word vectors (image + explamation)
In the following we want to discuss some important topics related to GloVe. After deriving the model with a more theoretical point of view in section 2. In section 3 we also want to take a look at how to evaluating given word vectors which is quite interesting since we handle an unsupervised task. After that we take a short look at the data and common sources for text corpora in section 4.

Then we know how to evaluate word vectors and have an idea about the data. With that knowledge we may ask how different methods to create word embeddings or different data sources influences the quality of the word vectors. This is discussed shortly in chapter 5. After that we take a short look how we can use GloVe embeddings for further text classification techniques followed from a small conclusion in section 6.

2 GloVe Model

2.1 Terminology

Notation	Symbol	Description
Vocabulary	V	Set of unique words of the given corpus
Word vector	w (w_i)	Word vector (of corresponding word i)
Context vector	$\tilde{w}$ ($\tilde{w_i}$)	Context vector (of corresponding word i)
Dimension	d	Dimension of word vectors w
Word-word co-occurence matrix	$X \in \mathbb{R}^{\|V\| \times \|V\|}$	Matrix of context counts X_{ij}
Loss function		
Empirical risk		

Table 2.1: Overview about the used terminology.

2.2 Word-Word Co-Occurence Matrix

Base of the model is the word-word co-occurrence matrix $X \in \mathbb{R}^{|V| \times |V|}$, where $|V|$ is the number of different words which occurs within a given corpus. In X every entry X_{ij} describes how often word j occurs in context of word i with a given window size. Therefore we have the following properties:

- $X_i = \sum_k X_{ik}$: Number of words which occur in the context of i.
- $P_{ij} = P(j|i) = X_{ij}/X_i$: "Probability" of word j occurs in context of i.
- The ratio P_{ik}/P_{jk} describes if word k is more related to ...
  - ... word i if $P_{ik}/P_{jk} > 1$
  - ... word j if $P_{ik}/P_{jk} < 1$
  - ... or similar related if $P_{ik}/P_{jk} \approx 1$

Example: word-word co-occurenece Matrix

To understand how an entry of the word-word co-occurence matrix is computed we take a look at the following sample corpus:

$$\text{A D C E A D E B A C E D} \;\Rightarrow\; V = \{\text{A B C D E}\},\; |V| = 5$$

Furthermore, we need to specify a window size which indicate how much words we want to look at around a specific word. We choose a window size of 2 (the 2 words of the either side) for this example. With the given corpus and the window size we now compute the counts of how often word D occurs in context of word A (X_{AD}):

A D C E A D E B A C E D ⇒ D occurs 1 time
A D C E A D E B A C E D ⇒ D occurs 1 time
A D C E A D E B A C E D ⇒ D occurs 0 times

⇒ D occurs 2 times in context of A: $X_{AD} = 2$

Finally, we can do this for every word combination of $(i, j) \in V \times V$ and calculate the ratio P_{AD}:

$$X = \begin{array}{c|ccccc} & \text{A} & \text{B} & \text{C} & \text{D} & \text{E} \\ \hline \text{A} & 0 & 1 & 3 & 2 & 4 \\ \text{B} & 1 & 0 & 1 & 1 & 1 \\ \text{C} & 3 & 1 & 0 & 2 & 2 \\ \text{D} & 2 & 1 & 2 & 0 & 4 \\ \text{E} & 4 & 1 & 2 & 4 & 0 \end{array} \quad \Rightarrow \quad P_{\text{AD}} = X_{\text{AD}}/X_{\text{A}} = 2/10$$

Sidenote:

Some packages (like `R`'s `text2vec`) have the option to give a weight vector with as many weights as the window size. It is then common to weight context words closer to the inspected word higher than words which aren't that close.

2.3 The Model

GloVe was introduced by Jeffrey Pennington, Richard Socher and Christopher D.Manning [PSDM14]:

> "... *GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.*"

Remember: We want to create word vectors $w_i \in \mathbb{R}^d$ and want to use the ratio P_{ik}/P_{jk} since the ratio is more appropriate to take words into context.

The idea is to take a function F which takes the three words i, j and k, given by using the ratio, and map F to that ratio:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

Here w_i, w_j are context vectors while $\tilde{w}_k$ is the context vector. F is unknown at this stage and basically can be any function. But we remember that we want to keep the linear structure of the vector space $\mathbb{R}^d$. So it is important to choose a function F which is able to hold the linear structure. For instance we can choose F to be a neural net. But that wouldn't guarantee that F keeps the linear structure.

