October 2022

# Modeling the Multi-mode Distribution in Self-Supervised Language Models

Haw-Shiuan Chang
*University of Massachusetts Amherst*

# MODELING THE MULTI-MODE DISTRIBUTION IN SELF-SUPERVISED LANGUAGE MODELS

A Dissertation Presented

by

HAW-SHIUAN CHANG

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2022

Robert and Donna Manning College of
Information and Computer Sciences

# MODELING THE MULTI-MODE DISTRIBUTION IN SELF-SUPERVISED LANGUAGE MODELS

A Dissertation Presented

by

HAW-SHIUAN CHANG

Approved as to style and content by:

Andrew McCallum, Chair

Mohit Iyyer, Member

Hamed Zamani, Member

Weibo Gong, Member

Zihang Dai, Member

James Allan, Chair of the Faculty
Robert and Donna Manning College of
Information and Computer Sciences

# DEDICATION

*To my wife, for believing in me.*

*To all my family members, for supporting me.*

*To Andrew, my Ph.D. advisor, for allowing me to pursue this research direction.*

# ACKNOWLEDGEMENT

I am really fortunate to have Andrew McCallum as my Ph.D. advisor. He values long-term and fundamental research and encourages his students to think big. Andrew believes in my potential and gives me lots of autonomy in my Ph.D. journey, while always providing very insightful guidance. Even if he disagrees with my arguments, he always respects my viewpoints and gives me sufficient time and resources to justify my perspective. He really cares about the development of his students and empathizes with the difficulties we face in our lives and in the COVID pandemic. Completing this thesis won't be possible without his full support. He is a role model that would continue guiding me to become a respected and considerate advisor for the rest of my life.

I am thankful to all the members of IESL (Andrew's lab) and UMass NLP. Everyone is super smart and nice. Their amazing research achievements teach me what a fundamental and impactful research problem looks like. I want to especially thank Nicholas Monath for his selfless help, Luke Vilnis for his support, Michael Boratko for his mathematical guidance, Jay Yoon Lee for his insightful suggestions, and Nader Akoury for your generous help. Thank you for making the journey so enjoyable. I would miss every joke, smile, and whiteboard discussion in the lab. I would also like to thank the kindness from the whole Amherst community in our everyday life.

I appreciate the opportunities to work with many amazing master's and undergraduate students, especially Ao Liu, Abdurrahman Munir, Johnny Tian-Zheng Wei, Aaron Traylor, Yang Jiao, ZiYun Wang, Vikram Pawar, Amol Agrawal, Ananya Ganesh, Rheeya Uppaal, Jiaming Yuan, Nikhil Agarwal, Alolika Gon, Hieu Phan, Purujit Goyal, Rohan Paul, Ruei-Yao Sun, Ronald Seoh, and Zonghai Yao. Thank you for believing in my research vision.

Without your help, I cannot show the effectiveness of multiple embeddings in so many applications.

I appreciate the help from the senior collaborators such as Erik Learned-Miller and Mohit Iyyer during my Ph.D., and I am grateful to the help from my thesis committee members (Andrew McCallum, Mohit Iyyer, Hamed Zamani, Weibo Gong, and Zihang Dai). Thank you for your time and valuable advice on our work. I also learned a lot from the advisors of my internship, including Shankar Vembu, Sunil Mohan, Xin Luna Dong, and Andrey Kan. Thank you for your time and effort.

I also want to thank the professors/researchers who help me to start the Ph.D. journey. Thank you, I-Chen Wu, for supervising my first research project, Yu-Chiang Frank Wang, for teaching me how to do research and write research papers, Kuan-Ta Chen, for giving me the courage to pursue my dream, Beverly P. Woolf, and Brendan O'Connor, for helping me to find my Ph.D. advisor.

Finally, I would like to thank my family: My beloved wife, son, and daughter, for their endless loving and caring. My mom and dad, for raising me in an environment that nurtures my dream. My father-in-law and mother-in-law, for their support, especially when we needed it the most.

# ABSTRACT

## MODELING THE MULTI-MODE DISTRIBUTION IN SELF-SUPERVISED LANGUAGE MODELS

SEPTEMBER 2022

HAW-SHIUAN CHANG

B.S., NATIONAL CHIAO TUNG UNIVERSITY

M.S., UNIVERSITY OF MASSACHUSETTS, AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew McCallum

Self-supervised large language models (LMs) have become a highly-influential and foundational tool for many NLP models. For this reason, their expressivity is an important topic of study. In near-universal practice, given the language context, the model predicts a word from the vocabulary using a single embedded vector representation of both context and dictionary entries. Note that the context sometimes implies that the distribution over predicted words should be multi-modal in embedded space. However, the context's single-vector representation provably fails to capture such a distribution. To address this limitation, we propose to represent context with multiple vector embeddings, which we term facets. This is distinct from previous work on multi-sense vocabulary embeddings, which employs multiple vectors for the dictionary entries, not the context.

In this dissertation, we first present the theoretical limitations of the single context embedding in LMs and how the theoretical analyses suggest new alternative softmax layers

that encode a context as multiple embeddings. The proposed alternatives achieve better perplexity than the mixture of softmax (MoS), especially given an ambiguous context, without adding significant computational cost to LMs. Our approaches also let GPT-2 learn to properly copy the entities from the context, which increases the coherence of the generated text without requiring any labels.

In addition to predicting the next word, we also use multiple CLS embeddings to improve state-of-the-art pretraining methods for BERT on natural language understanding (NLU) benchmarks without introducing significant extra parameters or computations, especially when the training datasets are small. Furthermore, we show that our multi-facet embeddings improve the sequential recommendation, scientific paper embeddings, measurement of sentence similarity, distantly supervised relation extraction, unsupervised text pattern entailment detection, and cold-start citation recommendation. Finally, we use the multiple vector embeddings to predict the future topics of a context, and build on the basis, we propose a novel interactive language generation framework.

# TABLE OF CONTENTS

**Page**

**ACKNOWLEDGEMENT** . . . . . **v**

**ABSTRACT** . . . . . **vii**

**LIST OF TABLES** . . . . . **xiv**

**LIST OF FIGURES** . . . . . **xx**

**CHAPTER**

**1. INTRODUCTION** . . . . . **1**

1.1 Relation to Previous Work and Our Contributions . . . . . 3
1.2 Thesis Goal . . . . . 7
1.3 Thesis Outline . . . . . 8
1.4 Declaration of Collaborations . . . . . 12

**2. BREAKING THE SOFTMAX BOTTLENECK FOR LANGUAGE GENERATION** . . . . . **14**

2.1 Introduction . . . . . 14
2.2 Theoretical Limitations of the Single Embedding in the Softmax Layer and Empirical Analyses . . . . . 18

2.2.1 Background . . . . . 18
2.2.2 Structural Weakness Theorems from Linear Dependency . . . . . 19
2.2.3 Measuring Linear Dependency among Words . . . . . 20
2.2.4 Softmax Bottleneck in GPT-3 . . . . . 22

2.3 Multi-facet Softmax . . . . . 24

2.3.1 Method . . . . . 24

2.3.1.1 Mixture of Softmax . . . . . 25
2.3.1.2 Multiple Input Hidden States . . . . . 25

2.3.1.3 Multiple Partitions . . . . . 26
2.3.2 Language Modeling Experiments . . . . . 27
2.3.2.1 Baselines . . . . . 28
2.3.2.2 Results . . . . . 28
2.4 Dynamic Partitioning . . . . . 30
2.4.1 Method . . . . . 31
2.4.1.1 Context Partition . . . . . 32
2.4.1.2 Reranker Partition . . . . . 33
2.4.1.3 Two-stage Reranker . . . . . 35
2.4.1.4 Local Context Word Embedding . . . . . 35
2.4.1.5 Hybrid Approach . . . . . 36
2.4.2 Experiments . . . . . 37
2.5 Applications on Sequential Recommendation . . . . . 40
2.5.1 Experiment Setup . . . . . 41
2.5.2 Results . . . . . 42
2.6 Related Work . . . . . 44
2.7 Chapter Conclusion . . . . . 46
**3. IMPROVEMENT UNSUPERVISED SIMILARITY ESTIMATION BY PREDICTING CLUSTER CENTERS . . . . . 48**
3.1 Introduction . . . . . 48
3.2 Applications on the Representation of Sentences and Phrases . . . . . 51
3.2.1 Method . . . . . 52
3.2.1.1 Self-supervision Signal . . . . . 52
3.2.1.2 Non-negative Sparse Coding Loss . . . . . 53
3.2.1.3 Sequence to Embeddings . . . . . 55
3.2.2 Experiments . . . . . 56
3.2.2.1 Experiment Setup . . . . . 57
3.2.2.2 Qualitative Evaluation . . . . . 57
3.2.2.3 Unsupervised Sentence Similarity . . . . . 58
3.2.2.4 Unsupervised Extractive Summarization . . . . . 62
3.2.2.5 Unsupervised Phrase Similarity . . . . . 64

3.2.3 Related Work . . . . . 66

3.3 Applications on Relation Extraction . . . . . 67

3.3.1 Method . . . . . 69

3.3.1.1 Background and Problem Setup . . . . . 70
3.3.1.2 Objective Function . . . . . 71
3.3.1.3 Scoring Functions . . . . . 72

3.3.2 Experiments . . . . . 73

3.3.2.1 Relation Extraction . . . . . 74
3.3.2.2 Entailment Detection . . . . . 76

3.3.3 Related Work . . . . . 78

3.4 Applications on Citation and Authorship Prediction . . . . . 79

3.4.1 Method . . . . . 80

3.4.1.1 Background and Problem Formulation . . . . . 80
3.4.1.2 Authorship and Citation Prediction . . . . . 81

3.4.2 Neural Architecture . . . . . 84
3.4.3 Experiments . . . . . 85

3.4.3.1 Evaluation Setup . . . . . 85
3.4.3.2 Comparing Methods . . . . . 85
3.4.3.3 Metrics . . . . . 86
3.4.3.4 Results and Discussion . . . . . 87

