UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ALEXANDRE TADEU SALLE

The Role of Negative Information when Learning Dense Word Vectors

Thesis presented in partial fulfillment of the requirements for the degree of Doctor of Computer Science

Advisor: Prof. Dr. Aline Villavicencio

Porto Alegre
December 2021

CIP — CATALOGING-IN-PUBLICATION

Salle, Alexandre Tadeu

The Role of Negative Information when Learning Dense Word Vectors / Alexandre Tadeu Salle. – Porto Alegre: PPGC da UFRGS, 2021.

133 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2021. Advisor: Aline Villavicencio.

1. Word vectors. 2. Matrix factorization. 3. Natural language processing. I. Villavicencio, Aline. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos André Bulhões
Vice-Reitora: Prof^{a}. Patricia Pranke
Pró-Reitor de Pós-Graduação: Prof. Celso Giannetti Loureiro Chaves
Diretora do Instituto de Informática: Prof^{a}. Carla Maria Dal Sasso Freitas
Coordenador do PPGC: Prof. Claudio Rosito Jung
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

ABSTRACT

By statistical analysis of the text in a given language, it is possible to represent each word in the vocabulary of the language as an m-dimensional *word vector* (also known as a word embedding) such that this vector captures semantic and syntactic information. Word embeddings derived from unannotated corpora can be divided into (1) counting methods which perform factorization of the word-context cooccurrence matrix and (2) predictive methods where neural networks are trained to predict word distributions given a context, generally outperforming counting methods. In this thesis, we hypothesize that the performance gap is due to how counting methods account for – or completely ignore – negative information: word-context pairs where observing one is informative of *not* observing the other, mathematically formulated as two events (words and contexts) having negative Pointwise Mutual Information. We validate our hypothesis by creating an efficient factorization algorithm, LexVec, scalable to web-size corpora, that accounts for negative information in a principled way, closing the performance gap with predictive methods. Additionally, we show that strategies for breaking up words into smaller units (subwords) – an important technique in predictive methods for representing rare words – can be successfully adapted to LexVec. We show that the explicit nature of LexVec – having access to the underlying cooccurrence matrix – allows us to selectively filter whether to account for negative information in the factorization and to what degree, and use this filtering to isolate the impact that negative information has on embeddings. Word and sentence-level evaluations show that only accounting for positive PMI in the factorization strongly captures both semantics and syntax, whereas using only negative PMI captures little of semantics but a surprising amount of syntactic information. Finally, we perform an in-depth investigation of the effect that increasing the relative importance of negative PMI compared to positive PMI has on the geometry of the vector space and its representation of frequent and rare words. Results reveal two rank invariant geometric properties – vector norms and direction – and improved rare word representation induced by incorporating negative information.

Keywords: Word vectors. matrix factorization. natural language processing.

O Papel da Informação Negativa na Aprendizagem de Vetores Palavra Densos

RESUMO

Pela análise estatística de textos em uma dada linguagem, é possível representar cada palavra contida no vocabulário desta linguagem por meio de um *vetor palavra* m-dimensional (também conhecido como *embedding* de palavra) de forma que esse vetor capture informações semânticas e sintáticas. Embeddings de palavras podem ser derivados de corpora não-anotados por meio de (1) métodos de contagem onde é efetuada explicitamente a fatoração da matriz de coocorrência e (2) métodos preditivos onde redes neurais são treinadas para predizer distribuições de palavras dado um contexto. Nesta tese, hipotetizamos que essa diferença de desempenho é devida à forma com que métodos baseados em contagem levam em consideração – ou ignoram completamente – *informação negativa*: pares palavra-contexto, nos quais o fato de observar um é informativo para a não observação do outro, formulado matematicamente como dois eventos (palavra e contexto) possuindo Pointwise Mutual Information negativa. Validamos nossa hipótese criando um novo método e eficiente método de fatoração de matrizes, o LexVec, altamente escalável, limitado apenas por espaço em disco e núcleos computacionais, que leva em consideração a informação negativa de forma embasada, eliminando a diferença de desempenho em relação a métodos preditivos. Adicionalmente, mostramos que estratégias para quebrar palavras em unidades menores (subpalavras) – uma técnica importante em métodos preditivos para a representação de palavras infrequentes – podem ser adaptadas ao LexVec. Se aproveitando do fato que o LexVec tem acesso à matriz de coocorrência sendo fatorada, efetuamos fatorações que filtram seletivamente o uso ou não de informação negativa, estudando assim o impacto que a informação negativa tem nos embeddings de palavras. Avaliações a nível de palavra e de frases mostram que o uso exclusivo de PMI positivo na fatoração captura fortemente a semântica e sintaxe de palavras, enquanto que o uso exclusivo de PMI negativo captura pouca informação semântica porém uma quantidade surpreendente de informação sintática. Finalmente, efetuamos uma investigação profunda sobre o efeito que o aumento do peso da informação negativa (em relação a informação positiva) tem na geometria dos espaços vetoriais dos embeddings e nas representações de palavras frequentes e infrequentes. Os resultados revelam duas invariantes geométricas – a norma e a direção vetorial – e melhorias nas representações de palavras raras que são induzidas pelo uso aumentado da informação negativa.

Palavras-chave: vetores palavra. fatoração de matriz. processamento de linguagem natural.

AGRADECIMENTOS

À minha orientadora Aline pelo apoio e parceria nos altos e baixos.

Ao Marco pela colaboração e pelas discussões junto ao quadro branco.

Ao António, Helena, Viviane e Marcelo por contribuírem para a melhora deste trabalho.

Aos meus pais por me trazerem até aqui.

Às minhas irmãs por estarem comigo sempre.

Aos meus colegas de laboratório Luis e Duda, que trouxeram muita vida ao nosso pequeno aquário.

Ao meu grande amigo Gustavo. Onde estiver, sinto muito tua falta.

À Erika, amiga conselheira desde meu primeiro ano de graduação.

To my friend and mentor Shervin, a light in the darkness.

Ao Fernando e a fuga da sarjeta, e à Fabiana pela calma no final.

À Tita pelos almoços e cafés que representam tanto mais que almoços e cafés. E ao Gilbertinho rabugento.

À minha mulher Júlia, amor da minha vida, que entrou de carona nessa montanha-russa.

E acima de tudo à minha pequena Pixu, amor do pai, que me deu fôlego para terminar, dividindo o baba com as cooccurrences e word vectors em seu primeiro ano de vida. És minha inspiração para viver.

LIST OF ABBREVIATIONS AND ACRONYMS

BERT	Bidirectional Encoder Representations from Transformers
CBOW	Continuous Bag-of-Words
FT	fastText
GSem	Google Semantic Analogies
GSyn	Google Syntactic Analogies
KDE	Kernel density estimation
LR	Learning rate
LV	LexVec
mnPMI	Maximally Negative Pointwise Mutual Information
MRR	Mean Reciprocal Rank
MSR	Microsoft Research Syntactic Analogies
nPMI	Negative Pointwise Mutual Information
NLI	Natural Language Inference
NLP	Natural Language Processing
OOV	Out-of-vocabulary
PCA	Principal Component Analysis
PMI	Pointwise Mutual Information
POS	Part-of-speech
PPMI	Positive Pointwise Mutual Information
RW	Rare Word
SG	Skip-gram
SGD	Stochastic Gradient Descent
STS-B	Semantic Textual Similarity Benchmark
SVD	Singular Value Decomposition
WSNS	Window Sampling Negative Sampling

LIST OF FIGURES

Figure 1.1 2-dimensional PCA projections of LexVec Common Crawl vectors: (a) Countries (red) and capitals (blue) (b) Adverbs (red), comparatives (blue), superlatives (green). Note how the geometry captures the semantic “capital-of” and syntactic “adverb-comparative-superlative” relations through linear offsets between vectors. 17

Figure 2.1 A cooccurrence matrix constructed from two sentences. When the window is centered on “also”, the corresponding cells for “she”, “loves”, and “my” are incremented........ 27

Figure 2.2 Conversational search refinement using CoSearcher. 44

Figure 3.1 LexVec running over a single target word “dogs”, window sampled words “friend”, “loves”, “and”, “cats”, and negatively sampled words “the” and “computer”. Highlighted $f(M)$ squares on the right represent lookups to the transformed PMI values between target word and every window sampled and negatively sampled contexts. Vectors being updated by SGD: (1) Red row: target word vector. (2) Purple columns: window sampled context vectors. (3) Green columns: negatively sampled context vectors. Equation (3.1) is minimized through SGD for each lookup........ 49

Figure 3.2 Histogram (bin width equal to 0.2) of 10^5 values *not in mnPMI* sampled from PMI_{wiki5}. The negative spectrum of PMI are the values to the left of the dashed line ($\sim 22.2\%$ of sampled values), which are collapsed to 0 when using the popular PPMI association measure. Note that we exclude mnPMI (sample only non-zero cooccurrences) otherwise the graph would be a single vertical line at -2 (if graphing $CPMI(-2)$) since 99.07% of values in M_{wiki5} are 0. 54

Figure 3.3 Example MapReduce steps used to construct the F stream used by the external memory variant of LexVec. 55

Figure 6.1 The distribution (bin width equal to 0.2) of sampled $CPMI_{wiki5}(-2)$ values when using window sampling and negative sampling. Histogram from Figure 3.2 included for comparison. These window sampling and negative sampling distributions of PMI values correspond to the reconstruction error weights $\lambda_{ws}, \lambda_{ns}$ in Equation (3.8) as a function of PMI. Negative sampling assigns high weights to values in nPMI, and window sampling to values in pPMI. 83

Figure 6.2 The distribution of vector L_2 norms for 100 words sampled from various frequency buckets for LexVec models using 0, 5, and 20 negative samples (LV-0, LV-5, LV-20). As the number of negative samples increases, the norms become more rank invariant, with means of the different buckets becoming increasingly closer and variance decreasing. The norm distributions for Skip-gram, GloVe, and SVD are shown for comparison. 85

Figure 6.3 In solid lines, the distributions of cosines (Equation (2.11)) of word vectors from different frequency buckets with the mean vector of word vectors from all buckets, and in dashed lines the distributions of cosines of *context* vectors from different frequency buckets with this *same mean word vector*. As the number of negative samples increases, word vectors increasingly point in the same direction and word-context vectors point in the opposite direction. ..86

Figure 6.4 (a) Word similarity results: RW improves as the number of negative samples increases. There is a small but consistent drop in performance on SimLex when increasing the number of negative samples. Scores in WSRel do indeed increase, but there is no clear drop in similarity scores in WSSim. (b) GSem/GSyn/MSR: Performance increases as the number of negative samples increases. Normalization has a minor effect, whereas premise exclusion is critical to performance. (c) GSem/GSyn/MSR Rare words: Consistent improvements in all conditions because rare words are better represented. (d) GSem/GSyn Frequent words: With both heuristics or only the normalization heuristic performance is nearly constant. For conditions without premise exclusion heuristic, there is remarkable improvement in analogy performance as negative samples increase, especially for semantic analogies. Metrics: for (a) Spearman rank correlation ($\times 100$), for (b,c,d), Accuracy. 90

Figure 6.5 (a, b, c): Word vector norm distributions for Skip-gram models with 1, 2 and 5 negative samples (SG-1, SG-2, SG-5). In contrast to the same figures for the LexVec models (Figures 6.2a to 6.2c), it is not clear if increasing the number of negative samples increases the rank invariance of vector norms. (d, e): Simple moving average of period 100 (with accompanying scatter plot of points used in calculating this average) of word vector norms as a function of word rank. We include additional SG-k and LV-k models for various k to make the trend clear. Although the shape of (d) and (e) is different, it is evident for both Skip-gram and LexVec that as the number of negative samples increases the functions become flatter, indicating the increase in rank invariance of vector norms. SVD displays the extreme case where nearly all loss weight is assigned to negative information, leading to ideal rank invariance. (f, g, h): Solid lines show distribution of cosines between vectors of words of different frequency buckets with the mean vector of all buckets, and dashed lines the of cosines of context vectors of these same words with the mean word vector, as in Figures 6.3a to 6.3c. Word vectors increasingly point in the same direction and word vectors point away from context vectors as the number of negative samples increases. 95

Figure C.1 Same as Figure 3.2, but using **positional contexts and symmetric context window of fixed size 2**. 119

Figure C.2 Same as Figure 6.1, but using **positional contexts and symmetric context window of fixed size 2**. 119

Figure C.3 Same as Figure 6.2, but using **positional contexts and symmetric context window of fixed size 2**. 120

Figure C.4 Same as Figure 6.3, but using **positional contexts and symmetric context window of fixed size 2**. 121

Figure C.5 Same as Figure 6.4, but using **positional contexts and symmetric context window of fixed size 2**. 122

Figure C.6 Same as Figure 6.5, but using **positional contexts and symmetric context window of fixed size 2**. 125

Figure C.7 Same as Figure 3.2, but using **subword information**, positional contexts and symmetric context window of fixed size 2. 127

Figure C.8 Same as Figure 6.1, but using **subword information**, positional contexts and symmetric context window of fixed size 2. 127

Figure C.9 Same as Figure 6.2, but using **subword information**, positional contexts and symmetric context window of fixed size 2. 128

Figure C.10 Same as Figure 6.3, but using **subword information**, positional contexts and symmetric context window of fixed size 2. 129
Figure C.11 Same as Figure 6.4, but using **subword information**, positional contexts and symmetric context window of fixed size 2. 130
Figure C.12 Same as Figure 6.5, but using **subword information**, positional contexts and symmetric context window of fixed size 2. 133

LIST OF TABLES

Table 2.1 Sample values from M_{wiki5} and corresponding PMI values. 29
Table 2.2 Set sizes computed over 10^5 cells sampled randomly from PMI_{wiki}, such that nPMI+pPMI sum to 100%. 30
Table 2.3 Word similarity examples. 36
Table 2.4 Word analogy samples, having the form “*a* is to *b* as *c* is *d*”. 38
Table 2.5 Samples from textual similarity datasets used in our evaluations. 40
Table 2.6 Text classification tasks. 41
Table 2.7 Natural language inference (NLI) datasets. 42

Table 3.1 Principled reconstruction error weighting and the models that respect them. Top row contains the Skip-gram model that performs implicit matrix factorization, and the bottom rows are models that perform explicit matrix factorization. 50

Table 4.1 Spearman rank correlation on word similarity tasks. 61
Table 4.2 Results on word analogy tasks, given as percent accuracy. 61
Table 4.3 Leave-two out cross-validation results on the fMRI task. Numbers are percent accuracy. 62
Table 4.4 Spearman rank correlation on word similarity tasks when using the external memory LexVec. 63
Table 4.5 Results on word analogy tasks, given as percent accuracy when using the external memory LexVec. 63
Table 4.6 Spearman rank correlation on word similarity tasks when incorporating subword information. 64
Table 4.7 Results on word analogy tasks, given as percent accuracy when incorporating subword information. 64
Table 4.8 We generate vectors for OOV using subword information and search for the nearest (cosine distance) words in the embedding space. The LV-M segmentation for each word is: $\{\langle$hell, o, o, o$\rangle\}$, $\{\langle$marvel, i, cious$\rangle\}$, $\{\langle$louis, ana$\rangle\}$, $\{\langle$re, re, read$\rangle\}$, $\{\langle$ tu, z, read$\rangle\}$. We omit the LV-N and FT n-grams as they are trivial and too numerous to list. 65
Table 4.9 Hyperparameter search on the SST5 dev set. Unless otherwise specified, models use PPMI, $lr = 2.5\text{e-}2$, positional contexts and $\alpha_{cds} = 1$. 66
Table 4.10 Comparing merging strategies on word similarity tasks. Numbers are Spearman’s rank correlation x100. 68
Table 4.11 Comparing merging strategies on word analogy tasks. Numbers are percent accuracy. 68
Table 4.12 Comparing merging strategies on classification tasks. Numbers are percent accuracy. 68
Table 4.13 Comparing merging strategies on NLI tasks. Numbers are percent accuracy. 69
Table 4.14 Comparing merging strategies on textual similarity tasks. Numbers are Pearson’s correlation coefficient. 69
Table 4.15 Comparing merging strategies on textual similarity tasks using regression and cosine (has suffix -Cos). Numbers are Pearson’s correlation coefficient. 69
Table 4.16 Percent accuracy on text classification tasks. 70
Table 4.17 Performance on textual similarity tasks using cosine similarity. Numbers are Pearson’s correlation coefficient x100. 71

Table 4.18 Comparing merging strategies on textual similarity tasks using regression and cosine (has suffix -Cos). Numbers are Pearson's correlation coefficient x100. 71
Table 4.19 Results on POS tagging and syntactic probing tasks. All numbers represent percent accuracy. 71
Table 4.20 Results on natural language inference tasks. Numbers represent percent accuracy. 72
Table 4.21 Facet ranking evaluation using precision-at-1 (P@1) and mean reciprocal rank (MRR) using unsupervised mean bag-of-vectors. 73
Table 4.22 Facet ranking evaluation using precision-at-1 (P@1) and mean reciprocal rank (MRR) using trained deep averaging network (DAN). 73
Table 4.23 Comparing lexical semantic performance between static word embeddings and contextualized word embeddings on word similarity tasks. Numbers are Spearman rank correlation x100. 75
Table 4.24 Comparing lexical semantic performance between static word embeddings and contextualized word embeddings on word analogy tasks. Numbers are percent accuracy. Missing scores are due to Lenci et al. (2021) only reporting aggregate (GSem+GSyn) Google analogy scores. 75
Table 4.25 Comparing text classification performance between unsupervised sentence embeddings using static word embeddings and state-of-the-art sentence embeddings based on Transformer models. Numbers are percent accuracy. 76
Table 4.26 Comparing textual similarity performance between unsupervised sentence embeddings using static word embeddings and state-of-the-art sentence embeddings based on Transformer models. Numbers are Pearson correlation coefficient x100. 77

Table 5.1 Performance on tasks focused on semantics or syntax by models that use only positive information (p*), negative information (n*), or both (no prefix), and the random baseline. Using negative information alone performs far better than the random baseline, especially on the syntactic tasks. Metrics: Spearman rank correlation ($\times 100$) for SimLex and RW word similarity; Pearson correlation for STS-B; % accuracy for GSem/GSyn/MSR word analogies, POS tagging and WC, Dep, TopC probing tasks. Best result for each column in bold, second best underlined. 80

Table 6.1 Set (rows) membership of samples for various sampling methods (columns). Cell values are the percentage of samples for a given method that fall within a set, such that nPMI+pPMI sum to 100. **Full**: computed over all cells in PMI_{wiki5}. **Hist**: computed over the same 10^5 sampled pairs plotted in Figure 3.2. **WS** and **NS**: computed over the same 10^5 sampled pairs used in Figure 6.1 using window sampling (ws) and negative sampling (ns). Observe that window sampling is heavily skewed towards pairs in pPMI, and negative sampling heavily skewed towards pairs in nPMI. 83
Table 6.2 Nearest neighbors of words sampled from SimLex. Subscripts denote the percentile rank of a word (0.0 is most frequent word, 100.0 is rarest). All models exhibit semantically coherent neighbors, with the exception of the SG model which has some unrelated *intruders*, notably for the word "interest". 88
Table 6.3 Same as Table 6.2, but with words sampled from RW. LV-0 neighbors have no clear semantic connection to the target word. In the LV-20 model, have clear semantic relations to the targets. GloVe behaves like LV-0, and Skip-gram and SVD like LV-20. 89

Table 6.4 Comparing task performance of LexVec, Skip-gram, GloVe, and SVD models in Word similarity and analogy tasks (-: no heuristics excluded, -N: norm heuristic excluded, -P: premise heuristic excluded, -N-P: both heuristics excluded). The LV-0 model is similar to the GloVe model: strong SimLex results, weaker RW results, and similar across all analogy evaluations, barring MSR where GloVe is considerably stronger than all other models. Analogously, the LV-20 model is similar to the Skip-gram and models: strong SimLex and RW results, and consistently strong performance on all analogy evaluations. SVD has the lowest performance, but note that it suffers the smallest drop when the -P heuristic is excluded. Metrics: Spearman rank correlation ($\times 100$) for SimLex and RW, Accuracy for analogies. 93

Table B.1 Percentile rank statistics for words in the datasets used Chapters 5 and 6. Note: μ- mean, σ- standard deviation, Qk- k-th quartile. 117

Table C.1 Same as Table 6.1, but using **positional contexts and symmetric context window of fixed size 2**. 118
Table C.2 Same as Table 5.1, but using **positional contexts and symmetric context window of fixed size 2**. 119
Table C.3 Same as Table 6.2, but using **positional contexts and symmetric context window of fixed size 2**. 123
Table C.4 Same as Table 6.3, but using **positional contexts and symmetric context window of fixed size 2**. 124
Table C.5 Same as Table 6.4, but using **positional contexts and symmetric context window of fixed size 2**. 126
Table C.6 Same as Table 6.1, but using **subword information**, positional contexts and symmetric context window of fixed size 2. 126
Table C.7 Same as Table 5.1, but using **subword information**, positional contexts and symmetric context window of fixed size 2. 127
Table C.8 Same as Table 6.2, but using **subword information**, positional contexts and symmetric context window of fixed size 2. 131
Table C.9 Same as Table 6.3, but using **subword information**, positional contexts and symmetric context window of fixed size 2. 132
Table C.10 Same as Table 6.4, but using **subword information**, positional contexts and symmetric context window of fixed size 2. 133

CONTENTS

1 INTRODUCTION..16
1.1 Contributions..19
1.1.1 Reconstruction error weighting..19
1.1.2 Positional contexts..20
1.1.3 Subwords..20
1.1.4 Association measures..21
1.1.5 Downstream Performance..21
1.1.6 An Application in Information Retrieval..22
1.1.7 Information contained positive/negative PMI..23
1.1.8 Vector space geometry..24
1.2 Relevant work during PhD..24
1.2.1 Publications..24
1.2.2 Software and Models..26
2 BACKGROUND..27
2.1 Cooccurrence Matrix and Positional Contexts..27
2.2 Pointwise Mutual Information (PMI)..28
2.3 Related Work..31
2.3.1 Cooccurrence Matrix Transformations..31
2.3.2 Matrix Factorization..31
2.3.3 Subword information..33
2.3.4 Analysis..33
2.3.5 Contextualized Word Embeddings and Language Models..34
2.4 Evaluation..35
2.4.1 Intrinsic Evaluation..35
2.4.1.1 Word similarity..35
2.4.1.2 Word analogy..37
2.4.1.3 fMRI..37
2.4.2 Downstream Evaluation..39
2.4.2.1 Text Similarity..39
2.4.2.2 Text Classification..39
2.4.2.3 Natural Language Inference..41
2.4.2.4 Syntactic Tasks..41
2.4.2.5 Facet Ranking..42
2.5 Conclusion..43
3 THE LEXVEC MODEL..45
3.1 Principled Reconstruction Error Weighting..45
3.2 LexVec..47
3.3 Positional Contexts..52
3.3.1 Combining Word and Context Vectors..52
3.4 Spectrum of PMI..53
3.4.1 Preserving the spectrum of negative information..53
3.5 Scaling to Larger Corpora..54
3.6 Incorporating Subword Information..58
3.7 Conclusion..59
4 EVALUATION..60
4.1 Base Model..60
4.1.1 Materials..60

4.1.2 Results.....61
4.1.2.1 LexVec.....61
4.1.2.2 Positional Contexts.....62
4.2 Scaling using External Memory.....62
4.2.1 Materials.....62
4.2.2 Results.....62
4.3 Subwords.....63
4.3.1 Materials.....63
4.3.2 Results.....64
4.4 Downstream Performance.....66
4.4.1 Hyperparameter Selection.....66
4.4.2 Materials.....67
4.4.3 Merging.....68
4.4.4 Text Classification.....70
4.4.5 Text Similarity.....70
4.4.6 Syntactic Tasks.....71
4.4.7 Natural Language Inference.....72
4.4.8 Facet Ranking.....72
4.5 Comparison to Transformers.....75
4.5.1 Lexical Semantics.....75
4.5.2 Sentence Embedding.....76
4.6 Conclusion.....78
5 SEMANTICS/SYNTAX IN NEGATIVE/POSITIVE INFORMATION.....79
5.1 Materials.....79
5.1.1 Results.....80
5.2 Conclusion.....81
6 NEGATIVE INFORMATION AND GEOMETRY.....82
6.1 Increasing relative weight of negative information.....82
6.2 Materials.....84
6.3 Results.....84
6.4 Connection to other models.....91
6.4.1 Materials.....92
6.4.2 Results.....94
6.4.2.1 Skip-gram.....94
6.4.2.2 GloVe and SVD.....96
6.5 Conclusion.....98
7 CONCLUSIONS AND FUTURE WORK.....99
7.1 Future Work.....101
FUNDING.....102
REFERENCES.....103
APPENDIX A — EXTENDED ABSTRACT IN PORTUGUESE.....113
APPENDIX B — RANK STATISTICS.....117
B.1 Dataset Rank Statistics.....117
APPENDIX C — COMPLEMENTARY EXPERIMENTS.....118
C.1 Fixed Window Size $l = 2$ and Positional Contexts.....118
C.2 Subword Information.....118

1 INTRODUCTION

One approach to represent words in a given language's vocabulary is through low-dimensional *word vectors* (also known as word embeddings) such that these vectors capture semantic and syntactic information. The similarity of two vectors serves as a proxy for the similarity between the corresponding words. These low dimensional (dense) word vectors have become a key component in modern NLP systems for information retrieval, language modeling, parsing, sentiment classification, and many others (LEE; CHUANG; SEAMONS, 1997; BENGIO et al., 2003; SOCHER et al., 2013a; TURNEY; LITTMAN, 2003; TURNEY; PANTEL, 2010). Figure 1.1 shows a 2-dimensional PCA projection of a set of words vectors drawn from our LexVec Common Crawl vectors (the LexVec model is presented in Chapter 3), illustrating how a vector space model represents meaning. Words that are semantically similar cluster in space, and the relations "capital-of" and "adverb-comparative-superlative" are represented by linear offsets in vector space.

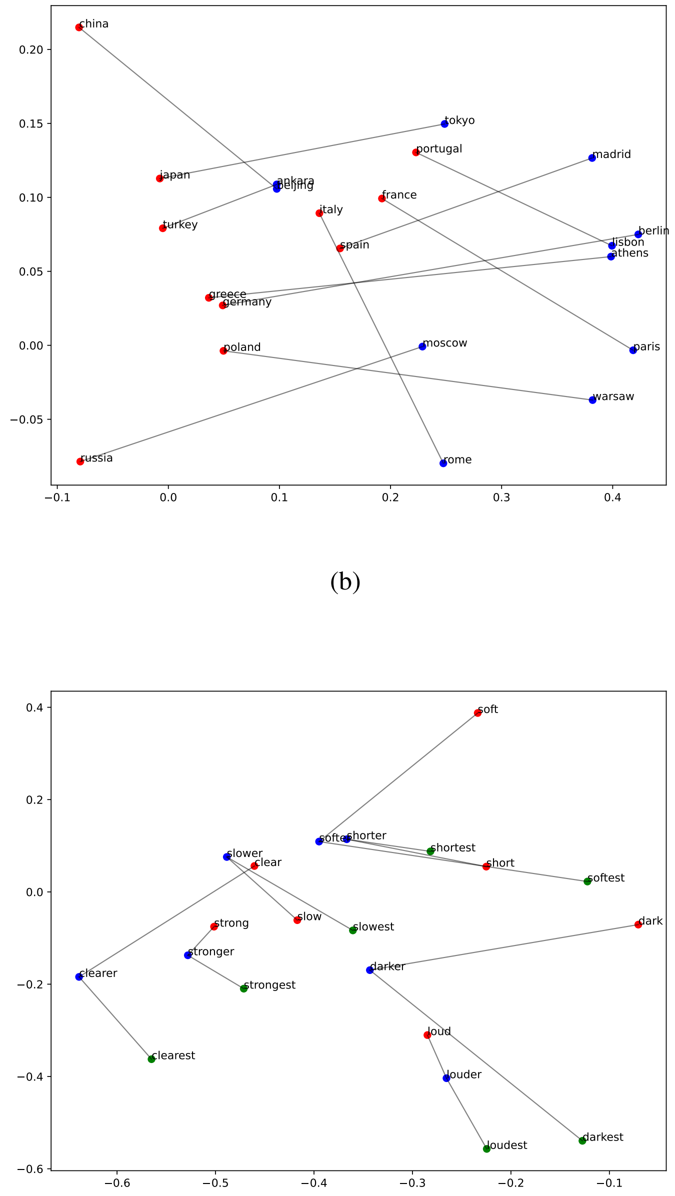
These word vectors are usually created in an unsupervised manner from corpus statistics by exploiting the distributional hypothesis: that similar words appear in similar contexts (HARRIS, 1954), paraphrased by Firth (1957) as "a word is characterized by the company that it keeps". A word-to-context *cooccurrence matrix* is constructed by counting the number of times each word in the vocabulary is observed with a given context. The context could be other words (SCHUTZE, 1992; LUND; BURGESS, 1996) or documents (DEERWESTER et al., 1990). When considering words as context (the focus of this thesis), two words are said to cooccur, and their count incremented, if they appear within some given distance of each other. This distance is referred to as the *context window*.

The rows of this highly sparse matrix can be used directly as word vectors, but this is problematic from a computational standpoint, where even with efficient sparse matrix methods, vector operations are costly and storing the matrix in memory is prohibitive for commonly used vocabulary sizes.

Sparsity also poses an issue for measuring similarity between vectors: the compression induced by techniques that perform matrix factorization such as latent semantic analysis (DEERWESTER et al., 1990) unveil similarities in dense latent space which might otherwise be missing in the surface/observed space of sparse vectors. Performing matrix factorization has become the norm for generating word vectors from cooccurrence counts (SCHÜTZE, 1993; TURNEY; PANTEL, 2010; PENNINGTON; SOCHER;

Figure 1.1 – 2-dimensional PCA projections of LexVec Common Crawl vectors: (a) Countries (red) and capitals (blue) (b) Adverbs (red), comparatives (blue), superlatives (green). Note how the geometry captures the semantic “capital-of” and syntactic “adverb-comparative-superlative” relations through linear offsets between vectors.

(a)



MANNING, 2014).

A parallel line of work uses neural networks to train dense word vectors, foregoing the construction of a cooccurrence matrix. Bengio et al. (2003) train a language model using trainable low-dimensional vectors as input, where each input word has its own vector. This is in stark contrast to prior work that used static one-hot encoding vectors for each word. The intuition is that the network will learn similar vectors for similar concepts, and so learning to make predictions for one input word *transfers* to making predictions about other words having similar vectors. This idea is the realization in NLP of *distributed representations* (HINTON, 1986), which is why word vectors are commonly referred to as *distributed word representations*.

Ten years later, word embeddings from the word2vec (MIKOLOV et al., 2013) package took the NLP community by storm with a number of striking results. Its Skip-gram (SG) model strips away the non-linearities of the Bengio et al. (2003) language model, and rather than predicting the next token in a sequence, it predicts the surrounding context words given a *center* word. Its other model, Continuous Bag-of-Words (CBOW), instead predicts the center word given the sum of context words. From here onward we focus on the SG model since it consistently outperforms CBOW (LEVY; GOLDBERG; DAGAN, 2015) and has a nice mathematical equivalence (LEVY; GOLDBERG, 2014) to implicit matrix factorization. This simpler model, combined with the introduction of negative sampling in a followup paper (MIKOLOV et al., 2013), allowed word embeddings to be trained on corpora multiple orders larger than previous neural network models. Evaluation on word similarity tasks and the jointly introduced (MIKOLOV et al., 2013)[1] task of solving word analogies – such as "Paris is to France as Tokyo is to ?" – using only vector arithmetic showed significant performance gains over older models based on cooccurrence counting and matrix factorization. This gap in performance is highlighted by Baroni, Dinu and Kruszewski (2014), which create the nomenclature we use in this thesis for referring to models based on cooccurrence matrix construction and factorization as *counting*, and methods such as those in word2vec that are trained to predict words as *predictive*.

Among the most prominent counting methods were the truncated Singular Value Decomposition of the Positive Pointwise Mutual Information (PPMI) matrix (NIWA; NITTA, 1994; BULLINARIA; LEVY, 2007; LEVY; GOLDBERG; RAMAT-GAN, 2014),

---

[1] Often unreferenced, Rumelhart and Abrahamson (1973) proposed analogical reasoning over vectors decades earlier, but using vectors derived from human word similarity judgements rather than corpus cooccurrence.

and the GloVe model (PENNINGTON; SOCHER; MANNING, 2014). *We hypothesized they lagged the word2vec predictive methods in performance because of incorrect weighting of or totally discarding information contained in zero count cells of the cooccurrence matrix.*

Word-to-word cooccurrence matrices tend to be extremely sparse: most cells have a count of 0. For example, the matrix used throughout this thesis, constructed from the English Wikipedia, is 99.71% zeros. We believe a good matrix factorization should approximately recover these values when factor matrices are multiplied, as zero counts contain information – in this case *negative information* – that a word and context tend to *not* appear together.

In the work that follows, we make the following contributions:

1.1 Contributions

1.1.1 Reconstruction error weighting

Matrix factorization word embedding models differ in the way they prioritize information when compressing word representations into dense word vectors. Better understanding this prioritization or *weighting* of information can assist in designing better models.

(Q1) What is a principled approach to account for negative information in explicit matrix factorization, and how do existing models fail to account for it?

We motivate and define the set of *principled reconstruction error* properties that matrix factorization models should respect and situate existing models within this framework (Section 3.1).

(Q2) Can this principled approach be implemented efficiently?

We propose LexVec (SALLE; VILLAVICENCIO; IDIART, 2016), a new method for generating word embeddings that uses low-rank, weighted factorization of the arbitrary transformations of the word-context cooccurrence matrix via stochastic gradient descent, respecting the full set of *principled reconstruction error* properties (Section 3.2). LexVec closes the performance gap in word similarity tasks between counting and predictive word embedding models, creating both a state-of-the-art static word embedding model and a more transparent alternative to Skip-gram due to its explicit nature (Section 4.1.2).

(Q3) How can this approach be scaled to web-sized corpora?

We propose an external memory variant of LexVec (SALLE; IDIART; VILLAVICENCIO, 2016) that is based on MapReduce operations that is highly scalable (Section 3.5). Results show its equivalence to the in-memory variant (Section 4.2). We showcase its scalability by training it on the larger Common Crawl corpus (Section 4.4.2).

1.1.2 Positional contexts

The use of positional information is a recurring theme in NLP models, present in foundational works through to modern Transformer models. For example, if position (or ordering) is ignored and a bag-of-words approach is used, without additional context it becomes impossible to determine the subject in sentences such as "John read Harry's PhD thesis". Did John or Harry do the reading? Additional evidence of the importance of positional contexts both strengthens this theme, and paves the way for improved model performance.

(Q4) In counting cooccurrence, can we use the relative position of words and contexts and does this improve word representations?

We augment the cooccurrence matrix with positional information and use this information to learn position-dependent context vectors (Section 3.3) (SALLE; IDIART; VILLAVICENCIO, 2016). We compare existing and novel ways of merging word and context vectors to obtain single word vector representations (Section 3.3.1).

1.1.3 Subwords

As with positional contexts, breaking words into subwords permeates all areas of NLP, from the study of morphology to optimization of softmax layers in large language models.

(Q5) Given that the cooccurrence matrix is word-to-word, how can we learn vectors for subwords?

We represent word vectors by the sum of their subword vectors and successfully incorporate this into the LexVec model (Section 3.6).

(Q6) Are linguistically aware subwords better than naive character subwords?

We perform novel evaluation of the robustness of naive character n-grams versus morphemes, with results showing that despite their simplicity, n-grams are an adequate alternative (Section 4.3.2).

1.1.4 Association measures

PMI is yet another staple of NLP with applications that extend beyond word embedding. Though theoretically sound, alternatives are required to deal with pairs of events that have 0 probability.

(Q7) How can we address computationally PMI going to $-\infty$ *without discarding the spectrum of negative values?*

Since it is unclear what is lost by collapsing the spectrum of negative PMI (nPMI) values to 0 when using PPMI, we look at alternative ways of dealing with nPMI by proposing two new PMI variants (CPMI and NNEGPMI) which partially preserve the negative spectrum (Section 3.4.1).

1.1.5 Downstream Performance

Pretrained and widely-distributed word embedding models sparked a revolution in NLP. Despite the recent popularity of Transformer models, pretrained static word embeddings remain the state-of-the-art in lexical semantic tasks and remain competitive in certain tasks that make use of sentence embeddings (LENCI et al., 2021; YANG; ZHU; CHEN, 2019; WANG et al., 2020a).

(Q8) When trained on similarly-sized corpora, how does LexVec compare to other popular word embeddings in downstream tasks?

We train LexVec models on a 63B token dump of Common Crawl. LexVec's average downstream performance is (1) superior to a GloVe model trained on 840B tokens (13x more data) in all evaluations (2) superior to Skip-gram's trained on 100B tokens (1.3x more data) in all evaluations excepting textual similarity (3) superior to the state-of-the-art fastText subword model trained on 630B tokens (10x more data) in syntactic, natural language inference, and facet ranking tasks.

(Q9) On what tasks are static word embeddings competitive with recent Transformer models?

In lexical semantic tasks, results show that static word embeddings remain the dominant approach (Section 4.5.1). As universal sentence encoders (where rather than being fine-tuned on a specific task, Transformer outputs are used as sentence embeddings for arbitrary tasks), sentence embedding methods using as input static word embeddings (including LexVec) are competitive in textual similarity tasks and are only marginally worst in mean text classification accuracy (Section 4.5.2).

1.1.6 An Application in Information Retrieval

Conversational search is a rapidly growing field in IR and NLP. Human evaluation of conversational search systems is expensive, and prohibitive as the number of systems to be tested grows.

(Q10) Can word embeddings be used as a building block in such systems?