We now parameterize every word vectors, therefore we have $d \cdot |V|$ parameter to estimate to get word vectors. But how should we estimate those parameters without a specific function F? There are several steps to come up with a solution which fulfilles the desierd behaviour:

1. Reduce number of possible outputs of F by taking $w_i - w_j$ instead of two vectors w_i and w_j:
$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$
2. We want to keep the linear structure between the word vectors and context vector:
$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$$
3. Keep exchange symmetry. We want be able to exchange word vectors w with context vectors $\tilde{w}$ without getting different results. To obtain this property we apply to tricks:
   a) We restrict F to be a homomorphism between $(\mathbb{R}, f) = (\mathbb{R}, +)$ and $(\mathbb{R}_+, g) = (\mathbb{R}_+, \cdot)$. That means we expect F to fulfill:
$$F(a+b) = \underbrace{F(f(a,b)) = g(F(a), F(b))}_{\text{Definition of homomorphism}} = F(a)F(b)$$
   With $a = w_i^T \tilde{w}_k$ and $b = -w_j^T \tilde{w}_k$.
   We notice, that the upper equation implies a functional equation which has just one solution:
$$F = \exp_a$$
   We choose $a = \exp(1)$, therefore $F = \exp$.
$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \exp\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{\exp(w_i^T \tilde{w}_k)}{\exp(w_j^T \tilde{w}_k)} = \frac{P_{ik}}{P_{jk}}$$
$$\begin{aligned} &\Rightarrow \ \exp(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i} \\ &\Leftrightarrow w_i^T \tilde{w}_k = \log(X_{ik}) - \log(X_i) \\ &\Leftrightarrow w_i^T \tilde{w}_k + \log(X_i) = \log(X_{ik}) \end{aligned}$$
   But: How to handle $X_{ik} = 0$? We handle this later.

   b) Since $\log(X_i)$ is independent of k we can put this in a bias term. This bias term can be decomposed into a term b_i from w_i and $\tilde{b}_k$ from $\tilde{w}_k$:
$$\Rightarrow \ w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik})$$
   This is now an model equation which we can use for training the word vectors. We also note, that we have transformed the unsupervised task into a supervised task which we know how to handle.

Sidenote:

Point 2. looks a bit like the model equation of a generalized linear model (glm) with linear predictor $(x_i - w_j)^T \tilde{w}_k$, response function F and response P_{ik}/P_{jk}. The glm takes a linear structure and transforms the output on an interval of plausible values. This also holds for GloVe by using a homomorphism as F which preserves the structure between the two used algebraic structures. Therefore, it is possible to keep the linearity of the used vector space $\mathbb{R}^d$.

Model Equation

A problem appears for $X_{ik} = 0$. This is definitely the case since X is a sparse matrix. We don't want to drop the sparsity due to convenient memory handling. Therefore, we use an additive shift in the logarithm:

$$\log(X_{ik}) \;\to\; \log(X_{ik} + 1)$$

This maintains the sparsity:

$$X_{ik} = 0 \;\Rightarrow\; \log(X_{ik} + 1) = \log(1) = 0$$

The final model equation which we use for training is

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik} + 1)$$

Empirical Risk

Kurz Loss funktion erwähnen!

To estimate the word vectors GloVe uses a weighted least squares approach:

$$J = \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} f(X_{ij}) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij} + 1) \right)^2$$

[PSDM14] proposed as weight function:

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha}, & x < x_{\max} \\ 1, & x \geq x_{\max} \end{cases}$$

Plotten wie funktion aussieht!

Using this function also introduces two new hyperparameters α and $x_{\max}$. The ordinary way to find good values for α and $x_{\max}$ would be to use a tuning method. The problem now is that for all tuning methods we need to evaluate the model. Since we want to handle an unsupervised task evaluating isn't straightforward.

Algorithm used for Fitting

Basically, GloVe uses a gradient descend technique to minimize the objective J. Nevertheless, using ordinary gradient descend would be way too expensive in practice. A more appropriate gradient based method is used here called adaptive gradient descent (AdaGrad). A short description of the most important points of AdaGrad transferred to text mining are:

- Individual learning rate in each iteration.
- Adaption of the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameter.
- Well-suited for sparse data, improve robustness of (stochastic) gradient descent.
- For GloVe, infrequent words require much larger updates than frequent ones.

For a more detailled explanation see [Rud16].

3 Evaluating Word Vectors

- Distances
- Curse of Dimensionality

4 About the Data

5 Real Word Vectors

TEXT HERE

6 Outlook and Conclusion

6.1 Outlook

6.2 Conclusion

List of Figures

List of Tables

Bibliography

[PSDM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[Rud16] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.