3.4.4 Related Work . . . . . 88

3.5 Chapter Conclusion . . . . . 89

**4. INTERACTIVE LANGUAGE GENERATION . . . . . 90**

4.1 Introduction . . . . . 90
4.2 Method . . . . . 93

4.2.1 Option Generator . . . . . 93

4.2.1.1 Model Prediction . . . . . 93
4.2.1.2 Model Training . . . . . 96

4.2.2 Conditional Text Generator . . . . . 96

4.2.2.1 Model Prediction . . . . . 97
4.2.2.2 Model Training . . . . . 97

4.3 Experiments . . . . . 99

4.3.1 Datasets . . . . . 99
4.3.2 Option Generator Evaluation . . . . . 99

4.3.2.1 Automatic Evaluation Metrics . . . . . 100
4.3.2.2 Human Evaluation . . . . . 100
4.3.2.3 Option Generator Baselines . . . . . 101
4.3.2.4 Results . . . . . 101

4.3.3 Conditional Text Generator Evaluation . . . . . 103

4.3.3.1 Automatic Evaluation Metrics . . . . . 103
4.3.3.2 Human Evaluation . . . . . 104
4.3.3.3 Conditional Text Generator Baselines . . . . . 104
4.3.3.4 Results . . . . . 105

4.3.4 Option Generator Comparison Using Generated Continuations . . . . . 105

4.3.4.1 Automatic Evaluation Metrics . . . . . 105
4.3.4.2 Human Evaluation . . . . . 106
4.3.4.3 Results . . . . . 107

4.4 Related Work . . . . . 107
4.5 Chapter Conclusion . . . . . 108

**5. CONTRASTIVE LEARNING ON TWO-TOWER MODELS . . . . . 109**

5.1 Introduction . . . . . 109
5.2 Applications on Natural Language Understanding Benchmarks . . . . . 110

5.2.1 Method . . . . . 112

5.2.1.1 Multi-task Pretraining . . . . . 113
5.2.1.2 Quick Thoughts Loss . . . . . 113
5.2.1.3 Multiple CLS Embeddings . . . . . 114
5.2.1.4 Hard Negative . . . . . 115
5.2.1.5 Architecture-based Diversification . . . . . 116
5.2.1.6 Fine-Tuning . . . . . 117

5.2.2 Experiments . . . . . 118

5.2.2.1 Experiment Setup . . . . . 118
5.2.2.2 Main Results . . . . . 120

5.2.2.3 Ablation Study . . . . . 120
5.2.2.4 Ensembling Analysis . . . . . 122

5.2.3 Related Work . . . . . 124

5.3 Applications on Scientific Paper Representation Benchmarks . . . . . 125

5.3.1 Multi$^2$SPE: Multi-Domain $\times$ Multi-CLS Scientific Paper Encoder . . . . . 127

5.3.1.1 Multiple CLS Encoder . . . . . 127
5.3.1.2 Contrastive Citation Prediction Loss . . . . . 128
5.3.1.3 Measuring Document Similarity with Multiple Embeddings . . . . . 129

5.3.2 Multi-SciDocs . . . . . 129
5.3.3 Experiments and Analyses . . . . . 130

5.3.3.1 Results . . . . . 131
5.3.3.2 Ablation Studies . . . . . 131

5.3.4 Related Work . . . . . 132

5.4 Chapter Conclusion . . . . . 133

**6. CONCLUSION AND FUTURE WORK . . . . . 134**

6.1 Take Home Messages . . . . . 134
6.2 Limitations and Future Work . . . . . 135

**BIBLIOGRAPHY . . . . . 139**

**APPENDICES**

**A. APPENDIX FOR CHAPTER 2 . . . . . 172**
**B. APPENDIX FOR CHAPTER 3 . . . . . 181**
**C. APPENDIX FOR CHAPTER 4 . . . . . 185**
**D. APPENDIX FOR CHAPTER 5 . . . . . 187**

# LIST OF TABLES

**Table** **Page**

1.1 Comparison to the previous work that demonstrates the effectiveness of multiple embeddings. In each row, we only cite one representative paper. Predicting embeddings means using a neural encoder rather than a lookup table to generate the multiple embeddings from an input sequence. Self-supervision means that the models do not require labels annotated by humans. *The applications we include in this thesis handle the text input except for the sequential recommendation in Section 2.5. . . . . 3

1.2 Task summarization. The input specifies the data we required to train the model. We focus on the self-supervised pretraining, but would also show the benefits of multiple embeddings after using the labels to fine-tune the pre-trained models. Please refer to Figure 1.1 to see the meaning of *embedding(s) 1* (E1) and *embedding(s) 2* (E2). Each text sequence in Chapter 2 usually only co-occurs with one item (e.g., a word), while each text sequence in Chapter 3 and 4 usually co-occurs with multiple items (e.g., words). In Chapter 5, a text sequence co-occurs with other similar text sequences. . . . . 6

1.3 Method summarization. Chapter 2 trains the models using cross entropy and softmax, a kind of one-tower co-occurrence learning. Chapter 3 and 4 train the models using the one-tower co-occurrence learning with negative sampling. Chapter 5 studies the models using the two-tower co-occurrence learning framework (i.e., contrastive learning). . . . . 7

1.4 Representative empirical result summary. The improvement ratio is the improvement divided by the baseline score. Please refer to the corresponding sections to see the setup of the experiments. . . . . 8

1.5 The contributions of other students to the chapters of this thesis. . . . . 12

2.1 Perplexity comparison between MFS (Ours) and baselines. #S, #I, #P are the number of softmaxes (i.e., $K$), input hidden states, and partitions, respectively. The top four baselines use a single softmax. OWT and Wiki are the test set perplexity of OpenWebText and Wikipedia 2021, respectively. The standard errors of all models are smaller than 0.02 perplexity. We also compare the number of parameters and the inference time on one batch. . . . . 29

2.2 Perplexity of the *GPT-2 Small* in OpenWebText. The percentages of the perplexity reduction compared to Softmax are presented in the parentheses. . . . . 29

2.3 Prediction visualization using a context in each dataset. We show the top three words with the highest prediction probabilities of each method. In the last three rows, we visualize the outputs of the softmax grey boxes in Figure 2.4 (d), which model different modes of the next word distribution. The prediction target is boldfaced in the context and the predictions. ## indicates there is no space before the word. . . . . 30

2.4 Comparison of perplexity (PPL) in a validation set of Wikipedia 2021 with 100k tokens. All models are built on GPT-2 Small, including CopyNet, Pointer Generator, and Pointer Sentinel. *R:100* means the reranker partition using $n = 100$. *Rs2:100* means the 2nd stage reranker uses $n = 100$. *R:20,100,500* means three layers of the ranker partition whose $n_1$, $n_2$, and $n_3$ values are 20,100, and 500, respectively. *C* means a context partition. *L* means a local context embedding. *Mi* means multiple input hiddent state (i.e., *#I > 1*). That is, *Softmax + Mi* refers to *Softmax + Multi-input*. . . . . 38

2.5 Comparison of ROUGE 1 F1 between the generated text and the context or continuation. We present the ROUGE scores of all tokens, uppercased tokens, and proper nouns. *Multi-softmax* is the same as *MoS* and *Multi-softmax + Mi* is the same as *MFS w/o Multi-partition*. . . . . 39

2.6 Prediction visualization of three input contexts. We show the top five words with the highest prediction probabilities of each model. The reasonable next word predictions are boldfaced. . . . . 40

2.7 The validation scores of the four datasets and their geometric average (*G mean*). In all the model names with *R*, we use 3 reranker partitions with $n_1 = 20$, $n_2 = 100$, $n_3 = 500$. Due to the memory constraints of our GPU, we haven't had the results of *Multi-Softmax + CLR* at *Amazon-beauty*. Other notations are the same as the notations used in Table 2.4. . . . . 43

3.1 Examples of the codebook embeddings predicted by our models with different $K$. The embedding in each row is visualized by the three words whose GloVe embeddings have the highest cosine similarities (also presented) with the codebook embedding. . . . . 58

3.2 Pearson correlation (%) in the development and test sets in the STS benchmark. The performance of all sentence pairs is indicated as All. Low means the performance on the half of sentence pairs with lower similarity (i.e., STSB Low). Our c means our codebook embeddings and Our a means our attention vectors. * indicates a supervised method. † indicates the methods which use training distribution to approximate testing distribution. The best score with and without † are highlighted. . . . . 61

3.3 The ROUGE F1 scores of different methods on CNN/Daily Mail dataset. The results with † are taken from Zheng and Lapata [260]. The results with * are taken from Celikyilmaz et al. [26]. . . . . 64

3.4 Performance of phrase similarity tasks. Every model is trained on a lowercased corpus. In SemEval 2013, AUC (%) is the area under the precision-recall curve of classifying similar phrase pairs. In Turney, we report the accuracy (%) of predicting the correct similar phrase pair among 5 or 10 candidate pairs. The results with † are taken from Yu and Dredze [252]. . . . . 65

3.5 Distantly supervised relation extraction using different versions of the universal schema. All numbers are %. CUSchema refers to compositional universal schema. Trans is an abbreviation of transformer. The best scores of the single models and ensemble models are highlighted. *The performance of TAC 2013 and 2014 is copied from Verga et al. [225]. . . . . 75

3.6 Example of sentence pattern pairs, its label, and our predictions in our entailment experiment. **Ours** and **Ours Diff** are the predictions from **Ours (Trans)**. **Freq Diff** is the frequency difference baseline. . . . . 76

3.7 Comparison of entailment detection methods. AP and Acc are average precision and accuracy, respectively. All numbers are %. Our methods use a transformer as their encoder. . . . . 78

3.8 Results for cold-start authorship and citation prediction in our test set. AUC is the area under the ROC curve. We highlight the best values of cold-start recommendation methods with and without using weighted average of CBOW. K=1 (no auto) is an extension from Bansal et al. [14]. . . . . 87

4.1 Comparison of the option generators using automatic metrics. The best numbers within each scope are highlighted. . . . . 102

4.2 Comparison of option generators using human judgment (mean $\pm$ standard error). L and TP refer to likelihood and topic promotion, respectively. . . . . 102

4.3 Comparison of all $K$ topics for the input prompt using $M = 2$ words closest to each topic. . . . . 102

4.4 The continuations that are generated by conditioning on all of $K$ topics from different option generators. The input prompt comes from STSb. . . . . 102

4.5 Comparison of conditional text generators. The numbers in Dist-1, Dist-2, Recall, and Precision are percentages. Lower perplexity (PPL) and inference time are better. The better performance between PPLM and our method is highlighted. In human evaluation, we report the mean $\pm$ standard error of each method. . . . . 104

4.6 Comparison of the continuations generated by different option generators using automatic metrics. The values are percentages except in Word Hit. Higher numbers are better except in Self-BLEU. The best numbers within each scope are highlighted. . . . . 106