We propose and evaluate a user simulator for conversational search that enables scalable evaluation of conversational search systems. One of the components of this simulator is a facet ranker: given the current dialogue, rank what are the most likely intents (or facets) of the user. Using as a starting point an existing dialogue dataset, we use heuristics to automate the construction of a facet ranking dataset to enable the evaluation of different facet rankers (Section 2.4.2.5). Results show that static word embeddings are an effective approach for facet ranking, competitive with Transformer models when used for sentence embedding, with LexVec models outperforming other static word embeddings (Section 4.4.8).

1.1.7 Information contained positive/negative PMI

The commonly used PPMI measure discards negative information by collapsing it to 0, raising the question of exactly what information is being discarded. From a practical point of view, understanding what can be learned from different spectrums of PMI can further our understanding of word embedding models and assist in their design. On the theoretical side, this same understanding helps better characterize distributional semantics.

(Q11) Is it possible to isolate the effects of positive or negative PMI when learning dense word vectors?

We show that thanks to LexVec's explicit access to the PMI matrix being factorized, SGD steps can be skipped for pairs containing positive or negative target values, achieving the desired isolation and resulting in models that use only positive or negative information (Section 5.1). We also observe that the negative sampling procedure used in both LexVec and Skip-gram samples primarily negative PMI pairs, allowing the effects of negative information to be amplified by increasing the number of negative samples used (Section 6.1).

(Q12) Does positive/negative information encode semantic and/or syntactic properties of words? What type of linguistic information is predominantly represented by which part of the PMI spectrum?

Word and sentence-level evaluations show that only accounting for positive PMI (pPMI) in the factorization strongly captures both semantics and syntax, whereas using only nPMI captures little of semantics but a surprising amount of syntactic information (Section 5.1.1). This deepens our understanding of distributional semantics and of computational linguistics by extending the distributional hypothesis to "a word is not only characterized by the company that it keeps, *but also by the company it rejects*". We hypothesize that grammar is systematically generating negative cooccurrence (equivalently evidence *not seen*, negative evidence, or negative information), and from this negative evidence our word embeddings capture significant syntactical information.

1.1.8 Vector space geometry

Vector normalization is known to improve word analogy performance. It would be interesting to know under what conditions this geometry-altering heuristic becomes unnecessary. In our work, we focus on how negative information impacts vector space geometry.

(Q13) How is the geometry of the word vector space affected by negative information?

We perform an in-depth investigation of the effect that increasing the relative importance of nPMI compared to pPMI has on the geometry of the vector space and its representation of frequent and rare words. We find that increasing the relative importance of negative information strengthens *geometric rank invariant properties* – vector norms and direction – of word vectors and improves the representation of rare words (Section 6.3). Experiments reveal similar results for Skip-gram, GloVe, and SVD models, showing that the important role played by negative information in LexVec transfers well to these other models (Section 6.4).

Additionally, our analysis reveals that when word analogies are evaluated correctly (without the use of heuristics), performance improves as more negative information is used, suggesting that these geometric properties are connected to more strongly capturing the linear vector offsets used in answering analogies (Section 6.3).

1.2 Relevant work during PhD

1.2.1 Publications

As of this publication, the following works have been cited 141 times:

- SALLE, A.; VILLAVICENCIO, A.; IDIART, M. Matrix factorization using window sampling and negative sampling for improved word representations. In: **Pro-**

ceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Berlin, Germany: Association for Computational Linguistics, 2016. p.419–424. Available from Internet: <https://www.aclweb.org/anthology/P16-2068>. Quali: A1. *Section 3.2.*

- SALLE, A.; IDIART, M.; VILLAVICENCIO, A. Enhancing the LexVec distributed word representation model using positional contexts and external memory. **CoRR**, abs/1606.01283, 2016. Available from Internet: <https://arxiv.org/abs/1606.01283>. *Sections 3.3 and 3.5.*
- SALLE, A.; VILLAVICENCIO, A. Incorporating subword information into matrix factorization word embeddings. In: **Proceedings of the Second Workshop on Subword/Character LEvel Models**. New Orleans: Association for Computational Linguistics, 2018. p. 66–71. Available from Internet: <https://www.aclweb.org/anthology/W18-1209>. *Section 3.6.*
- SALLE, A.; VILLAVICENCIO, A. Restricted recurrent neural tensor networks: Exploiting word frequency and compositionality. In: **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**. Melbourne, Australia: Association for Computational Linguistics, 2018. p. 8–13. Available from Internet: <https://www.aclweb.org/anthology/P18-2002>. Quali: A1.
- SALLE, A.; VILLAVICENCIO, A. Why so down? The role of negative (and positive)pointwise mutual information in distributional semantics. **CoRR**, abs/1908.06941, 2019. Available from Internet: <http://arxiv.org/abs/1908.06941>. *Chapter 5.*
- SALLE, A.; VILLAVICENCIO, A. Understanding the Effects of Negative (and Positive) Pointwise Mutual Information on Word Vectors. **Under review**, 2021. *Chapters 5 and 6.*
- SALLE, A.; MALMASI, S.; ROKHLENKO, O.; AGICHTEIN, E.; Studying the effectiveness of conversational search refinement through user simulation. **43rd European Conference On Information Retrieval**. Lucca, Italy: Advances in Information Retrieval, Springer International Publishing, 2021. p. 587–602. Available from Internet: <https://link.springer.com/chapter/10.1007%2F978-3-030-72113-8_39>. Quali: A2. *Sections 2.4.2.5 and 4.4.8.*
- SALLE, A.; MALMASI, S.; ROKHLENKO, O.; AGICHTEIN, E.; Studying the effectiveness of conversational search refinement and clarification through user sim-

ulation. **Under review**, 2021. *Chapters 5 and 6.*

1.2.2 Software and Models

- LexVec: <https://github.com/alexandres/lexvec>
- terashuf: <https://github.com/alexandres/terashuf>
- Models (English):
    - LexVec-63B (Mean). Vectors: <https://www.dropbox.com/s/ztbj824y779w385/lexvec.commoncrawl.300d.vectors.gz?dl=1>.
    - LexVec-63B (Concat). Vectors: <https://www.dropbox.com/s/a26u7j1i9459b2w/lexvec.commoncrawl.1500d.vectors.gz?dl=1>.
    - SubLexVec-63B (Mean). Vectors: <https://www.dropbox.com/s/vwywt301aw0ebxw/lexvec.commoncrawl.ngramsubwords.300d.vectors.gz?dl=1>.
    - SubLexVec-63B (Concat). Vectors: <https://www.dropbox.com/s/kkkbc695j7dfs7y/lexvec.commoncrawl.ngramsubwords.1500d.vectors.gz?dl=1>.
    - SubLexVec-Crawl. Vectors: <https://www.dropbox.com/s/mrxn933chn5u37z/lexvec.commoncrawl.ngramsubwords.300d.W.pos.vectors.gz?dl=1>. Binary model: <https://www.dropbox.com/s/buix0deqlks4312/lexvec.commoncrawl.ngramsubwords.300d.W.pos.bin.gz?dl=1>.
    - LexVec Common Crawl. Word Vectors: <https://www.dropbox.com/s/flh1fjynqvdsj4p/lexvec.commoncrawl.300d.W.pos.vectors.gz?dl=1>. Merged Vectors: <https://www.dropbox.com/s/zkiajh6fj0hm0m7/lexvec.commoncrawl.300d.W%2BC.pos.vectors.gz?dl=1>.
    - LexVec Wikipedia + NewsCrawl. Word Vectors: <https://www.dropbox.com/s/kguufyc2xcdi8yk/lexvec.enwiki%2Bnewscrawl.300d.W.pos.vectors.gz?dl=1>. Merged Vectors: <https://www.dropbox.com/s/u320t9bw6tzlwma/lexvec.enwiki%2Bnewscrawl.300d.W%2BC.pos.vectors.gz?dl=1>.

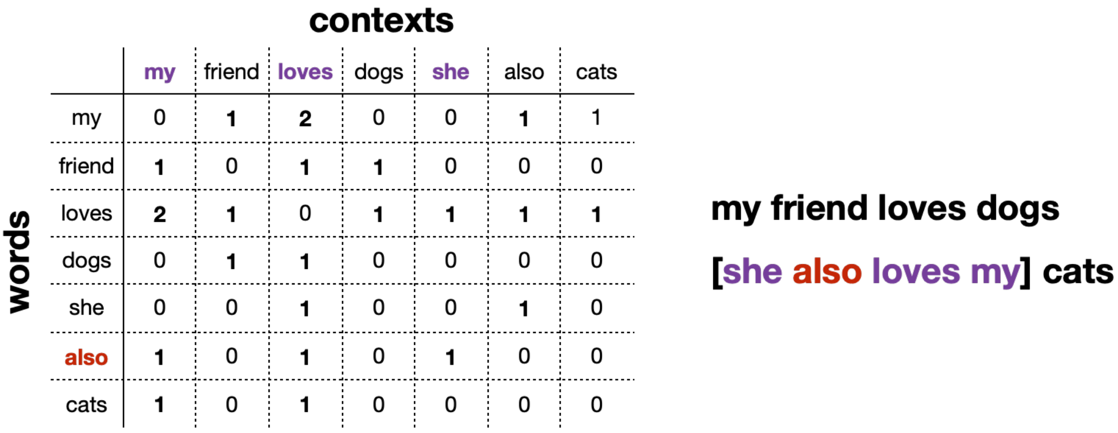
2 BACKGROUND

In this chapter, we present an overview of word embedding models, starting from their building blocks – the cooccurrence matrix (Section 2.1) and its transformations (Section 2.2) – through to related work in matrix transformations (Section 2.3.1), factorization (Section 2.3.2), subword information (Section 2.3.3), analysis (Section 2.3.4) and contextualized word embeddings (Section 2.3.5). We then describe the set of evaluations used in testing their performance, both intrinsic (Section 2.4.1) and downstream (Section 2.4.2).

2.1 Cooccurrence Matrix and Positional Contexts

The building block for *counting* word embedding methods is the cooccurrence matrix. This matrix M is constructed by initializing it to 0 and sliding a symmetric window over the training corpus and for each center word $w \in V$ (V is the word vocabulary) and context word $c \in V$ within the window, incrementing $M_{w,c}$. A row M_w in this matrix represents the raw count *context distribution* of the word w. Figure 2.1 gives an example of a cooccurrence matrix constructed from two sentences.

Figure 2.1 – A cooccurrence matrix constructed from two sentences. When the window is centered on “also”, the corresponding cells for “she”, “loves”, and “my” are incremented.



contexts
words

	my	friend	loves	dogs	she	also	cats
my	0	1	2	0	0	1	1
friend	1	0	1	1	0	0	0
loves	2	1	0	1	1	1	1
dogs	0	1	1	0	0	0	0
she	0	0	1	0	0	1	0
also	1	0	1	0	1	0	0
cats	1	0	1	0	0	0	0

my friend loves dogs

[she also loves my] cats

As observed by Ling et al. (2015b) and more recently Salle, Idiart and Villavicencio (2016), Vaswani et al. (2017), Mikolov et al. (2018), Devlin et al. (2019), using *positional information* (SCHÜTZE, 1993) consistently improves word representations: the *positional contexts* for the target word “dogs” in “My friend loves *dogs* and cats” for a symmetric window of size $l = 2$ (2 words to each side of the target word) are $\{friend_{-2}$, $loves_{-1}$, and_1, $cats_2\}$. This leads to a context vocabulary $V_c = \bigcup_{w \in V_w} \{w_{-l}, \ldots, w_{-1},$

$w_1, \ldots, w_l\}$ of size $|V_w| \times 2 \times l$. After sliding the symmetric window over the entire corpus, the value $M_{dogs,cats_2}$, for example, is the number of times "cats" appeared two words to the right of "dogs".

For concreteness throughout this thesis, we refer to the cooccurrence matrix M_{wiki} constructed from a lowercased, alphanumerical 2015 English Wikipedia dump with $3.8e9$ tokens, discarding tokens with frequency < 100, window size $l = 2$, *positional contexts*, for a word vocabulary V_w of $303,517$ words and context vocabulary of V_c of $1,214,068$ words.

Additionally, using this same configuration but *without positional contexts*, we follow Mikolov et al. (2013) in using a symmetric window of size 5 drawn from $U(1,5)$ for each target word. We refer to this matrix as M_{wiki5}.

In constructing both M_{wiki} and M_{wiki5}, we use the additional heuristic of token *subsampling* (MIKOLOV et al., 2013) the training corpus: tokens for word w are randomly discarded with probability $p_w = \max(0, 1 - \sqrt{t/f_w})$, where t is the subsampling threshold (we follow Mikolov et al. (2013) and set $t = 10^{-5}$ throughout this paper) and f_w is the unigram frequency (tokens of w divided by total number of tokens in training corpus). For Skip-gram and LexVec, which perform factorization by sampling word-context pairs from the training corpus, subsampling accelerates training significantly. Mikolov et al. (2013) also observe empirically that it improves the representation of uncommon words.

2.2 Pointwise Mutual Information (PMI)

Rather than using the raw frequency count matrix, it is common to reweight it using PMI (FANO, 1961; CHURCH; HANKS, 1990). Table 2.1 illustrates the difference between cooccurrence and PMI values by showing 10 context words with the highest cooccurrences/PMI with a set of sample words. These context words correspond to peaks in the context distributions of these sample words. We can interpret these peaks as the distributional features that *should* most strongly represent the semantics of the target word. However, even with subsampling, raw cooccurrences lead to frequent features that have little semantic information, such as "the", "at", "of". In contrast, PMI downweights these features and emphasizes features that have clear semantic relationship to the target words.

The PMI transformation (association measure) of the cooccurrence matrix is

Table 2.1 – Sample values from M_{wiki5} and corresponding PMI values.

Word	Coocs	PMI
time	the, at, of, to, a, for, in, and, same, that	spend, spent, waste, spending, heavyweight, periods, trial, nepal, same, full
system	system, operating, the, solar, of, a, pass, and, to, in	operating, solar, nintendo, transit, classification, system, pass, navigation, linux, integrated
early	life, in, the, century, late, s, born, th, career, nine	childhood, life, stages, late, morning, centuries, career, century, ages, medieval
member	of, parliament, a, the, board, committee, founding, member, council, elected	founding, parliament, legislative, honorary, board, elected, committee, assembly, representatives, mp
original	research, the, recording, holder, of, original, affect, resolution, to, copyright	holder, affect, poster, recording, soundtrack, research, limit, resolution, sample, copies
works	works, published, of, the, and, in, his, art, novels, by	works, literary, influenced, novels, collections, collected, shakespeare, published, translated, fiction
strong	keep, a, zero, utc, support, two, the, oppose, and, delete	ties, oppose, keep, weak, strong, opinions, support, opposition, influence, enough
hand	hand, other, right, the, on, left, a, to, and, of	hand, luke, cool, combat, right, cards, holding, picked, palm, arm
continued	until, to, the, and, he, his, in, grow, of, as	grow, operate, until, expand, decline, perform, serve, struggle, throughout, retirement
problem	the, problem, is, a, that, solution, i, with, this, to	solution, fix, problem, serious, solutions, biggest, fixed, addressed, finding, isn
features	the, and, features, of, a, which, album, video, include, also	interviews, bonus, features, architectural, soundtrack, unique, guest, dvd, interface, disc
outside	the, of, inside, and, topic, to, a, just, in, walls	inside, walls, topic, limits, dispute, scope, views, jurisdiction, outside, residents

equal to:

$$PMI_{w,c} = \log \frac{P(w,c)}{P(w)P(c)} = \log \frac{\frac{M_{w,c}}{M_{*,*}}}{\frac{M_{w,*}}{M_{*,*}} \frac{M_{*,c}}{M_{*,*}}} = \log \frac{M_{w,c}M_{*,*}}{M_{w,*}M_{*,c}} \tag{2.1}$$

where $*$ denotes summation over the corresponding index. We refer to the set $\{(w,c) \mid PMI_{w,c} \leq 0\}$ as negative pointwise mutual information (nPMI), the set $\{(w,c) \mid PMI_{w,c} > 0\}$ as positive pointwise mutual information (pPMI), and the set $\{(w,c) \mid M_{w,c} = 0\}$ as maximally negative pointwise mutual information (mnPMI). Table 2.2 gives the sizes of these sets for PMI_{wiki}, the transformation of M_{wiki}.

Table 2.2 – Set sizes computed over 10^5 cells sampled randomly from PMI_{wiki}, such that nPMI+pPMI sum to 100%.

Name	Description	Set	Size (%)
nPMI	Negative information	$\{(w,c) \mid PMI_{w,c} \leq 0\}$	99.75
pPMI	Positive information	$\{(w,c) \mid PMI_{w,c} > 0\}$	0.25
mnPMI	Maximally-negative information	$\{(w,c) \mid M_{w,c} = 0\}$	99.71
nPMI\mnPMI	Collapsed negative information under PPMI	$\{(w,c) \mid PMI_{w,c} \leq 0 \wedge M_{w,c} > 0\}$	0.04

An issue with PMI is that rare words/contexts can lead to very high PMI, since these terms lead to small denominators in Equation (2.1). A workaround proposed in Levy, Goldberg and Dagan (2015) is to smooth the context distribution $P(c)$ via an exponential term α (this reassigns probability mass from frequent events to rare events, while still preserving the ordering of probabilities, i.e., if $P(c_1) > P(c_2)$, after smoothing this still holds), giving the context distribution smoothed (CDS) version of PMI:

$$PMI^{\alpha_{cds}}_{w,c} = \log \frac{P(w,c)}{P(w)P^{\alpha_{cds}}(c)} \tag{2.2}$$

$$= \log \frac{\frac{M_{w,c}}{M_{*,*}}}{\frac{M_{w,*}}{M_{*,*}} \frac{M^{\alpha_{cds}}_{*,c}}{\sum_{c'} M^{\alpha_{cds}}_{c',c}}} \tag{2.3}$$

$$= \log \frac{M_{w,c}}{M_{w,*} \frac{M^{\alpha_{cds}}_{*,c}}{\sum_{c'} M^{\alpha_{cds}}_{*,c'}}} \tag{2.4}$$

Even with smoothing, there is still an issue with unobserved word-context pair cooccurrences, as PMI goes to negative infinity. This problem is aggravated by unreliable statistics from finite corpora which lead to numerous such pairs. A common workaround is to clip *all* negative PMI values at 0, including those for cooccurring pairs; this is known as Positive PMI (PPMI, not to be confused with the pPMI set defined above):

$$PPMI_{w,c}^{\alpha_{cds}} = \max(PMI_{w,c}^{\alpha_{cds}}, 0) \tag{2.5}$$

This indiscriminate clipping of negative values motivated this thesis, making us question what information it discards. Or in other words, what can we learn from negative information?

2.3 Related Work

2.3.1 Cooccurrence Matrix Transformations

There is a long history of studying transformations (also known as association measures) of cooccurrence matrices in general, not only of word-context pairs; see Schütze (1993), Manning, Manning and Schütze (1999), Jurafsky (2000) for an overview and Curran and Moens (2002) for comparison of different transformations. One widely adopted measure is PMI, and in fact, Bullinaria and Levy (2007) show that word vectors derived from PPMI matrices perform better than alternative transformations for word-context cooccurrence. Moreover, Levy and Goldberg (2014) show theoretically that the popular Skip-gram model (MIKOLOV et al., 2013) performs implicit factorization of shifted PMI. Another PMI variant is normalized PMI, which Bouma (2009) proposed for dealing with negative infinity ($-\infty$), for collocation extraction.

2.3.2 Matrix Factorization

Low-rank word vectors obtained through the factorization of the PMI matrix are advantageous computationally and arguably lead to better generalization than directly using rows from PMI as word vectors.

Although the SVD provably (ECKART; YOUNG, 1936) provides factorizations with the lowest possible squared loss $L_2(w, c) = \frac{1}{2}\lambda_{SVD}(w, c)(W_w C_c^\top - f(M)_{w,c})^2$ – where $\lambda_{SVD}(w, c) = 1$ and $f(\cdot)$ is an arbitrary transformation – Salle, Villavicencio and Idiart (2016) show that, in word embedding where f is some variant of PMI, uniform weights λ significantly reduce the quality of the word vectors.

GloVe (PENNINGTON; SOCHER; MANNING, 2014) does perform weighting but ignores values in the mnPMI set (where $M_{w,c} = 0$):

$$L_{GloVe}(w,c) = \frac{1}{2}\lambda_{GloVe}(w,c)(W_w C_c^\top - \log M_{w,c} + b_w + \tilde{b}_c)^2 \tag{2.6}$$

$$\lambda_{GloVe}(w,c) = \min(M_{w,c}^{\alpha}/x_{max}^{\alpha}, 1) \tag{2.7}$$

where α and x_{max} are constants, and b and $\tilde{b}$ are bias terms.

Swivel (SHAZEER et al., 2016) adjusts the GloVe objective to account for values in mnPMI:

$$L_{Swivel}(w,c) = \begin{cases} \frac{1}{2}\lambda_{Swivel}(w,c)(W_w C_c^\top - PMI(w,c))^2 & \text{if } (w,c) \notin mnPMI \\ \frac{1}{2}\log(1 + \exp(W_w C_c^\top - PMI^*(w,c))) & \text{otherwise} \end{cases} \tag{2.8}$$

$$\lambda_{Swivel}(w,c) = M_{w,c}^{1/2} \tag{2.9}$$

where PMI^* is a Laplace smoothed PMI matrix (replacing zero coocurrences with ones). Note that although accounting for values in mnPMI, Swivel does so by indiscriminately assigning uniform weight to all negative cooccurrences, similar to the SVD.[1]

The Skip-gram (MIKOLOV et al., 2013) model performs *implicit* matrix factorization by predicting contexts given target words. For each (w, c) pair observed in sliding a window over the training corpus, k negative samples are drawn from the unigram context distribution and the following objective function is *maximized*:

$$J_{sg}(w,c) = \log\sigma(W_w C_c^\top) + \sum_{i=1}^{k} \mathbf{E}_{c_i \sim P_n(c)} \log\sigma(-W_w C_{c_i}^\top) \tag{2.10}$$

where $\sigma(\cdot)$ is the logistic function. Under the assumption that the embedding dimension is high enough such that $J(\cdot)$ terms can be maximized independently for different word-context pairs, Levy and Goldberg (2014) show that the matrix $WC^T = PMI - \log k$ maximizes J. In other words, Skip-gram implicitly factorizes a shifted PMI matrix.

---

[1] Swivel is not used in our evaluations because of its $O(|V|^2)$ computational complexity from calculating loss terms for *every cell* in the matrix being factorized, thus requiring a large distributed computing environment to be feasible: in our initial experiments $|V|^2 = 9.2e10$, and $|V_w||V_c| = 3.7e11$ when using positional contexts. In the downstream experiments in Section 4.4, this problem is aggravated as $|V|^2 = 4e12$.

2.3.3 Subword information

Word embeddings that leverage subword information were first introduced by Schütze (1993) which represented a word as the sum of four-gram vectors obtained running an SVD of a four-gram to four-gram cooccurrence matrix. Our model differs by learning the subword vectors and resulting representation jointly as weighted factorization of a word-context cooccurrence matrix is performed.

There are many models that use character-level subword information to form word representations (LING et al., 2015a; CAO; REI, 2016; KIM et al., 2016; WIETING et al., 2016; VERWIMP et al., 2017), as well as fastText (the model on which we base our work). Closely related are models that use morphological segmentation in learning word representations (LUONG; SOCHER; MANNING, 2013; BOTHA; BLUNSOM, 2014; QIU et al., 2014; MITCHELL; STEEDMAN, 2015; COTTERELL; SCHÜTZE, 2015; BHATIA; GUTHRIE; EISENSTEIN, 2016). Our model also uses n-grams and morphological segmentation, but it performs explicit matrix factorization to learn subword and word representations, unlike these related models which mostly use neural networks.

Finally, Cotterell, Schütze and Eisner (2016) and Vúlic et al. (2017) *retrofit* morphological information onto *pre-trained* models. These differ from our work in that we incorporate morphological information at training time.

2.3.4 Analysis

The continued relevance of classic word embeddings has led to a number of recent papers that aim to understand their properties. These include research on why word analogies ("*a* is to *b* as *c* is to *?*") hold (HASHIMOTO; ALVAREZ-MELIS; JAAKKOLA, 2016; ETHAYARAJH; DUVENAUD; HIRST, 2019; ALLEN; HOSPEDALES, 2019), or conversely why they do not hold (SCHLUTER, 2018; LINZEN, 2016; ROGERS; DROZD; LI, 2017); on their geometry (MIMNO; THOMPSON, 2017); and on possible biases they incorporate (BOLUKBASI et al., 2016; NISSIM; NOORD; GOOT, 2019; GONEN; GOLDBERG, 2019). This paper follows this line of research into understanding the workings of these models.

Mimno and Thompson (2017) analyze the geometry of the Skip-gram model, concluding that it is "strange": word vectors occupy a narrow cone in space diametrically opposed to context vectors. Our analysis of model geometry is directly inspired by their work, but rather than analyzing the implicit PMI factorization of Skip-gram, we look at explicit factorizations, under which the "strange" geometry of Skip-gram can be explained by looking at the underlying PMI counts.

Miaschi and Dell'Orletta (2020) use a set of sentence representation probing tasks to investigate the linguistic knowledge encoded in sentence embeddings obtained from BERT and word2vec, observing comparable results for both models. Different from our work, their focus is on evaluating sentence representations, and they do not isolate the effects of negative information.

Schluter (2018) observed that in analogies of the form "*a* is to *b* as *c* is to ?" if *a,b,c* are not excluded from the set of possible answers, performance of the Skip-gram and GloVe models plummets. Linzen (2016), Rogers, Drozd and Li (2017) made similar observations that these models do not quite seem to capture the geometry necessary to correctly answer analogies. However, none of these works *simultaneously* relate (a) increasing negative information with (b) word vector geometry for different frequencies and (c) analogy performance. Although Ethayarajh, Duvenaud and Hirst (2019), Allen and Hospedales (2019) theoretically show that these linear analogies should hold in PMI factorizations, there is no investigation into how the geometry of different factorizations affects results.

2.3.5 Contextualized Word Embeddings and Language Models

The most significant performance jump in NLP tasks since word2vec has been due to *deep* contextualized word embeddings (PETERS et al., 2018) – where rather than having a single vector for a word as described in the factorization above, models use neural networks to compute vectors for words *in context*. For instance, the computed vector for the word "bank" in "the river bank was teeming with frogs" is different from the one in "I went to the bank to make a deposit".

More recently, gains in performance have also been obtained due to language model pre-training and fine-tuning (HOWARD; RUDER, 2018; RADFORD et al., 2019; DEVLIN et al., 2019) – where a *deep* neural network language model is pre-trained on very large corpora and the token-prediction output layer is replaced with an output layer fine-tuned for the target task.

Task leaderboards such as SuperGLUE[2] are dominated by the latter, specifically

---

[2] https://super.gluebenchmark.com/leaderboard

language models based on the Transformer (VASWANI et al., 2017) architecture.

Interestingly, there are ways in which our thesis work on static word embeddings complements that in contextualized word embeddings and language models: for example, (LIU; MCCARTHY; KORHONEN, 2020) show how static word embeddings can improve *contextual* word representations in lexical semantic tasks that are context-dependent, and (HOOVER et al., 2021) use our result on the connection between negative PMI and syntactic information to justify preserving the negative spectrum of PMI measures extracted from Transformer models.

2.4 Evaluation

If an evaluation serves only to evaluate the information contained in a vector space model, it is said to be *intrinsic*. On the other hand, if the evaluation measures the utility in some downstream task, it is said to be *extrinsic or downstream*. Clearly there is room for debate as to which category an evaluation belongs to: for example, Part-of-Speech (POS) tagging both serves to evaluate whether POS information is captured by a vector space model, but it is also useful for downstream tasks such as separating object (Noun) from attributes (Adjectives and adverbs) in an eCommerce search query. Nevertheless, we follow the convention in the literature and leave discussions on taxonomy to future debate.

Evaluations can be further categorized into whether they evaluate individual word representations, or phrase/sentence representations where word vectors must be composed to form a single vector representation. Commonly used intrinsic *word-level* evaluations include word similarity, word analogy, and fMRI. For sentence-level evaluation, we look at text classification, POS tagging, and sentence-level similarity, natural language inference, and various probing tasks from the SentEval (CONNEAU; KIELA, 2018) suite.

2.4.1 Intrinsic Evaluation

2.4.1.1 Word similarity

In the word similarity task, given a pair of words, human judges are tasked with assigning a high score if the words are similar, and a low score if they are dissimilar, creating a word similarity dataset. Table 2.3 gives examples of pairs and scores from the datasets used in this paper.

Word vectors are evaluated using such word similarity datasets by comparing the agreement – using Spearman rank correlation – between its human assigned score and *some function* of the two word vectors. We use the cosine function as the measure of word similarity between word vectors u and v in vector space:

$$Cos(u, v) = \frac{u \cdot v}{|u||v|} \tag{2.11}$$

This discards vector norms, and considers only the angle between vectors.

Table 2.3 – Word similarity examples.

dataset	pairs	word 1	word 2	judgement
MEN	3000	black	pigeon	13.00
		bloom	daisy	41.00
		fabric	wool	43.00
		held	theatre	9.00
		lamb	sheep	40.00
MTurk	287	coin	awards	2.17
		gambling	money	3.75
		navy	withdrawn	1.57
		pacific	ocean	4.27
		reichstag	germany	2.29
RW	2034	correspondence	write	7.83
		engineering	design	8.33
		excitation	exciting	8.20
		leadership	high	4.33
		resigning	office	5.43
SimLex	999	bed	hospital	0.92
		bread	flour	3.33
		butter	potato	1.22
		choose	elect	7.62
		diet	apple	1.18
WS-R	252	computer	laboratory	6.78
		cup	food	5.00
		population	development	3.75
		practice	institution	3.19
		soap	opera	7.94
WS-S	203	journey	voyage	9.29
		music	project	3.63
		peace	insurance	2.94
		stock	egg	1.81
		vodka	gin	8.46

2.4.1.2 Word analogy

Analogies of the form “*a* is to *b* as *c* is *d*” are evaluated by finding the word d^* such that:

$$d^* = argmax_{w \in V_w} Cos(W_w, W_c + W_b - W_a) \tag{2.12}$$

$$\tag{2.13}$$

If $d^* = d$, the analogy is said to hold in the vector space. Schluter (2018) points out two flaws in the way this evaluation is conducted in works such as Mikolov et al. (2013), Pennington, Socher and Manning (2014):

(1) Normalization (Norm): The vector space is *distorted* by normalizing all word vectors to unit length before the term $W_c + W_b - W_a$ is calculated.

(2) Premise exclusion (Prem): The set $\{w \in V_w\}$ in Equation (2.12) is replaced by $\{w \in V_w \backslash \{a, b, c\}\}$ – the analogy’s premises are excluded from the set of candidate answers. In practice this improves performance because it is often the case that $W_b - W_a \approx 0$ and so $d* = c$ if c is not excluded from the candidates (LINZEN, 2016).

Although these heuristics work well in practice, significantly improving accuracy, they mask whether the linear relationship $W_a - W_b \approx W_c - W_d$ is really present in the vector space. This is particularly problematic as it may lead to wrong conclusions. For instance, using the popular GoogleNews word2vec vectors, the answer to the analogy “man is doctor as woman is to ?” was “nurse” (BOLUKBASI et al., 2016) if both Norm and Prem are performed, when in fact, if analogies are evaluated correctly, the actual answer is “doctor” as well (NISSIM; NOORD; GOOT, 2019).

In this work, we perform incorrect evaluation where both heuristics are used (e.g., GSem, GSyn) partially correct evaluation where one heuristic is excluded (e.g., GSem-Norm, GSem-Prem), and correct evaluation where both heuristics are excluded (e.g., GSem-Norm-Prem). Some example analogies from each dataset are shown in Table 2.4.

2.4.1.3 fMRI

Though there is no hard requirement that techniques in the field of AI mimic human biology, it is of theoretical interest to know when such connections do exist. To this end, Mitchell et al. (2008) evaluated how closely semantic representations from NLP

Table 2.4 – Word analogy samples, having the form “*a* is to *b* as *c* is *d*”.

dataset	pairs	a	b	c	d
GSem	8869	Dhaka	Bangladesh	Madrid	Spain
		Fresno	California	Plano	Texas
		Helsinki	Finland	London	England
		Moscow	Russia	Tirana	Albania
		Nigeria	naira	Argentina	peso
		dad	mom	he	she
GSyn	10675	convenient	inconvenient	consistent	inconsistent
		deep	deeper	smart	smarter
		dollar	dollars	woman	women
		reasonable	unreasonable	possible	impossible
		swimming	swam	paying	paid
		typical	typically	rare	rarely
MSR	8000	bright	brighter	strong	stronger
		closest	closer	earliest	earlier
		feel	feels	prove	proves
		followed	follow	needed	need
		night’s	night	party’s	party
		schools	school	artists	artist

predicted cortical activations in the human brain. Nine human subjects were primed with target words and fMRI images taken of brain activation. The task is then, for each subject, given two target words and the corresponding fMRI images, use a regressor that takes as input a semantic representation of the target words and predicts voxel activations, then match each predicted image to the correct labeled image. The regressor is trained on all but the two target words, this repeated for every two-word combination from the dataset of 60 words, also known as “leave-two-out” cross-validation. Matching is done using cosine distance between the images represented as vectors. Accuracy is given by the fraction of correct pair matchings over all two-word combinations. A random regressor has expected accuracy of .5. Mitchell et al. (2008) showed that using human labeled semantic features as a semantic representation performs significantly better than a random baseline. Subsequent work (BRANCO et al., 2020) showed that the same holds true for representations derived from semantic networks and distributional semantic models. We use the evaluation setup as Abnar et al. (2018) and the corresponding software package[3].

---

[3] https://github.com/samiraabnar/NeuroSemantics

2.4.2 Downstream Evaluation

Although individual word representations are of theoretical interest, natural language comes in sequences of words. Transforming a sequence of word vectors into a fixed length vector representation is known as sentence embedding. Given a sequence $w_1, w_2, ...w_n$, the simplest of these models is averaging word vectors, known as a mean bag-of-vectors:

$$BoV(w_1, ..., w_N) = \frac{1}{N}\sum_{i=1}^{N} W_{w_i} \tag{2.14}$$

where W_{w_i} is the word vector for w_i. For most of our experiments, we use this simple sentence embedding model since it is computationally efficient and remains a strong baseline for unsupervised sentence representation.

2.4.2.1 Text Similarity

A desideratum of sentence representations is that two sentences with the same meaning but differing surface form share a similar semantic representation. Agreement between human judgements and computed scores (either cosine between sentence embeddings or a trained regressor) is measured using Pearson's correlation coefficient. To this end, we use the Semantic Textual Similarity (STS) datasets from years 2012-2017 (AGIRRE et al., 2012; AGIRRE et al., 2013; AGIRRE et al., 2014; AGIRRE et al., 2015; AGIRRE et al., 2016; CER et al., 2017) referred to as STS12-16 (2012-2016) and STS-B (STS-Benchmark, 2017), and the SICK relatedness (SICK-R, Marelli et al. (2014)) dataset. Table 2.5 gives some examples of sentence pairs and the associated human similarity score.

2.4.2.2 Text Classification

A common application in NLP is text classification: for example, identify if a movie review is positive or negative, or classify an email as spam or not. There can be more than two classes, such as classifying a sentence's sentiment as positive, negative, neutral, etc. In text classification, the sentence embedding is used as input to a classification network (which can be a deep neural network or a simple logistic/softmax layer) trained on labeled data. We use the SentEval suite for sentence classification evalua-

Table 2.5 – Samples from textual similarity datasets used in our evaluations.

Name	N	Text 1	Text 2	Judgement
STS12	3.1k	ryan harvey , an outfielder from dunedin high school in florida , was selected with the sixth pick by the chicago cubs .	ryan harvey , a high school outfielder from florida , was chosen sixth by the cubs .	4.2
STS13	1.5k	germany dodges recession with growth in first quarter	eurozone avoids recession in q1 , thanks to germany	3.2
STS14	3.7k	zimmerman replied , " what are you doing here ? "	martin said , " why are you following me ? "	0.9
STS15	8.5k	the woman holding a black purse is smoking a cigarette .	a woman with a black bag is smoking a cigarette"	4.8
STS16	9.2k	how the dow jones industrial average did wednesday	how the dow jones industrial average fared on monday	2
STS-B	5.7k	the man set up his camera to take sunset pictures at the beach .	a man is taking pictures of a lake .	2.2
SICK-R	4.5k	Two dogs are fighting	Two dogs are wrestling and hugging	4

tion which comprises the following datasets: MR Pang and Lee (2005), CR (HU; LIU, 2004), SUBJ (PANG; LEE, 2004), MPQA (WIEBE; WILSON; CARDIE, 2005), SST2/5 (SOCHER et al., 2013b), and TREC (VOORHEES; TICE, 2000). Examples are given in Table 2.6.

Table 2.6 – Text classification tasks.

Name	Task	N	examples	label(s)	classes
MR	movie review	11k	"Too slow for a younger crowd , too shallow for an older one."	neg	2
CR	product review	4k	"We tried it out christmas night and it worked great ."	pos	2
SUBJ	subjectivity status	10k	"A movie that doesn't aim too high , but doesn't need to."	subj	2
MPQA	opinion-polarity	11k	"don't want"; "would like to tell";	neg. pos	2
SST	binary sentiment analysis	67k	"Audrey Tautou has a knack for picking roles that magnify her [..]"	pos	2
SST	fine-grained sentiment analysis	8.5k	"nothing about this movie works."	0	5
TREC	question-type classification	6k	"What are the twin cities ?"	LOC:city	6

2.4.2.3 Natural Language Inference

Natural language inference is the task of determining the logical relation between a pair of sentences. This logical relation can be that of *positive entailment* (or simply entailment), *negative entailment* (contradiction), or *non-entailment*. To evaluate performance on this task, we use the SICK entailment (SICK-E, Marelli et al. (2014)) and Standford Natural Language Inference (SNLI, Bowman et al. (2015)) datasets:

2.4.2.4 Syntactic Tasks

Previously described evaluation, except for syntactic analogies, focus on semantics. As we will show, negative information mostly captures syntactic information, so we need a set of evaluations focused on syntax. For word level evaluation, we use syntactic analogies. For sentence-level evaluation, we train and test part-of-speech (POS) tag-

Table 2.7 – Natural language inference (NLI) datasets.