4.7 Comparison of the continuations generated by different option generators using human judgment (mean $\pm$ standard error). F, NP, and A refer to fluency, narrative promotion, and overall, respectively. . . . . 107

5.1 The macro average scores on the development set. All numbers are percentages. The standard errors are shown as the confidence intervals. We make the best scores of the model built on $\text{BERT}_{\text{Base}}$ boldface and similar for the models built on $\text{BERT}_{\text{Large}}$. †The number is much higher than $81.4$, the GLUE score reported by Aroca-Ouellette and Rudzicz [6] because we continue training from the pretrained BERT and we use better fine-tuning hyperparameters. *The scores do not contain ReCoRD in SuperGLUE.[1] . . . . . 119

5.2 The macro average scores on the development set for our ablation study. We highlight the best performance after excluding the ensemble baselines, which require much more computation. The scores are different in Table 5.1 because we use two pretraining random seeds instead of four in the ablation study. SWA refers to Stochastic weight averaging [92]. *SuperGLUE score does not contain ReCoRD. . . . . 122

5.3 The comparison of inference time and expected calibration error (ECE). The confidence intervals are standard errors. *Only includes the classification tasks (i.e., excludes STS-b). . . . . 123

5.4 The overlapping ratio of the top 20% most uncertain examples using different uncertainty estimation methods. ENS is ensemble of Ours (K=5, $\lambda = 0.1$) with different fine-tuning seeds. *Only includes the classification tasks (i.e., excludes STS-b). . . . . 123

5.5 Results of our methods and baselines on Multi-SciDocs. All scores are averaged over four random seeds. We show standard errors as their confidence interval. Percentages indicate relative error reduction over the baselines (SPECTER or SciNCL). . . . . 130

5.6 Ablation studies conducted on SPECTER and multiple domain training data. All scores are averaged over four random seeds. Percentages indicate relative error reduction over the baseline (3 CLS, $\lambda = 0.1$). . . . . 132

A.1 Perplexity comparison of different *GPT-2 Small* models on the words with different types of analogy relations. The validation set (valid) includes all four types of relations. . . . . 173

A.2 Prediction visualization using a context in each dataset. Each row visualizes a model as in Table 2.3. The models are built on *GPT-2 Medium* in OpenWebText and Wikipedia and on *GPT-2 Small* in the synthesized dataset. *MFS Avg* shows the words that are closest to the average facet embedding in *MFS*. See the details in Appendix A.2.3. We underline the words that appear in the top predictions of both *MFS* and *MFS Avg*. . . . . 174

A.3 The loss improvement comparison between the *Improvement Models* and *Reference Models*. The models are named using their number of softmaxes, input hidden states, and partitions. Thus, S3I9P4 is *MFS*, S3I9P1 is *MFS w/o Multi-partition*, S1I9P1 is *Softmax + Multi-input*, S3I1P1 is *MoS (3)*, and S1I1P1 is *Softmax*. *Multi-mode Percentage* is the percentage of the contexts where the *Improvement Models* output multimodal distribution. *Multi-mode Loss Improvement* refers to the average improvement when *Improvement Models* outputs multimodal distribution and *Other Loss Improvement* refers to the improvement of the contexts where the facets of *Improvement Models* are close to each other. *Improvement Ratio* divides *Multi-mode Loss Improvement* by *Other Loss Improvement*. . . . . 175

D.1 Training and testing dataset statistics. Note: Since some papers are categorized under multiple MAG fields in S2ORC, they are counted more than once in this table. Unknown refers to the papers without MAG information in S2ORC. . . . . 192

# LIST OF FIGURES

**Figure** **Page**


1.1 Illustration of the differences between the single embedding and the proposed multiple embeddings in the two kinds of co-occurrence learning. For the models studied in this thesis, the embedding(s) 1 (E1) always come from a text encoder (a tower), while the embedding(s) 2 (E2) could come from a static item embedding table/matrix or also from the text encoder (the second tower). Dot or L2 refers to dot products or Euclidean distance. In Chapter 2, 3, and 4, we study the one-tower co-occurrence learning. In Chapter 5, we study the two-tower co-occurrence learning. . . . . 2

2.1 Comparison between the softmax layers using a single embedding and multiple embeddings when the next word should be either *woman* or *king*. In GPT-2 and multi-embedding GPT-2, the hidden states of the context are visualized by the single facet ◻ and multiple facets ⧫, respectively. The word embeddings are visualized using ●. GPT-2 cannot output *woman* and *king* as the top two words because ***queen*** and ***man*** are close to the middle of ***woman*** and ***king***. The improvement in this type of ambiguous context will be quantified in Appendix A.2.1. . . . . 15

2.2 Minimal eigenvalue ratios of different groups of $N$ word embeddings. Lower ratios indicate that the corresponding word embeddings are more linearly dependent and thus the probabilities of the words cannot be determined arbitrarily by the hidden state of the language models. The number of parameters and the size of hidden states are shown in caption beside every model name. . . . . 21

2.3 The next word probabilities outputted by GPT-3. Notice that this is a raw probability before being modified using the temperature. . . . . 23

2.4 Comparison between different architectures. The #S, #I, and #P are the number of softmaxes, input hidden states, and partitions, respectively. The green boxes refer to embeddings/vectors. The vocab means the embeddings of all words in the vocabulary. $\oplus$ refers to concatenation. $L^h$, $L^f$, and $L^\pi$ are linear projection layers. . . . . 24

2.5 Architecture of dynamic partitioning that computes $\text{Logit}_{CLR}$ in Equation 2.11. The architecture combines the idea of the context partition, reranker partition, and local context word embedding. *Top n - Context* means the embeddings of the top n prediction words that are not in the context. . . . . 32

2.6 Two-stage reranker baseline architecture. This architecture performs similarly compared to the reranker partition approach but requires much more training time. . . . . 34

3.1 The input phrase *real property* is represented by $K = 5$ cluster centers. The previous work discovers the multiple senses by clustering the embedding of observed co-occurring words. Instead, our compositional model learns to predict the embeddings of cluster centers from the sequence of words in the input phrase so as to reconstruct the (unseen) co-occurring distribution well. . . . . 49

3.2 Our model for sentence representation. We represent each sentence as multiple codebook embeddings (i.e., cluster centers) predicted by our sequence to embeddings model. Our loss encourages the model to generate codebook embeddings whose linear combination can well reconstruct the embeddings of co-occurring words (e.g., *music*), while not able to reconstruct the negatively sampled words (i.e., the co-occurring words from other sentences). . . . . 52

3.3 Comparison of our attention weights and the output embeddings between two similar sentences from STSB. A darker red indicates a larger attention value in Equation 3.6 and the output embeddings are visualized using the same way in Table 3.1. . . . . 61

3.4 Comparison between the multi-facet and compositional universal schema. In our training loss, we encourage one of the facet embeddings from a sentence pattern to be similar to its co-occurred entity pair. . . . . 68

3.5 An illustration of the proposed method. The training signal comes from the co-occurrence matrices of the KB and training text corpus on the right. On the lower left, we visualize our neural encoder, which captures the compositional meaning of tokens in the sentence pattern, and our neural decoder, which models the dependency among multiple facet embeddings. When a sentence pattern co-occurs with an entity pair, the training loss minimizes the distance between the entity pair embedding and the closest facet embedding of the sentence pattern (e.g., 0.2 between $s_{i,2}$ and $e_1$). Trainable parameters in our model are highlighted using red borders. On the upper left, we visualize the embedding space to establish the connection between our method and clustering. . . . . 70

3.6 Comparison of the asymmetric similarities. $Asym(\{\tilde{\underline{\boldsymbol{s}}}_{i,k}\}, \{\tilde{\underline{\boldsymbol{s}}}_{j,m}\}) > Asym(\{\tilde{\underline{\boldsymbol{s}}}_{j,m}\}, \{\tilde{\underline{\boldsymbol{s}}}_{i,k}\})$ because the average cosine distance on the left is smaller than that on the right. . . . . 72

3.7 The proposed learning framework. We train the components with red bolder (i.e., $F$, $\boldsymbol{A}$, and $\boldsymbol{C}$) by minimizing the difference between our predictions and the interaction matrices, $\boldsymbol{Y}^{\boldsymbol{a}}$ and $\boldsymbol{Y}^{\boldsymbol{c}}$. During testing, we encode an unseen paper using our neural model into $K$ embeddings ($K = 5$ in this case) and predict the relevancy scores by choosing the best one among the $K$ embeddings. . . . . 80

3.8 The architecture of our neural model and its predicted $K = 5$ paper embeddings when its input is a lowercased robotics paper [49]. All the embeddings of input papers (colored dots), authors (white dots), and citing papers (black dots) are mapped to the same vector space. The top-left purple box outputs paper embedding $\underline{\boldsymbol{p}}_{i,k}$ and the top-right yellow box outputs paper embedding $\underline{\boldsymbol{p}}^{a}_{i,k}$ for authorship prediction and $\underline{\boldsymbol{p}}^{c}_{i,k}$ for citation prediction. We manually tag each paper embedding (e.g., security and surgery) according to the citing papers closest to them for the visualization purpose. . . . . 83

4.1 Given an input prompt, the transformer-based language model (LM) provides $K = 10$ topics that might be mentioned next and each topic is represented by $M = 3$ words. The user could guide the generation process by choosing a subset of topics. . . . . 91

4.2 Examples of our generated options and continuations. We highlight the words in the continuation that are related to the chosen topics or to the specified word. . . . . 91

4.3 Our model architectures for (a) conditional text generator and (b) option generator. During testing, the information flows from the bottom to the top. . . . . 94

4.4 Training our two components using the same sentence. (a) We randomly pick $n = 3$ words in the actual continuation as our conditions for the text generator, and the null labels mean their predicted probabilities are ignored in our loss. (b) We visualize 5 out of $K = 10$ generated topics in a normalized GloVe space. Red words are the ones that appear in the continuation and pull the nearby cluster centers closer during training. . . . . 95

5.1 Comparison of Multi-CLS BERT and the classic BERT ensemble. Multi-CLS BERT only ensembles the multiple CLS embeddings in one BERT encoder rather than ensemble multiple BERT encoders with different parameter weights. . . . . 111

5.2 Our MCQT, SO, MLM, and TFIDF loss, which are a modification of multi-task pretraining proposed in Aroca-Ouellette and Rudzicz [6]. The multi-CLS quick thought (MCQT) loss maximizes the CLS similarities between a sequence (sentences 1 and 2) and the next sequence (sentences 3 and 4) while minimizing the CLS similarities to other random sequences and the sequence after the next one (sentences 5 and 6). Notice that sentence 4 is inputted before sentence 3 because the sentence order is swapped for the SO loss. . . . . 112

5.3 The architecture of Multi-CLS BERT encoder that is built on $\text{BERT}_{\text{Base}}$ model. The different linear layers are applied to the hidden states corresponding to different CLS tokens to increase the diversity of the resulting CLS embeddings. . . . . 116

5.4 An overview of our two-parted solution. 1) Multi$^2$SPE is our modified Multi-CLS BERT model that better utilizes multi-domain citation data through multiple diversified CLS embeddings. 2) Multi-SciDocs is our new benchmark for testing embeddings of scientific papers under multi-domain settings. . . . . 126

5.5 The architecture of Multi$^2$SPE and its similarity measurement during training $\text{S}^{MC}_{\mathcal{P}^A,\mathcal{P}^B}$. . . . . 128

A.1 Illustration of the MFS predictions given the Wikipedia context in the second column of Table A.2. The green circles mean the facet embeddings from MFS. The orange circle is the average of the facet embeddings (**MFS Avg**). The blue circles are the word embeddings that are close to the facet embeddings and **MFS Avg**. The word *project* is highlighted because it is the next word in our ground truth. . . . . 174

A.2 An example for explaining the connection between our Theorem 1 and the theorem from Demeter et al. [45]. . . . . 179

# CHAPTER 1

# INTRODUCTION

Neural language models (LMs) are usually trained by a form of self-supervised co-occurrence learning [105]: After encoding every input sequence into an embedding, we encourage the embeddings of the co-occurred text and item closer with each other while pushing the embeddings of dissimilar text and item farther away. For example, a next word co-occurs with a context in a corpus. When training GPT-2 [177], we encourage the embedding of the context close to the embedding of the next word[1] in the output softmax layer while pushing the context embedding away from the other word embeddings in the vocabulary.

Although largely effective, the co-occurrence learning framework has a fundamental limitation: One text could co-occur with various items but the single embedding of the text might not be able to close to many different embeddings of the co-occurred items simultaneously.

In this thesis, we study how to use multiple embedding representations to overcome the limitation. Different embeddings often capture the different semantic aspects of the input or different modes of the co-occurrence item distribution. Hence, we often call the predicted multiple embeddings *multi-facet embeddings*, where a facet means a mode of co-occurrence distribution or the embedding representing the mode.

In Figure 1.1, we illustrate the high-level architecture differences between the single embedding and the proposed multiple embedding representations. In the one-tower co-

[1]To be more precise, GPT-2 uses word pieces. However, to make the discussion more concise, we would use "word" to refer to word pieces in this thesis unless their differences influence our discussion.

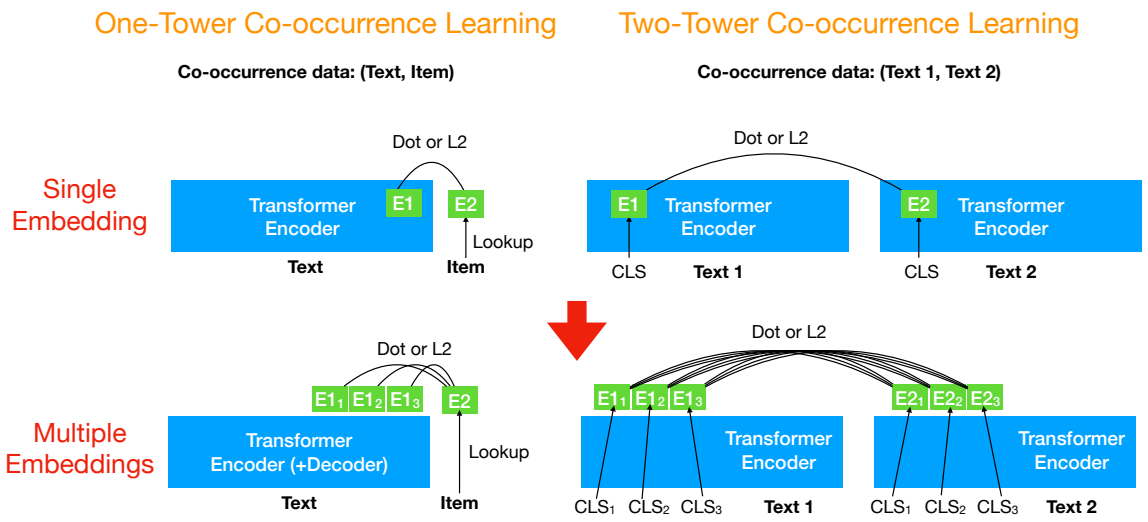



Figure 1.1: Illustration of the differences between the single embedding and the proposed multiple embeddings in the two kinds of co-occurrence learning. For the models studied in this thesis, the embedding(s) 1 (E1) always come from a text encoder (a tower), while the embedding(s) 2 (E2) could come from a static item embedding table/matrix or also from the text encoder (the second tower). Dot or L2 refers to dot products or Euclidean distance. In Chapter 2, 3, and 4, we study the one-tower co-occurrence learning. In Chapter 5, we study the two-tower co-occurrence learning.

occurrence learning framework on the left side, we encode the text into multiple embeddings and maximize their dot products to the co-occurred item embedding. For example, in GPT-2, we maximize the dot products of multiple hidden states and the embedding of the next word. Notice that we only use a single embedding to represent each next word, so the goal of our approach is not modeling the multiple senses of the words as in Miao et al. [147]. Instead, our goal is to model the multiple possibilities of the next word given this text context. In the two-tower co-occurrence learning framework on the right side (e.g., contrastive learning), we are given two related texts (e.g., similar sentences or similar documents) and we maximize the dot products of the multiple text embeddings.

| | Predicting Embeddings for each Input Sequence | Self-Supervision by Co-occurrence | Transformer | Input Text |
|---|---|---|---|---|
| Word Embedding [159] | | V | | V |
| Graph Embedding [53] | | V | | |
| RNN-based LM [248] | V | V | | V |
| LM for IR [134] | V | | V | V |
| LM for Entity Linking [138] | | | V | V |
| LM for RE [48] | | V | V | V |
| Sentence Embeddings [77] | | | | V |
| Style Transfer [140] | V | | V | V |
| Recommendation [108] | V | V | | |
| Object Detection [25] | V | | V | |
| Ours | V | V | V | V* |

Table 1.1: Comparison to the previous work that demonstrates the effectiveness of multiple embeddings. In each row, we only cite one representative paper. Predicting embeddings means using a neural encoder rather than a lookup table to generate the multiple embeddings from an input sequence. Self-supervision means that the models do not require labels annotated by humans. *The applications we include in this thesis handle the text input except for the sequential recommendation in Section 2.5.

## 1.1 Relation to Previous Work and Our Contributions

Multiple embedding representation has been previously proposed to improve several models and applications such as word embedding [159, 9, 147], node embedding in a graph [246, 128, 53], RNN-based LM [248, 68], LM for information retrieval (IR) [100, 134, 106], LM for relation extraction (RE) [48], LM for entity linking [138], sentence embeddings [77], style transfer [140], sequential recommendation model [108, 234, 233, 124], and object detection model [25]. However, this thesis focuses on improving self-supervised LMs that use a transformer to encode the text into multiple embeddings, and most of the previous studies focus on a different setting. See a comparison in Table 1.1.

Narang et al. [158] and Tay et al. [218] recently show that the multiple embedding representation is one of the few modifications of the modern transformer-based LM that are effective in downstream applications. However, most of the state-of-the-art NLP models are built on LMs, and LMs still nearly universally adopt the single embedding representation in the co-occurrence learning framework due to the following concerns of the existing multiple embedding approaches.

- **Insufficient Theoretical Support**: Even though Narang et al. [158] and Tay et al. [218] demonstrate that multiple embeddings such as *Mixture of Softmax* (MoS) [248] could indeed empirically improve LMs, Parthiban et al. [168] show that the current *softmax bottleneck* theory [248] is not sufficient to explain its improvement on the perplexity metric. This shows that we lack the understanding of why multiple embeddings work well, and in what cases, multiple embeddings would be significantly better than single embedding.
- **Optimization Difficulties**: In many co-occurrence learning tasks, the co-occurrence data is very sparse. For example, most of the contexts only appear once in a corpus, so when training BERT, we usually can only observe one masked word. In the task, training a LM to output diverse multiple embeddings representation is difficult because multiple embeddings often collapse into an identical embedding, especially when the LM is deep.
- **Efficiency Cost**: Some existing approaches such as MoS are computationally costly [94, 66, 249, 158], especially when the vocabulary size and hidden state size are large.
- **Unknown Effectiveness and Applicability**: Dubossarsky et al. [51] question the effectiveness of multiple word embedding approaches for estimating word similarity. Without knowing why multiple embeddings are better, we do not know we should use multiple embeddings to replace the single embedding in which kinds of applications or self-supervised LMs.

In this thesis, we propose several solutions to respond to the challenges. We briefly summarize our contributions on addressing the above concerns as follows.

- **Theoretical Support**: We advance our understanding of *softmax bottleneck* by showing that multimodal distribution must exist among a subset of the word embeddings in a low dimensional subspace and make the single embedding in the softmax layer not able to arbitrarily rank the words according to their probabilities in Section 2.2.

Furthermore, we establish a connection between the multi-facet embedding representation and clustering in Chapter 3; we propose to learn the multi-facet embedding representation by predicting the cluster centers of the embeddings of co-occurred items in Chapter 3 and Chapter 4.

- **Optimization Techniques**: We diversify facets (i.e., multiple embeddings) by using non-negative sparse coding (NNSC) [89] in the one-tower co-occurrence learning framework when modeling the distribution of the co-occurred word embeddings in Section 3.2; when training BERT using two-tower co-occurrence learning (i.e., contrastive learning) in Chapter 5, we insert different linear layers for each facet after some transformer layers and propose a novel way of aggregating the multiple embeddings that prevents the facets from collapsing by forcing the weights of the final linear layer to be different during the fine-tuning.