Name	N	Premise	Hypothesis	Label	Outputs
SNLI	560k	“A small girl wearing a pink jacket is riding on a carousel.”	“The carousel is moving.”	entailment	3
SICK-E	10k	“A man is sitting on a chair and rub- bing his eyes”	“There is no man sitting on a chair and rubbing his eyes”	entailment	3

ging using data from the Penn Treebank (MARCUS; SANTORINI; MARCINKIEWICZ, 1994). We also run two syntactic probing tasks from the SentEval suite: given a sentence, learn to classify the top constituents (TopC) and depth (Dep) of the constituency-based parse tree using only the sentence’s embedding as input.

2.4.2.5 Facet Ranking

Often, a searcher (user) provides an under-specified query to the search system, which may reflect multiple information needs, or different facets of the same intent. A conversational search refinement system attempts to pinpoint the user’s search intent via a series of *clarification questions*, which the searcher can *choose* to answer cooperatively (by volunteering additional information about their intent), lazily (“yes/no”) or not respond to the system at all, e.g., if the searcher ran out of time or patience. After each turn, the search system may choose to ask additional clarification questions, or return search results, or both. In Salle et al. (2021), we develop a *user simulator* called CoSearcher for conversational search refinement and clarification. Figure 2.2 shows two example dialogs between a search system (Bot) and CoSearcher. The dialogue starts with the user inputting a topic (an underspecified search query). The system retrieves a set of *candidate facets* for this topic. Although Salle et al. (2021) consider multiple types of candidate facets, here we focus on the topic and set of human annotated facets from the TREC09-12 diversity track (CLARKE; CRASWELL; SOBOROFF, 2009; CLARKE; CRASWELL; VOORHEES, 2012). At every turn, the system maintains a set of candidate facets and uses the dialogue context to rank them. It then proposes the top ranked facet to the user simulator. The user simulator compares the question to the true intent (the target facet), and responds with either a “Yes” or a “No”. In our work, we use as context any information received through *informative* “No”’s, where in addition to responding in the negative, the user simulator also supplies additional information to assist the search system, for example the user response “no i just want to know a few common korean phrases” from Figure 2.2. If a user rejects a facet, it is removed from the set of candidate facets.

We perform facet ranking using a semantic similarity strategy, which assigns a score for each candidate facet c by computing the similarity between it and the conversation context:

$$score(c, db) = \sum_{s \in db} Cos(rep(c), rep(s))/|db| \tag{2.15}$$

where db is the set of information “No”’s, and $rep(s)$ is the embedding of sentence s. For the purpose of evaluating facet rankers below, we consider only single contexts ($|db| = 1$).

Our evaluation dataset is derived from the Qulac dataset (ALIANNEJADI et al., 2019), which was created by having humans generate clarification question-answer pairs for each of the TREC09-12 topics and facets. For the topic “South Africa” and target facet “Find pictures of South Africa”, an example human Qulac question-answer pair instance is the question “would you like the history of south africa” and the answer “no i would like pictures of south africa”. For every human answer containing the word “no” in its first three words, we extract as the context every word after the “no”, in this case “i would like pictures of south africa”.

Candidate facets for the topic “South Africa” are (1) “Find pages that give the history of South Africa.” (2) “Find maps of South Africa.” (3, the target facet) “Find pictures of South Africa.”. The context, along with the target facet (3) and the negative candidate facets (1,2) represent one instance of our facet ranking dataset. The goal is for $score$(“Find pictures of South Africa.”, “i would like pictures of south africa”) to be higher than the score for the negative facets.

Evaluation metrics are Precision@1 (P@1, in what fraction of instances is the target facet ranked first) and Mean Reciprocal Rank (MRR, the mean inverse rank of the target facet). We split Qulac’s 198 topics (and corresponding context-facet pairs) into 100 training, 25 validation, and 73 test topics. This dataset can then be used to evaluate sentence embeddings $rep(\cdot)$, be they supervised (use training data) or unsupervised.

2.5 Conclusion

In this chapter, we reviewed existing methods for word embedding, starting from the fundamentals. We also presented the evaluations that will be used throughout the rest

Figure 2.2 – Conversational search refinement using CoSearcher.

(a)

USER: Korean language *(the topic)*

BOT: **Are you looking to find resources on translating Korean to other languages?**

COSEARCHER: no

BOT: **Ok, are you looking to find resources on learning and teaching Korean?**

COSEARCHER: no

BOT: **Ok, are you looking to find a Korean/English bilingual dictionary?**

COSEARCHER: no i just want to know a few common korean phrases

BOT: **Are you looking for what are some common Korean phrases?**

COSEARCHER: YES

True intent: "What are some common Korean phrases?"

(b)

USER: South Africa *(the topic)*

BOT: **Are you looking to find pictures of South Africa?**

CoSearcher: no but historical maps along with current ones would be good

BOT: **Are you looking to find pages that give the history of South Africa?**

CoSearcher: no i want maps of south africa

BOT: **Are you looking to find maps of South Africa?**

CoSearcher: yes

True intent = "Find maps of South Africa"

of this thesis.

3 THE LEXVEC MODEL

In this chapter, we present our LexVec word embedding model. We first define a set of desiderata or principles that a word embedding model should satisfy (Section 3.1, in answer to Q1). We then describe LexVec in detail, showing that it satisfies these principles, while also drawing connections to other word embedding models (Section 3.2, in answer to Q2). In the sections that follow, we describe a series of improvements to the base model, namely the use of positional information (Section 3.3, in answer to Q4), improved association measures (Section 3.4, in answer to Q7), scalability (Section 3.5, in answer to Q3), and the incorporation of subword information (Section 3.6, in answer to Q5).

3.1 Principled Reconstruction Error Weighting

As was shown in Section 2.3.2, different matrix factorization techniques assign different loss weights $\lambda_{w,c}$ to different word-context pairs (w, c). Consider the pair $(\textit{reigning}, \textit{champion}_1)$ – that is the expression "reigning champion" being observed in the training corpus – and the arbitrary pair (that is, having no semantic/syntactic reason for cooccurring) $(\textit{cromford}, \textit{deseronto}_1)$ – the phrase "cromford deseronto" for the small cities Deseronto in Canada and Cromford in England. In our English Wikipedia corpus, $PMI_{\textit{wiki}}$, $PMI_{\textit{reigning,champion}_1} = 6.497$ and $PMI_{\textit{cromford,deseronto}_1} = -\infty$, cooccurring more often than by chance and not cooccurring at all respectively.

SVD assigns loss weights uniformly, so $\lambda_{\textit{reigning,champion}_1} = \lambda_{\textit{cromford,deseronto}_1}$. Given that model capacity is limited – the multiplication of word and context matrices cannot preserve all the information in the matrix being factorized – loss weights determine which information should be better preserved. And it stands to reason that the cooccurrence of $(\textit{reigning}, \textit{champion}_1)$, a common expression in the English language, is more meaningful than the non-cooccurrence of $(\textit{deseronto}, \textit{cromford}_1)$ for at least two reasons: (1) in downstream applications, for example, having a better meaning representation of "reigning" is more important than that of "deseronto" because it is much more likely to appear, and so dedicating more model capacity to correctly recovering its context distribution is more important (2) there is little information in the non-cooccurrence of two rare words such as "deseronto" and "cromford" – two random words of low frequency are almost certain not to cooccur, so the model does not learn much from this observation, whereas observing "reigning champion" reveals an important expression in the English language. In designing loss error weights, such frequent cooccurrences should be upweighted.

Let us now consider the pair $(\textit{should}, \textit{began}_1)$: observing the phrase "should began" in the corpus. This pair does not occur at all in the English Wikipedia, having $PMI_{\textit{should},\textit{began}_1} = -\infty$. GloVe assigns a reconstruction error weight of zero, thus making no use of this information when learning word and context embeddings. But there is certainly valuable information here: the non-cooccurrence of "should began" tells us that this construction is grammatically incorrect. It captures syntactic information about the word "should": that it cannot be followed by a certain verb in past tense, in this case "began". Therefore, the loss weight $\lambda_{\textit{should},\textit{began}_1}$ should be greater than zero.

As a final example, consider the pair $(\textit{should}, \textit{deseronto}_1)$, which is also non-cooccurring in the English Wikipedia. The phrase "*should deseronto* raise taxes?" is both syntactically and semantically correct, so the non-cooccurrence can be attributed to the limited size of the corpus and the infrequent word "deseronto". Contrast this with "should began", where both words are very frequent: the phrase is unobserved because it is not syntactically valid. We thus argue that the non-cooccurrence of a pair of frequent words is more likely to carry information than the non-cooccurrence involving a rare word. Thus, when designing loss weights, $\lambda_{\textit{should},\textit{began}_1}$ should be greater than $\lambda_{\textit{should},\textit{deseronto}_1}$. Neither GloVe nor SVD have this property, since the former ignores non-cooccurrences, and the later assigns uniform weights.

Drawing upon these observations, we argue that *principled reconstruction error weighting* $\lambda_{w,c}$ should have the following properties:

- **Bigram Monotonic:** $\lambda_{w,c}$ is a strictly monotonically increasing function of coocurrences $M_{w,c}$.
- **Non-zero:** $\lambda_{w,c} > 0$.
- **Unigram Monotonic:** $\lambda_{w,c}$ is a strictly monotonically increasing function of unigram counts $M_{w,*}$ and $M_{*,c}$.

We can also frame these same observations from the perspective of statistical reliability of PMI estimates. That is, if we were to recompute PMI values from a resampling of a corpus, $PMI_{w,c}$ estimates for certain pairs (w, c) are likely to suffer less change (to be more *reliable* or consistent) than for other pairs. The more frequent a word w and context c are in the corpus, the more confident we can be about the corpus estimated $PMI_{w,c}$ (the more reliable the estimate). Suppose both $\#(w)$ and $\#(c)$ are high, but $PMI_{w,c}$ is low.

This is unequivocal evidence of negative correlation between them, and so we should put more effort into approximating their PMI. The argument is analogous for high PMI. If on the other hand $\#(w)$ and $\#(c)$ are low, we cannot be too confident about the corpus estimated $PMI_{w,c}$, and so less effort should be spent on its approximation.

3.2 LexVec

We now propose our matrix factorization model LexVec, then show that it performs principled reconstruction error weighting.

LexVec performs matrix factorization by sliding a symmetric window over the training corpus (window sampling), in the exact same way as when M was constructed in Section 2.2, and performing one Stochastic Gradient Descent (SGD) step every time a (w, c) pair is observed, minimizing

$$l(w, c) = \frac{1}{2}(W_w C_c^\top - f(M)_{w,c})^2 \tag{3.1}$$

Additionally, *negative sampling* is performed, where for every center word w, k negative samples (MIKOLOV et al., 2013) are drawn from the unigram context distribution:

$$P_n(c) = (M_{*,c})^{\alpha_{pow}} / \sum_c (M_{*,c})^{\alpha_{pow}} \tag{3.2}$$

where α_{pow} is a smoothing factor (set to .75 in this thesis) and SGD steps are taken to minimize

$$l_{neg}(w) = \sum_{i=1}^{k} \mathbf{E}_{c_i \sim P_n(c)} l(w, c_i) \tag{3.3}$$

Figure 3.1 illustrates a step of LexVec over a single target word. Algorithm 1 describes the entire algorithm, which has time complexity $O(|corpus|(2l + k)d)$.

The expected loss for a pair (w, c) in a single pass over the training corpus is equal to the pairwise loss $l(w, c)$ (Equation (3.1)) multiplied by (1) the number of times the pair is observed using window sampling ($M_{w,c}$), *and* (2) the expected number of times the pair is sampled using negative sampling, which is equal to the number of times w is a target word ($M_{w,*}/2l$[1]) multiplied by the number of negative samples per target word and the

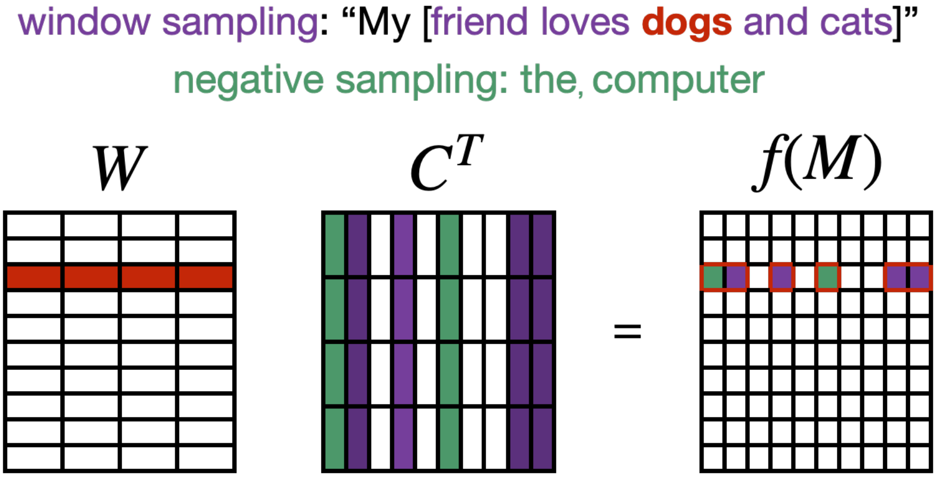
---

[1] $M_{w,*}$ is incremented by $2l$ every time w is a target word, thus the division by $2l$ recovers the number of times it is a target word.

Algorithm 1 LexVec

```
procedure LexVec($corpus, l, d, k, \alpha_{pow}, iters, initialLr, f$)    ▷ $l$: window size, $d$: embedding dimension, $k$: negative samples, $\alpha_{pow}$: smoothing factor, $iters$: training iterations, $initialLr$: initial learning rate, $f$: matrix transformation
    $M \leftarrow ConstructCoocMatrix(corpus, l)$    ▷ Construct cooccurrence matrix by sliding symmetric window of size $l$ over corpus
    $T \leftarrow f(M)$    ▷ Transformed cooccurrence matrix
    $P_n \leftarrow (M_{*,c})^{\alpha_{pow}} / \sum_c (M_{*,c})^{\alpha_{pow}}$ for every context $c$    ▷ Negative sampling distribution
    $W \leftarrow$ Initialize $(M.rows \times d)$ matrix with values drawn from $U(-\frac{1}{2d}, \frac{1}{2d})$
    $C \leftarrow$ Initialize $(M.cols \times d)$ matrix with values drawn from $U(-\frac{1}{2d}, \frac{1}{2d})$
    for $iter \leftarrow 1, iters$ do
        for $w, contexts, progress \leftarrow IterateOverWindows(corpus, l)$ do    ▷ Slide symmetric window of size $l$ over corpus, yielding for each window: the target word $w$, $2l$ surrounding $contexts$, and $progress$ indicating relative corpus position from 0 to 1
            $lr \leftarrow (1 - \frac{iter-1+progress}{iters}) * initialLr$    ▷ Linear learning rate decay
            for $c \leftarrow contexts$ do    ▷ Window sampling
                $SGDStep(T_{w,c}, W, C, w, c, lr)$
            end for
            for $c_n \leftarrow$ Draw k samples from $P_n$ do    ▷ Negative sampling
                $SGDStep(T_{w,c_n}, W, C, w, c_n, lr)$
            end for
        end for
    end for
    return $W, C$
end procedure
procedure SGDStep($t, W, C, w, c, lr$)    ▷ $W, C$ passed by reference
    $err = W_w C_c^\top - t$
    $\Delta W_w \leftarrow err * C_c$
    $\Delta C_c \leftarrow err * W_w$
    $W_w \leftarrow W_w - lr * \Delta W_w$
    $C_c \leftarrow C_c - lr * \Delta C_c$
end procedure
```

Figure 3.1 – LexVec running over a single target word “dogs”, window sampled words “friend”, “loves”, “and”, “cats”, and negatively sampled words “the” and “computer”. Highlighted $f(M)$ squares on the right represent lookups to the transformed PMI values between target word and every window sampled and negatively sampled contexts. Vectors being updated by SGD: (1) Red row: target word vector. (2) Purple columns: window sampled context vectors. (3) Green columns: negatively sampled context vectors. Equation (3.1) is minimized through SGD for each lookup.


window sampling: “My [friend loves dogs and cats]”
negative sampling: the, computer
W
C^T
f(M)
=

probability of c being sampled:

$$\begin{aligned}
\mathbf{E}[L_{LexVec}(w,c)] &= \underbrace{M_{w,c} l(w,c)}_{ws} + \underbrace{\frac{M_{w,*}}{2l}\; k\; P_n(c)\; l(w,c)}_{ns} && (3.4)\\
&= (M_{w,c} + \frac{M_{w,*}}{2l}\; k\; P_n(c))\; l(w,c) && (3.5)\\
&= (M_{w,c} + \frac{M_{w,*}}{2l}\; k\; P_n(c))\; \frac{1}{2}(W_w C_c^\top - f(M)_{w,c})^2 && (3.6)\\
&= \frac{1}{2}\lambda_{LexVec}(w,c)(W_w C_c^\top - f(M)_{w,c})^2 && (3.7)\\
\lambda_{LexVec}(w,c) &= \underbrace{M_{w,c}}_{\lambda_{ws}(w,c)} + \underbrace{\frac{M_{w,*}}{2l}\; k\; P_n(c)}_{\lambda_{ns}(w,c)} && (3.8)
\end{aligned}$$

In λ_{LexVec}, the first term prioritizes the correct approximation of frequently cooccurring pairs (window sampling), and the second term of pairs where either word occurs with high frequency (negative sampling). It is easy to see that λ_{LexVec} is a principled reconstruction error weighting:

- **Bigram Monotonic:** λ_{LexVec} is a strictly monotonically increasing function of $M_{w,c}$ because of the term λ_{ws}, by definition.
- **Non-zero:** $\lambda_{LexVec} > 0$ because of the term λ_{ns} which is strictly positive (all words and contexts in the vocabulary appear at least once the training corpus).
- **Unigram Monotonic:** λ_{LexVec} is a strictly monotonically increasing function of $M_{w,*}$ and $M_{*,c}$ because of the term λ_{ns}, by definition.

Table 3.1 shows which of the above 3 properties are satisfied by explicit or implicit matrix factorization models. The only explicit matrix factorization model that satisfies all 3 properties is LexVec.

Table 3.1 – Principled reconstruction error weighting and the models that respect them. Top row contains the Skip-gram model that performs implicit matrix factorization, and the bottom rows are models that perform explicit matrix factorization.

model	Bigram Monotonic	Non-zero	Unigram Monotonic
Skip-gram	✓	✓	✓
SVD		✓	
GloVe	✓[2]		
SwiVel	✓	✓	
LexVec	✓	✓	✓

---

[2] GloVe is not *strictly* monotonic since error weights in Equation (2.7) are capped.

Connection to Skip-gram: The Skip-gram loss function (Equation (2.10)) for a single pass over the training corpus for a specific (w, c) is very similar to the LexVec loss function (Equation (3.4)). Restating Equation (2.10) here and performing substitution using LexVec's λ terms (Equation (3.8) we get:

$$\mathbf{E}[J_{sg}(w,c)] = M_{w,c} \log \sigma(W_w C_c^\top) + M_{w,*}\ k\ P_n(c)\ \log \sigma(-W_w C_c^\top) \tag{3.9}$$

$$= \lambda_{ws}(w,c) \log \sigma(W_w C_c^\top) + 2l\ \lambda_{ns}(w,c) \log \sigma(-W_w C_c^\top) \tag{3.10}$$

Skip-gram shares the same LexVec weight terms $\lambda_{ws}, \lambda_{ns}$ in J. However, in contrast to this apparent similarity, there is a subtle difference between the models in the way errors are weighted. Suppose we have found parameters that perfectly factorize the PMI matrices, such that for LexVec $WC^T = CPMI(-2)$ and for Skip-gram $WC^T = PMI - \log k$ (we use the same W, C for notational convenience, but the factorizations are necessarily different). We then introduce a small error ϵ into the dot product $W_w C_c^T$ so that for LexVec:

$$\epsilon = W_w C_c^T - CPMI_{w,c}(-2) \tag{3.11}$$

$$L_{LexVec+\epsilon}(w,c) = \frac{1}{2} \lambda_{LexVec}(w,c)\ (W_w C_c^T - CPMI(-2))^2 \tag{3.12}$$

$$= \frac{1}{2} (\lambda_{ws}(w,c) + \lambda_{ns}(w,c))\ \epsilon^2 \tag{3.13}$$

$$\tag{3.14}$$

and for Skip-gram:

$$\epsilon = W_w C_c^T - (PMI_{w,c} - \log k) \tag{3.15}$$

$$W_w C_c^T = \epsilon + (PMI_{w,c} - \log k) \tag{3.16}$$

$$J_{sg+\epsilon}(w,c) = \lambda_{ws}(w,c) \log \sigma(W_w C_c^\top) + 2l\ \lambda_{ns}(w,c) \log \sigma(-W_w C_c^\top) \tag{3.17}$$

$$= \lambda_{ws}(w,c) \log \sigma(\epsilon + (PMI_{w,c} - \log k)) \tag{3.18}$$

$$+\ 2l\ \lambda_{ns}(w,c) \log \sigma(-(\epsilon + (PMI_{w,c} - \log k))) \tag{3.19}$$

The LexVec loss deviation from 0 depends only on the weights λ and the error ϵ introduced, whereas the Skip-gram deviation from the objective's maximum depends on the weights λ, the error ϵ, *and the PMI value*; because of the logistic function, reconstruction errors for PMI values near zero affect the objective much more than for high and low PMI values. If we re-word small reconstruction errors as *sharp* and large reconstruction errors as *fuzzy*, the fuzziness of LexVec approximations is entirely determined by the weights. The Skip-gram objective, in contrast, has opposing forces, with these same "sharpnesses" encouraged by λ_{ws} and λ_{ns} being opposed by the fuzziness of the logistic function at extreme PMI values. Despite this subtle difference in error weightings, as we will see in Chapter 6, the models behave very similarly empirically.

Connection to GloVe: GloVe factorizes the logarithm of the cooccurrence matrix with added bias terms (Equation (2.7)). Although it is not clear what the optimal values of these bias terms are, Shazeer et al. (2016) observe that these terms are highly correlated to the respective word and context corpus frequencies, such that the matrix being approximated could resemble PMI (Equation (2.1)). This, combined with the fact that GloVe's loss weighting resembles the window sampling weight λ_{ws}, suggests a connection between the LexVec model with no negative sampling and GloVe.

Connection to SVD: Since $\lambda_{SVD}(w,c) = 1$ is uniform for all word-context pairs, the fraction of $|nPMI|/|nPMI \cup pPMI|$ gives the fraction of total loss weights assigned to negative information, which in the case of PMI_{wiki5} is equal to 99.27%. The SVD is thus an extreme case of prioritizing negative information.

3.3 Positional Contexts

Positional contexts can be incorporated into LexVec with two minor modifications: 1) The context embedding C takes on dimensions $2 * win * |V| \times d$, 2) Negative sampling must now sample positional contexts rather than simple contexts. This latter point requires that the distribution from which negative samples are drawn become:

$$P_n(c^i) = M^{\alpha}_{*c^i} / \sum_{c^i} M^{\alpha}_{*c^i} \tag{3.20}$$

3.3.1 Combining Word and Context Vectors

Without positional contexts, either W or $W+C$ can be used as embeddings. Since positional contexts make the dimensions of both matrices incompatible, C cannot be used directly. We propose three ways to combine word and context embeddings for a word w:

Sum: $W_w + \sum_{-win \leq i \leq win, i \neq 0} C_{w_i}$

Context-only: $\sum_{-win \leq i \leq win, i \neq 0} C_{w_i}$

Mean: $W_w + \frac{1}{2win} \sum_{-win \leq i \leq win, i \neq 0} C_{w_i}$

Concat: Concatenate $(W_w, C_{w_{-win}}, ..., C_{w_{win}})$.

Note that while Sum and Mean preserve the original dimensionality d of the word embedding, Concat increases it to $d(1 + 2win)$.

All three techniques can be used without positional contexts. In that case, Sum and Mean are equivalent, and are identical to the way GloVe combines word and context vectors.

3.4 Spectrum of PMI

To better understand the distribution of PMI, we plot in Figure 3.2 a histogram of PMI_{wiki5} values of 10^5 non-zero pairs randomly sampled from M_{wiki5}. We sample only non-zero pairs because M_{wiki5} is sparse: only 0.93% of cells are non-zero.

To the left of the 0 line, we can clearly see the spectrum of nPMI that is collapsed when using the PPMI measure, which maps these negative values – $\sim$ 22.2% of *cooccurring pairs* – to 0.

3.4.1 Preserving the spectrum of negative information

To deal with values in mnPMI, we propose clipped PMI,

$$CPMI_{w,c}(z) = max(z, PMI_{w,c}) \tag{3.21}$$

which is equivalent to PPMI when $z = 0$, and captures most of the nPMI spectrum when $z \leq 2$.

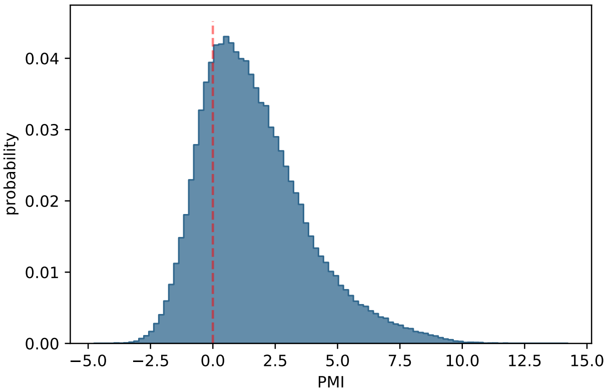
We also experiment with normalized PMI ($NPMI$) (BOUMA, 2009):

$$NPMI_{w,c} = PMI_{w,c} / - log(M_{w,c}/M_{**})$$

such that $NPMI(w, c) = -1$ when $(w, c) \in mnPMI$ (never cooccur), $NPMI(w, c) = 0$ when they are independent, and $NPMI(w, c) = 1$ when they always cooccur together. This effectively captures the entire negative spectrum, but has the downside of normalization which discards scale information. In practice, we find this works poorly if done symmetrically, so we introduce a variant called $NNEGPMI$ which only normalizes nPMI:

$$NNEGPMI_{w,c} = \begin{cases} NPMI_{w,c} & \text{if } PMI_{w,c} < 0 \\ PMI_{w,c} & \text{otherwise} \end{cases}$$

Figure 3.2 – Histogram (bin width equal to 0.2) of 10^5 values *not in mnPMI* sampled from PMI_{wiki5}. The negative spectrum of PMI are the values to the left of the dashed line ($\sim 22.2\%$ of sampled values), which are collapsed to 0 when using the popular PPMI association measure. Note that we exclude mnPMI (sample only non-zero cooccurrences) otherwise the graph would be a single vertical line at -2 (if graphing $CPMI(-2)$) since 99.07% of values in M_{wiki5} are 0.

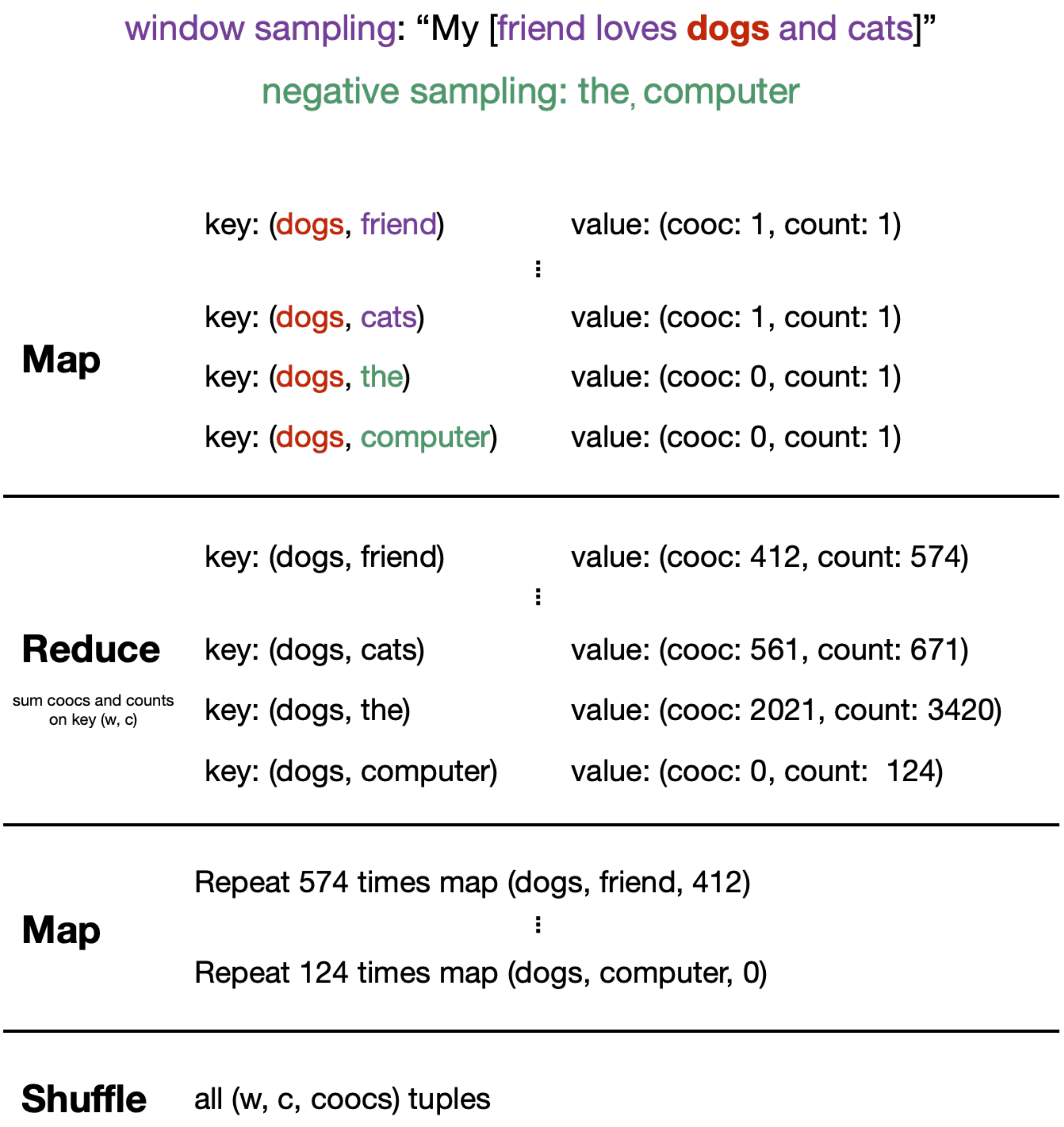


We also experimented with Laplace smoothing as in Turney and Littman (2003) for various pseudocounts but found it to work consistently worse than both $CPMI$ and $NNEGPMI$, so we omit further discussion in this thesis.

3.5 Scaling to Larger Corpora

As window sampling scans over the training corpus and negative sampling selects random contexts, (w, c) pairs are generated and the corresponding $PMI_{w,c}$ cell must be accessed so that Equation (3.4) can be minimized. Unfortunately, this results in random access to the PMI matrix which requires it to be kept in main memory. Pennington, Socher and Manning (2014) show that the under certain assumptions, this sparse matrix grows as $O(|corpus|^{0.8})$, which bounds the maximum corpus size that can be processed by LexVec. To overcome this limitation, the PMI matrix must be represented in external memory (such as a hard drive), where random access is prohibitive, and we must instead use sequential access.

Figure 3.3 – Example MapReduce steps used to construct the F stream used by the external memory variant of LexVec.


window sampling: “My [friend loves dogs and cats]”
negative sampling: the, computer
Map
key: (dogs, friend) value: (cooc: 1, count: 1)
⋮
key: (dogs, cats) value: (cooc: 1, count: 1)
key: (dogs, the) value: (cooc: 0, count: 1)
key: (dogs, computer) value: (cooc: 0, count: 1)
Reduce
sum coocs and counts on key (w, c)
key: (dogs, friend) value: (cooc: 412, count: 574)
⋮
key: (dogs, cats) value: (cooc: 561, count: 671)
key: (dogs, the) value: (cooc: 2021, count: 3420)
key: (dogs, computer) value: (cooc: 0, count: 124)
Map
Repeat 574 times map (dogs, friend, 412)
⋮
Repeat 124 times map (dogs, computer, 0)
Shuffle
all (w, c, coocs) tuples

We can derive an exact equivalent of LexVec that uses external memory rather than main memory if we represent both matrix construction as matrix access through a MapReduce paradigm (DEAN; GHEMAWAT, 2008). Doing so simultaneously enables the use of external memory and the use of highly scalable MapReduce infrastructure.

We can obtain $M_{w,*}$ through a $map(key : w, value : 1)$ every time a window sampling operation (w, c) is performed, followed by a sum reduction. Computing $M_{*,c}$ is analogous. We can store these values in-memory since the resulting arrays only have length $|V_w|$ and $|V_c|$.

Computing $M_{w,c}$ is only marginally more complicated. We start by defining the window sampling operation as $map(key : (w, c), value : (cooc : 1, count : 1))$. Likewise, we define the negative sampling operation as $map(key : (w, c), value : (cooc : 0, count : 1))$. We then reduce on the key with the function $reduce((cooc_a, count_a), (cooc_b, count_b)) \to (cooc_a + cooc_b, count_a + count_b)$. The result is that for every pair (w, c), we have $(totalcoocs, totalcount)$. Note that $totalcoocs = M_{w,c}$. We then perform $totalcount$ operations of $map(key : (w, c), value : totalcoocs)$. Finally, we perform a shuffle operation, which is trivial under MapReduce. Let us refer to this stream of shuffled values $((w, c), totalcoocs)$ as F. By construction, this stream is identical to the sequence of random accesses LexVec would perform in a single iteration over the training corpus, differing only in the order of the observed pairs. Having $M_{w,c}$, $M_{w,*}$, $M*, c$, we can compute $PMI_{w,c}$ for each of these pairs in F, such that the resulting total loss function being optimized in this external memory variant of LexVec is identical to Equation (3.4). Figure 3.3 gives an example of an external memory LexVec run. The SGD optimization that uses F to train W, C can also be distributed using techniques such as Hogwild (RECHT et al., 2011).

For reference, we present in Algorithm 2 a single machine version of this external memory approach. F's construction and storage requires $O(|corpus|(2l + k))$ disk space and $O(|corpus|(2l+k)\log(|corpus|(2l+k)))$ time[3], but only $O(|V|)$ main memory, making it scalable to large corpora. Otimization of W, C has the same $O(|corpus|(2l + k)d)$ time complexity as Algorithm 1.

---

[3] Sorting and shuffling in external memory are $O(N \log N)$ (AGGARWAL; VITTER JEFFREY, 1988) and $O(N)$ (SANDERS, 1998) respectively.

Algorithm 2 External memory LexVec

```
procedure LexVec($corpus, l, d, k, \alpha_{pow}, iters, initialLr, f$)    ▷ $l$: window size, $d$: embedding dimension, $k$: negative samples, $\alpha_{pow}$: smoothing factor, $iters$: training iterations, $initialLr$: initial learning rate, $f$: matrix transformation
    $MW \leftarrow$ zero vector of dimension equal to # of words
    $MC \leftarrow$ zero vector of dimension equal to # of contexts
    for $w, contexts, progress \leftarrow IterateOverWindows(corpus, l)$ do    ▷ As in Algorithm 1
        for $c \leftarrow contexts$ do
            $MW[w] \leftarrow MW[w] + 1$
            $MC[c] \leftarrow MC[c] + 1$
        end for
    end for
    $totalCoocs \leftarrow Sum(MW)$
    $P_n \leftarrow MC[c]^{\alpha_{pow}} / \sum_c MC[c]^{\alpha_{pow}}$ for every context $c$    ▷ Negative sampling distribution
    $S \leftarrow$ empty file of (word, context, cooc, count) records
    for $w, contexts, progress \leftarrow IterateOverWindows(corpus, l)$ do
        for $c \leftarrow contexts$ do    ▷ Window sampling
            Write $(w, c, 1, 1)$ to $S$
        end for
        for $c_n \leftarrow$ Draw k samples from $P_n$ do    ▷ Negative sampling
            Write $(w, c_n, 0, 1)$ to $S$
        end for
    end for
    Sort $S$ records lexicographically    ▷ merge sort using external memory
    Collapse adjacent lines in $S$ where $(word, context)$ match, summing $(cooc, count)$
    $F \leftarrow$ empty file of (word, context, cooc) records
    for $(w, c, cooc, count), progress \leftarrow IterateOverRecords(S)$ do
        for $i \leftarrow 1, count$ do
            Write $(w, c, cooc)$ to $F$
        end for
    end for
    Shuffle $F$    ▷ Using external memory
    $W \leftarrow$ Initialize $(|MW| \times d)$ matrix with values drawn from $U(-\frac{1}{2d}, \frac{1}{2d})$
    $C \leftarrow$ Initialize $(|MC| \times d)$ matrix with values drawn from $U(-\frac{1}{2d}, \frac{1}{2d})$
    for $iter \leftarrow 1, iters$ do
        for $(w, c, cooc), progress \leftarrow IterateOverRecords(F)$ do
            $lr \leftarrow (1 - \frac{iter-1+progress}{iters}) * initialLr$    ▷ Linear learning rate decay
            $t \leftarrow f(cooc, MW[w], MC[c], totalCoocs)$    ▷ Compute PMI transform; note that $cooc = M_{w,c}, MW[w] = M_{w,*}, MC[c] = M_{*,c}, totalCoocs = M_{*,*}$
            $SGDStep(t, W, C, w, c, lr)$    ▷ From Algorithm 1
        end for
    end for
    return $W, C$
end procedure
```

3.6 Incorporating Subword Information

If our model contains an embedding for the word “true” but not for the word “*un*true”, how can we recover the meaning of the latter? The issue with learning word vectors which do not account for subword information mirrors the problem with using one-hot vectors to represent words: although we have distributed rather than symbolic representations, we still use words as symbols to access distinct dense vectors.