- **Efficiency Improvement**: MoS (*mixture of softmax*) [248] needs to compute the dot product between every facet embedding and the embeddings of all the words in the vocabulary. To save the computational resources, we split the vocabulary into several partitions and only compute the dot product between each facet and a partition of the word embeddings in Section 2.3.1.3 when predicting next/masked words. In Section 2.4, we further show that when one partition is dynamically formed by the words that have already been mentioned in the context or the words with high probability, we can achieve better perplexity than MoS without adding significant computational cost to GPT-2 (much faster than MoS). In Chapter 5, we use multiple CLS embeddings to represent a text sequence and train the LM by contrastive learning. The computation cost of multiple embeddings is not significantly higher than the single embedding baseline because we do not need to compute the probabilities over all words in the vocabulary in contrastive learning.

| Chapter | Application | Input | Embedding(s) 1 (E1) | Embedding(s) 2 (E2) |
|---|---|---|---|---|
| 2.3 | Language Generation | Corpus | Multiple Hidden States of a Context | Next Word Embedding |
| 2.4 | | | | Hidden State or Next Word Embedding |
| 2.5 | Sequential Recommendation | Corpus (History) | Multiple Embeddings of a Sequence | Next Item Embedding |
| 3.2 | Sentence Similarity | Corpus | Multiple CLS of a Sentence | Nearby Pre-trained Word Embeddings |
| 3.3 | Relation Extraction | Corpus + NER + EL | Multiple CLS of a Sentence Pattern | Co-occurred Entity Pairs |
| 3.4 | Citation/Authorship Prediction | Paper + Citation | Multiple CLS of Paper's Title+Abstract | Embeddings of Citing Papers and Authors |
| 4 | Interactive Language Generation | Corpus | Multiple Hidden States of a Context | Future Word Embeddings |
| 5.2 | Natural Language Understanding | Corpus (+ Labels) | Multiple CLS of a Sentence | Same as E1 |
| 5.3 | Scientific Paper Representation | Paper + Citation | Multiple CLS of Paper's Title+Abstract | |

Table 1.2: Task summarization. The input specifies the data we required to train the model. We focus on the self-supervised pretraining, but would also show the benefits of multiple embeddings after using the labels to fine-tune the pre-trained models. Please refer to Figure 1.1 to see the meaning of *embedding(s) 1* (E1) and *embedding(s) 2* (E2). Each text sequence in Chapter 2 usually only co-occurs with one item (e.g., a word), while each text sequence in Chapter 3 and 4 usually co-occurs with multiple items (e.g., words). In Chapter 5, a text sequence co-occurs with other similar text sequences.

- **Effectiveness and Applicability**: We demonstrate the effectiveness and wide applicability of multi-facet embeddings on various transformer-based and self-supervised LMs. We summarize the co-occurrence learning tasks in Table 1.2. Then, we summarize the high-level training/testing methods and the text encoders each model used in Table 1.3.

  Our studies suggest that multiple embeddings are better than single embedding especially when the co-occurred item distribution is multimodal. In Section 3.2, we discover that longer input text sequences are often more likely to have multiple aspects. Different aspects could consistently attract the co-occurred items that are semantically different, so the multimodal co-occurrence distribution is more likely to happen when the text sequences are sentences or documents rather than phrases or words.

  In addition to better predicting the co-occurrence probability, we demonstrate the different advantages of multiple embeddings in different sections. For example, in Section 2.4, we show that multiple embeddings can not only improve the next word prediction (especially given ambiguous contexts) but also reduce the hallucinated entity names generated by GPT-2. In Section 2.5, we show that multiple embeddings improve the next product recommendation by learning to copy or exclude the products

| Chapter | Application | Training | Testing | Encoder |
|---|---|---|---|---|
| 2.3 | Language Generation | E1s to E2 (Dot Product) | E1s to E2 (Dot Product) | GPT-2 |
| 2.4 | | Dynamic Partitioning | Dynamic Partitioning | |
| 2.5 | Sequential Recommendation | | | GPT-2-like |
| 3.2 | Sentence Similarity | E1s to E2 (L2) | E1s to E1s (Similarity) | Transformer, LSTM, or GRU (not Pretrained) |
| 3.3 | Relation Extraction | | | |
| 3.4 | Citation/Authorship Prediction | | E1s to E2 (L2) | |
| 4 | Interactive Language Generation | E1s to E2 (L2) | Embeddings as Cluster Centers | GPT-2 |
| 5.2 | Natural Language Understanding | E1s to E1s (Dot Product) | Fine-tuning E1s | BERT |
| 5.3 | Scientific Paper Representation | | E1s to E1s (Dot Product) | SPECTER |

Table 1.3: Method summarization. Chapter 2 trains the models using cross entropy and softmax, a kind of one-tower co-occurrence learning. Chapter 3 and 4 train the models using the one-tower co-occurrence learning with negative sampling. Chapter 5 studies the models using the two-tower co-occurrence learning framework (i.e., contrastive learning).

in the input product history. In Chapter 3, we discover that multiple embeddings could improve the sentence similarity estimation without supervision. In Chapter 4, we show that multiple embeddings could be used as future topics to build a novel interactive and topical-guided language generation framework. In Section 5.2, we show that multiple CLS embeddings representation significantly outperforms the state-of-the-art pre-trained single CLS embedding of BERT on the natural language understanding (NLU) benchmarks, especially when only a limited amount of human labels is available. Finally, in Section 5.3, we show that multiple CLS embeddings improve SPECTER [42] on the benchmarks for evaluating the scientific document embeddings, especially when the model is trained using the papers from multiple domains.

## 1.2 Thesis Goal

The major goal of this thesis is to mitigate the concerns that prevent the multiple embedding representation from becoming a mainstream pre-training approach and demonstrate that the representation is effective and applicable to the various LMs that have different sizes and that are pre-trained by different self-supervised learning tasks. Specifically, we want to develop a set of pre-training techniques or architecture modifications for self-supervised

| Chapter | Application | Dataset(s) | Metric | Baseline | Impr. Ratio |
|---|---|---|---|---|---|
| 2.3 | Next Word Prediction | Wikipedia | Perplexity | GPT-2 Small | 2.5% |
| 2.4 | Text Generation | | Proper Noun ROUGE-1 | | 20.0% |
| 2.5 | Sequential Recommendation | 4 datasets | NDCG@10 | SASRec [96] | 14.4% |
| 3.2 | Sentence Similarity | STSB Low | Pearson Cor. | SIF [7] | 7.1% |
| 3.3 | Relation Extraction | TAC2014 | F1 | CUSchema [225] | 6.0% |
| 3.4 | Citation Prediction | ML S2ORC | MAP | Bansal et al. [14] | 30.1% |
| 4 | Future Topic Prediction | Wikipedia | Similarity to Words | Kmeans-global | 19.0% |
| 5.2 | Natural Language Understanding | GLUE 100 | Overall | $\text{BERT}_{\text{Base}}$ MTL [6] | 4.2% |
| 5.3 | Scientific Paper Representation | Multi-SciDocs | | SPECTER [42] | 4.0% |

Table 1.4: Representative empirical result summary. The improvement ratio is the improvement divided by the baseline score. Please refer to the corresponding sections to see the setup of the experiments.

LMs such that the LMs can learn to output the diverse embeddings without significantly increasing the model size or computational costs.

There are several ways to use the multi-facet embeddings after pretraining, including predicting co-occurrence probability according to the E1s and E2 in Figure 1.1, estimating unsupervised text similarity based on their E1s, serving as the multiple options to help users to inject their intentions to LMs, and classifying the input text after fine-tuning LMs through E1. In the above usages, we want to show that the multi-facet embeddings outperform the single embedding alternatives and understand where the improvement comes from theoretically and/or empirically.

## 1.3 Thesis Outline

As shown in Table 1.3, some projects share the same applications and some use very similar models architecture and training methods. In this thesis, we put the projects using the similar training signal and similar methods into the same chapter. In Chapter 2, each sequence usually co-occurs with one item. We introduce the multi-facet softmax (MFS), its extensions using dynamic partitioning, and its applications. In Chapter 3 and Chapter 4, each sequence usually co-occurs with multiple items. We describe multi-facet embedding, its connection to clustering, and its applications. In Chapter 5, each sequence co-occurs with other sequence(s). We investigate how to use multiple embeddings to improve state-

of-the-art models that use contrastive learning. Please see Table 1.4 for the summary of representative results in every chapter.

In Section 2.2, we advance our understanding of *softmax bottleneck* and propose *multi-facet softmax* (MFS) to improve *mixture of softmax* (MoS) [248]. Theoretically, we show multimodal distribution must exist if many word embeddings lie on a low dimensional subspace/hyperplane. Empirically, we show that multiple embeddings improve the quality of predicting the next word using GPT-2 especially when the context is ambiguous, the next word distribution has multiple modes, or the context comes from a rare language in our corpus. The results are published in Chang and McCallum [32].

In Section 2.4, we unify three popular softmax alternatives: MFS/MoS, copying mechanism/pointer network [74, 146, 72, 191], and reranker/verifier. We propose three dynamic partitioning methods: context partition, reranker partition, and local word embedding, to improve the softmax layer without inducing significant extra computational cost. We use context partition and local word embedding to improve the pointer network and use the reranker partition as a very efficient alternative of the reranker. Our experiments show that one of the major improvement sources from these alternatives is their ability to model the multimodal distribution and the proposed dynamic partitioning approaches can encourage GPT-2 to copy more entity names from the context and accordingly improve the consistency of generated text.

In Section 2.5, we replace the input word sequence with the input product sequence and modify GRU4Rec [85] and SASRec [96], a GPT-2-like architecture, to recommend the next product based on the previous shopping history. Our preliminary experiments show that our softmax alternatives, especially the context partition, significantly improves the sequential recommendation. The results further indicate that the context partition, which learns to copy and exclude the previous context products, not only improve the datasets with duplicated products in the record of each user but also the datasets without duplicated products.

In Section 3.2, we demonstrate that transformers can encode a sentence and predict the cluster centers of its unseen nearby word embeddings well. By modeling the multimodal distribution in a static word embedding space, we can estimate the word importance and improve the unsupervised sentence similarity estimation based on the GloVe [174] space in Section 3.2. We also discover that multiple embeddings do not outperform single embedding when the input is a phrase rather than a sentence, which suggests that multimodal co-occurrence distribution might not be prominent in some applications. The results are published in Chang et al. [34].