We would like to instead break “untrue” into the constituents “un” (negation) and “true” and somehow combine the two vectors to recover the meaning of the entire word even if it was not seen at training time, resulting in semantics similar to the word “false”. In other words, we would like our model to be aware of *morphology*: the study of combining subword units known as *morphemes* (“un”, “true”) to represent words. This would not only solve the meaning-altering derivational morphology example given above – “un” negates the meaning of “true” – but also of inflectional morphology such as recovering the meaning of “presented” if the morphemes “present” and “ed” are in the model’s vocabulary.

Another motivating example is representing new words arising from word formation, such as “marvelicious”. Although never having seen the word before, a native speaker is able to guess that the underlying meaning is related to that of “marvelous” and perhaps “delicious”. Closely related is representing misspelled forms of words which might be absent in training corpora such as Wikipedia but might be common in downstream applications such as sentiment classification on social media.

Finally, a model that is aware of subword information is not only able to represent words unseen at training time, but can potentially better represent rare words through information sharing with more frequent words or forms, such as the rarer adjective “constitutive” sharing information with the more frequent verb “constitutes”.

fastText (BOJANOWSKI et al., 2017) addresses these issues in the Skip-gram word2vec model by representing a word by the sum of a unique vector and a set of shared character n-grams (from hereon simply referred to as n-grams) vectors. This addresses both issues above as learned information is shared through the n-gram vectors and from these OOV word representations can be constructed.

The LexVec objective is modified such that a word’s vector is the sum of all its subword vectors. We compare 1) the use of n-gram subwords, like fastText, and 2) unsupervised morphemes identified using Morfessor (VIRPIOJA et al., 2013) to learn whether more linguistically motivated subwords offer any advantage over simple n-grams.

Given a set of subwords S_w for a word w, we follow fastText and replace W_w in Equation (3.1) by W'_w:

$$W'_w = \frac{1}{|S_w| + 1}(W_w + \sum_{s \in S_w} q_{hash(s)}) \tag{3.22}$$

such that a word is the sum of its word vector and its d-dimensional subword vectors q_x. The number of possible subwords is very large so the function $hash(s)^4$ hashes a subword to the interval $[1, buckets]$. For OOV words,

$$W'_w = \frac{1}{|S_w|} \sum_{s \in S_w} q_{hash(s)} \tag{3.23}$$

We compare two types of subwords: simple n-grams (like fastText) and unsupervised morphemes. For example, given the word "cat", we mark beginning and end with angled brackets and use all n-grams of length 3 to 6 as subwords, yielding $S_{\text{cat}} = \{\langle\text{ca}, \text{at}\rangle, \text{cat}\}$. Morfessor (VIRPIOJA et al., 2013) is used to probabilistically segment words into morphemes. The Morfessor model is trained using raw text so it is entirely unsupervised. For the word "subsequent", we get $S_{\text{subsequent}} = \{\langle\text{sub}, \text{sequent}\rangle\}$.

3.7 Conclusion

In this chapter, we defined the set of principled reconstruction error weighting properties and used these to design the LexVec model, also showing how existing models fit into this framework. We then made a series of refinements – positional information, subwords, and scalability – to the LexVec model. With its explicit access to the underlying PMI matrix being factorized, we have the necessary instrument for our investigation into the role of negative information in word embedding models to follow in Chapters 5 and 6.

---

[4] http://www.isthe.com/chongo/tech/comp/fnv/

4 EVALUATION

We study in this chapter whether LexVec, which follows the principled reconstruction error weighting properties, closes the performance gap between counting and predictive methods in intrinsic and downstream evaluations. In Section 4.1, we compare the base LexVec model, with no positional or subword information and no hyperparameter tuning, to Skip-gram, GloVe, and SVD models (in answer to Q2). We then look at how positional information affects task performance Section 4.1.2.2 (in answer to Q4). Next, in Section 4.2, we evaluate whether the highly-scalable external memory variant does indeed approximate the standard in-memory model (in answer to Q3). Section 4.3 compares different types of subwords and their impact on task performance (in answer to Q5 and Q6). In Section 4.4, we compare the downstream task performance of LexVec and popular word embeddings models (in answer to Q8). Finally, in Section 4.5, we study in what tasks static word embeddings remain competitive (or not) with Transformer models (in answer to Q9 and Q10).

4.1 Base Model

4.1.1 Materials

Unless otherwise noted, all models were trained on a dump of Wikipedia from June 2015, split into sentences, with punctuation removed, numbers converted to words, and lower-cased. Words with less than 100 counts were removed, resulting in a vocabulary of 303,517 words. All models generate embeddings of 300 dimensions.

The PPMI matrix used by both PPMI-SVD and LexVec was constructed using smoothing of $\alpha_{cds} = .75$ and a fixed window of size 2. Subsampling of the corpus is adopted for PPMI and Skip-gram with threshold of $t = 10^{-5}$. Additionally, Skip-gram uses 5 negative samples, a window of size 10, for 5 iterations with initial learning rate set to the default 0.025. We run LexVec and Skip-gram for 5 iterations over the training corpus. Both LexVec and Skip-gram use $\alpha_{pow} = .75$ for the negative sampling distribution. GloVe is run with a window of size 10, $x_{max} = 100$, $\beta = .75$, for 50 iterations and initial learning rate of 0.05.

All methods generate both word and context matrices (W and C): W is used for LexVec, Skip-gram, PPMI-SVD and $W + C$ for GloVe.

4.1.2 Results

Table 4.1 – Spearman rank correlation on word similarity tasks.

model	SimLex	RW	WS-S	WS-R	MEN	MTurk	Mean
PPMI-SVD	30.3	42.7	72.9	61.2	73.1	62.7	57.2
GloVe	33.8	40.0	71.5	62.3	73.6	64.3	57.6
Skip-gram	33.9	**46.5**	**77.1**	65.7	**76.3**	**67.5**	**61.2**
LexVec	33.6	45.8	76.4	**65.9**	76.0	65.5	60.5
LexVec (Pos)	**35.8**	46.4	74.4	61.9	74.4	64.5	59.5

Table 4.2 – Results on word analogy tasks, given as percent accuracy.

model	GSem	GSyn	MSR	Mean
PPMI-SVD	46.0	44.5	30.3	40.2
GloVe	**81.8**	63.0	**53.9**	**66.3**
Skip-gram	77.3	**64.2**	48.1	63.2
LexVec	79.4	54.3	37.8	57.2
LexVec (Pos)	80.8	63.2	49.6	64.5

4.1.2.1 LexVec

Results for word similarity and for the analogy tasks are given in Tables 4.1 and 4.2, respectively. Compared with PPMI-SVD, LexVec performs better in all tasks. As they factorize the same PPMI matrix, PPMI-SVD's lower performance must be due to uniform loss weights. LexVec is competitive with Skip-gram in several word similarity tasks. It outperforms Skip-gram on the semantic analogy task, nearly approaching GloVe's performance. LexVec's poor syntactic analogy performance is a result of the PPMI transformation (PPMI-SVD also struggled with syntactic analogies more than any other task), which is remedied in sections Sections 4.1.2.2 and 5.1.1 with the use of positional contexts and alternative PMI transformations.

LexVec outperforms GloVe on all but one word similarity task. Of particular note is the large improvement on the RW dataset when compared to GloVe. As will be further explored in Chapter 6, the incorporation of negative information is crucial to strong rare word representations. Overall, LexVec closes the gap in word similarity performance between GloVe and Skip-gram, or equivalently, between counting and predictive methods.

Results for the fMRI task are shown in Table 4.3. Although Skip-gram has the highest mean accuracy, we observe that there is no best model for *all subjects*, which

Table 4.3 – Leave-two out cross-validation results on the fMRI task. Numbers are percent accuracy.

model	1	2	3	4	5	6	7	8	9	Mean
PPMI-SVD	88.5	76.0	76.1	**91.2**	78.0	**74.7**	76.8	68.8	76.7	78.5
GloVe	89.5	74.4	**83.1**	89.0	76.3	67.3	68.3	**75.3**	75.3	77.6
Skip-gram	**90.4**	76.9	80.9	90.6	74.1	72.1	**80.7**	66.0	**77.7**	**78.9**
LexVec	87.9	**78.0**	79.0	88.9	**78.2**	70.7	67.2	64.1	72.4	76.3
LexVec (Pos)	89.4	75.9	79.9	89.6	72.7	74.5	63.9	65.0	74.7	76.2

makes it hard to nominate a single best model. Similar non-unanimity was observed in Abnar et al. (2018). Abnar et al. (2018) and Branco et al. (2020) report strong results on this task with embeddings trained using syntactic dependencies as contexts. Training LexVec using syntactic dependencies is an interesting experiment for future work.

4.1.2.2 Positional Contexts

Positional contexts improved performance in both similarity (Table 4.1) and analogy tasks (Table 4.2). Their use significantly improved LexVec's performance on syntactic analogies. This confirms the relevance of using positional contexts to capture syntactic information.

4.2 Scaling using External Memory

4.2.1 Materials

We use the same setup and LexVec models as in Section 4.1, but now train external memory (EM) variants.

4.2.2 Results

Results are shown in Tables 4.4 and 4.5. The external memory implementation very closely approximates the *standard* variant (without the use of external memory), which was expected given that they minimize the exact same loss function. There is, however, a small gap in performance in favor of the standard variant. We attribute this to the subsampling that the standard variant performs in *each iteration*: in each pass over

Table 4.4 – Spearman rank correlation on word similarity tasks when using the external memory LexVec.

model	SimLex	RW	WS-S	WS-R	MEN	MTurk
LexVec	33.6	45.8	**76.4**	**65.9**	**76.0**	**65.5**
LexVec EM	32.3	44.4	74.4	62.1	73.0	61.9
LexVec (Pos)	**35.8**	**46.4**	74.4	61.9	74.4	64.5
LexVec EM (Pos)	35.6	45.6	76.1	62.2	74.0	63.6

Table 4.5 – Results on word analogy tasks, given as percent accuracy when using the external memory LexVec.

model	GSem	GSyn	MSR
LexVec	79.4	54.3	37.8
LexVec EM	76.0	52.8	33.8
LexVec (Pos)	**80.8**	**63.2**	**49.6**
LexVec EM (Pos)	78.3	61.1	45.6

the training corpus, subsampling is performed, meaning that the (w, c) pairs seen over each pass are likely to differ. The external memory variant sees the same (w, c) pairs in each iteration as were seen during cooccurrence matrix construction, thus SGD steps are performed on a smaller set of pairs than are performed by the standard variant.

4.3 Subwords

Our experiments aim to measure if the incorporation of subword information into LexVec results in similar improvements as observed in moving from Skip-gram to fastText, and whether unsupervised morphemes offer any advantage over n-grams. For IV words, we perform intrinsic evaluation via word similarity and word analogy tasks. OOV word representation is tested through qualitative nearest-neighbor analysis.

4.3.1 Materials

Model parameters are identical to those in Section 4.1, but use only LexVec with positional contexts. Morfessor is trained on that same vocabulary of $303,517$ words.

Subword models include (1) LexVec using n-grams (LV-N) (2) LexVec using unsupervised morphemes (LV-M) (3) fastText (FT) trained using the reference implementation[1] of fastText with the hyper-parameters given by Bojanowski et al. (2017) (window $=$

---

[1] https://github.com/facebookresearch/fastText

5, initial learning rate $= .025$, subsampling $= 10^{-4}$, negative samples $= 5$).

All five models are run for 5 iterations over the training corpus and generate 300 dimensional word representations. LV-N, LV-M, and FT use 2000000 buckets when hashing subwords.

Finally, we use LV-N, LV-M, and FT to generate OOV word representations for the following words: 1) "hellooo": a greeting commonly used in instant messaging which emphasizes a syllable. 2) "marvelicious": a made-up word obtained by merging "marvelous" and "delicious". 3) "louisana": a misspelling of the proper name "Louisiana". 4) "rereread": recursive use of prefix "re". 5) "tuzread": made-up prefix "tuz".

4.3.2 Results

Table 4.6 – Spearman rank correlation on word similarity tasks when incorporating subword information.

model	SimLex	RW	WS-S	WS-R	MEN	MTurk
FT	36.7	50.0	78.1	65.7	75.5	65.1
Skip-gram	37.1	48.1	**78.8**	**66.8**	**76.0**	**66.7**
LexVec	35.9	46.1	74.8	61.2	74.8	64.0
LexVec Morphs	36.6	47.9	74.6	61.2	74.6	62.8
LexVec n-gram	**37.4**	**52.2**	75.0	61.7	75.3	63.9

Table 4.7 – Results on word analogy tasks, given as percent accuracy when incorporating subword information.

model	GSem	GSyn	MSR
FT	77.0	**71.1**	**59.6**
Skip-gram	78.9	68.2	57.8
LexVec	**80.7**	62.8	49.6
LexVec Morphs	80.7	63.8	53.8
LexVec n-gram	73.8	68.6	55.0

Results for in-vocabulary evaluation are shown in Tables 4.6 and 4.7. Like in FT, the use of subword information in both LV-N and LV-M results in 1) better representation of rare words, as evidenced by the increase in RW correlation, and 2) significant improvement on the GSyn and MSR tasks, in evidence of subwords encoding information about a word's syntactic function (the suffix "ly", for example, suggests an adverb). There seems to be a trade-off between capturing semantics and syntax as in both LV-N and FT there is an accompanying decrease on the GSem tasks in exchange for gains on the GSyn and

Table 4.8 – We generate vectors for OOV using subword information and search for the nearest (cosine distance) words in the embedding space. The LV-M segmentation for each word is: $\{\langle \text{hell, o, o, o} \rangle\}$, $\{\langle \text{marvel, i, cious} \rangle\}$, $\{\langle \text{louis, ana} \rangle\}$, $\{\langle \text{re, re, read} \rangle\}$, $\{\langle \text{ tu, z, read} \rangle\}$. We omit the LV-N and FT n-grams as they are trivial and too numerous to list.

Word	Model	5 Nearest Neighbors
"hellooo"	LV-N	hellogoodbye, hello, helloworld, helloween, helluva
	LV-M	kitsos, finos, neros, nonono, theodoroi
	FT	hello, helloworld, hellogoodbye, helloween, joegazz
"marvelicious"	LV-N	delicious, marveled, marveling, licious, marvellous
	LV-M	marveling, marvelously, marveled, marvelled, loquacious
	FT	delicious, deliciously, marveling, licious, marvelman
"louisana"	LV-N	luisana, pisana, belisana, chiisana, rosana
	LV-M	louisy, louises, louison, louiseville, louisiade
	FT	luisana, louisa, belisana, anabella, rosana
"rereread"	LV-N	reread, rereading, read, writeread, rerecord
	LV-M	alread, carreer, whiteread, unremarked, oread
	FT	reread, rereading, read, reiterate, writeread
"tuzread"	LV-N	tuzi, tuz, tuzla, prizren, momchilgrad, studenica
	LV-M	tuzluca, paczk, goldsztajn, belzberg, yizkor
	FT	pazaryeri, tufanbeyli, yenipazar, leskovac, berovo

MSR tasks. Morphological segmentation in LV-M appears to favor syntax less strongly than do simple n-grams.

In OOV representation (Table 4.8), LV-N and FT work almost identically, as is to be expected. Both find highly coherent neighbors for the words "hellooo", "marvelicious", and "rereread". Interestingly, the misspelling of "louisana" leads to coherent name-like neighbors, although none is the expected correct spelling "louisiana". All models stumble on the made-up prefix "tuz". A possible fix would be to downweight very rare subwords in the vector summation. LV-M is less robust than LV-N and FT on this task as it is highly sensitive to incorrect segmentation, exemplified in the "hellooo" example.

Finally, we see that nearest-neighbors are a mixture of similarly pre/suffixed words. If these pre/suffixes are semantic, the neighbors are semantically related, else if syntactic they have similar syntactic function. This suggests that it should be possible to get *tunable* representations which are more driven by semantics or syntax by a *weighted* summation of subword vectors, given we can identify whether a pre/suffix is semantic or syntactic in nature and weight them accordingly. Such identification might be possible without supervision using corpus statistics as syntactic subwords are likely to be more frequent, and so could be down-weighted for more semantic representations.

In summary, like fastText, subword LexVec learns better representations for rare words than its word-level counterpart. All models generated coherent representations for OOV words, with simple n-grams demonstrating more robustness than unsupervised morphemes.

4.4 Downstream Performance

4.4.1 Hyperparameter Selection

Table 4.9 – Hyperparameter search on the SST5 dev set. Unless otherwise specified, models use PPMI, lr = 2.5e-2, positional contexts and $\alpha_{cds} = 1$.

model	SST5 Dev
LexVec $\alpha_{cds} = .75$	40.9
LexVec	41.0
LexVec -Pos	40.6
LexVec NNEGPMI	40.5
LexVec NPMI	40.3
LexVec CPMI(-2)	<u>41.3</u>
LexVec CPMI(-2) lr = 2e-4	40.0
LexVec CPMI(-2) lr = 1e-3	41.0
LexVec CPMI(-2) lr = 5e-3	**41.6**

Before performing evaluation on sequence tasks, we perform hyperparameter selection using LexVec models trained on the same Wikipedia corpus as in Section 4.1. We use this training corpus because training multiple models on the much larger corpus (Common Crawl) used later in this section would be prohibitive. As the target task, we select the most difficult classification task in the SentEval suite: SST-5.

We first consider whether or not to use context distribution smoothing (CDS) of the PPMI matrix. Results are marginally better without CDS. Throughout the rest of this thesis, we do not use CDS. We then consider whether or not to use positional contexts. There is a performance drop when *not* using positional contexts. Next we evaluate different PMI variants. $CPMI(-2)$ outperforms all other variants. Finally, we tune the learning rate from values in $\{2\text{e-}4, 1\text{e-}3, 5\text{e-}3, 2.5\text{e-}2\}$. Higher learning rates are not reported since they caused SGD to diverge. 5e-3 is the top performer.

4.4.2 Materials

We train two external memory LexVec models, with and without subwords, on the publicly available Common Crawl[2] dump from 2016 published by the WMT group[3]. We perform no preprocessing of any kind other than tokenization using the spaCy toolkit[4]. The resulting corpus has a total of 65 billions tokens. We restrict the vocabulary to the 2 million most frequent words. Aside from the hyperparameters selected in the previous section, we use the same configuration from Section 4.1, except for the number of negative samples which is decreased to 3 to reduce training time. For subword LexVec, we had to reduce the learning rate to 1e-3 to keep SGD from diverging. Character n-grams of size 3-6 were used as subwords. Both models were trained for 5 iterations. We refer to the model without subwords as LexVec-65B and to the model with subwords as SubLexVec-65B.

For comparison, we use publicly available word embeddings trained on similarly large corpora: (1) GoogleNews[5]: Skip-gram embeddings trained on 100B word GoogleNews corpus. (2) GloVe-840B[6]: GloVe embeddings trained on a 840B Common Crawl dump. (3) fastText-630B[7]: fastText Continuous Bag of Words (CBOW) embeddings *using positional embeddings*, trained on a carefully preprocessed Common Crawl dump containing 630B tokens. All vectors are 300-dimensional.

Our models are trained on a corpus are more than 10x smaller than the corpora on which GloVe and fastText were trained. As evidenced in Mikolov et al. (2018), both intrinsic and downstream performance increase with training corpus size, making the comparison between these and our LexVec models trained on much smaller corpora unfair. However, the LexVec results presented here serve as a *lower bound on performance*, and as we will see, already show that LexVec outperforms the GoogleNews and GloVe vectors on most tasks, and is competitive with the fastText vectors on many tasks.

Classifiers for all SentEval tasks are multilayer perceptrons with a single hidden layer of 100 units and dropout of 0.1.

POS tagging is performed using FLAIR (AKBIK et al., 2019) with the same BiLSTM-CRF setup as Huang, Xu and Yu (2015) but using only word embeddings (no hand-engineered features) as input, trained on the WSJ section of the Penn Treebank

---

[2] https://commoncrawl.org/

[3] https://www.statmt.org/wmt18/translation-task.html

[4] https://spacy.io/

[5] https://code.google.com/archive/p/word2vec/

[6] https://nlp.stanford.edu/projects/glove/

[7] https://dl.fbaipublicfiles.com/fasttext/vectors-english/crawl-300d-2M-subword.zip

(MARCUS; MARCINKIEWICZ; SANTORINI, 1993).

4.4.3 Merging

In this section, we compare the 5 merging strategies discussed in Section 3.3.1 on word similarity, word analogy, text classification, NLI, and textual similarity. Results are shown in Tables 4.10 to 4.14.

Table 4.10 – Comparing merging strategies on word similarity tasks. Numbers are Spearman's rank correlation x100.

model	SimLex	RW	WS-S	WS-R	MEN	MTurk	Mean
LexVec-63B (Word)	**45.4**	49.6	80.9	70.1	79.7	70.8	66.1
LexVec-63B (Sum)	41.2	50.0	80.2	71.7	80.0	73.1	66.0
LexVec-63B (Context-only)	40.8	49.6	79.0	68.2	77.9	72.2	64.6
LexVec-63B (Mean)	42.8	**50.5**	**81.5**	**73.7**	**81.4**	**73.3**	**67.2**
LexVec-63B (Concat)	43.1	49.7	79.8	68.4	78.3	72.3	65.3

Table 4.11 – Comparing merging strategies on word analogy tasks. Numbers are percent accuracy.

model	GSem	GSyn	MSR	Mean
LexVec-63B (Word)	74.2	**73.3**	**71.5**	73.0
LexVec-63B (Sum)	77.2	69.2	67.6	71.3
LexVec-63B (Context-only)	69.0	67.8	67.8	68.2
LexVec-63B (Mean)	**80.6**	71.7	69.9	**74.1**
LexVec-63B (Concat)	72.6	72.2	71.1	71.9

The Mean strategy has the highest mean score on word similarity and analogy tasks, followed by Word. Preserving word vectors is important in these tasks: Context-only merging has the lowest score of all strategies.

Table 4.12 – Comparing merging strategies on classification tasks. Numbers are percent accuracy.

model	MR	CR	SUBJ	MPQA	TREC	SST2	SST5	Mean
LexVec-63B (Word)	76.5	78.2	90.8	87.7	85.2	81.4	**44.6**	77.8
LexVec-63B (Sum)	77.0	78.2	91.6	88.0	83.0	**81.7**	44.6	77.7
LexVec-63B (Context-only)	76.9	79.2	91.0	87.8	81.2	81.4	43.8	77.3
LexVec-63B (Mean)	76.2	79.3	91.2	87.4	83.6	81.2	43.4	77.5
LexVec-63B (Concat)	**77.1**	**79.8**	**91.8**	**88.4**	**86.6**	81.3	43.8	**78.4**

In classification and NLI, concatenation has the highest scores in 5 of the 9 tasks, and has the highest mean score. The lowest score comes from using only word vectors, altogether ignoring values in context vectors.

Table 4.13 – Comparing merging strategies on NLI tasks. Numbers are percent accuracy.

model	SNLI	SICK-E	Mean
LexVec-63B (Word)	68.8	79.1	74.0
LexVec-63B (Sum)	72.8	77.8	75.3
LexVec-63B (Context-only)	**73.5**	77.8	75.7
LexVec-63B (Mean)	71.7	78.6	75.1
LexVec-63B (Concat)	73.0	**81.1**	**77.1**

Table 4.14 – Comparing merging strategies on textual similarity tasks. Numbers are Pearson's correlation coefficient.

model	STS12	STS13	STS14	STS15	STS16	Mean
LexVec-63B (Word)	56.2	59.4	59.1	64.0	55.5	58.8
LexVec-63B (Sum)	**58.2**	**62.0**	61.4	64.5	56.5	60.6
LexVec-63B (Context-only)	57.7	61.4	60.3	63.6	55.2	59.6
LexVec-63B (Mean)	58.2	61.9	**61.7**	**65.3**	**57.5**	**60.9**
LexVec-63B (Concat)	56.9	59.9	59.3	63.5	55.3	59.0

Table 4.15 – Comparing merging strategies on textual similarity tasks using regression and cosine (has suffix -Cos). Numbers are Pearson's correlation coefficient.

model	STS-B	STS-B-Cos	SICK-R	SICK-R-Cos
LexVec-63B (Word)	64.0	50.0	79.3	65.8
LexVec-63B (Sum)	66.9	**54.0**	79.8	**69.6**
LexVec-63B (Context-only)	67.0	53.0	79.8	69.0
LexVec-63B (Mean)	65.5	53.8	79.7	68.9
LexVec-63B (Concat)	**67.6**	50.1	**82.5**	66.6

In STS12-16, using only word vectors is again the weakest heuristic, and surprisingly Concat is second worst. In these tasks, textual similarity is estimated using the cosine between sentence embeddings. If we now look at STS-B and SICK-R scores in Table 4.15, where a regressor is trained when given as input sentence embeddings, Concat is the top scorer. In this same table we also include scores for STS-B and SICK-R using cosine between sentence embeddings, and the results follow those from STS12-16. The trend is that the higher dimensional Concat heuristic – 1500d vs. 300d for the other strategies – should be used when training a classifier/regressor, and the other strategies should be used otherwise.

Based on these results, we include only Mean and Concat in the evaluations that follow in this chapter.

4.4.4 Text Classification

Table 4.16 – Percent accuracy on text classification tasks.

model	MR	CR	SUBJ	MPQA	TREC	SST2	SST5	Mean
LexVec-63B (Mean)	76.2	79.3	91.2	87.4	83.6	81.2	43.4	77.5
LexVec-63B (Concat)	77.1	79.8	91.8	88.4	86.6	81.3	43.8	78.4
SubLexVec-63B (Mean)	76.6	79.5	91.2	88.0	84.0	80.5	43.4	77.6
SubLexVec-63B (Concat)	76.7	79.1	91.9	**88.5**	**89.4**	82.0	44.2	78.8
GoogleNews-100B	77.1	79.1	90.8	88.0	83.4	80.3	43.5	77.5
GloVe-840B	77.6	79.1	91.5	87.7	83.4	81.7	44.9	78.0
fastText-630B	**78.2**	**80.1**	**92.1**	87.7	85.0	**83.5**	**46.0**	**78.9**

Despite being trained on much smaller corpora, results in Table 4.16 show that both LexVec (Concat) models have higher mean scores than the GoogleNews and GloVe vectors. The subword LexVec model outperforms fastText on MPQA and TREC tasks, and has only marginally weaker mean performance. Given a greater computational budget, it would be interesting to train LexVec on larger corpora to see if this gap disappears.

4.4.5 Text Similarity

On STS12-16 results in Table 4.17, the GoogleNews and fastText vectors have a large advantage over LexVec and GloVe models. However, LexVec (Concat) strategy overtakes GoogleNews on the STS-B and SICK-R tasks and outperforms fastText on SICK-R *when regressors are trained* (results in Table 4.18). We conclude that while

Table 4.17 – Performance on textual similarity tasks using cosine similarity. Numbers are Pearson's correlation coefficient x100.

model	STS12	STS13	STS14	STS15	STS16	Mean
LexVec-63B (Mean)	58.2	61.9	61.7	65.3	57.5	60.9
LexVec-63B (Concat)	56.9	59.9	59.3	63.5	55.3	59.0
SubLexVec-63B (Mean)	56.1	59.4	57.9	63.0	53.3	58.0
SubLexVec-63B (Concat)	53.6	56.4	54.1	59.7	49.1	54.6
GoogleNews-100B	54.7	63.3	**67.2**	**69.9**	**64.4**	63.9
GloVe-840B	53.1	54.2	55.3	59.9	51.5	54.8
fastText-630B	**59.1**	**64.0**	66.1	69.8	64.1	**64.6**

Table 4.18 – Comparing merging strategies on textual similarity tasks using regression and cosine (has suffix -Cos). Numbers are Pearson's correlation coefficient x100.

model	STS-B	STS-B-Cos	SICK-R	SICK-R-Cos	Mean
LexVec-63B (Mean)	65.5	53.8	79.7	68.9	67.0
LexVec-63B (Concat)	67.6	50.1	**82.5**	66.6	66.7
SubLexVec-63B (Mean)	64.9	49.8	79.8	67.0	65.4
SubLexVec-63B (Concat)	67.3	45.1	82.1	65.0	64.9
GoogleNews-100B	64.3	**62.1**	80.1	**70.2**	69.2
GloVe-840B	64.7	41.5	79.9	65.1	62.8
fastText-630B	**70.2**	58.1	82.0	69.8	**70.0**

the information necessary for this task is present to a higher degree in LexVec (Concat) when compared to GoogleNews vectors, this information is better recovered with a trained neural network, whereas recovery using sentence embedding cosine similarity is less effective.

4.4.6 Syntactic Tasks

Table 4.19 – Results on POS tagging and syntactic probing tasks. All numbers represent percent accuracy.

model	POS	Dep	TopC	Mean
LexVec-63B (Mean)	96.8	33.3	62.6	64.2
LexVec-63B (Concat)	96.9	**36.1**	**73.6**	**68.9**
SubLexVec-63B (Mean)	96.8	34.2	63.4	64.8
SubLexVec-63B (Concat)	96.8	36.0	72.3	68.3
GoogleNews-100B	92.1	27.0	61.2	60.1
GloVe-840B	**96.9**	35.6	65.3	65.9
fastText-630B	96.8	32.9	65.4	65.0

Results for syntactic sequence tasks are presented in Table 4.19. Both Concat

LexVec models have the highest overall performance. In all these tasks, taggers (POS) and classifiers (Dep, TopC) are trained. This reiterates the importance of the Concat strategy, which preserves distinct representations in the input, rather than conflating all representations in a single vector as with the Mean strategy. This holds for semantic tasks (Table 4.18), and holds here for syntactic tasks as well.

4.4.7 Natural Language Inference

Table 4.20 – Results on natural language inference tasks. Numbers represent percent accuracy.

model	SNLI	SICK-E	Mean
LexVec-63B (Mean)	71.7	78.6	75.1
LexVec-63B (Concat)	<u>73.0</u>	<u>81.1</u>	<u>77.1</u>
SubLexVec-63B (Mean)	72.0	78.4	75.2
SubLexVec-63B (Concat)	**73.0**	**81.3**	**77.2**
GoogleNews-100B	67.7	79.2	73.4
GloVe-840B	71.8	79.0	75.4
fastText-630B	71.2	79.1	75.2

NLI results are shown in Table 4.20. LexVec (Concat) models dominate these two tasks by a large margin. Given that fastText achieves higher STS scores than LexVec (Concat), we attribute LexVec's leading NLI performance to its leading results in syntactic tasks. This hypothesis is supported by improvements in SNLI results when incorporating explicit syntactic information into more sophisticated compositional models (CHEN et al., 2017).

4.4.8 Facet Ranking

In addition to the LexVec models used in previous sections which were not available at the time the CoSearcher paper was published, we include here the SubLexVec-Crawl[8] model which was available and used in that work. SubLexVec-Crawl uses a PPMI matrix with context distribution smoothing of $.75$, different preprocessing of the same CommonCrawl corpus (naive punctuation removal and whitespace tokenization). Results are shown in Table 4.21.

---

[8] <https://www.dropbox.com/s/mrxn933chn5u37z/lexvec.commoncrawl.ngramsubwords.300d.W.pos.vectors.gz>

Table 4.21 – Facet ranking evaluation using precision-at-1 (P@1) and mean reciprocal rank (MRR) using unsupervised mean bag-of-vectors.

model	P@1			MRR		
	Train	Dev	Test	Train	Dev	Test
LexVec-63B (Mean)	79.7	80.4	76.5	88.1	88.2	86.0
LexVec-63B (Concat)	77.4	78.8	75.1	86.8	87.3	85.2
SubLexVec-63B (Mean)	77.0	79.0	73.8	86.5	87.3	84.6
SubLexVec-63B (Concat)	74.5	76.6	71.7	85.0	85.9	83.2
fastText-630B	81.4	83.3	80.6	89.2	90.0	88.5
GoogleNews-100B	81.3	80.0	78.7	89.2	88.2	87.4
GloVe-840B	69.8	75.0	68.2	81.7	84.9	80.7
SubLexVec-Crawl	83.3	83.3	80.7	90.3	89.9	88.6
SBERT-large	**98.1**	85.5	85.5	**99.0**	91.1	91.2
BERT-large CE	95.8	**89.7**	**91.7**	97.6	**93.7**	**95.0**

Table 4.22 – Facet ranking evaluation using precision-at-1 (P@1) and mean reciprocal rank (MRR) using trained deep averaging network (DAN).

model	P@1			MRR		
	Train	Dev	Test	Train	Dev	Test
LexVec-63B (Mean)	91.2	84.9	86.2	95.0	91.2	91.8
LexVec-63B (Concat)	96.7	86.4	88.7	98.3	92.1	93.2
SubLexVec-63B (Mean)	90.9	85.0	86.2	94.8	91.3	91.7
SubLexVec-63B (Concat)	95.7	86.6	87.7	97.7	92.2	92.8
fastText-630B	90.8	85.9	86.8	94.8	91.8	92.1
GoogleNews-100B	91.2	86.7	85.6	95.0	92.0	91.4
GloVe-840B	89.7	83.5	84.3	94.2	90.4	90.7
SubLexVec-Crawl	89.8	84.5	85.1	94.2	91.0	91.1
SBERT-large	**98.1**	85.5	85.5	**99.0**	91.1	91.2
BERT-large CE	95.8	**89.7**	**91.7**	97.6	**93.7**	**95.0**

The SubLexVec-Crawl embeddings outperform all other unsupervised embeddings. There is a small gap in performance between its P@1 and that of the SBERT model, and yet smaller gap in MRR. As an upper bound, we also evaluate a supervised cross encoder (CE) BERT model which does not generate sentence embeddings, but is instead trained to calculate $score(.)$ (Equation (2.15)) directly from an input context-facet pair. This model has a larger performance gap compared to the BoV models.

If the computation cost of $score(.)$ for a single context-facet pair is r, the number of turns in a dialogue is p, and the set of candidate facets is C, facet ranking over a whole dialogue incurs cost $O(rp|C|)$. This makes the CE model prohibitive since it requires running the entire BERT model (high r) for every context-facet pair and performance is a crucial aspect of simulation. In contrast, sentence embedding models perform only $O(r(p + |C|))$ computations to generate sentence embeddings for contexts and candidate facets, making $O(rp|C|)$ at ranking time cheap since r is but a cosine between vectors.

In summary, SubLexVec-Crawl vectors were used as $rep(.)$ in Salle et al. (2021) because (1) the performance gap to the SBERT model is small, in particular for MRR (2) we hypothesize that being unsupervised will help it generalize to other kinds of facets (such as search engine keyword suggestions as used in Salle et al. (2021)) which look less like natural language than do Qulac facets used in our training set.

The gap between the static word embedding models and the Transformer models decreases further if we allow some supervision. Rather than computing the cosine between the BoV, we train a Deep Averaging Network (DAN) (IYYER et al., 2015) to compute sentence embeddings which are then used in the cosine computation. A DAN is simply a sequence of two dimension-preserving $\tanh$ feed-forward layers which transform the input BoV sentence embedding into an output sentence embedding. Results are shown in Table 4.22. The same phenomenon occurs as in Table 4.15: learning a non-linear transformation of the larger dimensional Concat vectors significantly improves performance. The improvement is such that both LexVec-63B (Concat) and SubLexVec-63B(Concat) outperform the SBERT trained using the same data. The BERT cross-encoder is still the strongest performing model, but in scenarios where cross encoding is prohibitively expensive and sentence embedding is required, Concat models under a DAN transformation are a competitive and cheap approach.

4.5 Comparison to Transformers

Given the recent success of Transformer models in NLP, in this section we investigate whether static word embeddings (including LexVec) remain competitive in word-level and sentence-level tasks.

4.5.1 Lexical Semantics

Table 4.23 – Comparing lexical semantic performance between static word embeddings and contextualized word embeddings on word similarity tasks. Numbers are Spearman rank correlation x100.

model	SimLex	RW	WS-S	WS-R	MEN	MTurk	Mean
LexVec-63B (Mean)	42.8	50.5	81.5	73.7	81.4	**73.3**	67.2
LexVec-63B (Concat)	43.1	49.7	79.8	68.4	78.3	72.3	65.3
SubLexVec-63B (Mean)	43.3	52.2	81.2	70.8	81.1	71.6	66.7
SubLexVec-63B (Concat)	40.6	48.6	75.9	60.6	76.2	69.2	61.8
GoogleNews-100B	44.2	53.4	77.2	63.5	78.2	68.4	64.2
GloVe-840B	40.8	46.2	80.3	68.8	80.5	69.3	64.3
fastText-630B	50.3	**59.5**	**84.0**	**73.7**	**84.6**	72.6	**70.8**
BERT.F4	49.0	37.0	67.0	56.0	70.0	59.0	56.3
BERT.L4	**55.0**	48.0	70.0	51.0	69.0	56.0	58.2
BERT.L	50.0	36.0	63.0	47.0	64.0	52.0	52.0

Table 4.24 – Comparing lexical semantic performance between static word embeddings and contextualized word embeddings on word analogy tasks. Numbers are percent accuracy. Missing scores are due to Lenci et al. (2021) only reporting aggregate (GSem+GSyn) Google analogy scores.

model	GSem+GSyn	GSem	GSyn	MSR	Mean
LexVec-63B (Mean)	75.8	80.6	71.7	69.9	74.5
LexVec-63B (Concat)	72.4	72.6	72.2	71.1	72.0
SubLexVec-63B (Mean)	75.1	75.7	74.6	74.0	74.9
SubLexVec-63B (Concat)	73.3	71.9	74.5	73.2	73.2
GoogleNews-100B	72.8	72.3	73.2	72.5	72.7
GloVe-840B	76.0	78.8	73.7	73.5	75.5
fastText-630B	**82.1**	**83.1**	**81.2**	**82.6**	**82.3**
BERT.F4	38.0	–	–	76.0	57.0
BERT.L4	66.0	55.0	71.0	69.0	65.2
BERT.L	64.0	–	–	68.0	66.0

For lexical semantic tasks, we include BERT results from Lenci et al. (2021). To compute a non-contextualized representation for BERT, they average the contextualized embeddings of a target word across ten or fewer sentences in which the word appears within a Wikipedia corpus. They report results for 3 variants:

- **BERT.F4:** the sum of the embeddings from the first four layers;
- **BERT.L4:** the sum of the embeddings from the last four layers;
- **BERT.L:** the sum of the embeddings from the last layer;

Word similarity results are shown in Table 4.23. Except for the SimLex dataset, BERT embeddings underperform static word embeddings in all other datasets by wide margin. In word analogy evaluations reported in Table 4.24, BERT embeddings have the lowest average performance.

Lenci et al. (2021) only report distinct semantic/syntactic Google analogy scores for the BERT.L4 embeddings which has higher aggregate scores than both BERT.F4 and BERT.L. Even though it is the best of the BERT embeddings on this task, its semantic analogy score is far below those of the static word embeddings. Although the positive SimLex and syntactic analogy scores are encouraging, lexical semantics are still dominated by static word embeddings.

Finally, we note that LexVec models are trained on a far smaller training corpus than the other static word embeddings. As stated in Section 4.4.2, we believe these results serve as a lower-bound for its performance in lexical semantic tasks.

4.5.2 Sentence Embedding

Table 4.25 – Comparing text classification performance between unsupervised sentence embeddings using static word embeddings and state-of-the-art sentence embeddings based on Transformer models. Numbers are percent accuracy.

model	MR	CR	SUBJ	MPQA	SST2	TREC	MRPC	Mean
LexVec-63B (Mean)	76.2	79.3	91.2	87.4	81.2	83.6	72.3	81.6
LexVec-63B (Concat)	77.1	79.8	91.8	88.4	81.3	86.6	73.5	82.7
SubLexVec-63B (Mean)	76.6	79.5	91.2	88.0	80.5	84.0	74.3	82.0
SubLexVec-63B (Concat)	76.7	79.1	91.9	88.5	82.0	89.4	73.3	83.0
fastText-630B	78.2	80.1	92.1	87.7	83.5	85.0	74.5	83.0
GEM (L.F.P.)	79.8	82.5	93.8	89.9	84.7	**91.4**	75.4	85.4
S3E (L.F.P.)	79.4	81.4	92.9	89.4	83.5	89.0	75.6	84.5
SBERT-large	84.8	90.5	**94.7**	**90.6**	91.0	88.2	76.9	88.1
SRoBERTa-WK-base	**85.8**	**91.4**	94.5	89.7	**92.3**	91.0	**78.9**	**89.1**

We compare recent state-of-the-art Transformer sentence encoders – Sentence-BERT (SBERT, Reimers and Gurevych (2019)) and SRoBERTa-WK-base (WANG; KUO,

Table 4.26 – Comparing textual similarity performance between unsupervised sentence embeddings using static word embeddings and state-of-the-art sentence embeddings based on Transformer models. Numbers are Pearson correlation coefficient x100.

model	SICK-R	STS-B	Mean
LexVec-63B (Mean)	79.7	65.5	72.6
LexVec-63B (Concat)	82.5	67.6	75.0
SubLexVec-63B (Mean)	79.8	64.9	72.4
SubLexVec-63B (Concat)	82.1	67.3	74.7
fastText-630B	82.0	70.2	76.1
GEM (L.F.P.)	<u>86.5</u>	78.4	<u>82.5</u>
S3E (L.F.P.)	84.7	<u>78.6</u>	81.7
SBERT-large	84.7	75.6	80.2
SRoBERTa-WK-base	**87.6**	**80.7**	**84.2**

2020) – to state-of-the-art sentence embeddings based on static word embeddings – GEM (WANG et al., 2019) and S3E (WANG et al., 2020a) – on the same text classification, NLI, and textual similarity tasks used previously. Both GEM and S3E use a concatenation of LexVec, fastText, and PPDB (WIETING et al., 2015) (L.F.P.) as input. As a baseline, we also include the mean bag of vector sentence embeddings used previously. GEM results are from Wang et al. (2019), S3E results from Wang et al. (2020a), and SBERT-large and SRoBERTa-WK-base results from Wang and Kuo (2020). Text classification results are shown in Table 4.25. As expected, Transformer models dominate most tasks. However, the gap in average performance is small when compared to GEM, and except for the MR and CR tasks, the difference is marginal.

This same trend repeats in textual similarity results in Table 4.26, but here the gap is even smaller. Sentence embedding is more important in these tasks than in text classification: a common use case for sentence embeddings is dense retrieval where given an input item (query, passage, etc.) a set of output items (documents, passages, etc.) must be ranked using some similarity function. Computing an expensive similarity function (such as a full run of BERT) for every candidate input-output item can be prohibitive when the number of possible input-output combinations is large. A cheap similarity function such as the cosine function alleviates this problem. This is the reason the BERT Cross Encoder (an expensive similarity function) was not used for facet ranking in Section 4.4.8. Additionally, embedding inputs and outputs in a common vector space allows for efficient approximate nearest-neighbors methods (GIONIS et al., 1999; BÖHM; BERCHTOLD; KEIM, 2001; DONG; MOSES; LI, 2011) such that not all output items have to be considered when searching for most similar results.

4.6 Conclusion

We performed extensive evaluation of our proposed LexVec model. Results show that LexVec closes the gap between counting and predictive methods. LexVec is competitive with the state-of-the-art fastText model despite being trained on 13x less data. Additionally, we observe that static word embeddings remain competitive with Transformer models in many tasks, notably outperforming the latter in lexical semantic tasks.

5 SEMANTICS/SYNTAX IN NEGATIVE/POSITIVE INFORMATION

In this chapter, we investigate what kind of information is captured in word vectors if we exclusively consider nPMI *or* pPMI values when performing matrix factorization (in answer to Q11 and Q12). In other words, if all we know is that "word w tends *not* to occur with words $c_i, \ldots, c_j$" (nPMI, remove "not" for similar statement about pPMI), what can we learn about w? This can help us train better models, but perhaps more importantly, improve our understanding of natural language.

5.1 Materials

Models: In order to identify the role that nPMI and pPMI play in distributional semantics, we train two LexVec models:

- one that only considers negative information, nPMI, i.e., any pair in pPMI is skipped during factorization, or equivalently, $\lambda_{LexVec} = 0$ if $(w, c) \in pPMI$, and
- one model that only considers positive information, pPMI, i.e., any pair in nPMI is skipped during factorization, or equivalently $\lambda_{LexVec} = 0$ if $(w, c) \in nPMI$.

We compare these to models that include both negative and positive information ($nPMI \cup pPMI$) to see how the two interact. To account for values in mnPMI, we use the four PMI variants described in Section 2.2: PPMI, CPMI(-2), NPMI, NNEGPMI.

We use the following LexVec configuration for all PMI variants: window size $l = 5$, embedding dimension of 300, 5 negative samples, no positional contexts, learning rate of 0.025, no subword information, no context distribution smoothing, and negative distribution power $\alpha_{pow} = 0.75$.

For all experiments, we use the English Wikipedia corpus described in Section 2.2, resulting in the same underlying M_{wiki5} matrix for all models.

For comparison, we include results for a randomly initialized, untrained embedding to establish task baselines.

Semantic tasks: To evaluate word-level semantics, we use the SimLex and Rare Word (RW) word similarity datasets. To evaluate word analogies, we use the Google Semantic (GSem) analogies. We evaluate sentence-level semantics on STS Benchmark.

Syntactic tasks: Similarly, we use the Google Syntactic analogies (GSyn) and MRS syntactic analogies to evaluate word-level syntactic information. Google Syntactic analogies are in fact morphological but many categories test for POS relations and are therefore syntactic in nature. We employ the Depth (Dep) and Top Constituents (TopC) (of the input sentence's constituent parse tree) probing tasks from SentEval to evaluate sentence-level syntax. Our final syntactic task is part-of-speech (POS) tagging.

5.1.1 Results

Table 5.1 – Performance on tasks focused on semantics or syntax by models that use only positive information (p*), negative information (n*), or both (no prefix), and the random baseline. Using negative information alone performs far better than the random baseline, especially on the syntactic tasks. Metrics: Spearman rank correlation ($\times 100$) for SimLex and RW word similarity; Pearson correlation for STS-B; % accuracy for GSem/GSyn/MSR word analogies, POS tagging and WC, Dep, TopC probing tasks. Best result for each column in bold, second best underlined.

model	SimLex	RW	GSem	STSB	GSyn	MSR	POS	Dep	TopC
pPPMI	**37.0**	40.1	58.8	**65.4**	52.7	35.1	92.0	27.1	30.4
nPPMI	4.0	1.8	0.0	48.6	0.0	0.0	16.3	17.9	5.0
nCPMI(-2)	22.6	25.2	18.3	41.4	24.5	18.3	90.6	**32.9**	**33.7**
nNPMI	9.9	21.8	8.2	38.3	9.6	5.8	89.0	31.1	32.3
PPMI	34.0	**45.3**	76.5	61.6	55.1	36.7	91.7	25.5	26.6
CPMI(-2)	34.0	41.8	**78.4**	61.9	**58.7**	**42.6**	**92.2**	27.3	28.4
NPMI	26.0	39.4	60.0	60.6	44.4	30.2	91.4	26.3	27.9
NNEGPMI	34.0	43.0	78.3	61.7	56.3	39.8	92.0	25.1	26.3
Random	4.0	1.9	0.0	45.3	0.0	0.0	59.1	17.9	5.0

The results shown in Table 5.1 provide insights into the role of negative and positive PMI for capturing semantic and syntactic information.

Negative PMI: The $nPPMI$ model is almost identical to the random baseline; this is to be expected as random initialization gives in expectation perpendicular vectors or equivalently dot products equal to zero, and if the only learning signal is to make dot products equal to zero nothing changes. The other nPMI models ($nCPMI(-2)$ and $nPMI$) exhibit more interesting behavior. They perform similarly to all others in POS tagging and both syntactic probing tasks (Dep and TopC), but very poorly on all semantic tasks, suggesting that nPMI mostly encodes syntactic information. Our hypothesis to explain this phenomenon is that the grammar that generates language implicitly creates negative cooccurrence and so nPMI encodes this syntactic information. Interestingly, this idea creates a bridge between distributional semantics and the argument by Regier and Gahl (2004), Foraker et al. (2009) that indirect negative evidence might play an important role in human language acquisition of grammar.

Positive PMI: The $pPPMI$ model which accounts for only values in pPMI performs similarly to the full spectrum models on most tasks, clearly indicating that pPMI encodes both semantic and syntactic information.

Why incorporate nPMI? $pPPMI$ falters on the RW and analogy tasks, and accounting for nPMI significantly improves performance on both tasks. Section 6.3 explores how increasing the relative importance of negative information increases rank invariance in the word vectors and improves results on both these tasks relative to using only pPMI.

Full spectrum models: Of the models which account for all PMI values ($nPMI \cup pPMI$), the PPMI, $CPMI_{-2}$, and $NNEGPMI$ models perform similarly, whereas the $NPMI$ model is significantly worse on nearly all tasks. We thus conclude that accounting for scale in the positive spectrum is more important than in the negative spectrum.

Collapsing the negative spectrum: The PPMI model, which collapses the negative spectrum to zero, performs almost identically to the $CPMI_{-2}$ and $NNEGPMI$ models that account for the range of negative values. We believe this is because the set $nPMI \setminus mnPMI$, which contains the collapsed values, is much smaller than $nPMI$ (in PMI_{wiki5}, 483 times smaller) because of the sparsity of the underlying cooccurrence matrix, thus almost all negative information is in mnPMI: $nPMI \approx mnNPMI$.

5.2 Conclusion

In this chapter, we isolated the effects of negative and positive information on word embeddings. Results show that only accounting for positive information strongly captures both semantics and syntax, whereas using only negative information captures little of semantics but a surprising amount of syntactic information.

6 NEGATIVE INFORMATION AND GEOMETRY

One hypothesis related to negative information is that it helps to “scatter” unrelated words in the vector space (SALLE; VILLAVICENCIO; IDIART, 2016). As we will show, this scattering does indeed happen between *words and contexts*, but surprisingly the opposite happens between words since they are pulled together in space. In this chapter, we first study whether increasing the number of negative samples in the LexVec and Skip-gram models does indeed increase the relative importance of negative information (Section 6.1). Confirming this to be true, we investigate the effect of increasing negative information on the geometry of LexVec (Section 6.3) and of other word embedding models (Section 6.4), in answer to Q13.

6.1 Increasing relative weight of negative information

In our study, we need a way to gradually increase the loss weights in the negative spectrum of PMI, λ_{LexVec} for $(w, c) \in nPMI$, relative to pairs in the positive spectrum, $(w, c) \in pPMI$, to identify how increasing the relative importance of negative information affects the geometry of word vector spaces. Window sampling weights λ_{ws} are fixed given the data, so the only control we have over λ_{LexVec} is k, the number of negative samples. We need to show that increasing the number of negative samples in the LexVec factorization increases the relative weight of negative information.

To show this, we sample 10^5 values from $CPMI(-2)_{wiki5}$ using window sampling and negative sampling and plot the distribution of these values in Figure 6.1. This plot shows the reconstruction error weights λ_* different sampling regimes assign to different PMI values. For example, the peak at -2 tells us that negative sampling weights λ_{ns} will be higher for (w, c) pairs in nPMI than for (w, c) pairs in pPMI. The opposite is observed for window sampling weights, which assign more weight to values in pPMI. Table 6.1 shows this same result by aggregating values in Figure 6.1.

These results confirm that, as the number of *negative samples* is increased, so is the relative weight of *negative information* in the factorization. This enables us to use the LexVec model to investigate the impact of increasing negative information on resulting word vector geometry.

Figure 6.1 – The distribution (bin width equal to 0.2) of sampled $CPMI_{wiki5}(-2)$ values when using window sampling and negative sampling. Histogram from Figure 3.2 included for comparison. These window sampling and negative sampling distributions of PMI values correspond to the reconstruction error weights $\lambda_{ws}, \lambda_{ns}$ in Equation (3.8) as a function of PMI. Negative sampling assigns high weights to values in nPMI, and window sampling to values in pPMI.

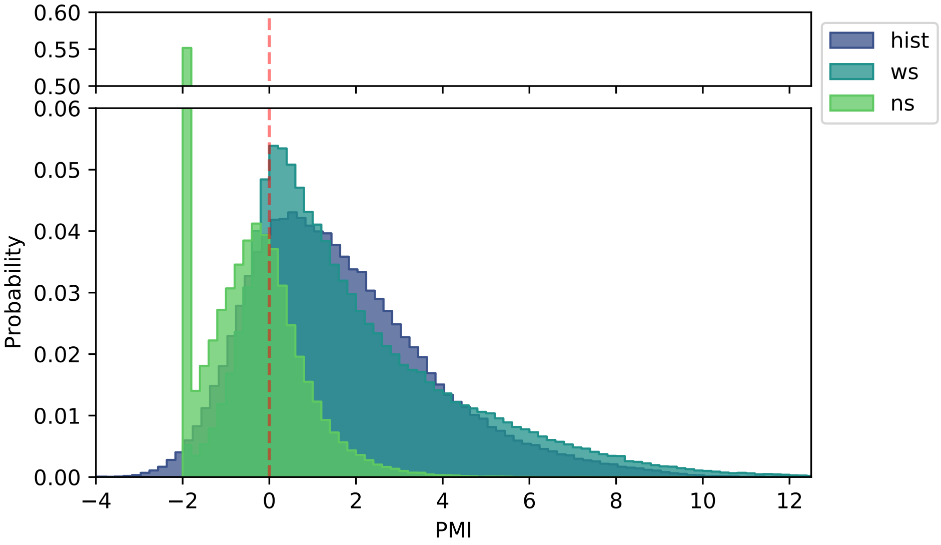


Table 6.1 – Set (rows) membership of samples for various sampling methods (columns). Cell values are the percentage of samples for a given method that fall within a set, such that nPMI+pPMI sum to 100. **Full**: computed over all cells in PMI_{wiki5}. **Hist**: computed over the same 10^5 sampled pairs plotted in Figure 3.2. **WS** and **NS**: computed over the same 10^5 sampled pairs used in Figure 6.1 using window sampling (ws) and negative sampling (ns). Observe that window sampling is heavily skewed towards pairs in pPMI, and negative sampling heavily skewed towards pairs in nPMI.

Name	Set	Full	Hist	WS	NS
nPMI: Negative information	$\{(w,c) \mid PMI_{w,c} \leq 0\}$	99.27	22.24	19.35	81.75
pPMI: Positive information	$\{(w,c) \mid PMI_{w,c} > 0\}$	0.73	77.76	80.65	18.25
mnPMI: Maximally-negative information	$\{(w,c) \mid M_{w,c} = 0\}$	99.07	0.00	0.00	51.33
nPMI\mnPMI: Collapsed negative information under PPMI	$\{(w,c) \mid PMI_{w,c} \leq 0 \wedge M_{w,c} > 0\}$	0.21	22.24	19.35	30.42

6.2 Materials

Here we describe the LexVec models and tasks used in our investigations into vector space geometry.

Increasing negative samples: As shown in Section 6.1, when using both window sampling and negative sampling, increasing the number of negative samples effectively increases the relative importance of negative information. We use the default LexVec setting in which both nPMI and pPMI are used (no steps are skipped), and increase the number of negative samples from 0 to 1, 2, 4, 5, 10, 15, and 20. We focus on $CPMI(-2)$ since as described in Section 2.2 it closely mimics the measure of ultimate interest which is PMI.

Geometry × frequency: To understand how increased negative sampling affects the geometry of words of different frequencies, we evaluate performance using SimLex (which consists of frequent words) and RW (which consists of frequent-rare word) word similarity datasets. To perform the same frequency analysis on analogies, we order the analogies in the analogy datasets by the highest rank of any of the words in each analogy. We take the first 10% and last 10% analogies to create frequent (GSemF, GSynF, MSRF) and rare (GSemR, GSynR, MSRR) word analogy datasets, respectively, with * and ** variants when heuristics are excluded as described above. Table B.1 in the Appendix gives percentile rank statistics for all datasets we use in this chapter.

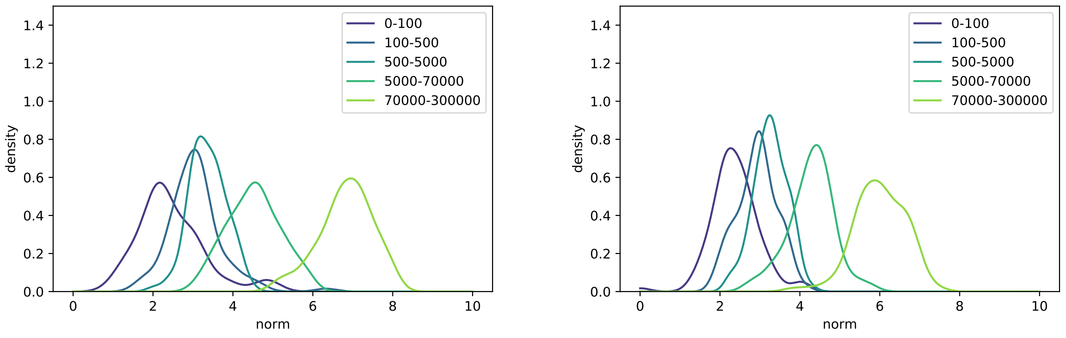
6.3 Results

Norms: In Figure 6.2, we plot the distribution of vector L_2 norms for 100 words sampled from different frequency buckets for LexVec models using 0, 5, and 20 negative samples (LV-k denote the model with k negative samples). We use the same buckets as Mimno and Thompson (2017), indexing words by inverse frequency (most frequent first), 0-100, 100-500, 500-5000, 5000-70000, and defining an additional bucket 70000-300000 for extremely rare words. With an increasing number of negative samples, the relative weight of negative information is increased, and *vectors norms become rank invariant*; the means of the different buckets becomes increasingly closer and variance decreases as negative samples are increased.

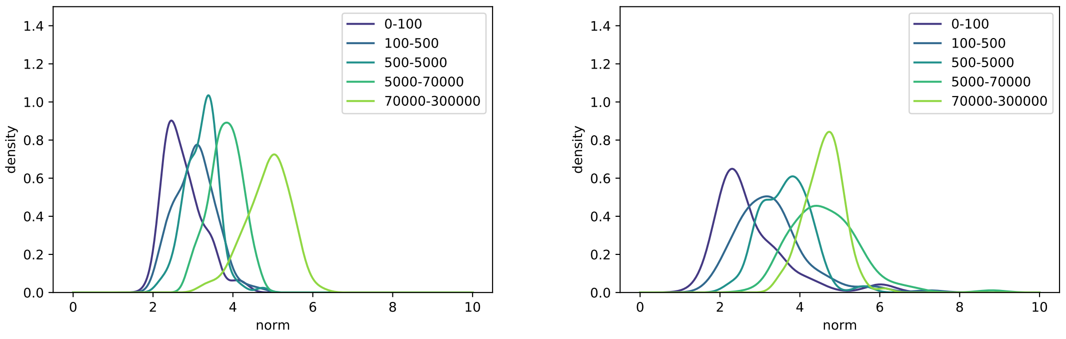
Direction: In Figure 6.3, we perform the experiment of Mimno and Thompson (2017), where using the same sampled words and frequency buckets as in Figure 6.2, we

Figure 6.2 – The distribution of vector L_2 norms for 100 words sampled from various frequency buckets for LexVec models using 0, 5, and 20 negative samples (LV-0, LV-5, LV-20). As the number of negative samples increases, the norms become more rank invariant, with means of the different buckets becoming increasingly closer and variance decreasing. The norm distributions for Skip-gram, GloVe, and SVD are shown for comparison.

(a) LV-0 (b) LV-5



(c) LV-20 (d) Skip-gram



(e) GloVe (f) SVD

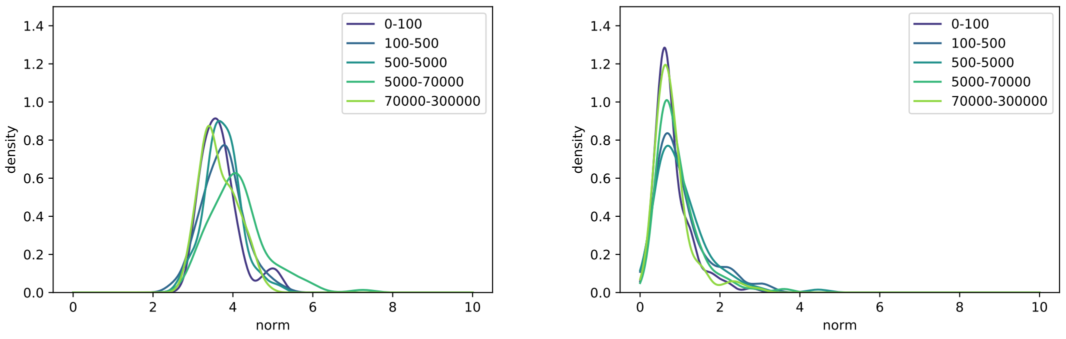
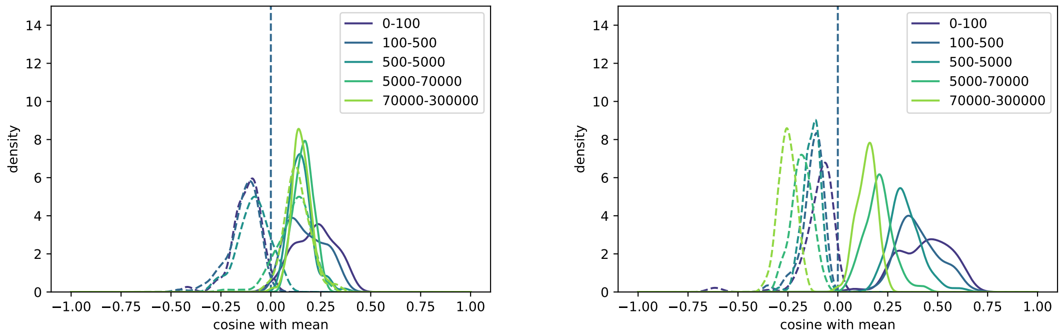
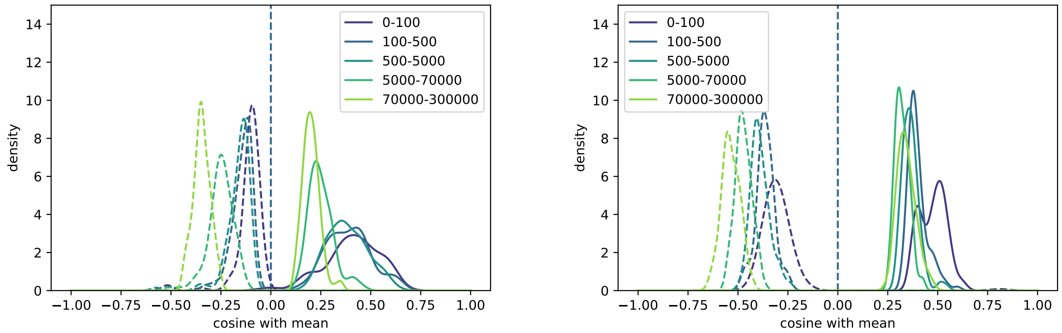


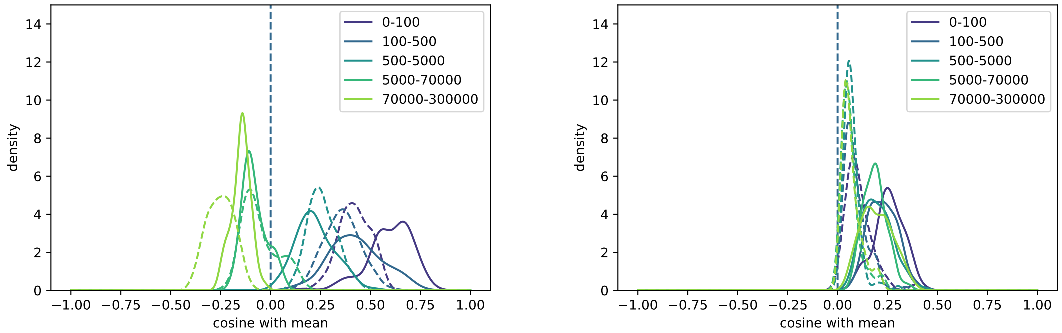
Figure 6.3 – In solid lines, the distributions of cosines (Equation (2.11)) of word vectors from different frequency buckets with the mean vector of word vectors from all buckets, and in dashed lines the distributions of cosines of *context* vectors from different frequency buckets with this *same mean word vector*. As the number of negative samples increases, word vectors increasingly point in the same direction and word-context vectors point in the opposite direction.



(a) LV-0 (b) LV-5



(c) LV-20 (d) Skip-gram



(e) GloVe (f) SVD

calculate the mean vector of all sampled words from all buckets, and plot the distribution of the cosine (Equation (2.11)) of sampled word vectors (solid line) and corresponding context vectors (dashed lines) with this mean vector. Here we observe that as the number of negative samples increases, word vectors increasingly point in the same direction, and context vectors point in opposite directions. As a preface to Section 6.4.2, these are the same effects observed by Mimno and Thompson (2017) for the Skip-gram model.

Nearest neighbors: We perform qualitative analysis of nearest neighbors of words sampled from SimLex in Table 6.2 and RW in Table 6.3, where neighbors are ordered by descending cosine similarity (Equation (2.11)) – similarity decreases from left to right – and subscripts denote the percentile rank of a word (0.0 is most frequent word, 100.0 is rarest).

In both the LV-0 and LV-20 model, frequent words have semantically related, frequent word neighbors, showing that increasing negative information has no effect on semantic similarity of frequent words. Results change completely with rare words, where LV-0 neighbors are rare words and – barring a few exceptions – have no obvious semantic connection to the target word, e.g., “monocultures”’s (an agricultural practice) nearest neighbor is “fieldensis” (a species of arthropod). With the LV-20 model, on the other hand, the neighbors of target rare words are generally of higher frequency than the targets, and have clear semantic relations, as with “monocultures” and “monoculture” (singular) or “seedlings” (young plants). Qualitatively, and as we will see in the next section, quantitatively, increasing the importance of negative information has a very positive impact on the representation of rare words.

Word similarity tasks: Shown in Figure 6.4a, results on RW improve consistently as the number of negative samples is increased. There is a small but consistent drop in performance on SimLex when increasing the number of negative samples. Looking at the SimLex nearest neighbor examples in Table 6.2, the semantic similarities are indistinguishable between the LV-0 and LV-20 models, even though the LV-0 model has a marginally higher SimLex score. We tested the WS353 word similarity dataset (FINKELSTEIN et al., 2001), split into relatedness (WSRel) and similarity (WSSim), to see if this drop on SimLex scores, which measures similarity, is due to an increase in relatedness; that is, if related pairs such as (psychology,Freud) are being drawn closer together in space to the detriment of semantically similar pairs such as (psychology,psychiatry). We observe in Figure 6.4a that relatedness scores in WSRel do indeed increase, and that there is a small drop in similarity scores in WSSim. This suggests that as relatedness increases

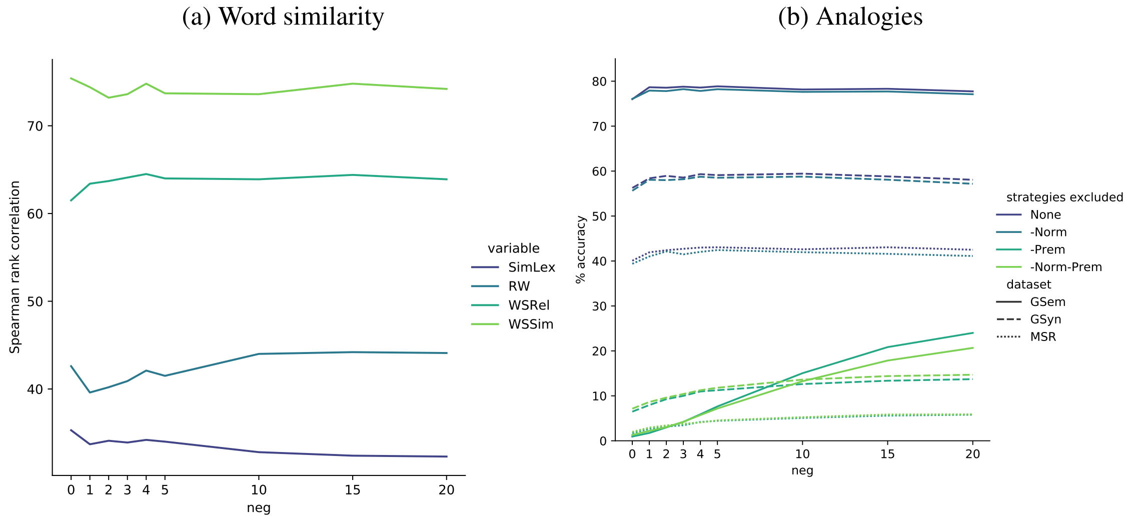
Table 6.2 – Nearest neighbors of words sampled from SimLex. Subscripts denote the percentile rank of a word (0.0 is most frequent word, 100.0 is rarest). All models exhibit semantically coherent neighbors, with the exception of the SG model which has some unrelated *intruders*, notably for the word “interest”.

word	model	neighbors
interest $_{0.3}$	LV-0	interests $_{1.0}$, conflict $_{0.5}$, interested $_{0.7}$, scientific $_{0.5}$, particular $_{0.3}$, expertise $_{2.3}$, reason $_{0.2}$, wizardimps $_{67.2}$, attention $_{0.5}$, desire $_{1.3}$
	LV-20	interests $_{1.0}$, interested $_{0.7}$, scientific $_{0.5}$, attention $_{0.5}$, particular $_{0.3}$, conflict $_{0.5}$, focus $_{0.5}$, concerned $_{1.0}$, nature $_{0.4}$, piqued $_{26.1}$
	SG	bshsu $_{90.2}$, conflict $_{0.5}$, btheeuropeanlibrary $_{87.6}$, bwral $_{65.5}$, bepochtimes $_{41.4}$, richarddawkins $_{74.5}$, thegauntlet $_{78.0}$, interested $_{0.7}$, wizardimps $_{67.2}$, towsonedu $_{95.1}$
	GloVe	interests $_{1.0}$, interested $_{0.7}$, concern $_{0.9}$, attention $_{0.5}$, focus $_{0.5}$, knowledge $_{0.5}$, conflict $_{0.5}$, influence $_{0.6}$, involvement $_{1.2}$, subject $_{0.2}$
	SVD	interests $_{1.0}$, interested $_{0.7}$, attention $_{0.5}$, substantial $_{1.1}$, share $_{0.7}$, own $_{0.1}$, credit $_{0.8}$, benefit $_{1.0}$, debt $_{1.6}$, financial $_{0.5}$
cup $_{0.1}$	LV-0	champions $_{0.5}$, uefa $_{1.3}$, championship $_{0.2}$, cups $_{2.7}$, league $_{0.1}$, finals $_{0.7}$, trophy $_{1.1}$, tournament $_{0.4}$, fifa $_{1.1}$, championships $_{0.3}$
	LV-20	champions $_{0.5}$, cups $_{2.7}$, championship $_{0.2}$, trophy $_{1.1}$, finals $_{0.7}$, league $_{0.1}$, competitions $_{0.9}$, runners $_{1.5}$, scorer $_{2.2}$, tournament $_{0.4}$
	SG	cups $_{2.7}$, champions $_{0.5}$, championship $_{0.2}$, finals $_{0.7}$, trophy $_{1.1}$, league $_{0.1}$, supercup $_{8.5}$, uefa $_{1.3}$, scorers $_{4.7}$, intertoto $_{12.4}$
	GloVe	championship $_{0.2}$, cups $_{2.7}$, champions $_{0.5}$, tournament $_{0.4}$, league $_{0.1}$, uefa $_{1.3}$, finals $_{0.7}$, championships $_{0.3}$, trophy $_{1.1}$, matches $_{0.4}$
	SVD	runners $_{1.5}$, nextseason $_{4.9}$, champions $_{0.5}$, prevseason $_{4.9}$, cups $_{2.7}$, scorers $_{4.7}$, competitions $_{0.9}$, scorer $_{2.2}$, fifa $_{1.1}$, matches $_{0.4}$
soul $_{0.9}$	LV-0	love $_{0.2}$, blues $_{0.8}$, heaven $_{1.5}$, funk $_{2.7}$, album $_{0.1}$, mind $_{0.5}$, spirit $_{0.8}$, souls $_{3.0}$, god $_{0.4}$, gospel $_{1.6}$
	LV-20	heaven $_{1.5}$, funk $_{2.7}$, love $_{0.2}$, essence $_{3.0}$, souls $_{3.0}$, mind $_{0.5}$, eternal $_{2.5}$, forever $_{1.7}$, spirit $_{0.8}$, dreams $_{1.6}$
	SG	soulful $_{11.0}$, funk $_{2.7}$, love $_{0.2}$, heaven $_{1.5}$, blues $_{0.8}$, essence $_{3.0}$, funky $_{6.1}$, souls $_{3.0}$, temptations $_{9.3}$, changeless $_{89.4}$
	GloVe	blues $_{0.8}$, funk $_{2.7}$, mind $_{0.5}$, hop $_{1.1}$, hip $_{1.0}$, love $_{0.2}$, rap $_{2.1}$, spirit $_{0.8}$, pop $_{0.4}$, heaven $_{1.5}$
	SVD	heaven $_{1.5}$, forever $_{1.7}$, dreams $_{1.6}$, eternal $_{2.5}$, dream $_{0.9}$, love $_{0.2}$, souls $_{3.0}$, funk $_{2.7}$, spirit $_{0.8}$, destiny $_{3.0}$

Table 6.3 – Same as Table 6.2, but with words sampled from RW. LV-0 neighbors have no clear semantic connection to the target word. In the LV-20 model, have clear semantic relations to the targets. GloVe behaves like LV-0, and Skip-gram and SVD like LV-20.

word	model	neighbors
rooters $_{77}$	LV-0	sonorella $_{98}$, bhavas $_{67}$, recreative $_{86}$, wwwjskscoin $_{98}$, wwwlegion $_{95}$, damsels $_{44}$, hassane $_{72}$, maniraptorans $_{99}$, abolboda $_{52}$, sympycnus $_{47}$
	LV-20	cheered $_{15}$, cheering $_{11}$, howled $_{73}$, jubilant $_{26}$, hissed $_{75}$, bosox $_{81}$, mcgreevy $_{53}$, rogell $_{62}$, jeers $_{43}$, mobbed $_{33}$
	SG	nuxhall $_{73}$, sparky $_{11}$, yanks $_{15}$, strupper $_{87}$, whitey $_{12}$, chisox $_{83}$, schoendienst $_{56}$, altrock $_{96}$, clendenon $_{85}$, campaneris $_{62}$
	GloVe	paiks $_{93}$, trashmen $_{96}$, wampanoags $_{93}$, mycenaeans $_{79}$, highnesses $_{60}$, perseids $_{89}$, clubmen $_{92}$, thalians $_{98}$, guelf $_{99}$, housecarls $_{93}$
	SVD	ballplayers $_{31}$, shibe $_{31}$, semipro $_{60}$, ebbets $_{29}$, phanatic $_{71}$, kekiongas $_{80}$, mudville $_{73}$, comiskey $_{18}$, mutuals $_{33}$, nabbp $_{48}$
monocultures $_{76}$	LV-0	pegomya $_{94}$, atara $_{81}$, shebang $_{40}$, subsidization $_{69}$, kiyomori $_{38}$, lucullus $_{23}$, intercal $_{76}$, paaerduag $_{99}$, dagbon $_{80}$, voluntas $_{89}$
	LV-20	monoculture $_{35}$, crops $_{2}$, agroforestry $_{34}$, replanting $_{35}$, silviculture $_{45}$, replanted $_{27}$, overgrazing $_{26}$, clearcutting $_{47}$, rainfed $_{65}$, seedlings $_{11}$
	SG	monoculture $_{35}$, overgrazed $_{81}$, polyculture $_{88}$, silvicultural $_{84}$, crops $_{2}$, woodlots $_{73}$, intercropping $_{90}$, overstory $_{91}$, allelopathic $_{83}$, croplands $_{60}$
	GloVe	chlamydospores $_{88}$, microbubbles $_{94}$, renunciations $_{92}$, insectoids $_{91}$, monoculture $_{35}$, plasmodesmata $_{87}$, relaxations $_{79}$, contactors $_{66}$, urocystis $_{57}$, vortexes $_{90}$
	SVD	monoculture $_{35}$, polyculture $_{88}$, dryland $_{33}$, silviculture $_{45}$, fuelwood $_{62}$, clearcutting $_{47}$, replanting $_{35}$, agroforestry $_{34}$, seedlings $_{11}$, swidden $_{59}$
flighted $_{83}$	LV-0	ablabesmyia $_{31}$, hydroptila $_{40}$, sphex $_{69}$, semiotus $_{82}$, sympycnus $_{47}$, coelichneumon $_{60}$, prajapati $_{32}$, diorhabda $_{98}$, quizzer $_{73}$, wwwtsuru $_{96}$
	LV-20	ratite $_{63}$, ratites $_{35}$, dromaeosaurids $_{50}$, flightless $_{14}$, tinamous $_{38}$, raptorial $_{60}$, parrots $_{8}$, psittaciformes $_{41}$, maniraptorans $_{99}$, psittacidae $_{32}$
	SG	raptorial $_{60}$, flightlessness $_{92}$, tibiotarsus $_{88}$, dromaeosaurids $_{50}$, zygodactyl $_{57}$, maniraptorans $_{99}$, apomorphic $_{93}$, ratite $_{63}$, rectrices $_{65}$, hindlimbs $_{39}$
	GloVe	sunbathe $_{92}$, hypnotise $_{99}$, maniraptorans $_{99}$, illidan $_{80}$, languorous $_{98}$, githyanki $_{95}$, dichotomius $_{73}$, tmesisternus $_{54}$, chloroceryle $_{89}$, quadroon $_{72}$
	SVD	zygodactyl $_{57}$, raptorial $_{60}$, beaks $_{13}$, flightless $_{14}$, avians $_{73}$, opposable $_{42}$, featherless $_{60}$, ratite $_{63}$, forelimbs $_{17}$, pronated $_{92}$

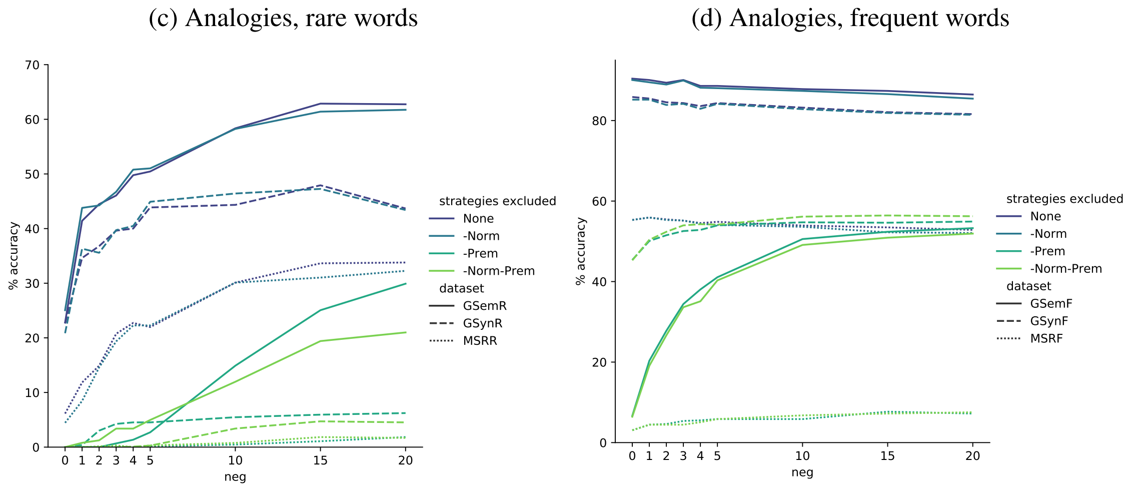
Figure 6.4 – (a) Word similarity results: RW improves as the number of negative samples increases. There is a small but consistent drop in performance on SimLex when increasing the number of negative samples. Scores in WSRel do indeed increase, but there is no clear drop in similarity scores in WSSim. (b) GSem/GSyn/MSR: Performance increases as the number of negative samples increases. Normalization has a minor effect, whereas premise exclusion is critical to performance. (c) GSem/GSyn/MSR Rare words: Consistent improvements in all conditions because rare words are better represented. (d) GSem/GSyn Frequent words: With both heuristics or only the normalization heuristic performance is nearly constant. For conditions without premise exclusion heuristic, there is remarkable improvement in analogy performance as negative samples increase, especially for semantic analogies. Metrics: for (a) Spearman rank correlation (×100), for (b,c,d), Accuracy.