Next, in Section 3.3, we modify the above approach to encode a sentence pattern or a knowledge base (KB) relation and predict the cluster centers of their co-occurred entity pairs. The cluster centers are the embeddings that represent sentence patterns and KB relations. The similarity between the sentence pattern embeddings and the embeddings of KB relation can be used to improve the distantly supervised relation extraction. Furthermore, we can detect the entailment relation between sentence patterns using the similarity between their embeddings. We publish the work in Paul et al. [171].

Similarly, the approach can also be used to encode the title and abstract of a paper and predict the cluster centers of its citing papers or its authors. In Section 3.4, we show that the improvement of multiple embeddings on the citation prediction is more significant than its improvement on the authorship prediction. This finding verifies that the benefits of multiple embeddings depend on the applications.

In Chapter 4, we use the above cluster center prediction model to predict the future topics given a context prompt. Then, a user could choose a subset of the topics and ask a conditional LM to generate the continuation that is more likely to contain the chosen topics. We publish the work in Chang et al. [35].

In Chapter 5, we improve the state-of-the-art text encoders that are pre-trained using two-tower co-occurrence learning (i.e., contrastive learning). In Section 5.2, we propose Multi-CLS BERT that takes multiple CLS tokens with the input sentences of BERT and

show that multiple CLS embeddings improve the state-of-the-art pretraining method for BERT [6] in GLUE [229] and SuperGLUE [228], especially when the finetuning dataset is small. Our experimental analyses further suggest that Multi-CLS BERT could be viewed as an efficient BERT ensemble model.

In Section 5.3, we simplify Multi-CLS BERT and use the multiple CLS embeddings to improve state-of-the-art scientific document encoders. The current evaluation benchmark for the encoders is dominated by the computer science papers. To address the limitation, we propose Multi-SciDocs, which tests the embedding qualities of the papers from multiple domains. We show that our proposed encoder, Multi$^2$SPE outperform SPECTER [42], especially in Multi-SciDocs.

This thesis includes work from the following research papers. For more method details and experiment details, please refer to their appendixes.

- **Haw-Shiuan Chang**, Amol Agrawal, Andrew McCallum. 2021. Extending Multi-Sense Word Embedding to Phrases and Sentences for Unsupervised Semantic Applications. In AAAI ([34])

- Rohan Paul*, **Haw-Shiuan Chang***, Andrew McCallum. 2021. Multi-facet Universal Schema. In EACL (Oral) ([171])

- **Haw-Shiuan Chang**, Jiaming Yuan, Mohit Iyyer, Andrew McCallum. 2021. Changing the Mind of Transformers for Topically-Controllable Language Generation. In EACL (Oral) ([35])

- **Haw-Shiuan Chang**, Andrew McCallum. 2022. Softmax Bottleneck Makes Language Models Unable to Represent Multi-mode Word Distributions. In ACL ([32])

- **Haw-Shiuan Chang***, Ruei-Yao Sun*, Kathryn Ricci*, Andrew McCallum. 2022. Multi-CLS BERT: An Efficient Alternative to Traditional Ensembling. In submission to EMNLP

| Chapter | Applications | Other Main Contributors | Their Contributions |
|---|---|---|---|
| 2.3 | Language Generation | None | NA |
| 2.4 | | Zonghai Yao and Alolika Gon | Most implementation and running most experiments |
| 2.5 | Sequential recommendation | Nikhil Agarwal | Some implementation and running some experiments |
| 3.2 | Sentence Similarity | Amol Agrawal | Baseline implementation and running exploratory experiments |
| 3.3 | Relation Extraction | Rohan Paul | All implementation and running most of experiments |
| 3.4 | Citation/Authorship Prediction | None | NA |
| 4 | Interactive Language Generation | Jiaming Yuan | Implementing some evaluation metrics and conducting some evaluations |
| 5.2 | Natural Language Understanding | Ruei-Yao Sun and Kathryn Ricci | Some implementation and running some experiments |
| 5.3 | Citation Prediction | Ronald Seoh | All implementation, running all experiments, and writing half of the paper. |

Table 1.5: The contributions of other students to the chapters of this thesis.

- Ronald Seoh*, **Haw-Shiuan Chang***, Andrew McCallum. 2022. Multi-domain Paper Encoder with Multiple CLS Tokens. In submission to EMNLP

Following work is not directly included, but they inspire some main ideas of this thesis.

- **Haw-Shiuan Chang**, ZiYun Wang, Luke Vilnis, Andrew McCallum. 2018. Distributional Inclusion Vector Embedding for Unsupervised Hypernymy Detection. In HLT/NAACL

- **Haw-Shiuan Chang**, Amol Agrawal, Ananya Ganesh, Anirudha Desai, Vinayak Mathur, Alfred Hough, Andrew McCallum. 2018. Efficient Graph-based Word Sense Induction by Distributional Inclusion Vector Embeddings. In TextGraphs-12

- **Haw-Shiuan Chang**, Abdurrahman Munir, Ao Liu, Johnny Tian-Zheng Wei, Aaron Traylor, Ajay Nagesh, Nicholas Monath, Patrick Verga, Emma Strubell, Andrew McCallum. 2016. Extracting Multilingual Relations under Limited Resources: TAC 2016 Cold-Start KB construction and Slot-Filling using Compositional Universal Schema. In TAC/KBP

## 1.4 Declaration of Collaborations

Most of the projects in this thesis are collaboratively done by me and (former) UMass master's students. In all of the projects, I am a supervisor who provides the ideas and the steps for implementing the ideas. My contributions might also include debugging the code,

designing the experiments, running experiments and analyses, surveying the related work, writing a paper, and presenting the work. The master students' contributions might include writing preprocessing codes, implementing the ideas by modifying my codes or the codes of state-of-the-art models, and running experiments to test the ideas. Please see Table 1.5 for the details.

# CHAPTER 2

# BREAKING THE SOFTMAX BOTTLENECK FOR LANGUAGE GENERATION

"The greater the ambiguity, the greater the pleasure." — Milan Kundera

## 2.1 Introduction

Recently, researchers have found that transformer-based language models (LMs), such as GPT-2, can predict the next word distribution better as their sizes grow [177, 21, 97]. Compared to greedily outputting the most probable next word, sampling the next word from the predicted distribution allows LMs to generate more diverse and high-quality text sequences [87]. By autoregressively sampling the next word, LMs can assist creative writing [1], reduce the cost of building datasets [239, 126], generate codes [119], solve math problems [41], etc. As a result, one natural question arises: Do modern language modeling architectures still have restrictions in their ability to represent the appropriate distribution over next words?

In this chapter, we discover that, when predicting the next word probabilities given an ambiguous context, GPT-2 is sometimes incapable of assigning the highest probabilities to the appropriate non-synonym candidates. For example, given the input prompt *"After debating whether to bow to the **woman** or the **king** first, the jester decided on the [MASK]"*, we would expect the distribution over the *[MASK]* fillers to put high probabilities on *woman* or *king* or their synonyms. However, GPT-2 might incorrectly assign the second highest probability to "*queen*" as in Figure 2.1.

In the final softmax layer of GPT-2, the log probabilities of the *woman* and *king* are computed based on the dot product between a single hidden state embedding and the global

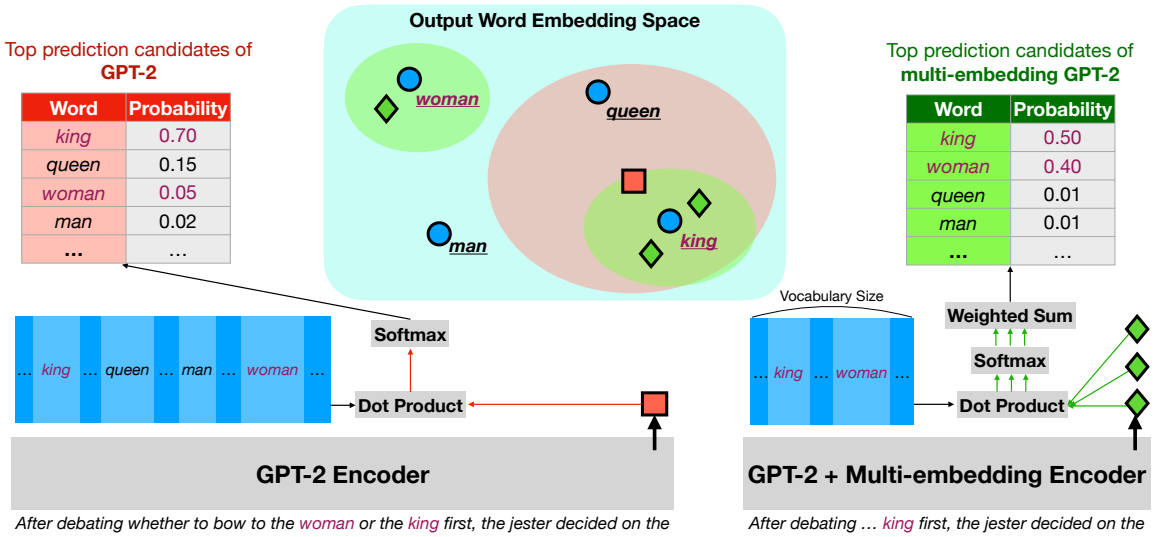



Figure 2.1: Comparison between the softmax layers using a single embedding and multiple embeddings when the next word should be either *woman* or *king*. In GPT-2 and multi-embedding GPT-2, the hidden states of the context are visualized by the single facet ■ and multiple facets ◆, respectively. The word embeddings are visualized using ●. GPT-2 cannot output *woman* and *king* as the top two words because ***queen*** and ***man*** are close to the middle of ***woman*** and ***king***. The improvement in this type of ambiguous context will be quantified in Appendix A.2.1.

word embeddings of the *woman* and *king*, respectively. To have the highest but similar dot products for the two options, the transformer encoder in GPT-2 wants to output the hidden state that is close to the average of the *woman* embedding and the *king* embedding. However, the words *queen*, *king*, *woman*, and *man* tend to form a parallelogram in the embedding space [148, 54, 235][1], which means the *man* and *queen* also have a similar average. Therefore, GPT-2 is forced to also output the *man* or *queen* when it wants to output the *woman* or *king*.

The problem not only happens to GPT-2 or the words whose embeddings form a parallelogram shape. Even though the hidden state embeddings of LMs are contextualized, the embedding of each word in the softmax layer is global and static during the inference

[1]Section 3.3.1.1 provides more background knowledge about the parallelogram shape and the softmax bottleneck.

time. Globally dissimilar words could all become the suitable next word in a context while other interfering words might be between them, which makes the ideal next word embedding distribution to have multiple modes and cannot be modeled by the single embedding representation.