(a) Word similarity

(b) Analogies

(c) Analogies, rare words

(d) Analogies, frequent words



– which happens as the relative importance of negative information is increased – true similarity drops.

Word analogies: Figure 6.4b shows how analogy performance varies as the number of negative samples is increased. Clearly performance increases as the number of negative samples increases, in particular for the semantic analogies. Removing the normalization heuristic has a minor effect on task performance. Removing the premise exclusion heuristic, however, leads to a tremendous drop in performance. This is why it is crucial to perform correct evaluation of word analogies without additional heuristics, for an accurate assessment of the semantic and syntactic information represented by the word embeddings.

The strong results with the premise exclusion heuristic might lead one to believe that the linear offsets corresponding to the analogies are straightforwardly accessible in the vector space when they clearly are not. However, if the number of negative samples is increased, we see a stark improvement under the correct evaluation, suggesting that these linear offsets/regularities manifest more clearly in the vector space.

To determine if this improvement is not merely due to better representations of rare words induced by increased negative sampling, as discussed previously, we look at the (GSem/GSyn/MSR)F/R datasets which contain the analogies with most/least frequent words. For (GSem/GSyn/MSR)R, we see the expected results in Figure 6.4c: consistent improvements with and without heuristics because rare words are better represented. The surprising result is for (GSem/GSyn/MSR)F in Figure 6.4d: with both heuristics or only the normalization heuristic performance is nearly constant. Without the premise exclusion heuristic, however, there is a remarkable improvement in analogy performance, especially for semantic analogies. This suggests that negative sampling is altering the geometry of the vector space in such a way that the linear offsets used to solve analogies hold more strongly, without heuristics.

6.4 Connection to other models

In this section, we study whether similar effects observed for our LexVec models are present in the popular Skip-gram, GloVe, and SVD models.

6.4.1 Materials

Given that Skip-gram implicitly factorizes a shifted PMI matrix, GloVe's factorization is related to a PMI factorization, and SVD performs optimal unweighted factorization, we investigate how these factorizations compare to the explicit PMI factorization performed by LexVec as regards increasing negative information. In particular, we are interested in the (dis)similarities of the geometry of the resulting word vector spaces.

Skip-gram: We train a Skip-gram model using the same parameters from the original paper (MIKOLOV et al., 2013), with window size $l = 5$ and number of negative samples in $\{1, 2, 5\}$ (we refer to these models as SG-1, SG-2 and SG-5). Note that Skip-gram with 5 negative samples performs the same amount of computation as the LexVec model with 20 negative samples: Skip-gram draws 5 negative samples per target-context pair ($2 \times 2 \times 5 = 20$ for each window), whereas LexVec draws 20 negative samples per window. Analogous parameters have the same values as the LexVec models: embedding dimension of size 300, learning rate of 0.025, negative distribution power of 0.75, subsampling threshold of $1e - 5$.

GloVe: The GloVe configuration follows the configuration of the original paper (PENNINGTON; SOCHER; MANNING, 2014), but with three changes to make the results directly comparable to LexVec: (1) unlike in the original paper, the corpus is subsampled using a threshold of $1e - 5$ before constructing the cooccurrence matrix. (2) Window size $l = 5$ to match LexVec and Skip-gram models. (3) Word vectors are output without averaging with context vectors, so that word vectors and context vectors can be analyzed separately. All other parameters are kept: embeddings of size 300, 100 training epochs, and learning rate of 0.05.

SVD: A limitation of the truncated SVD is that its computational efficiency is contingent on the sparsity of the input matrix. This sparsity is lost when using the $CPMI(-2)$, so we must use the zero-preserving transform PMI. Given the truncated SVD, $PPMI = U_d \Sigma_d V_d^T$, which discards all but the top d singular values, we follow Levy, Goldberg and Dagan (2015) and set word and context matrices to $W = U_d \sqrt{\Sigma_d}$, $C = V_d \sqrt{\Sigma_d}$ respectively. We factorize the $PPMI$ transform of M_{wiki5}, setting $d = 300$.

Table 6.4 – Comparing task performance of LexVec, Skip-gram, GloVe, and SVD models in Word similarity and analogy tasks (-: no heuristics excluded, -N: norm heuristic excluded, -P: premise heuristic excluded, -N-P: both heuristics excluded). The LV-0 model is similar to the GloVe model: strong SimLex results, weaker RW results, and similar across all analogy evaluations, barring MSR where GloVe is considerably stronger than all other models. Analogously, the LV-20 model is similar to the Skip-gram and models: strong SimLex and RW results, and consistently strong performance on all analogy evaluations. SVD has the lowest performance, but note that it suffers the smallest drop when the -P heuristic is excluded. Metrics: Spearman rank correlation (×100) for SimLex and RW, Accuracy for analogies.

model	SimLex	RW	GSem	GSem$^{\text{-N}}$	GSem$^{\text{-P}}$	GSem$^{\text{-N-P}}$
LV-0	35.3	42.6	76.0	76.1	0.9	1.3
LV-5	34.0	41.5	78.9	78.2	7.6	7.2
LV-20	32.3	44.1	77.7	77.1	**24.0**	**20.7**
SG-1	36.0	45.5	75.3	75.3	11.2	10.5
SG-2	**36.4**	46.5	77.4	76.9	11.7	10.6
SG-5	35.9	**46.9**	**78.9**	**78.4**	11.2	10.1
GloVe	35.4	40.6	74.4	74.3	5.6	6.0
SVD	28.6	41.9	43.9	41.7	20.5	20.6

model	GSyn	GSyn$^{\text{-N}}$	GSyn$^{\text{-P}}$	GSyn$^{\text{-N-P}}$	MSR	MSR$^{\text{-N}}$	MSR$^{\text{-P}}$	MSR$^{\text{-N-P}}$
LV-0	56.2	55.6	6.5	7.1	40.1	39.4	1.7	2.0
LV-5	59.1	58.5	11.2	11.8	43.0	42.4	4.4	4.5
LV-20	58.1	57.2	**13.7**	**14.7**	42.5	41.1	5.8	5.9
SG-1	61.9	60.4	11.4	12.2	45.6	44.8	3.7	3.7
SG-2	63.3	61.8	11.3	12.4	47.3	46.6	4.0	4.1
SG-5	63.0	61.1	10.9	11.8	46.3	45.2	3.1	3.7
GloVe	**64.5**	**64.0**	13.3	14.0	**58.1**	**57.6**	**7.9**	**8.5**
SVD	42.0	40.8	9.2	8.9	26.9	23.1	4.8	3.9

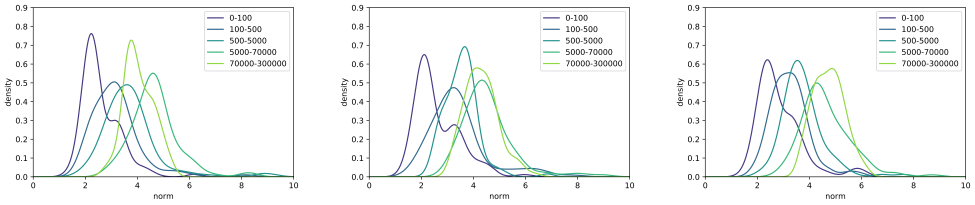
6.4.2 Results

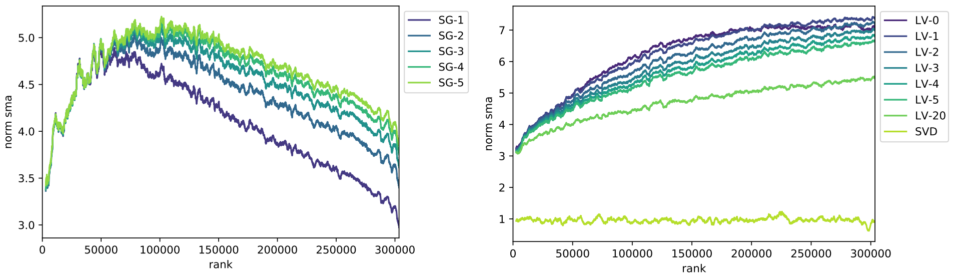
Given the similarities in loss functions presented in Section 3.2 – that LexVec and Skip-gram both contains loss weights λ_{ws} and λ_{ns}, while GloVe also contains loss weight λ_{ws} – we expect the Skip-gram model trained using 5 negative samples to resemble the LexVec LV-20 model that draws 20 negative samples (since, as stated in Section 6.4.1, this leads to the same number of total negative samples) and the GloVe model to resemble the LV-0 which uses only window sampling. We expect the patterns that emerge as the number of negative samples increase to manifest clearly in the SVD model, which assigns the vast majority of its loss weights to negative values. Table 6.4 shows task results for all compared models.

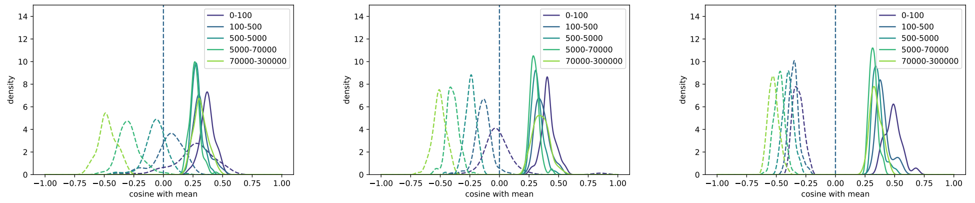
6.4.2.1 Skip-gram

- **Norms:** Figures 6.5a to 6.5c show the word vector norm distributions for the Skip-gram models. In contrast to the same figures for the LexVec models (Figures 6.2a to 6.2c), for Skip-gram it is not *clear* if increasing the number of negative samples increases the rank invariance of vector norms. Delving deeper, we plot a simple moving average of period 100 ($SMA_{100}(i) = \sum_{t=0}^{99} |W_{i-t}|/100$) of vector norms as a function of word rank in Figures 6.5d and 6.5e. Although the shape of both functions is different, it is clear for both Skip-gram and LexVec that as the number of negative samples increases the functions become flatter, indicating the increase in rank invariance of vector norms.
- **Directions:** In Figures 6.5f to 6.5h, we plot the distribution of cosines between words of different frequency buckets with the mean vector of all buckets for the SG models, as was drawn in Figures 6.3a to 6.3c for the LexVec models. Just as with the LexVec models, word vectors point in the same direction and word vectors point away from context vectors as the number of negative samples increases. This is precisely what was observed as the "strange" geometry of Skip-gram in Mimno and Thompson (2017), here explained by increasing importance of negative information.
- **Word similarity and analogies:** Overall SG results in Table 6.4 are similar to the LV-20 models, which is to be expected given the similarity in loss functions. All SG models achieve similar results, with the only clear trend being marginal im-

Figure 6.5 – (a, b, c): Word vector norm distributions for Skip-gram models with 1, 2 and 5 negative samples (SG-1, SG-2, SG-5). In contrast to the same figures for the LexVec models (Figures 6.2a to 6.2c), it is not clear if increasing the number of negative samples increases the rank invariance of vector norms. (d, e): Simple moving average of period 100 (with accompanying scatter plot of points used in calculating this average) of word vector norms as a function of word rank. We include additional SG-k and LV-k models for various k to make the trend clear. Although the shape of (d) and (e) is different, it is evident for both Skip-gram and LexVec that as the number of negative samples increases the functions become flatter, indicating the increase in rank invariance of vector norms. SVD displays the extreme case where nearly all loss weight is assigned to negative information, leading to ideal rank invariance. (f, g, h): Solid lines show distribution of cosines between vectors of words of different frequency buckets with the mean vector of all buckets, and dashed lines the of cosines of context vectors of these same words with the mean word vector, as in Figures 6.3a to 6.3c. Word vectors increasingly point in the same direction and word vectors point away from context vectors as the number of negative samples increases.


(a) SG-1 norms (b) SG-2 norms (c) SG-5 norms


(d) Skip-gram norms SMA (e) LexVec norms SMA


(f) SG-1 cosines with mean (g) SG-2 cosines with mean (h) SG-5 cosines with mean

provements in RW performance as the number of negative samples increases. Thus increasing negative information has little effect on task results. We attribute this to the fact that even with the minimum number of negative samples ($k = 1$), there is one negative sample per observed word-context pair, in contrast to LexVec where with $k = 1$ there are $2l$ observed pairs. In other words, the *minimum* relative importance of negative information within the Skip-gram model is sufficiently high so as not to observe the poor rare word representations observed in the LV-0 and GloVe models. Note that setting $k = 0$ makes the Skip-gram objective (Equation (3.9)) ill-defined since it can be made arbitrarily high by aligning all word vectors and increasing their norms.

- **Nearest neighbors:** We performed qualitative analysis of word neighbors in Tables 6.2 and 6.3 as was done with the LexVec models. Skip-gram neighbors are semantically related for both frequent and rare words, as is the case with other models that use negative information (LV-20 and SVD). The exception is the frequent word "interest" which has some incoherent neighbors. This irregularity deserves future investigation, but we suspect it is due to a weakness in the local nature of the Skip-gram model where, independent of the global PMI value for an observed pair (even if it is negative), a single step of optimization draws together the corresponding vectors. We omit results for SG-1,2 for there is no qualitative difference between the SG models as negative samples increase (our hypothesis for this is described above).

6.4.2.2 GloVe and SVD

- **GloVe:** In Figures 6.3a and 6.3e, LV-0 and GloVe behave similarly: vectors do not have a directional preference in respect to the mean vector and to context vectors. The similarity breaks in the distribution of vector norms in Figure 6.3e, which in LV-0 are far less rank invariant than in GloVe. We hypothesize that GloVe's vector norm rank invariance is due to bias terms which are responsible for scaling word-context dot products to approximate log cooccurrence count, allowing word/context vectors to have a similar norm.

  Word similarity and analogy results are given in Table 6.4. The LV-0 model is similar to the GloVe model: good SimLex results, weak RW results, and consistently similar across all analogy evaluations, except for MSR where the GloVe model outperforms all other models by a wide margin. The similarity is even clearer in the nearest neighbor samples in Tables 6.2 and 6.3. For both models, frequent words have semantically related neighbors, and rare words have incoherent neighbors. Overall, despite minor differences in their objectives, the GloVe and LV-0 models – which perform only window sampling – behave similarly.

- **SVD:** As can be seen in Figure 6.2f, the SVD vector norms are invariant to rank. As seen in moving from the LV-0 to LV-20 model, increasing the relative importance of negative information increases rank invariance of vector norms, and the SVD model which weighs negative information more heavily than any other model shows this effect to the extreme.

  Figure 6.3f shows less separation of word and context vectors than seen in the LV-20 and SG models. However, there is a clear separation in modes, with all word vector buckets having positive modes and all context vector buckets nearing zero. This is explained by the SVD model using the $PPMI$ transform, which drives dot products of negative cooccurring pairs to 0 rather than to negative values as with the $CPMI(-2)$ transform. Cosines of context vectors and the mean word vector are thus distributed near zero, rather than at negative values.

  Looking at Table 6.4, the SVD model is significantly weaker on the SimLex task than all other models. We attribute this to its uniform weighting of reconstruction errors. Weak results on RW are similarly attributed. Note that one might suspect the $PPMI$ transform to be at fault, but observe that in Table 5.1 the $PPMI$ variant of LexVec is on par with the other transforms that do not collapse the negative spectrum of PMI.

  Under incorrect evaluation – which includes norm and premise exclusion heuristics – SVD has the weakest performance on analogies of all models tested. However, if both heuristics are excluded, it performs nearly as well as the LV-20 model on GSem$^{\text{-N-P}}$, and marginally worse on GSyn$^{\text{-N-P}}$ and MSR$^{\text{-N-P}}$. We attribute the weaker performance on syntactic analogies to the $PPMI$ metric (in Table 5.1, $PPMI$ underperforms both $CPMI(-2)$ and $NNEGPMI$ on GSyn and MSR), and the strong performance *with heuristic exclusion* to the majority weighting of negative information in the loss function.

  Under qualitative analysis of nearest neighbors in Tables 6.2 and 6.3, the SVD model returns semantically related words for both frequent and rare words, similar to the LV-20 model.

6.5 Conclusion

In this chapter, we showed that increasing the number of negative samples in the LexVec and Skip-gram models increases the relative importance of negative information. This increase has a direct impact on the geometry of the resulting word vector spaces, increasing the rank invariance of vector norms and directions. Additionally, we observe that accounting for negative information significantly improves the rare word representations. Results also suggest the changes in geometry induced by increased negative information positively impact word analogy performance under correct evaluation (when no heuristics are used).

7 CONCLUSIONS AND FUTURE WORK

This thesis showed the important role of negative information in word vector representations. To do so, we first designed a word embedding model that explicitly accesses the PMI matrix, then exploited its explicit nature to isolate the effects of positive and negative information and the resulting impact on word vector geometry.

To design our model, we started by defining the set of principled reconstruction error properties that matrix factorization models should respect and situated existing models within this framework (in answer to Q1). Guided by these properties, we proposed LexVec, a new, highly-scalable method for generating word embeddings that uses low-rank, weighted factorization of the arbitrary transformations of the word-context cooccurrence matrix (in answer to Q2 and Q3). We incorporated positional information into our model, and compared existing and novel ways of merging word and context vectors to obtain better single word vector representations (in answer to Q4). By adding subword information into LexVec, we showed that naive character n-grams are a robust alternative to linguistically-motivated morphemes (in answer to Q5 and Q6).

LexVec effectively closes the performance gap between counting and predictive word embedding models, creating both a state-of-the-art static word embedding model while being a more transparent alternative to Skip-gram thanks to its explicit nature. LexVec's average downstream performance is (1) superior to a GloVe model trained on 13x more data in all evaluations (2) superior to Skip-gram's trained on 1.3x more data in all evaluations excepting textual similarity (3) superior to the state-of-the-art fastText subword model trained on 10x more data in syntactic, natural language inference, and facet ranking tasks (in answer to Q8). Sentence embedding methods using LexVec and other static word embeddings are competitive in textual similarity tasks and are only marginally worse in mean text classification accuracy. In our proposed facet ranking task, results show that static word embeddings are an effective approach for our proposed facet ranking task, and are competitive with Transformer models when used as sentence encoders, with LexVec models outperforming other static word embeddings (in answer to Q9 and Q10).

We looked at alternatives to collapsing the spectrum of negative PMI values to 0 by proposing two new PMI variants which partially preserve the negative spectrum (in answer to Q7). We obtained models that use only positive or negative information by exploiting LexVec's explicit access to the underlying PMI matrix being factorized (in answer to Q11). Word and sentence-level evaluations show that only accounting for positive PMI the factorization strongly captures both semantics and syntax, whereas using only nPMI captures little of semantics but a surprising amount of syntactic information. Our work deepens our understanding of distributional semantics and of computational linguistics by extending the distributional hypothesis to "a word is not only characterized by the company that it keeps, *but also by the company it rejects*" (in answer to Q12). We hypothesize that grammar is systematically generating negative cooccurrence (or information), and by encoding negative information our models are indirectly encoding grammatical information.

Finally, we investigated the effects on word vector space geometry of increasing the relative importance of negative information. Increasing the relative importance of negative information strengthens geometric rank invariant properties – vector norms and direction – of word vectors and improves the representation of rare words (in answer to Q13). Experiments reveal similar results for Skip-gram, GloVe, and SVD models, showing that the important role played by negative information in LexVec is present in these other models as well.

Over the course of this PhD, the most significant performance jump in NLP tasks since word2vec has been due to *deep* contextualized word embeddings (in particular Transformer models) – where rather than having a single vector for a word as described in the factorization above, models use neural networks to compute vectors for words *in context*. However, static word embeddings still dominate most *lexical semantic tasks* (such as word synonymy, similarity, relatedness, categorization, and analogy completion) (LENCI et al., 2021) where words are given *out-of-context*, making static word embeddings the appropriate choice. From an application point of view, although consistently inferior to the deep neural network models above, classic word embeddings still serve as very strong baselines with significantly lower computational cost, for both training and inference, and there is ongoing research into making them stronger (KIELA; WANG; CHO, 2018; WIETING; KIELA, 2019; YANG; ZHU; CHEN, 2019; WANG et al., 2020b). Given the continued importance of static word embeddings, this thesis contributes to the line of work that aims to understand what leads to the strong performance of these models in lexical semantic tasks.

7.1 Future Work

We believe some interesting directions for future work are:

- In the same way that combining vectors using concatenation rather than summing improved downstream task performance, explore similarly concatenating LexVec models that incorporate only positive or negative information, thus reducing the conflation between semantic and syntactic information in the vector representations. This might allow trained neural networks to better probe the semantic or syntactic information separately in a way that optimizes for downstream tasks.
- Train LexVec on larger corpora to see whether downstream performance – which is already comparable to a fastText model trained on 10x more data – can further push the state-of-the-art for static word embeddings.
- The increase in word analogy performance under correct evaluation, as the relative importance of negative information increases, warrants further investigation. Our results lead us to hypothesize that this is due to the change in the vector space geometry, with words of all ranks clustering in a small region of space where linear offsets used to solve analogies are more likely to hold.
- Explore the connection between our results in distributional semantics that negative information strongly captures syntactic information and the role that indirect evidence plays in language acquisition.

FUNDING

This research was partly supported by CAPES (1556422), CNPq (312114/2015-0, 423843/2016-8, 140402/2018-7), EPSRC (EP/T02450X/1), AIM-WEST (FAPERGS-INRIA 1706-2551/13-7), and project "Simplificacão Textual de Expressões Complexas" (sponsored by Samsung Eletrônica da Amazônia Ltda. under the terms of Brazilian federal law No. 8.248/91).

REFERENCES

ABNAR, S. et al. Experiential, distributional and dependency-based word embeddings have complementary roles in decoding brain activity. In: **Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018)**. Salt Lake City, Utah: Association for Computational Linguistics, 2018. p. 57–66. Available from Internet: <https://aclanthology.org/W18-0107>.

AGGARWAL, A.; VITTER JEFFREY, S. The input/output complexity of sorting and related problems. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 31, n. 9, p. 1116–1127, sep 1988. ISSN 0001-0782. Available from Internet: <https://doi.org/10.1145/48529.48535>.

AGIRRE, E. et al. SemEval-2014 task 10: Multilingual semantic textual similarity. In: **Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)**. Dublin, Ireland: Association for Computational Linguistics, 2014. p. 81–91. Available from Internet: <https://aclanthology.org/S14-2010>.

AGIRRE, E. et al. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In: **Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)**. Denver, Colorado: Association for Computational Linguistics, 2015. p. 252–263. Available from Internet: <https://aclanthology.org/S15-2045>.

AGIRRE, E. et al. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In: **Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)**. San Diego, California: Association for Computational Linguistics, 2016. p. 497–511. Available from Internet: <https://aclanthology.org/S16-1081>.

AGIRRE, E. et al. SemEval-2012 task 6: A pilot on semantic textual similarity. In: ***SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)**. Montréal, Canada: Association for Computational Linguistics, 2012. p. 385–393. Available from Internet: <https://aclanthology.org/S12-1051>.

AGIRRE, E. et al. *SEM 2013 shared task: Semantic textual similarity. In: **Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity**. Atlanta, Georgia, USA: Association for Computational Linguistics, 2013. p. 32–43. Available from Internet: <https://aclanthology.org/S13-1004>.

AKBIK, A. et al. FLAIR: An easy-to-use framework for state-of-the-art NLP. In: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)**. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 54–59. Available from Internet: <https://www.aclweb.org/anthology/N19-4010>.

ALIANNEJADI, M. et al. Asking clarifying questions in open-domain information-seeking conversations. In: ACM. **Proceedings of the 42nd International ACM SIGIR**

Conference on Research and Development in Information Retrieval. [S.l.], 2019. p. 475–484.

ALLEN, C.; HOSPEDALES, T. M. Analogies explained: Towards understanding word embeddings. In: CHAUDHURI, K.; SALAKHUTDINOV, R. (Ed.). **Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA**. PMLR, 2019. (Proceedings of Machine Learning Research, v. 97), p. 223–231. Available from Internet: <http://proceedings.mlr.press/v97/allen19a.html>.

BARONI, M.; DINU, G.; KRUSZEWSKI, G. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In: **Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics**. [S.l.: s.n.], 2014. v. 1, p. 238–247.

BENGIO, Y. et al. A neural probabilistic language model. **The Journal of Machine Learning Research**, JMLR. org, v. 3, p. 1137–1155, 2003.

BHATIA, P.; GUTHRIE, R.; EISENSTEIN, J. Morphological priors for probabilistic neural word embeddings. In: **EMNLP**. [S.l.: s.n.], 2016.

BÖHM, C.; BERCHTOLD, S.; KEIM, D. A. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 33, n. 3, p. 322–373, 2001.

BOJANOWSKI, P. et al. Enriching word vectors with subword information. **TACL**, v. 5, p. 135–146, 2017. Available from Internet: <https://transacl.org/ojs/index.php/tacl/article/view/999>.

BOLUKBASI, T. et al. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In: LEE, D. D. et al. (Ed.). **Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain**. [s.n.], 2016. p. 4349–4357. Available from Internet: <http://papers.nips.cc/paper/6228-man-is-to-computer-programmer-as-woman-is-to-homemaker-debiasing-word-embeddings>.

BOTHA, J. A.; BLUNSOM, P. Compositional morphology for word representations and language modelling. In: **ICML**. [S.l.: s.n.], 2014.

BOUMA, G. Normalized (pointwise) mutual information in collocation extraction. **Proceedings of GSCL**, p. 31–40, 2009.

BOWMAN, S. R. et al. A large annotated corpus for learning natural language inference. In: **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing**. Lisbon, Portugal: Association for Computational Linguistics, 2015. p. 632–642. Available from Internet: <https://aclanthology.org/D15-1075>.

BRANCO, A. et al. Comparative probing of lexical semantics theories for cognitive plausibility and technological usefulness. In: **Proceedings of the 28th International Conference on Computational Linguistics**. Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020. p. 4004–4019. Available from Internet: <https://aclanthology.org/2020.coling-main.354>.

BULLINARIA, J. A.; LEVY, J. P. Extracting semantic representations from word co-occurrence statistics: A computational study. **Behavior research methods**, Springer, v. 39, n. 3, p. 510–526, 2007.

CAO, K.; REI, M. A joint model for word embedding and word morphology. In: **Rep4NLP@ACL**. [S.l.: s.n.], 2016.

CER, D. et al. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In: **Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)**. Vancouver, Canada: Association for Computational Linguistics, 2017. p. 1–14. Available from Internet: <https://aclanthology.org/S17-2001>.

CHEN, Q. et al. Enhanced LSTM for natural language inference. In: **Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. Vancouver, Canada: Association for Computational Linguistics, 2017. p. 1657–1668. Available from Internet: <https://aclanthology.org/P17-1152>.

CHURCH, K. W.; HANKS, P. Word association norms, mutual information, and lexicography. **Computational Linguistics**, MIT Press, Cambridge, MA, USA, v. 16, n. 1, p. 22–29, mar. 1990. ISSN 0891-2017. Available from Internet: <http://portal.acm.org/citation.cfm?id=89095>.

CLARKE, C. L.; CRASWELL, N.; SOBOROFF, I. **Overview of the trec 2009 web track**. [S.l.], 2009.

CLARKE, C. L.; CRASWELL, N.; VOORHEES, E. M. **Overview of the TREC 2012 web track**. [S.l.], 2012.

CONNEAU, A.; KIELA, D. Senteval: An evaluation toolkit for universal sentence representations. **arXiv preprint arXiv:1803.05449**, 2018.

COTTERELL, R.; SCHÜTZE, H. Morphological word-embeddings. In: **HLT-NAACL**. [S.l.: s.n.], 2015.

COTTERELL, R.; SCHÜTZE, H.; EISNER, J. Morphological smoothing and extrapolation of word embeddings. In: **ACL**. [S.l.: s.n.], 2016.

CURRAN, J. R.; MOENS, M. Improvements in automatic thesaurus extraction. In: **Proceedings of the ACL-02 Workshop on Unsupervised Lexical Acquisition - Volume 9**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002. (ULA '02), p. 59–66.

DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. **Communications of the ACM**, ACM New York, NY, USA, v. 51, n. 1, p. 107–113, 2008.

DEERWESTER, S. C. et al. Indexing by latent semantic analysis. **JAsIs**, v. 41, n. 6, p. 391–407, 1990.

DEVLIN, J. et al. BERT: pre-training of deep bidirectional transformers for language understanding. In: BURSTEIN, J.; DORAN, C.; SOLORIO, T. (Ed.). **Proceedings of the**

2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). [S.l.]: Association for Computational Linguistics, 2019. p. 4171–4186.

DONG, W.; MOSES, C.; LI, K. Efficient k-nearest neighbor graph construction for generic similarity measures. In: **Proceedings of the 20th international conference on World wide web**. [S.l.: s.n.], 2011. p. 577–586.

ECKART, C.; YOUNG, G. The approximation of one matrix by another of lower rank. **Psychometrika**, Springer, v. 1, n. 3, p. 211–218, 1936.

ETHAYARAJH, K.; DUVENAUD, D.; HIRST, G. Towards understanding linear word analogies. In: KORHONEN, A.; TRAUM, D. R.; MÀRQUEZ, L. (Ed.). **Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers**. [S.l.]: Association for Computational Linguistics, 2019. p. 3253–3262.

FANO, R. M. Transmission of information: A statistical theory of communications. **American Journal of Physics**, American Association of Physics Teachers, v. 29, n. 11, p. 793–794, 1961.

FINKELSTEIN, L. et al. Placing search in context: The concept revisited. In: ACM. **Proceedings of the 10th international conference on World Wide Web**. [S.l.], 2001. p. 406–414.

FIRTH, J. R. A synopsis of linguistic theory, 1930-1955. **Studies in linguistic analysis**, Basil Blackwell, 1957.

FORAKER, S. et al. Indirect evidence and the poverty of the stimulus: The case of anaphoric one. **Cognitive Science**, Wiley Online Library, v. 33, n. 2, p. 287–300, 2009.

GIONIS, A. et al. Similarity search in high dimensions via hashing. In: **Vldb**. [S.l.: s.n.], 1999. v. 99, n. 6, p. 518–529.

GONEN, H.; GOLDBERG, Y. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In: BURSTEIN, J.; DORAN, C.; SOLORIO, T. (Ed.). **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)**. [S.l.]: Association for Computational Linguistics, 2019. p. 609–614.

HARRIS, Z. S. Distributional structure. **Word**, 1954.

HASHIMOTO, T. B.; ALVAREZ-MELIS, D.; JAAKKOLA, T. S. Word embeddings as metric recovery in semantic spaces. **TACL**, v. 4, p. 273–286, 2016. Available from Internet: <https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/809>.

HINTON, G. E. Learning distributed representations of concepts. In: AMHERST, MA. **Proceedings of the eighth annual conference of the cognitive science society**. [S.l.], 1986. v. 1, p. 12.

HOOVER, J. L. et al. Linguistic dependencies and statistical dependence. In: **Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing**. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021. p. 2941–2963. Available from Internet: <https://aclanthology.org/2021.emnlp-main.234>.

HOWARD, J.; RUDER, S. Universal language model fine-tuning for text classification. In: GUREVYCH, I.; MIYAO, Y. (Ed.). **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers**. Association for Computational Linguistics, 2018. p. 328–339. Available from Internet: <https://www.aclweb.org/anthology/P18-1031/>.

HU, M.; LIU, B. Mining and summarizing customer reviews. In: **Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2004. (KDD ’04), p. 168–177. ISBN 1581138881. Available from Internet: <https://doi.org/10.1145/1014052.1014073>.

HUANG, Z.; XU, W.; YU, K. Bidirectional lstm-crf models for sequence tagging. **CoRR**, abs/1508.01991, 2015.

IYYER, M. et al. Deep unordered composition rivals syntactic methods for text classification. In: **Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**. Beijing, China: Association for Computational Linguistics, 2015. p. 1681–1691. Available from Internet: <https://aclanthology.org/P15-1162>.

JURAFSKY, D. **Speech & language processing**. [S.l.]: Pearson Education India, 2000.

KIELA, D.; WANG, C.; CHO, K. Dynamic meta-embeddings for improved sentence representations. In: **Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing**. Brussels, Belgium: Association for Computational Linguistics, 2018. p. 1466–1477. Available from Internet: <https://www.aclweb.org/anthology/D18-1176>.

KIM, Y. et al. Character-aware neural language models. In: **AAAI**. [S.l.: s.n.], 2016.

LEE, D. L.; CHUANG, H.; SEAMONS, K. Document ranking and the vector-space model. **IEEE software**, IEEE, v. 14, n. 2, p. 67–75, 1997.

LENCI, A. et al. A comprehensive comparative evaluation and analysis of distributional semantic models. **CoRR**, abs/2105.09825, 2021. Available from Internet: <https://arxiv.org/abs/2105.09825>.

LEVY, O.; GOLDBERG, Y. Neural word embedding as implicit matrix factorization. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2014. p. 2177–2185.

LEVY, O.; GOLDBERG, Y.; DAGAN, I. Improving distributional similarity with lessons learned from word embeddings. **Transactions of the Association for Computational Linguistics**, v. 3, p. 211–225, 2015.

LEVY, O.; GOLDBERG, Y.; RAMAT-GAN, I. Linguistic regularities in sparse and explicit word representations. **CoNLL-2014**, p. 171, 2014.

LING, W. et al. Finding function in form: Compositional character models for open vocabulary word representation. In: **EMNLP**. [S.l.: s.n.], 2015.

LING, W. et al. Two/too simple adaptations of Word2Vec for syntax problems. In: **Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**. Denver, Colorado: Association for Computational Linguistics, 2015. p. 1299–1304. Available from Internet: <https://aclanthology.org/N15-1142>.

LINZEN, T. Issues in evaluating semantic spaces using word analogies. In: **Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP, RepEval@ACL 2016, Berlin, Germany, August 2016**. [S.l.]: Association for Computational Linguistics, 2016. p. 13–18.

LIU, Q.; MCCARTHY, D.; KORHONEN, A. Towards better context-aware lexical semantics:adjusting contextualized representations through static anchors. In: **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Online: Association for Computational Linguistics, 2020. p. 4066–4075. Available from Internet: <https://aclanthology.org/2020.emnlp-main.333>.

LUND, K.; BURGESS, C. Producing high-dimensional semantic spaces from lexical co-occurrence. **Behavior Research Methods, Instruments, & Computers**, Springer, v. 28, n. 2, p. 203–208, 1996.

LUONG, M.-T.; SOCHER, R.; MANNING, C. D. Better word representations with recursive neural networks for morphology. **CoNLL-2013**, Citeseer, v. 104, 2013.

MANNING, C. D.; MANNING, C. D.; SCHÜTZE, H. **Foundations of statistical natural language processing**. [S.l.]: MIT press, 1999.

MARCUS, M. P.; MARCINKIEWICZ, M. A.; SANTORINI, B. Building a large annotated corpus of english: The penn treebank. **Comput. Linguist.**, MIT Press, Cambridge, MA, USA, v. 19, n. 2, p. 313–330, jun. 1993. ISSN 0891-2017. Available from Internet: <http://dl.acm.org/citation.cfm?id=972470.972475>.

MARCUS, M. P.; SANTORINI, B.; MARCINKIEWICZ, M. A. Building a large annotated corpus of english: The penn treebank. **Computational Linguistics**, v. 19, n. 2, p. 313–330, 1994. Available from Internet: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.57.694>.

MARELLI, M. et al. A sick cure for the evaluation of compositional distributional semantic models. In: REYKJAVIK. **Lrec**. [S.l.], 2014. p. 216–223.

MIASCHI, A.; DELL'ORLETTA, F. Contextual and non-contextual word embeddings: an in-depth linguistic investigation. In: **Proceedings of the 5th Workshop on Representation Learning for NLP**. Online: Association for Computational Linguistics, 2020. p. 110–119. Available from Internet: <https://aclanthology.org/2020.repl4nlp-1.15>.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

MIKOLOV, T. et al. Advances in pre-training distributed word representations. In: CALZOLARI, N. et al. (Ed.). **Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018**. European Language Resources Association (ELRA), 2018. Available from Internet: <http://www.lrec-conf.org/proceedings/lrec2018/summaries/721.html>.

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2013. p. 3111–3119.

MIMNO, D.; THOMPSON, L. The strange geometry of skip-gram with negative sampling. In: **Empirical Methods in Natural Language Processing**. [S.l.: s.n.], 2017.

MITCHELL, J.; STEEDMAN, M. Orthogonality of syntax and semantics within distributional spaces. In: **Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**. [S.l.: s.n.], 2015. v. 1, p. 1301–1310.

MITCHELL, T. M. et al. Predicting human brain activity associated with the meanings of nouns. **science**, American Association for the Advancement of Science, v. 320, n. 5880, p. 1191–1195, 2008.

NISSIM, M.; NOORD, R. van; GOOT, R. van der. Fair is better than sensational: Man is to doctor as woman is to doctor. **CoRR**, abs/1905.09866, 2019. Available from Internet: <http://arxiv.org/abs/1905.09866>.

NIWA, Y.; NITTA, Y. Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In: **Proceedings of the 15th Conference on Computational Linguistics - Volume 1**. USA: Association for Computational Linguistics, 1994. (COLING '94), p. 304–309. Available from Internet: <https://doi.org/10.3115/991886.991938>.

PANG, B.; LEE, L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In: **Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)**. Barcelona, Spain: [s.n.], 2004. p. 271–278. Available from Internet: <https://aclanthology.org/P04-1035>.

PANG, B.; LEE, L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: **Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)**. Ann Arbor, Michigan: Association for Computational Linguistics, 2005. p. 115–124. Available from Internet: <https://aclanthology.org/P05-1015>.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. **Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)**, v. 12, 2014.

PETERS, M. E. et al. Deep contextualized word representations. In: WALKER, M. A.; JI, H.; STENT, A. (Ed.). **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)**. [S.l.]: Association for Computational Linguistics, 2018. p. 2227–2237.

QIU, S. et al. Co-learning of word representations and morpheme representations. In: **Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers**. [S.l.: s.n.], 2014. p. 141–150.

RADFORD, A. et al. Language models are unsupervised multitask learners. 2019.

RECHT, B. et al. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In: SHAWE-TAYLOR, J. et al. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2011. v. 24. Available from Internet: <https://proceedings.neurips.cc/paper/2011/file/218a0aefd1d1a4be65601cc6ddc1520e-Paper.pdf>.

REGIER, T.; GAHL, S. Learning the unlearnable: The role of missing evidence. **Cognition**, Elsevier, v. 93, n. 2, p. 147–155, 2004.

REIMERS, N.; GUREVYCH, I. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)**. Hong Kong, China: Association for Computational Linguistics, 2019. p. 3982–3992. Available from Internet: <https://aclanthology.org/D19-1410>.

ROGERS, A.; DROZD, A.; LI, B. The (too many) problems of analogical reasoning with word vectors. In: **Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)**. Vancouver, Canada: Association for Computational Linguistics, 2017. p. 135–148. Available from Internet: <https://www.aclweb.org/anthology/S17-1017>.

RUMELHART, D. E.; ABRAHAMSON, A. A. A model for analogical reasoning. **Cognitive Psychology**, Elsevier, v. 5, n. 1, p. 1–28, 1973.

SALLE, A.; IDIART, M.; VILLAVICENCIO, A. Enhancing the lexvec distributed word representation model using positional contexts and external memory. **CoRR**, abs/1606.01283, 2016. Available from Internet: <http://arxiv.org/abs/1606.01283>.

SALLE, A. et al. Studying the effectiveness of conversational search refinement through user simulation. In: HIEMSTRA, D. et al. (Ed.). **Advances in Information Retrieval**. Cham: Springer International Publishing, 2021. p. 587–602. ISBN 978-3-030-72113-8.

SALLE, A.; VILLAVICENCIO, A. Incorporating subword information into matrix factorization word embeddings. In: **Proceedings of the Second Workshop on Subword/Character LEvel Models**. New Orleans: Association for Computational Linguistics, 2018. p. 66–71. Available from Internet: <https://www.aclweb.org/anthology/W18-1209>.

SALLE, A.; VILLAVICENCIO, A. Restricted recurrent neural tensor networks: Exploiting word frequency and compositionality. In: **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**. Melbourne, Australia: Association for Computational Linguistics, 2018. p. 8–13. Available from Internet: <https://www.aclweb.org/anthology/P18-2002>.

SALLE, A.; VILLAVICENCIO, A. Why so down? the role of negative (and positive) pointwise mutual information in distributional semantics. **CoRR**, abs/1908.06941, 2019. Available from Internet: <http://arxiv.org/abs/1908.06941>.

SALLE, A.; VILLAVICENCIO, A.; IDIART, M. Matrix factorization using window sampling and negative sampling for improved word representations. In: **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**. Berlin, Germany: Association for Computational Linguistics, 2016. p. 419–424. Available from Internet: <https://www.aclweb.org/anthology/P16-2068>.

SANDERS, P. Random permutations on distributed, external and hierarchical memory. **Information Processing Letters**, Elsevier, v. 67, n. 6, p. 305–309, 1998.

SCHLUTER, N. The word analogy testing caveat. In: **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)**. [S.l.: s.n.], 2018. p. 242–246.

SCHUTZE, H. Dimensions of meaning. In: IEEE. **Supercomputing'92: Proceedings of the 1992 ACM/IEEE Conference on Supercomputing**. [S.l.], 1992. p. 787–796.

SCHÜTZE, H. Part-of-speech induction from scratch. In: SCHUBERT, L. K. (Ed.). **31st Annual Meeting of the Association for Computational Linguistics, 22-26 June 1993, Ohio State University, Columbus, Ohio, USA, Proceedings**. ACL, 1993. p. 251–258. Available from Internet: <https://www.aclweb.org/anthology/P93-1034/>.

SCHÜTZE, H. Word space. In: **Advances in neural information processing systems**. [S.l.: s.n.], 1993. p. 895–902.

SHAZEER, N. et al. Swivel: Improving embeddings by noticing what's missing. **arXiv preprint arXiv:1602.02215**, 2016.

SOCHER, R. et al. Parsing with compositional vector grammars. In: **ACL (1)**. [S.l.: s.n.], 2013. p. 455–465.

SOCHER, R. et al. Recursive deep models for semantic compositionality over a sentiment treebank. In: **Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing**. Seattle, Washington, USA: Association for Computational Linguistics, 2013. p. 1631–1642. Available from Internet: <https://aclanthology.org/D13-1170>.

TURNEY, P. D.; LITTMAN, M. L. Measuring praise and criticism: Inference of semantic orientation from association. **ACM Transactions on Information Systems (TOIS)**, ACM, v. 21, n. 4, p. 315–346, 2003.

TURNEY, P. D.; PANTEL, P. From frequency to meaning: Vector space models of semantics. **Journal of artificial intelligence research**, v. 37, p. 141–188, 2010.

VASWANI, A. et al. Attention is all you need. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2017. p. 5998–6008.

VERWIMP, L. et al. Character-word lstm language models. In: **EACL**. [S.l.: s.n.], 2017.

VIRPIOJA, S. et al. **Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline**. [S.l.], 2013. 38 p. (Aalto University publication series SCIENCE + TECHNOLOGY; 25/2013). Available from Internet: <http://urn.fi/URN:ISBN:978-952-60-5501-5>.

VOORHEES, E. M.; TICE, D. M. Building a question answering test collection. In: **Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval**. New York, NY, USA: Association for Computing Machinery, 2000. (SIGIR '00), p. 200–207. ISBN 1581132263. Available from Internet: <https://doi.org/10.1145/345508.345577>.

VÚLIC, I. et al. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In: BARZILAY, R.; KAN, M. (Ed.). **Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers**. [S.l.]: Association for Computational Linguistics, 2017. p. 56–68.

WANG, B. et al. Efficient sentence embedding via semantic subspace analysis. **arXiv preprint arXiv:2002.09620**, 2020.

WANG, B. et al. **Efficient Sentence Embedding via Semantic Subspace Analysis**. 2020.

WANG, B.; KUO, C.-C. J. Sbert-wk: A sentence embedding method by dissecting bert-based word models. **IEEE/ACM Transactions on Audio, Speech, and Language Processing**, v. 28, p. 2146–2157, 2020.

WANG, Y. et al. A clinical text classification paradigm using weak supervision and deep representation. **BMC Med. Inf. & Decision Making**, v. 19, n. 1, p. 1:1–1:13, 2019.

WIEBE, J.; WILSON, T.; CARDIE, C. Annotating expressions of opinions and emotions in language. **Language resources and evaluation**, Springer, v. 39, n. 2, p. 165–210, 2005.

WIETING, J. et al. From paraphrase database to compositional paraphrase model and back. **Transactions of the Association for Computational Linguistics**, v. 3, p. 345–358, 2015. Available from Internet: <https://aclanthology.org/Q15-1025>.

WIETING, J. et al. Charagram: Embedding words and sentences via character n-grams. In: **EMNLP**. [S.l.: s.n.], 2016.

WIETING, J.; KIELA, D. No training required: Exploring random encoders for sentence classification. In: **7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019**. OpenReview.net, 2019. Available from Internet: <https://openreview.net/forum?id=BkgPajAcY7>.

YANG, Z.; ZHU, C.; CHEN, W. Parameter-free sentence embedding via orthogonal basis. In: **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)**. Hong Kong, China: Association for Computational Linguistics, 2019. p. 638–648. Available from Internet: <https://www.aclweb.org/anthology/D19-1059>.

APPENDIX A — EXTENDED ABSTRACT IN PORTUGUESE

Vetores palavra são um componente vital de sistemas modernos de processamento de linguagem natural (PLN). Uma abordagem para a obtenção de tais vetores se dá através da análise estatística de textos não anotados. Esta análise leva a uma matriz de coocorrência, onde linhas representam palavras, e colunas representam contextos (geralmente também palavras) com as quais as palavras coocorrem, e o valor corresponde ao número de coocorrências: o número de vezes em que se observa juntos uma palavra e um contexto dentro de uma distância (ou janela) de um ao outro. Para uma linha da matriz (o vetor de uma palavra), os contextos e seus valores descrevem a distribuição de contextos. Dada a hipótese clássica da linguística de que *palavras com significado semelhante possuem distribuição de contextos também semelhantes*, podemos computar a semelhança de vetores de palavras para comparar os significados dessas mesmas – daí o nome para esta área de estudo de *semântica distribucional*.

Usar valores de coocorrências diretamente em computações não funciona bem empiricamente, e de um ponto de vista teórico também se avista problemas, pois palavras frequentes como “a” e “um” passam a dominar as distribuições de contextos. Uma técnica comumente usada é a de aplicar a transformação de Pointwise Mutual Information (PMI) à matriz de coocorrência. O valor de PMI para dois eventos (aqui a observação de uma palavra e um contexto dentro de a mesma janela) indica a tendência de se observar *um* dos eventos, dado que se observou o outro, e vice-versa. Então, por exemplo, o par “Porto Alegre” possui PMI elevado, enquanto o par “Porto Teutônia” possui um PMI baixo. PMI trata o problema de palavras frequentes dominarem a distribuição de contextos já que palavras como “a” e um “um” coocorrem de forma não-discriminativa com um número muito grande de palavras, portanto observar uma destas palavras não restringe possíveis contextos (matematicamente, a probabilidade de observar um contexto não é alterada ao observar uma destas palavras frequentes). Neste caso, o PMI tende a zero.

No caso contrário, quando uma palavra rejeita um contexto, PMI assume valores negativos proporcional à rejeição. Ao longo desta tese, nos referimos a este fenômeno como *informação negativa*. Se ao construir a matriz de coocorrência não se observar nenhuma vez um dado par palavra-contexto, tem-se o caso extremo de rejeição total e PMI tende a menos infinito. Esta *não-coocorrência* é muito frequente, ainda mais se o vocabulário for grande e o corpus pequeno. Por este motivo, na prática, PMI é restrito a números positivos: qualquer valor menor do que zero (incluindo infinito negativo) é setado a zero. Restringir PMI desta maneira gera a medida chamada Positive Pointwise Mutual Information (PPMI). Embora funcione bem empiricamente, esta restrição leva ao tema de nossa tese: que informação sobre palavras são capturadas pelo espectro de informação negativa, e o que perdemos ao restringi-lo?

Para tornar a questão mais relevante, a abordamos no contexto de word embeddings: vetores palavra *densos*, ao contrário dos vetores altamente esparsos quando se usa linhas da matriz de coocorrência. Do ponto de vista computacional, vetores densos são muito mais práticos do que vetores esparsos, em particular para uso junto a redes neurais. E do ponto de vista teórico, a compressão de vetores esparsos em vetores densos leva a fatores latentes que reduzem o ruído nas representações, melhorando o poder de generalização. Métodos para a obtenção de vetores densos podem ser divididos em duas famílias: (1) métodos de contagem, onde é construída a matriz de coocorrência, aplicada alguma transformação como PPMI, e feita uma *fatoração* desta matriz resultante em produto de matrizes esquerda e direita menores e densas onde as linhas da matriz esquerda representam os word embeddings; (2) métodos preditivos, onde se treina alguma arquitetura de redes neurais para predizer a distribuição de contextos dado como entrada um word embedding. GloVe e word2vec são os métodos mais populares de contagem e predição, respectivamente. Empiricamente, word2vec (em particular o seu modelo Skip-gram) obtém resultados melhores em diversas avaliações quando comparado a métodos de contagem. Seguindo o tema da nossa tese, hipotetizamos que essa desvantagem de métodos de contagem é devida a forma como esses não consideram de forma *embasada* a informação negativa.

Dado que a compressão em vetores densos leva a uma perda de informação, definimos 3 critérios de priorização de informação respondendo à pergunta: na reconstrução de matriz a partir da fatoração, quais células (ou pares) são mais informativas e devem ser melhor aproximadas? Os critérios são (1) a informatividade de um par cresce monotonicamente com o número de coocorrências (2) a não-coocorrência de um par não deve ser ignorada (3) a informatividade de um par cresce monotonicamente com a frequência de cada um de seus constituintes (palavra e contexto). A partir destes critérios, criamos um método de contagem e fatoração chamado LexVec. Através de avaliações ao nível de palavra (lexicais) e ao nível de frase, mostramos que o LexVec tem desempenho superior a outros métodos de contagem e semelhante a métodos preditivos, confirmando nossa hipótese de que a desvantagem de métodos de contagem era devida à consideração inadequada da informação negativa. Além do modelo base, fazemos uma série de melhorias ao

LexVec. A posição relativa de um contexto em relação a uma palavra contém informação que é descartada se esta posição é desconsiderada. Mostramos como preservar esta informação, usando representações distintas para contextos para cada posição possível, como também diferentes formas de juntar vetores palavras e contexto para melhorar o desempenho. Incorporamos informação *sub-palavra*, quebrando palavras em unidades menores (com duas abordagens diferentes), permitindo assim que palavras que não foram vistas no treinamento sejam representadas através de sub-palavras conhecidas. Finalmente, elaboramos uma versão do LexVec que usa memória externa (disco em vez de memória principal), permitindo escalá-lo a vocabulários e corpora de escala web. Com a série de melhorias e treinando com um corpus de 63B de palavras, o desempenho médio do LexVec supera o do GloVe treinado com 840B (13x mais) em todas avaliações aplicadas, do Skip-gram treinado usando 100B de tokens em todas as avaliações aplicadas exceto a de similaridade textual, e do modelo de sub-palavras estado-da-arte fastText treinado usando 630B de palavras (10 vezes mais do que o LexVec) em tarefas sintáticas, de inferência textual, e de ordenamento de facetas. Dado que o desempenho é geralmente proporcional ao tamanho do corpus de treinamento, obter recursos computacionais para treinar o LexVec num corpus de grandeza semelhante aos usados nestes outros modelos é uma direção promissora.

Em seguida, usando o LexVec como ferramenta experimental, retomamos a questão motivadora da tese sobre o papel da informação negativa, isolando os efeitos de informação positiva e negativa. Isto é feito considerando *exclusivamente* informação positiva ou negativa ao efetuar a fatoração. Tal isolamento demonstra uma vantagem (e necessidade no caso do nosso estudo) de métodos de contagem em relação a métodos de predição: só é possível graças ao acesso explícito à matriz sendo fatorada. Definimos duas variantes de PMI que ao contrário de PPMI, preservam (em parte ou integralmente) o espectro de valores negativos, possibilitando assim o estudo sobre informação negativa. Dividimos o estudo dos vetores palavra resultantes em avaliações semânticas e sintáticas. Resultados mostram que o uso exclusivo de informação positiva captura ambas informações semânticas e sintáticas. Quando se considera apenas a informação negativa, capta-se pouca informação semântica, mas surpreendentemente muita informação sintática, ao ponto de superar em algumas tarefas sintáticas modelos que usam ambas informação negativa e positiva em suas fatorações.

Finalmente, estudamos o impacto que a informação negativa tem na geometria do espaço vetorial gerado pelo LexVec, Skip-gram, GloVe, e da Decomposição em Valores Singulares (SVD). Se aproveitando da natureza explícita do LexVec, comprovamos experimentalmente a intuição de que a amostragem negativa usada em ambos LexVec e Skip-gram seleciona principalmente pares com PMI negativo. Assim, através do aumento da amostragem negativa, podemos aumentar o peso de informação negativa na fatoração e observar o impacto deste aumento na geometria resultante. Os resultados revelam duas invariantes geométricas – a norma e direção vetorial – e melhorias nas representações de palavras raras que são induzidas pelo uso aumentado da informação negativa. Além disto, observa-se uma melhora surpreendente na capacidade de resolução de analogias sob avaliação correta (quando não são usadas heurísticas para aumentar o desempenho nesta tarefa), sugerindo que as alterações induzidas pela informação negativa na geometria são uma peça chave nesta tarefa que é tida como porta-voz da capacidade de modelos de PLN.

APPENDIX B — RANK STATISTICS

B.1 Dataset Rank Statistics

Table B.1 – Percentile rank statistics for words in the datasets used Chapters 5 and 6. Note: μ- mean, σ- standard deviation, Qk- k-th quartile.

dataset	μ	σ	min	Q1	median	Q3	max
Rare Words (RW)	19.3	24.4	0.0	1.8	7.9	27.8	100.0
SimLex	2.3	3.1	0.0	0.6	1.5	3.0	46.4
WordSim-Relatedness (WSRel)	1.6	2.3	0.0	0.3	0.8	1.9	22.4
WordSim-Similarity (WSSim)	2.0	2.9	0.0	0.3	0.7	2.3	21.8
MSR Syntactic Analogies (MSR)	5.2	13.2	0.0	0.3	0.7	2.5	97.0
Google Semantic Analogies (GSem)	10.4	15.3	0.0	0.5	2.8	13.5	69.2
Google Syntactic Analogies (GSyn)	3.7	5.2	0.0	0.7	1.8	4.6	38.4
Google Semantic Analogies, Frequent Split (GSemF)	0.9	0.6	0.0	0.3	0.7	1.3	2.3
Google Syntactic Analogies, Frequent Split (GSynF)	0.4	0.3	0.0	0.2	0.3	0.6	1.0
Google Semantic Analogies, Rare Split (GSemR)	6.1	9.3	0.0	1.1	2.7	6.1	69.2
Google Syntactic Analogies, Rare Split(GSynR)	3.5	5.3	0.0	0.5	1.5	3.6	38.4
Penn Treebank (POS)	12.3	16.5	0.0	2.2	6.1	15.0	99.8
Tree Depth (Dep)	14.4	17.5	0.0	2.8	7.8	18.8	99.6
Top Constituent (TopC)	14.1	17.2	0.0	2.8	7.6	18.2	99.9

APPENDIX C — COMPLEMENTARY EXPERIMENTS

C.1 Fixed Window Size $l = 2$ and Positional Contexts

We performed identical experiments to those in Chapters 5 and 6, but using positional contexts and fixed window size of 2. Results lead to matching conclusions as those for the larger randomized windows used in the main chapters, and further highlight the role negative information: accounting for the zero cooccurrence (pairs in mnPMI) is even more important when using this smaller window size and positional contexts, which increase the sparsity of the cooccurrence matrix. Under this increased sparsity, models which ignore pairs in mnPMI (such as GloVe and LexVec with no negative sampling) see severe degradation in rare word representations.

Table C.1 – Same as Table 6.1, but using **positional contexts and symmetric context window of fixed size 2**.

Name	Set	Full	Hist	WS	NS
nPMI: Negative information	$\{(w,c) \mid PMI_{w,c} \leq 0\}$	99.75	14.53	12.44	85.42
pPMI: Positive information	$\{(w,c) \mid PMI_{w,c} > 0\}$	0.25	85.47	87.56	14.58
mnPMI: Maximally-negative information	$\{(w,c) \mid M_{w,c} = 0\}$	99.71	0.00	0.00	69.08
nPMI\mnPMI: Collapsed negative information under PPMI	$\{(w,c) \mid PMI_{w,c} \leq 0 \land M_{w,c} > 0\}$	0.04	14.53	12.44	16.34

C.2 Subword Information

Here we repeat the experiments from Appendix C.1, but incorporate subword information into LexVec (Subword LexVec; SLV) and Skip-gram (fastText; FT). Results follow the same trend of Chapters 5 and 6, leading to matching conclusions. However, note that whereas in the main chapters and Appendix C.1 we are able to isolate the effects of negative information on words of different frequencies, using subword information

Figure C.1 – Same as Figure 3.2, but using **positional contexts and symmetric context window of fixed size 2**.

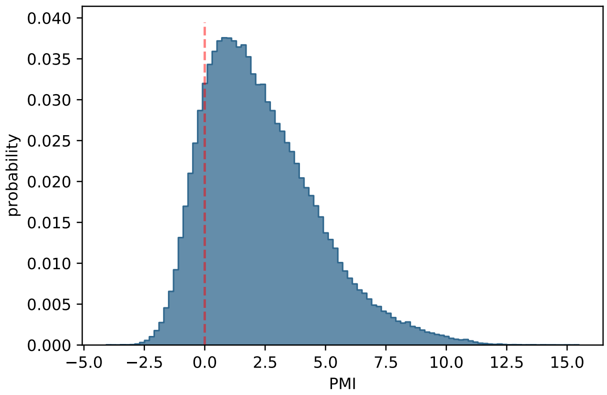


Figure C.2 – Same as Figure 6.1, but using **positional contexts and symmetric context window of fixed size 2**.

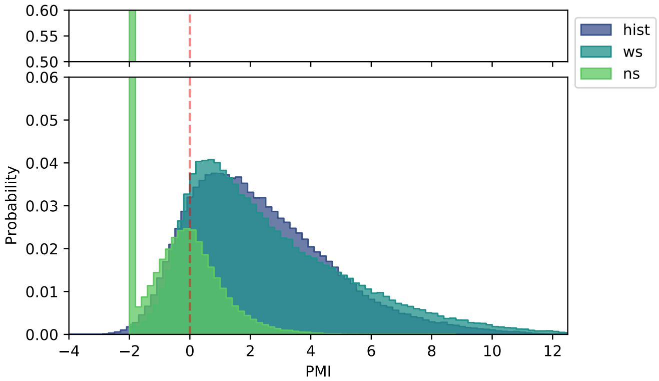
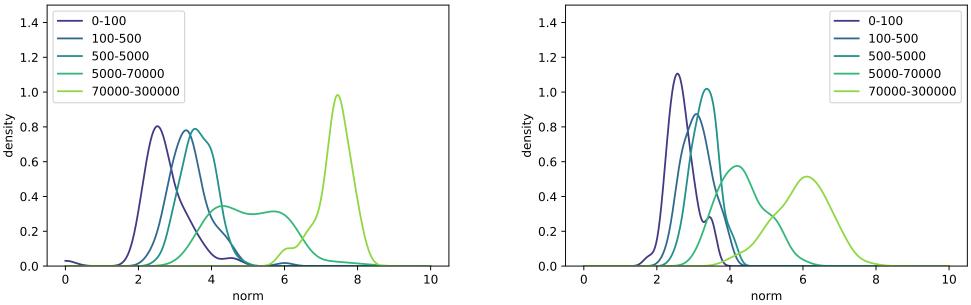


Table C.2 – Same as Table 5.1, but using **positional contexts and symmetric context window of fixed size 2**.

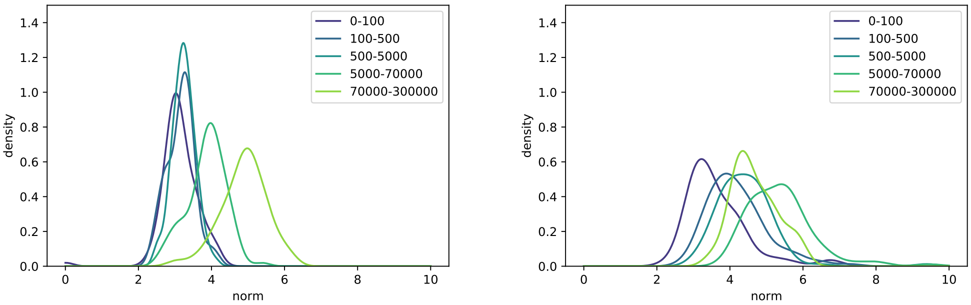
model	SimLex	RW	GSem	STSB	GSyn	MSR	POS	Dep	TopC
pPPMI	**36.9**	34.1	52.8	63.3	47.4	34.0	92.3	31.7	33.7
nPPMI	1.2	-1.2	0.0	48.1	0.0	0.0	16.3	17.9	5.0
nCPMI(-2)	18.3	24.0	6.4	41.0	13.9	13.9	90.8	**32.8**	**35.3**
nNPMI	13.7	23.9	4.5	40.1	8.9	7.8	89.8	31.7	34.0
PPMI	36.6	**45.1**	79.5	**63.4**	61.3	45.6	92.4	27.5	30.1
CPMI(-2)	35.8	43.1	80.4	63.0	**65.2**	**51.8**	**92.5**	28.0	31.3
NPMI	32.5	43.6	62.4	57.1	57.4	44.8	92.4	29.4	31.7
NNEGPMI	36.2	43.5	**80.7**	63.3	63.6	49.5	92.4	27.8	30.1
Random	1.2	-1.2	0.0	45.3	0.0	0.0	16.3	17.9	5.0

Figure C.3 – Same as Figure 6.2, but using **positional contexts and symmetric context window of fixed size 2**.

(a) LV-0 (b) LV-5



(c) LV-20 (d) Skip-gram



(e) GloVe (f) SVD

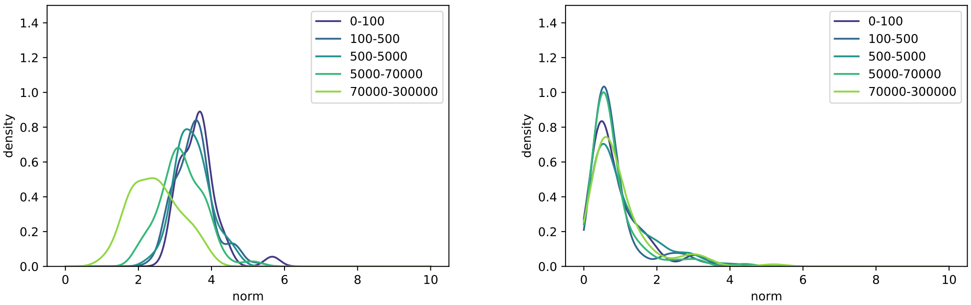
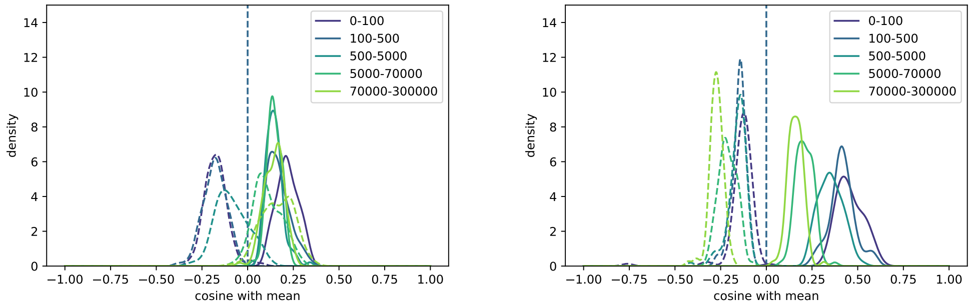
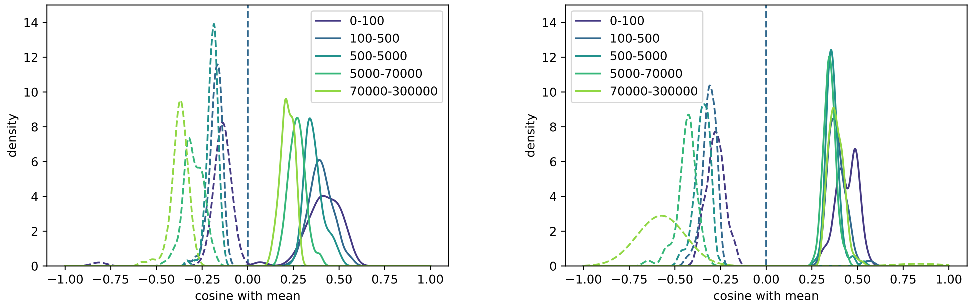


Figure C.4 – Same as Figure 6.3, but using **positional contexts and symmetric context window of fixed size 2**.

(a) LV-0 (b) LV-5



(c) LV-20 (d) Skip-gram



(e) GloVe (f) SVD

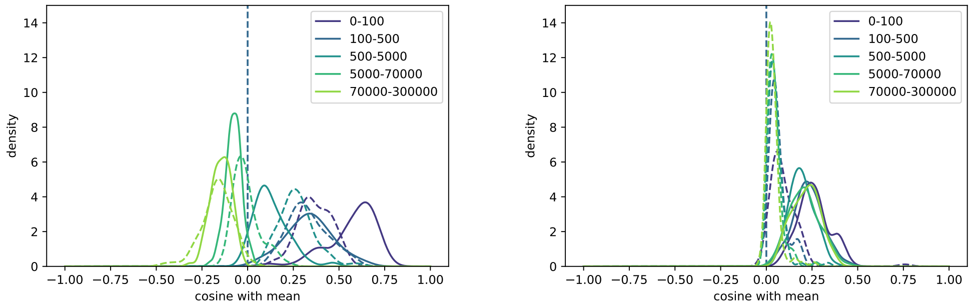
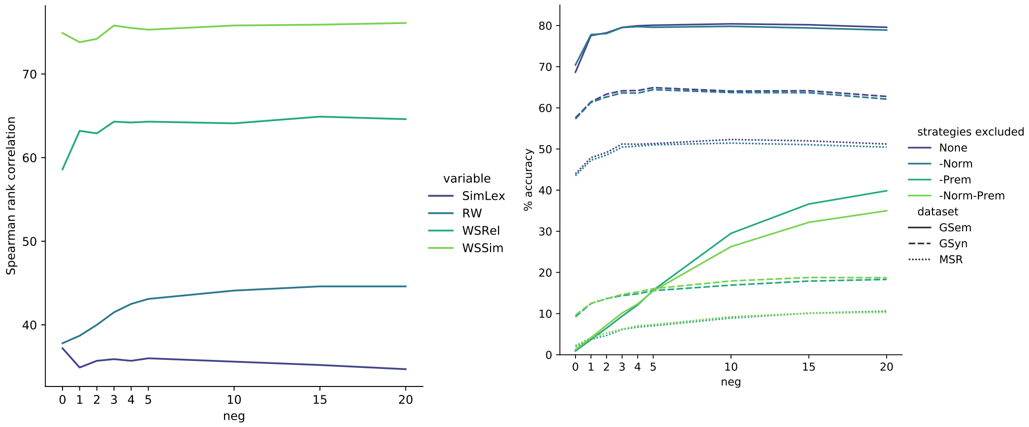


Figure C.5 – Same as Figure 6.4, but using **positional contexts and symmetric context window of fixed size 2**.



(a) Word similarity

(b) Analogies

(c) Analogies, rare words

(d) Analogies, frequent words

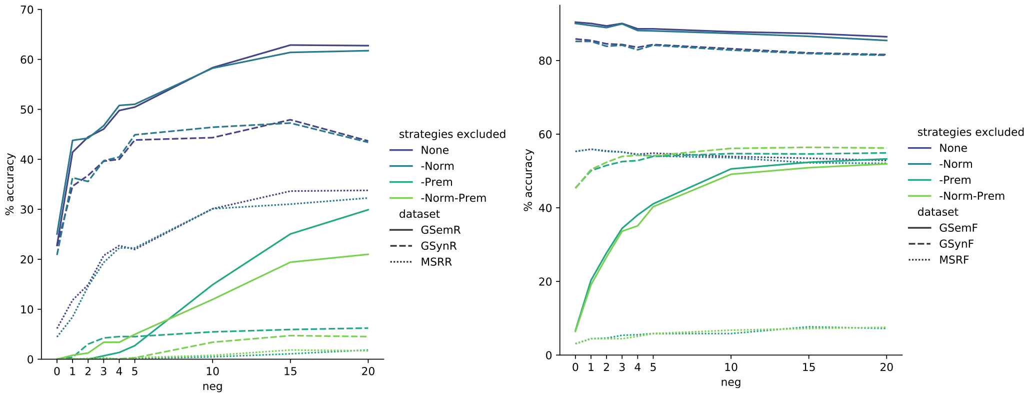


Table C.3 – Same as Table 6.2, but using **positional contexts and symmetric context window of fixed size 2**.