In this chapter, we propose theorems showing that given any LM using the output softmax layer, when there are more than $N$ word embeddings in a $N-1$ dimensional subspace/hyperplane (e.g., 4 embeddings in a two-dimensional plane), we can always find a set of possible next words (e.g., *woman* and *king*) such that there are some other interfering words between them (e.g., *man* or *queen*).

Recently, *mixture of softmax* (MoS) [248] regains attention as one of the few effective architecture modifications for transformer LM [158, 218]. In the meanwhile, Parthiban et al. [168] show that the *softmax bottleneck* [248] theory is not sufficient to explain the improvement of MoS. Our theorems not only provide geometrical intuitions of why and when the multiple embedding representation such as MoS would do better but also suggest that the *softmax bottleneck* might not be completely solved even if we adopt a very large hidden state size. For example, no matter how large the hidden state size is, as long as ***queen*** - ***king*** = ***woman*** - ***man*** in the embedding space, the LMs cannot output a pair of words in the longer diagonal of the parallelogram as the top two output words.

After better understanding why *mixture of softmax* (MoS) works well, we propose two enhancements over MoS. The first enhancement considers the hidden states of multiple positions and multiple transformer layers when determining the probability in each softmax; the second enhancement uses different contextualized embeddings to compute the probabilities of different global partitions of words.

The resulting method, *multi-facet softmax* (MFS), significantly outperforms the MoS and the GPT-2 with the softmax layer on the perplexity for predicting the next word, especially in ambiguous context and non-English text in OpenWebText [177]. For details of our methods and experiments, please see the appendix of Chang and McCallum [32].

MFS and MoS both use multiple softmaxes to break the softmax bottleneck and each softmax needs to compute the dot product between the facet embedding and the embeddings of all the words in the vocabulary. Although being able to handle the ambiguous contexts better, the approaches are much more computationally expensive compared to the single softmax baseline.

In Section 2.4, we propose three dynamic partitioning approaches to solve the problem: context partition, reranker partitions, and local word embedding. The context partition approach uses one facet to compute the logits of all the words in the context and uses another facet to compute the logits of the rest of the words in the vocabulary. The reranker partition approach uses another facet to re-estimate the logits of the top prediction words with the highest probabilities. The local word embedding approach uses another facet to compute the context-dependent word embeddings for all the words in the context.

The dynamic partitioning approaches are much more efficient compared to MoS and we find that after using dynamic partitioning, the improvement of MoS would become very small, which indicates that dynamic partitioning approaches can also solve the problem caused by the multimodal distribution.

Sequential recommendation problem is very similar to language modeling. By replacing the words with the products, predicting the next word based on the context becomes the task of predicting the next product based on the shopping/interaction history of a user. Recently, SASRec [96], a GPT-2-like architecture, has achieved good performance on the sequence-aware recommendation tasks. Our preliminary experiments in Section 2.5 suggest that the dynamic partitioning approaches significantly improve the SASRec and GRU4Rec [85] in four datasets and also outperform RepeatNet [181], a strong pointer network baseline for sequential recommendation.

## 2.2 Theoretical Limitations of the Single Embedding in the Softmax Layer and Empirical Analyses

In this section, we first review the softmax layer of GPT-2 formally and explain why ***queen*** - ***king*** = ***woman*** - ***man*** still tends to hold in contextualized LMs. Next, we present our theoretical analyses, which generalize the *woman* and *king* example by showing that the candidate words in a low dimensional subspace would induce the impossibility of ranking some candidates on top of other candidates. Finally, we empirically study how serious the limitations are in the language models with different sizes.

### 2.2.1 Background

The LMs typically use a softmax layer to predict $P_S(x|c_t)$, the probability of the next word $x$ given the context at the $t$th position $c_t$:

$$P_S(x|c_t) = \frac{\exp(\boldsymbol{h}_{c_t}^T \boldsymbol{w}_x)}{\sum_{x'} \exp(\boldsymbol{h}_{c_t}^T \boldsymbol{w}_{x'})}, \tag{2.1}$$

where $\boldsymbol{h}_{c_t}$ is the $t$th hidden state in the context $c$, and $\boldsymbol{w}_x$ is the output word embedding for the word $x$ (i.e., the linear weights that project the hidden state to the logit of the word $x$). Yang et al. [248] point out that the log probability distribution over all the words in the vocabulary $V$ is $\log\left(P_S(x|c_t)\right)|_{x\in V} = \boldsymbol{h}_{c_t}^T \boldsymbol{w}_x - \log\left(\sum_{x'} \exp(\boldsymbol{h}_{c_t}^T \boldsymbol{w}_{x'})\right)|_{x\in V}$. The distribution is a linear projection from the hidden state $\boldsymbol{h}_{c_t}$ with dimension $D$, so the degree of freedom in the distribution is only $D$ (i.e., there cannot be more than $D$ linearly independent log distributions). We call this restriction *softmax bottleneck* thoery.

During training, the ideal output word embedding $\boldsymbol{w}_x$ should be close to the hidden states of the contexts $\boldsymbol{h}_{c_t}$ that co-occur with the word $x$ while far away from the other hidden states. This objective is similar to the objective function of Word2Vec [148] except that the context embeddings are contextualized [105, 115].

If a context $c_t$ has a higher chance to co-occur with *queen* compared to *king*, the context also has a higher chance to co-occur with *woman* compared to *man* to a similar degree. This

is the main reason that makes ***queen*** - ***king*** = ***woman*** - ***man*** in the Word2Vec space [54]. Therefore, the same linear relations tend to hold in the output word embedding space of GPT-2 as well [235].

### 2.2.2 Structural Weakness Theorems from Linear Dependency

In addition to words satisfying the analogy relations, the following theorems imply that any linear dependency among the words causes the difficulties of LM in ranking the words in an arbitrary order according to their logits (i.e., dot products between the hidden state and the word embedding). For example, ***woman*** + ***king*** = ***queen*** + ***man*** makes a LM unable to output *woman* and *king* as the top two words in Figure 2.1.

**Theorem 1.** *If the nonzero output embeddings of $N$ words in a set $W$ are linearly dependent and on one side of a plane through the origin, the single embedding representation cannot produce positive logits for a subset of the word in $W$ that are higher than all the logits of the other words in $W$.*

Here, we provide an intuitive justification: if $N$ embeddings are in a subspace whose dimension is smaller than $N-1$, the $N$ embeddings are going to be linearly dependent and some set of words cannot have the top dot products due to the limited degree of freedom in the subspace. In Appendix A.3, we formally prove the theorem by identifying the sets of words that cannot be ranked top by the single embedding representation.

In practice, linear dependency holds approximately instead of exactly. For example, ***woman*** = ***queen*** + ***man*** - ***king*** + $\boldsymbol{\varepsilon}$. In this practical condition, the following theorem states that the logits of the bottom words (i.e., *man* and *queen*) cannot be much smaller than the logits of the top words (i.e., *woman* and *king*).

**Theorem 2.** *Let the output word embeddings in the set $W = \{\boldsymbol{w}_i \neq \mathbf{0} | i = 1...N\}$ satisfy $\boldsymbol{w}_1 = a_2\boldsymbol{w}_2 + ... + a_N\boldsymbol{w}_N + \boldsymbol{\varepsilon}$, where the constant $a_2, ..., a_N$ are neither all zero nor all negative and $||\boldsymbol{\varepsilon}|| < \epsilon$. Then, there must be a non-trivial partition $P = \{G, S\}$*

*of $W$ such that there is no hidden state $||\boldsymbol{h}|| \leq r$ and a threshold $\tau \geq r\epsilon$ that make $\min_{\boldsymbol{w}_g \in G} \boldsymbol{h}^T \boldsymbol{w}_g \geq (1+\delta)\tau$ and $\max_{\boldsymbol{w}_s \in S} \boldsymbol{h}^T \boldsymbol{w}_s < \tau$, where $\delta = \frac{2}{1+\sum_{i=2\ldots N}|a_i|}$.*

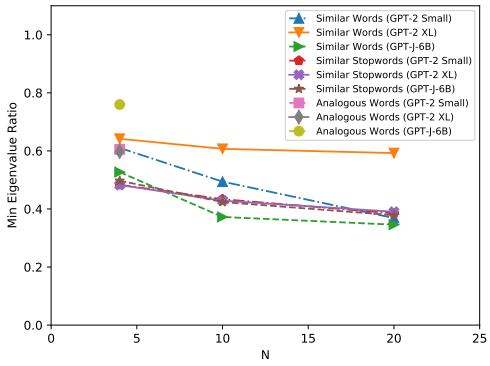
In the king-woman example, $(1+\delta) = (1+\frac{2}{4}) = 1.5$. Assuming $||\boldsymbol{\varepsilon}|| < \epsilon = 0.01$ and $||\boldsymbol{h}|| \leq r = 20$, we can get $\boldsymbol{h}^T\boldsymbol{\varepsilon} \leq 0.01 \times 20 = 0.2$. Then, we cannot find a hidden state $\boldsymbol{h}$ such that $\boldsymbol{h}^T\boldsymbol{w}_{king} \geq 1.5 \times 0.01 \times 20 = 0.3$ and $\boldsymbol{h}^T\boldsymbol{w}_{woman} \geq 0.3$ but $\boldsymbol{h}^T\boldsymbol{w}_{queen} < 0.2$ and $\boldsymbol{h}^T\boldsymbol{w}_{man} < 0.2$ because $\boldsymbol{h}^T\boldsymbol{w}_{king} + \boldsymbol{h}^T\boldsymbol{w}_{woman} = \boldsymbol{h}^T\boldsymbol{w}_{queen} + \boldsymbol{h}^T\boldsymbol{w}_{man} + \boldsymbol{h}^T\boldsymbol{\varepsilon}$. The formal proof can be found in Appendix A.3 and Section 2.2.3 estimates $\epsilon$ in LMs.

Even though, theoretically speaking, outputting *woman* and *king* as the top two words is possible due to the appearance of $\boldsymbol{\varepsilon}$, LMs may not successfully learn to output the optimal $\boldsymbol{h}$ and the optimal hidden state for these four words could lead to the wrong probabilities of the other words. Consequently, LMs sometimes still ranks *queen* or *man* higher than *woman* or *king* in practice.