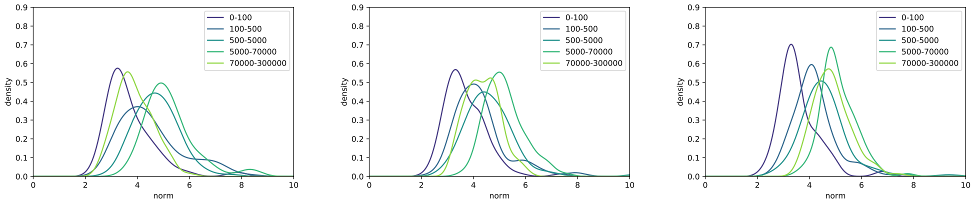
word	model	neighbors
interest $_{0.3}$	LV-0	interests $_{1.0}$, conflict $_{0.5}$, coi $_{1.2}$, interested $_{0.7}$, thegauntlet $_{78.0}$, scientific $_{0.5}$, passion $_{2.0}$, attention $_{0.5}$, concern $_{0.9}$, activity $_{0.6}$
	LV-20	interests $_{1.0}$, interested $_{0.7}$, conflict $_{0.5}$, attention $_{0.5}$, enthusiasm $_{3.9}$, expertise $_{2.3}$, concern $_{0.9}$, appreciation $_{3.5}$, scientific $_{0.5}$, involvement $_{1.2}$
	SG	interests $_{1.0}$, intrest $_{50.0}$, wizardimps $_{67.2}$, interested $_{0.7}$, richarddawkins $_{74.5}$, thegauntlet $_{78.0}$, interst $_{77.4}$, bluntsde $_{91.6}$, conflict $_{0.5}$, bizjournals $_{42.8}$
	GloVe	interests $_{1.0}$, interested $_{0.7}$, conflict $_{0.5}$, attention $_{0.5}$, influence $_{0.6}$, concern $_{0.9}$, expertise $_{2.3}$, involvement $_{1.2}$, popularity $_{1.2}$, passion $_{2.0}$
	SVD	interests $_{1.0}$, attention $_{0.5}$, interested $_{0.7}$, importance $_{0.4}$, profits $_{2.9}$, debt $_{1.6}$, expertise $_{2.3}$, benefit $_{1.0}$, contributions $_{0.5}$, contribution $_{1.0}$
cup $_{0.1}$	LV-0	championship $_{0.2}$, league $_{0.1}$, champions $_{0.5}$, cups $_{2.7}$, finals $_{0.7}$, trophy $_{1.1}$, uefa $_{1.3}$, final $_{0.1}$, tournament $_{0.4}$, championships $_{0.3}$
	LV-20	champions $_{0.5}$, cups $_{2.7}$, championship $_{0.2}$, finals $_{0.7}$, trophy $_{1.1}$, league $_{0.1}$, competitions $_{0.9}$, uefa $_{1.3}$, tournament $_{0.4}$, runners $_{1.5}$
	SG	cups $_{2.7}$, trophy $_{1.1}$, championship $_{0.2}$, champions $_{0.5}$, finals $_{0.7}$, supercup $_{8.5}$, pokal $_{9.4}$, championships $_{0.3}$, uhrencup $_{90.3}$, uefa $_{1.3}$
	GloVe	championship $_{0.2}$, champions $_{0.5}$, finals $_{0.7}$, cups $_{2.7}$, league $_{0.1}$, trophy $_{1.1}$, uefa $_{1.3}$, tournament $_{0.4}$, final $_{0.1}$, fifa $_{1.1}$
	SVD	runners $_{1.5}$, nextseason $_{4.9}$, champions $_{0.5}$, cups $_{2.7}$, trophy $_{1.1}$, scorers $_{4.7}$, competitions $_{0.9}$, matches $_{0.4}$, squad $_{0.7}$, fifa $_{1.1}$
soul $_{0.9}$	LV-0	souls $_{3.0}$, blues $_{0.8}$, funk $_{2.7}$, gospel $_{1.6}$, hop $_{1.1}$, mind $_{0.5}$, spirit $_{0.8}$, reggae $_{3.4}$, jazz $_{0.6}$, rap $_{2.1}$
	LV-20	funk $_{2.7}$, blues $_{0.8}$, heaven $_{1.5}$, essence $_{3.0}$, souls $_{3.0}$, mind $_{0.5}$, spirit $_{0.8}$, love $_{0.2}$, dreams $_{1.6}$, jazz $_{0.6}$
	SG	funk $_{2.7}$, souls $_{3.0}$, blues $_{0.8}$, soulful $_{11.0}$, seekerz $_{81.3}$, changeless $_{89.4}$, makossa $_{58.0}$, spirit $_{0.8}$, essence $_{3.0}$, jazzmatazz $_{98.9}$
	GloVe	blues $_{0.8}$, funk $_{2.7}$, souls $_{3.0}$, spirit $_{0.8}$, mind $_{0.5}$, gospel $_{1.6}$, hop $_{1.1}$, pop $_{0.4}$, jazz $_{0.6}$, love $_{0.2}$
	SVD	heaven $_{1.5}$, eternal $_{2.5}$, forever $_{1.7}$, dreams $_{1.6}$, dream $_{0.9}$, funk $_{2.7}$, love $_{0.2}$, souls $_{3.0}$, spirit $_{0.8}$, blues $_{0.8}$

Table C.4 – Same as Table 6.3, but using **positional contexts and symmetric context window of fixed size 2**.

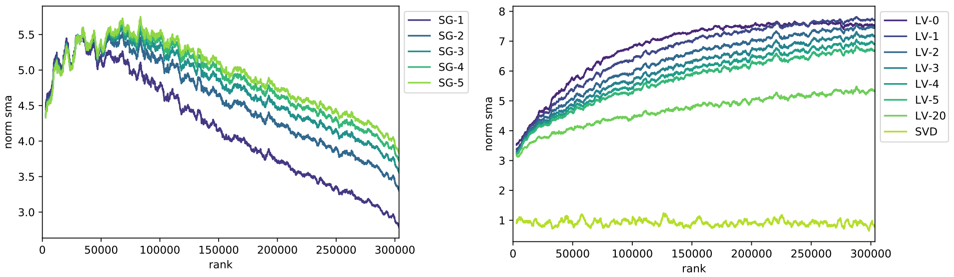
word	model	neighbors
rooters $_{77}$	LV-0	bajir $_{66}$, ravimy $_{87}$, hamerkop $_{83}$, argyrodes $_{68}$, thaumasia $_{90}$, brooklin $_{43}$, karuma $_{99}$, sorlle $_{86}$, valise $_{69}$, roaring $_{7}$
	LV-20	howled $_{73}$, cheering $_{11}$, roars $_{27}$, bosox $_{81}$, monumentals $_{68}$, cheered $_{15}$, nuxhall $_{73}$, booing $_{30}$, phillie $_{56}$, atlantics $_{26}$
	SG	cheering $_{11}$, bosox $_{81}$, howled $_{73}$, fans $_{0}$, lynah $_{86}$, cheered $_{15}$, landrith $_{57}$, clendenon $_{85}$, phillie $_{56}$, semipro $_{60}$
	GloVe	specifc $_{84}$, unserious $_{99}$, cusumano $_{87}$, whch $_{65}$, uninviting $_{91}$, overexcited $_{99}$, imnsho $_{89}$, preffered $_{91}$, alread $_{94}$, untypical $_{94}$
	SVD	ebbets $_{29}$, comiskey $_{18}$, ballplayers $_{31}$, batboy $_{62}$, crawfords $_{37}$, chisox $_{83}$, bosox $_{81}$, gothams $_{59}$, semipro $_{60}$, krichell $_{79}$
monocultures $_{76}$	LV-0	fieldensis $_{87}$, cantillans $_{87}$, berbers $_{13}$, ritsema $_{57}$, boutonii $_{76}$, tmutarakan $_{74}$, shuhada $_{67}$, chisocheton $_{74}$, poepp $_{77}$, approvals $_{12}$
	LV-20	monoculture $_{35}$, seedlings $_{11}$, saplings $_{24}$, conifers $_{13}$, hardwoods $_{19}$, crops $_{2}$, agroforestry $_{34}$, cultivations $_{64}$, understory $_{16}$, broadleaved $_{47}$
	SG	monoculture $_{35}$, polyculture $_{88}$, overgrazed $_{81}$, fuelwood $_{62}$, intercropping $_{90}$, rainfed $_{65}$, overharvesting $_{83}$, cucurbits $_{81}$, croplands $_{60}$, silvicultural $_{84}$
	GloVe	hereabouts $_{74}$, upend $_{97}$, controvertial $_{87}$, enlivening $_{79}$, pluralisation $_{92}$, selfsame $_{98}$, herrod $_{85}$, overspend $_{83}$, unserious $_{99}$, liquify $_{89}$
	SVD	monoculture $_{35}$, cropland $_{27}$, windbreaks $_{68}$, replanted $_{27}$, orchards $_{6}$, silviculture $_{45}$, plantations $_{3}$, intercropping $_{90}$, cultivations $_{64}$, seedlings $_{11}$
flighted $_{83}$	LV-0	uproot $_{34}$, okumoto $_{96}$, ratcheted $_{94}$, revealer $_{79}$, pandey $_{10}$, flagpole $_{15}$, halvard $_{66}$, bersetzungsstufen $_{74}$, swiftest $_{84}$, stairlift $_{86}$
	LV-20	feathered $_{11}$, flightless $_{14}$, quadrupedal $_{30}$, ratites $_{35}$, bipedal $_{16}$, raptorial $_{60}$, necked $_{6}$, beak $_{6}$, beaks $_{13}$, prehensile $_{26}$
	SG	digitigrade $_{84}$, plantigrade $_{84}$, zygodactyl $_{57}$, raptorial $_{60}$, woodcreepers $_{94}$, chelae $_{81}$, pronated $_{92}$, forelegs $_{30}$, apomorphic $_{93}$, stockier $_{86}$
	GloVe	wikispeak $_{87}$, similary $_{82}$, vandelism $_{97}$, specifc $_{84}$, validations $_{76}$, demagogic $_{77}$, imnsho $_{89}$, incentivise $_{100}$, smidge $_{97}$, geneological $_{98}$
	SVD	flightless $_{14}$, shoebill $_{71}$, tinamous $_{38}$, corvid $_{76}$, curassows $_{78}$, pratincoles $_{45}$, toucans $_{40}$, turacos $_{56}$, hoatzin $_{65}$, anseriformes $_{34}$

Figure C.6 – Same as Figure 6.5, but using **positional contexts and symmetric context window of fixed size 2**.

(a) SG-1 norms (b) SG-2 norms (c) SG-5 norms



(d) Skip-gram norms SMA (e) LexVec norms SMA



(f) SG-1 cosines with mean (g) SG-2 cosines with mean (h) SG-5 cosines with mean

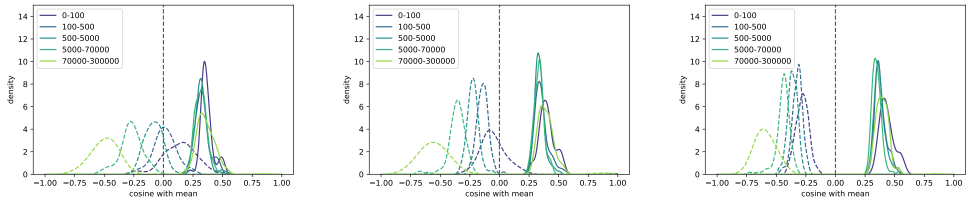


Table C.5 – Same as Table 6.4, but using **positional contexts and symmetric context window of fixed size 2**.

model	SimLex	RW	GSem	GSem$^{\text{-N}}$	GSem$^{\text{-P}}$	GSem$^{\text{-N-P}}$
LV-0	37.2	37.8	68.7	70.4	0.9	1.3
LV-5	36.0	43.1	**80.1**	79.6	15.7	15.5
LV-20	34.7	44.6	79.6	78.9	**39.9**	**35.0**
SG-1	38.5	46.4	74.7	74.3	18.0	17.8
SG-2	39.0	47.7	77.5	77.5	19.3	18.6
SG-5	**39.4**	**48.8**	79.8	**79.9**	18.9	18.2
GloVe	35.2	36.1	74.8	73.0	3.0	3.5
SVD	31.6	44.0	48.5	41.1	30.9	26.8

model	GSyn	GSyn$^{\text{-N}}$	GSyn$^{\text{-P}}$	GSyn$^{\text{-N-P}}$	MSR	MSR$^{\text{-N}}$	MSR$^{\text{-P}}$	MSR$^{\text{-N-P}}$
LV-0	57.6	57.3	9.2	9.6	44.0	43.5	2.0	2.3
LV-5	64.9	64.4	15.6	16.1	51.3	51.0	7.0	7.3
LV-20	62.8	62.1	**18.3**	**18.7**	51.2	50.5	**10.6**	**10.3**
SG-1	67.7	66.8	14.6	15.2	53.7	52.4	6.1	6.4
SG-2	**68.7**	**67.8**	14.8	15.6	54.8	53.5	6.0	6.7
SG-5	68.2	67.6	14.9	15.7	**56.0**	**54.7**	6.2	6.7
GloVe	59.2	58.2	9.7	9.7	47.5	45.7	2.8	3.2
SVD	49.3	46.5	17.1	15.8	37.3	31.5	9.9	8.5

breaks this isolation by sharing information between frequent and rare word forms. Nevertheless, despite this confounding factor, results follow a remarkably similar trend.

Table C.6 – Same as Table 6.1, but using **subword information**, positional contexts and symmetric context window of fixed size 2.

Name	Set	Full	Hist	WS	NS
nPMI: Negative information	$\{(w,c) \mid PMI_{w,c} \leq 0\}$	99.75	14.53	12.44	85.42
pPMI: Positive information	$\{(w,c) \mid PMI_{w,c} > 0\}$	0.25	85.47	87.56	14.58
mnPMI: Maximally-negative information	$\{(w,c) \mid M_{w,c} = 0\}$	99.71	0.00	0.00	69.08
nPMI\mnPMI: Collapsed negative information under PPMI	$\{(w,c) \mid PMI_{w,c} \leq 0 \wedge M_{w,c} > 0\}$	0.04	14.53	12.44	16.34

Figure C.7 – Same as Figure 3.2, but using **subword information**, positional contexts and symmetric context window of fixed size 2.

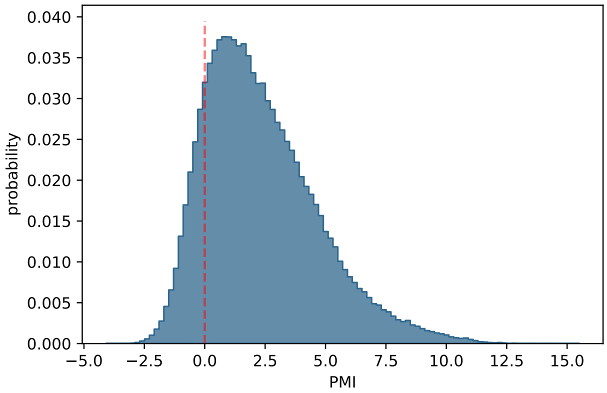


Figure C.8 – Same as Figure 6.1, but using **subword information**, positional contexts and symmetric context window of fixed size 2.

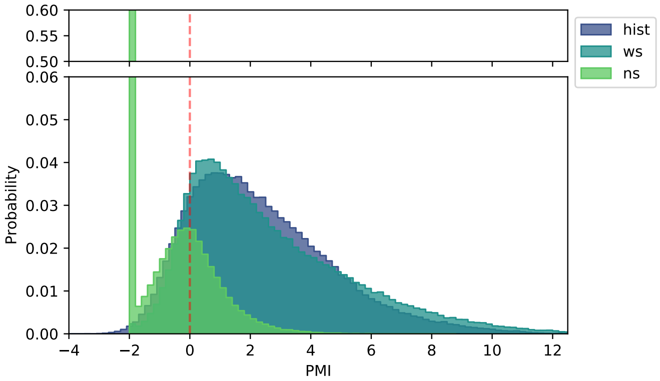
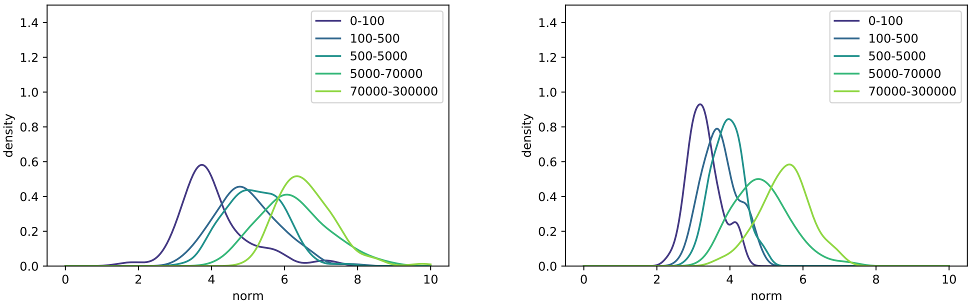


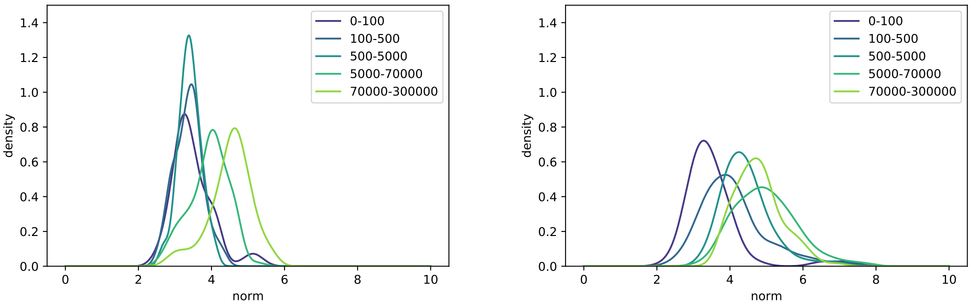
Table C.7 – Same as Table 5.1, but using **subword information**, positional contexts and symmetric context window of fixed size 2.

model	SimLex	RW	GSem	STSB	GSyn	MSR	POS	Dep	TopC
pPPMI	33.9	37.3	21.9	**64.3**	51.4	41.1	92.5	30.6	34.2
nPPMI	2.8	13.5	0.2	30.8	26.1	24.1	16.3	17.9	5.0
nCPMI(-2)	18.2	25.6	2.9	39.9	14.3	13.5	91.1	**33.0**	**35.5**
nNPMI	13.8	26.5	1.2	38.8	21.5	24.2	90.9	32.4	34.5
PPMI	**38.1**	**51.4**	72.2	63.5	67.5	52.7	92.5	27.9	31.1
CPMI(-2)	37.1	49.6	**77.3**	64.1	71.7	59.5	**92.8**	29.0	32.6
NPMI	32.8	46.6	32.8	54.1	**74.0**	**62.5**	92.4	31.5	32.4
NNEGPMI	37.2	49.6	76.5	64.0	70.5	57.3	92.6	28.5	31.5
Random	2.9	13.5	0.2	30.1	26.6	24.6	16.3	17.9	5.0

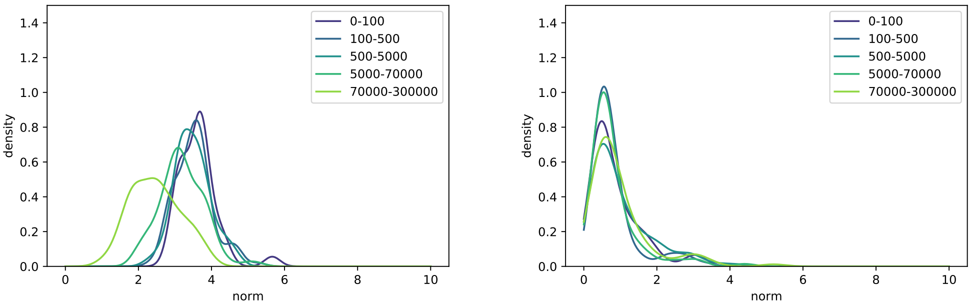
Figure C.9 – Same as Figure 6.2, but using **subword information**, positional contexts and symmetric context window of fixed size 2.



(a) SLV-0 (b) SLV-5



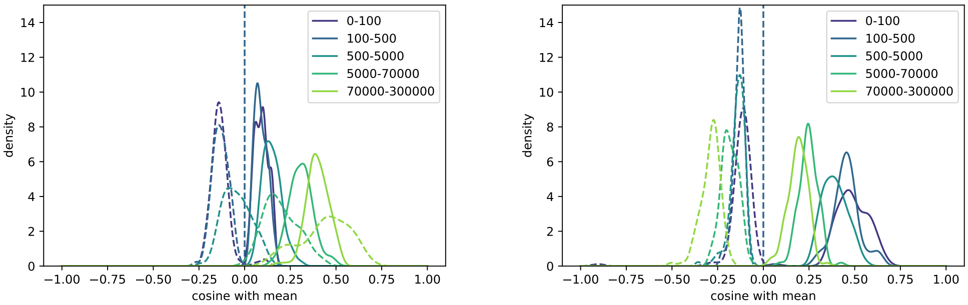
(c) SLV-20 (d) fastText



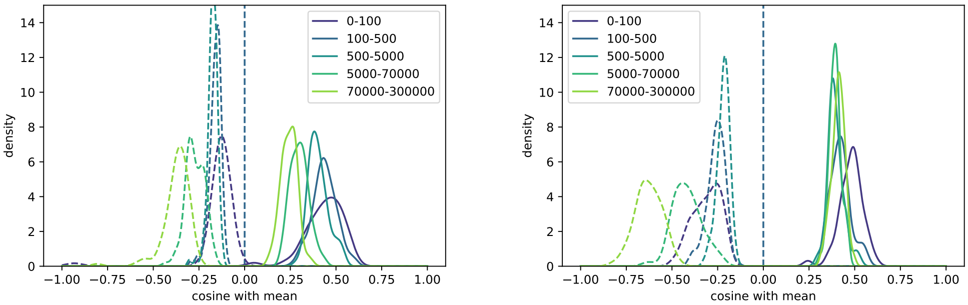
(e) GloVe (f) SVD

Figure C.10 – Same as Figure 6.3, but using **subword information**, positional contexts and symmetric context window of fixed size 2.

(a) SLV-0 (b) SLV-5



(c) SLV-20 (d) fastText



(e) GloVe (f) SVD

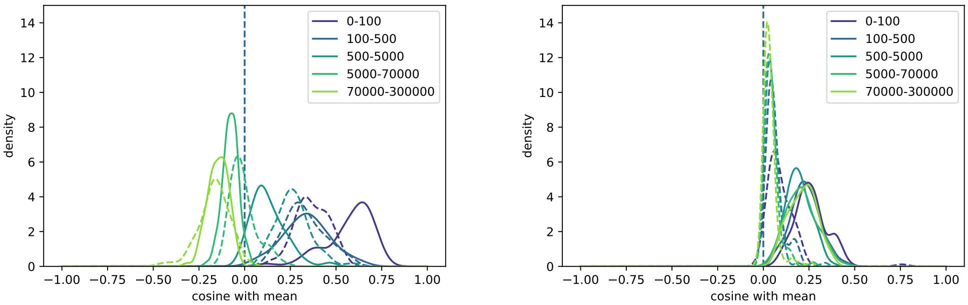
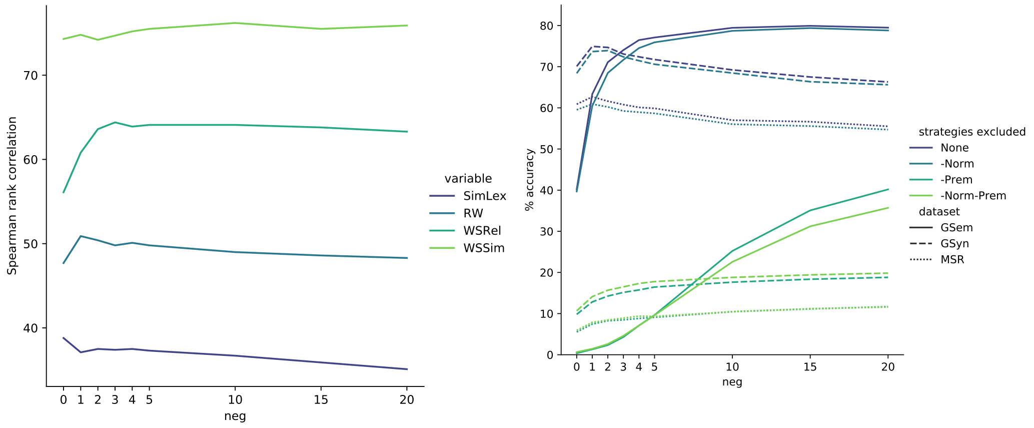


Figure C.11 – Same as Figure 6.4, but using **subword information**, positional contexts and symmetric context window of fixed size 2.



(a) Word similarity

(b) Analogies

(c) Analogies, rare words

(d) Analogies, frequent words

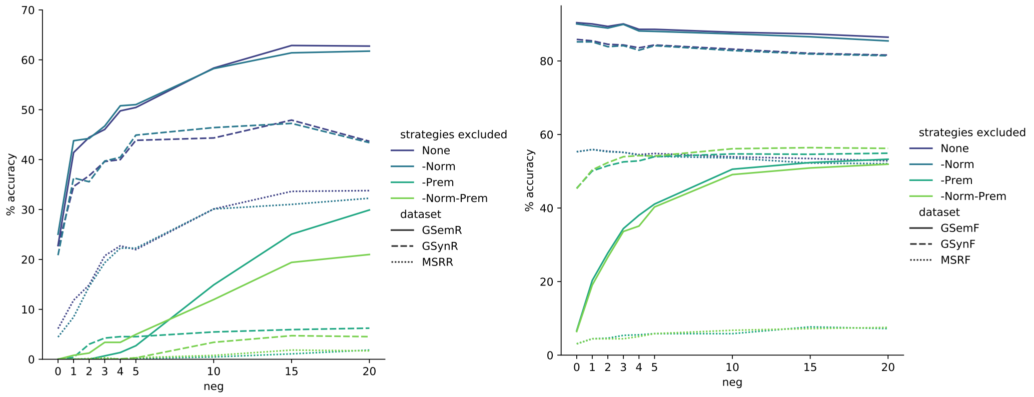


Table C.8 – Same as Table 6.2, but using **subword information**, positional contexts and symmetric context window of fixed size 2.

word	model	neighbors
interest $_{0.3}$	SLV-0	pinterest $_{29.2}$, interests $_{1.0}$, interested $_{0.7}$, disinterest $_{26.1}$, conflict $_{0.5}$, stuffofinterest $_{25.6}$, interesse $_{66.5}$, bitterest $_{51.3}$, merest $_{63.5}$, chemicalinterest $_{23.2}$
	SLV-20	interests $_{1.0}$, interested $_{0.7}$, conflict $_{0.5}$, attention $_{0.5}$, expertise $_{2.3}$, involvement $_{1.2}$, enthusiasm $_{3.9}$, disinterest $_{26.1}$, concern $_{0.9}$, appreciation $_{3.5}$
	FT	interests $_{1.0}$, disinterest $_{26.1}$, interested $_{0.7}$, pinterest $_{29.2}$, enthusiasm $_{3.9}$, fascination $_{7.0}$, attention $_{0.5}$, enthusiasms $_{56.1}$, disinterestedness $_{98.5}$, intrest $_{50.0}$
	GloVe	interests $_{1.0}$, interested $_{0.7}$, conflict $_{0.5}$, attention $_{0.5}$, influence $_{0.6}$, concern $_{0.9}$, expertise $_{2.3}$, involvement $_{1.2}$, popularity $_{1.2}$, passion $_{2.0}$
	SVD	interests $_{1.0}$, attention $_{0.5}$, interested $_{0.7}$, importance $_{0.4}$, profits $_{2.9}$, debt $_{1.6}$, expertise $_{2.3}$, benefit $_{1.0}$, contributions $_{0.5}$, contribution $_{1.0}$
cup $_{0.1}$	SLV-0	cups $_{2.7}$, championship $_{0.2}$, champions $_{0.5}$, cupfb $_{64.7}$, trophy $_{1.1}$, cupen $_{17.1}$, cupa $_{23.7}$, uefa $_{1.3}$, league $_{0.1}$, championships $_{0.3}$
	SLV-20	cups $_{2.7}$, champions $_{0.5}$, championship $_{0.2}$, finals $_{0.7}$, trophy $_{1.1}$, league $_{0.1}$, competitions $_{0.9}$, qualifiers $_{4.7}$, tournament $_{0.4}$, runners $_{1.5}$
	FT	cups $_{2.7}$, championship $_{0.2}$, supercups $_{45.3}$, supercup $_{8.5}$, trophy $_{1.1}$, champions $_{0.5}$, finals $_{0.7}$, supercupen $_{55.0}$, uhrencup $_{90.3}$, pokal $_{9.4}$
	GloVe	championship $_{0.2}$, champions $_{0.5}$, finals $_{0.7}$, cups $_{2.7}$, league $_{0.1}$, trophy $_{1.1}$, uefa $_{1.3}$, tournament $_{0.4}$, final $_{0.1}$, fifa $_{1.1}$
	SVD	runners $_{1.5}$, nextseason $_{4.9}$, champions $_{0.5}$, cups $_{2.7}$, trophy $_{1.1}$, scorers $_{4.7}$, competitions $_{0.9}$, matches $_{0.4}$, squad $_{0.7}$, fifa $_{1.1}$
soul $_{0.9}$	SLV-0	souls $_{3.0}$, soule $_{16.8}$, souli $_{48.4}$, nsoul $_{52.5}$, sould $_{40.0}$, souled $_{37.4}$, soulchild $_{51.7}$, soulive $_{72.5}$, soulful $_{11.0}$, soult $_{18.6}$
	SLV-20	funk $_{2.7}$, blues $_{0.8}$, souls $_{3.0}$, heaven $_{1.5}$, essence $_{3.0}$, spirit $_{0.8}$, mind $_{0.5}$, love $_{0.2}$, jazz $_{0.6}$, reggae $_{3.4}$
	FT	souls $_{3.0}$, funk $_{2.7}$, soulchild $_{51.7}$, soulful $_{11.0}$, blues $_{0.8}$, temptations $_{9.3}$, soulfulness $_{97.6}$, salsoul $_{42.7}$, soulmates $_{48.9}$, reggae $_{3.4}$
	GloVe	blues $_{0.8}$, funk $_{2.7}$, souls $_{3.0}$, spirit $_{0.8}$, mind $_{0.5}$, gospel $_{1.6}$, hop $_{1.1}$, pop $_{0.4}$, jazz $_{0.6}$, love $_{0.2}$
	SVD	heaven $_{1.5}$, eternal $_{2.5}$, forever $_{1.7}$, dreams $_{1.6}$, dream $_{0.9}$, funk $_{2.7}$, love $_{0.2}$, souls $_{3.0}$, spirit $_{0.8}$, blues $_{0.8}$

Table C.9 – Same as Table 6.3, but using **subword information**, positional contexts and symmetric context window of fixed size 2.

word	model	neighbors
rooters $_{77}$	SLV-0	cooters $_{97}$, looters $_{21}$, booters $_{75}$, rooter $_{61}$, hooters $_{18}$, footers $_{25}$, scooters $_{15}$, freebooters $_{74}$, rootes $_{24}$, troubleshooters $_{55}$
	SLV-20	rooter $_{61}$, cooters $_{97}$, booters $_{75}$, looters $_{21}$, truckers $_{16}$, revellers $_{44}$, stoners $_{46}$, greasers $_{63}$, tuckers $_{72}$, stokers $_{40}$
	FT	rooter $_{61}$, cooters $_{97}$, hooters $_{18}$, booters $_{75}$, looters $_{21}$, footers $_{25}$, rootes $_{24}$, crosscutters $_{63}$, roaders $_{91}$, freebooters $_{74}$
	GloVe	specifc $_{84}$, unserious $_{99}$, cusumano $_{87}$, whch $_{65}$, uninviting $_{91}$, overexcited $_{99}$, imnsho $_{89}$, preffered $_{91}$, alread $_{94}$, untypical $_{94}$
	SVD	ebbets $_{29}$, comiskey $_{18}$, ballplayers $_{31}$, batboy $_{62}$, crawfords $_{37}$, chisox $_{83}$, bosox $_{81}$, gothams $_{59}$, semipro $_{60}$, krichell $_{79}$
monocultures $_{76}$	SLV-0	monoculture $_{35}$, monocular $_{38}$, protoculture $_{61}$, uncultured $_{51}$, ethnocultural $_{59}$, monocarpic $_{97}$, monocoupe $_{73}$, cultureel $_{99}$, culturing $_{34}$, polyculture $_{88}$
	SLV-20	monoculture $_{35}$, polyculture $_{88}$, cultivations $_{64}$, crops $_{2}$, intercropping $_{90}$, agroforestry $_{34}$, cultivation $_{3}$, silviculture $_{45}$, seedlings $_{11}$, clearcutting $_{47}$
	FT	monoculture $_{35}$, polyculture $_{88}$, ethnocultural $_{59}$, silviculture $_{45}$, cultivations $_{64}$, crops $_{2}$, ecotypes $_{58}$, intercropping $_{90}$, cultures $_{1}$, overgrazing $_{26}$
	GloVe	hereabouts $_{74}$, upend $_{97}$, controvertial $_{87}$, enlivening $_{79}$, pluralisation $_{92}$, selfsame $_{98}$, herrod $_{85}$, overspend $_{83}$, unserious $_{99}$, liquify $_{89}$
	SVD	monoculture $_{35}$, cropland $_{27}$, windbreaks $_{68}$, replanted $_{27}$, orchards $_{6}$, silviculture $_{45}$, plantations $_{3}$, intercropping $_{90}$, cultivations $_{64}$, seedlings $_{11}$
flighted $_{83}$	SLV-0	alighted $_{52}$, flighty $_{43}$, lighted $_{9}$, slighted $_{25}$, flightplan $_{93}$, unlighted $_{83}$, benighted $_{57}$, flightaware $_{86}$, flightdeck $_{80}$, blighted $_{20}$
	SLV-20	flightdeck $_{80}$, flights $_{1}$, flight $_{0}$, flighty $_{43}$, flightless $_{14}$, taxiing $_{19}$, taxied $_{61}$, flightaware $_{86}$, flown $_{2}$, flightline $_{55}$
	FT	flight $_{0}$, flights $_{1}$, flighty $_{43}$, flightdeck $_{80}$, flightless $_{14}$, flightpath $_{67}$, flown $_{2}$, taxiing $_{19}$, alighted $_{52}$, taxied $_{61}$
	GloVe	wikispeak $_{87}$, similary $_{82}$, vandelism $_{97}$, specifc $_{84}$, validations $_{76}$, demagogic $_{77}$, imnsho $_{89}$, incentivise $_{100}$, smidge $_{97}$, geneological $_{98}$
	SVD	flightless $_{14}$, shoebill $_{71}$, tinamous $_{38}$, corvid $_{76}$, curassows $_{78}$, pratincoles $_{45}$, toucans $_{40}$, turacos $_{56}$, hoatzin $_{65}$, anseriformes $_{34}$

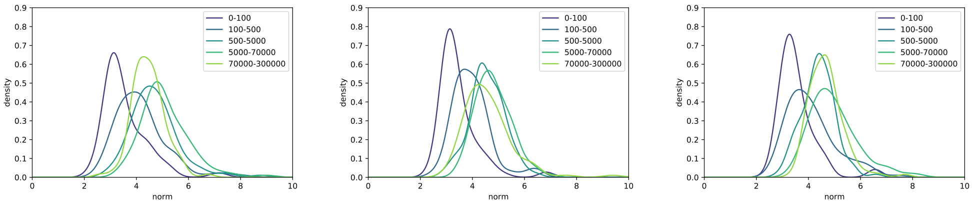
Table C.10 – Same as Table 6.4, but using **subword information**, positional contexts and symmetric context window of fixed size 2.

model	SimLex	RW	GSem	GSem$^{\text{-N}}$	GSem$^{\text{-P}}$	GSem$^{\text{-N-P}}$
SLV-0	38.8	47.7	40.2	39.7	0.4	0.7
SLV-5	37.3	49.8	77.1	75.9	9.7	9.6
SLV-20	35.1	48.3	**79.5**	**78.8**	**40.2**	**35.7**
FT-1	38.7	51.2	70.4	70.1	18.8	18.7
FT-2	39.4	51.7	74.3	74.1	21.3	21.4
FT-5	**40.3**	**52.1**	78.8	78.5	22.1	21.2
GloVe	35.2	36.1	74.8	73.0	3.0	3.5
SVD	31.6	44.0	48.5	41.1	30.9	26.8

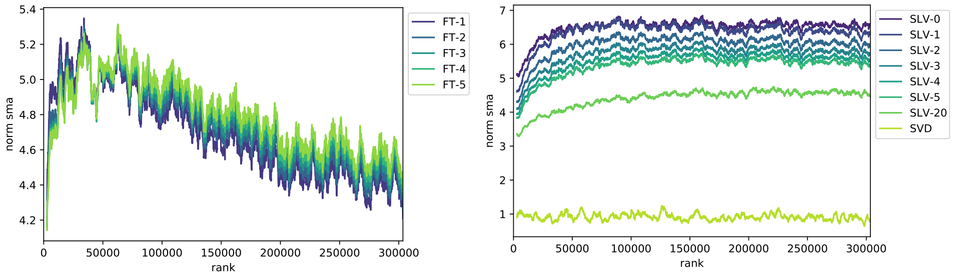
model	GSyn	GSyn$^{\text{-N}}$	GSyn$^{\text{-P}}$	GSyn$^{\text{-N-P}}$	MSR	MSR$^{\text{-N}}$	MSR$^{\text{-P}}$	MSR$^{\text{-N-P}}$
SLV-0	70.1	68.4	9.8	10.7	60.9	59.5	5.5	5.9
SLV-5	71.7	70.6	16.4	17.8	59.9	58.7	9.1	9.3
SLV-20	66.3	65.6	**18.8**	**19.8**	55.5	54.7	11.7	11.6
FT-1	**74.5**	**74.0**	17.8	19.3	61.2	60.2	**12.6**	**12.9**
FT-2	74.0	73.6	17.6	19.1	61.6	60.6	11.5	12.1
FT-5	74.2	73.6	17.4	19.0	**62.2**	**61.3**	11.2	11.7
GloVe	59.2	58.2	9.7	9.7	47.5	45.7	2.8	3.2
SVD	49.3	46.5	17.1	15.8	37.3	31.5	9.9	8.5

Figure C.12 – Same as Figure 6.5, but using **subword information**, positional contexts and symmetric context window of fixed size 2.

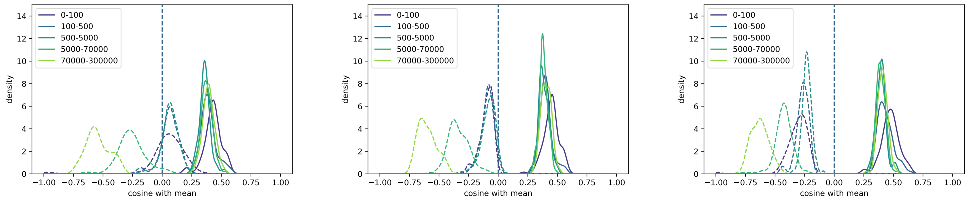
(a) FT-1 norms (b) FT-2 norms (c) FT-5 norms



(d) fastText norms SMA (e) Subword LexVec norms SMA



(f) FT-1 cosines with mean (g) FT-2 cosines with mean (h) FT-5 cosines with mean


density
cosine with mean