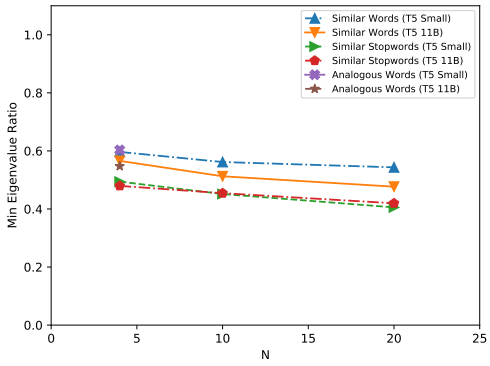
### 2.2.3 Measuring Linear Dependency among Words

Theorem 2 shows that when $N$ words are linearly dependent after moving one of the embeddings with a short distance $\epsilon$, the LM with the output softmax layer cannot output a large logit margin between two subsets of the $N$ words. We want to measure $\epsilon$ in the pretrained word embedding and compare the $\epsilon$ from different sets of words or from different LMs.

Given a set of $N$ words, we form a matrix by their word embeddings and estimate the $\epsilon$ value by the minimal eigenvalue of the matrix. We first want to verify that the four analogical words (e.g., *queen*, *king*, *man*, and *queen*) indeed have a smaller $\epsilon$ compared to a randomly selected four words. Thus, we define the min eigenvalue ratio as $\frac{\epsilon_S}{\epsilon_R}$, where $\epsilon_R$ is the average of minimal eigenvalues from 1,000 sampled $N$ word sets and $\epsilon_S$ is the average of minimal eigenvalues from sets of words (e.g., analogical words from the Google analogy dataset). We analyze the ratio instead of $\epsilon$ because the average word embedding magnitudes in different LMs would affect the absolute value of $\epsilon$.




(a) *GPT-2 Small* (0.1B, D=768), *GPT-2 XL* (1.5B, D=1600), and *GPT-J-6B* (6B, D=4096)




(b) *T5 Small* (0.06B, D=512) and *T5 11B* (11B, D=1024)

Figure 2.2: Minimal eigenvalue ratios of different groups of $N$ word embeddings. Lower ratios indicate that the corresponding word embeddings are more linearly dependent and thus the probabilities of the words cannot be determined arbitrarily by the hidden state of the language models. The number of parameters and the size of hidden states are shown in caption beside every model name.

In addition to analogical words, we also test sets of $N$ similar words, which are composed by the nearest $N-1$ words of every query word in the vocabulary, and test the $N$ similar stop words by finding the nearest $N-1$ words of every query word in a stop word list.[2]

We plot the min eigen value ratio versus $N$ in Figure 2.2 and compare the curves from three GPT LMs and two T5 LMs [178]. All the ratios are below $0$ and decrease as $N$ increase, which shows the analogical words and similar words indeed have significantly smaller $\epsilon$ especially for a large $N$. The low minimal eigenvalues and our theory support the recent empirical finds that LMs tend to be confused by the similar words [253]. This figure also provides a potential explanation why the candidates often include stop words when multiple embeddings outperform the single embedding in Table 2.3 and Table A.2.

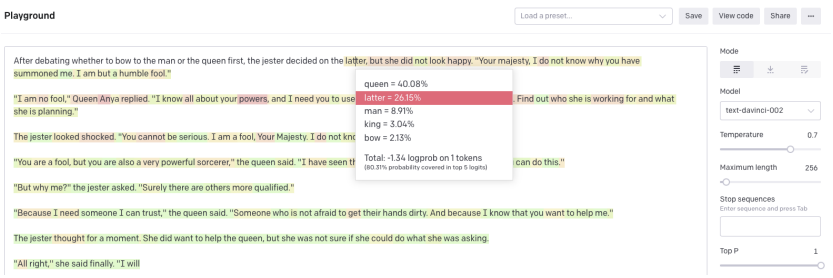
[2]We find that some rare words or special characters might have nearly identical word embeddings due to the lack of training instances, so we exclude the half of rarer word pieces in the vocabulary and exclude the word pieces whose first character is not a space. The rarity of a word piece is determined by the l2 norm of its word embedding.

Surprisingly, we find that a larger LM does not necessarily yield a larger ratio (i.e., embeddings of related words do not become more linearly independent as dimension or the size of the LM increases). All the LMs have very similar ratios of similar stop words. Compared to *GPT-small*, although *GPT-J-6B* [230] has a significantly higher ratio for analogical words, its ratio for similar words is significantly lower. Besides, *T5-11B* has a significantly lower ratios compared to *T5-small*. We need further investigation to understand the reason of this empirical finding and whether a larger LM suffers less from the limitation caused by the single embedding.

### 2.2.4 Softmax Bottleneck in GPT-3

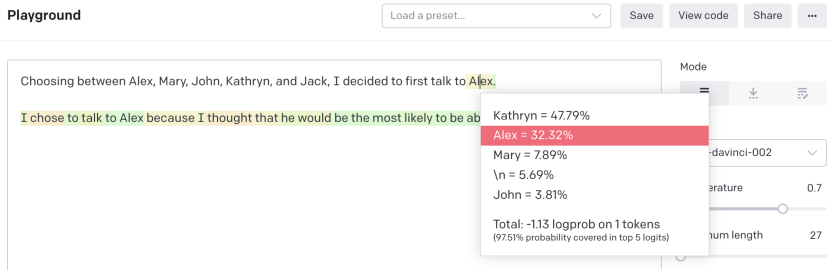
We show the three predictions of GPT-3 [21] in Figure 2.3 to see if a huge language model with a large hidden state size (12k) also suffers from the softmax bottleneck. In Figure 2.3a, we replace the *woman/king* with the *man/queen* example because the problem is more serious in this GPT-3 version than the *woman* or *king* example.

Figure 2.3a shows that although the incorrect answer *king* is not ranked higher than *queen* and *man*, GPT-3 put around 66% probability on *queen/latter*, while the word *man* only gets around 9%, which is not too much higher than the probability of the incorrect answer *king*. In Figure 2.3b, we construct our prompt such that the next word should be the five names we mentioned and each name should be similarly probable. However, we can see that around 68% probability is concentrated in the word *John* and the word *Alex* is not even in the top 5 list. In Figure 2.3c, we can see that GPT-3 completely change the probability distribution after the order of the name changes. In Figure 2.3d, the next word should be the five things we mentioned and each thing should receive similar probability. However, we can see that one of the choices (*scissors*) still take around 67% probability. In the meanwhile, the word *balloons* and *toys* receive less than 1% of probabilities. The results suggest that the softmax bottleneck is still a problem for a very large language model. The

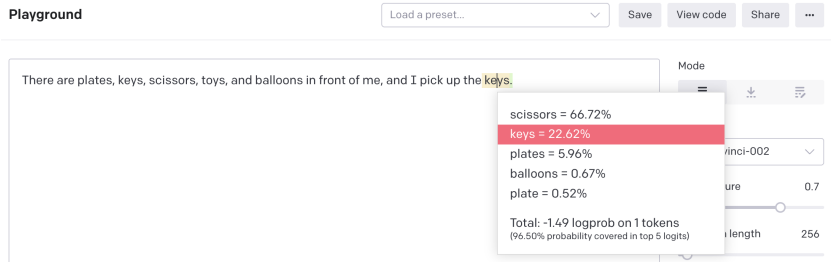



(a) The example where the next word should be either *man* or *queen* (or their synonym such as former and latter).


(b) The example where the next word *John*, *Alex*, *Mary*, *Kathryn*, and *Jack* should receive similar probabilities.




(c) Same as above except that the order of the names in the context is different.




(d) The example where the next word *plates*, *keys*, *scissors*, *toys*, and *balloons* should receive similar probabilities.

Figure 2.3: The next word probabilities outputted by GPT-3. Notice that this is a raw probability before being modified using the temperature.

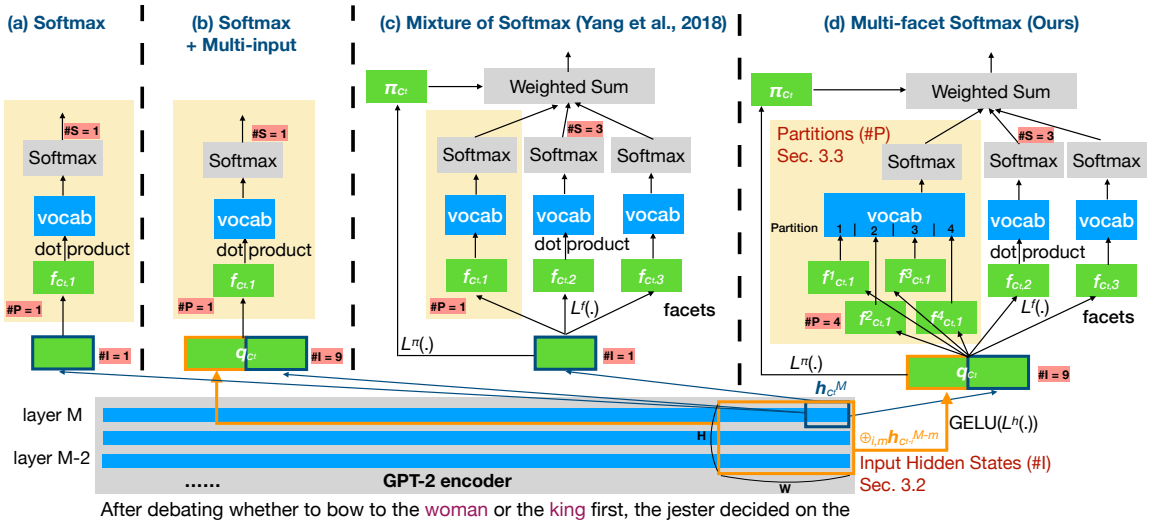



Figure 2.4: Comparison between different architectures. The #S, #I, and #P are the number of softmaxes, input hidden states, and partitions, respectively. The green boxes refer to embeddings/vectors. The vocab means the embeddings of all words in the vocabulary. $\oplus$ refers to concatenation. $L^h$, $L^f$, and $L^\pi$ are linear projection layers.

problem is especially serious when we want to get all the possible story developments by sampling from the next word distribution.

## 2.3 Multi-facet Softmax

Using multiple embeddings is a natural solution of modeling a multimodal distribution. For instance, we can use three embeddings to capture the high probability on the *woman* and *king* but low probability on the *man* and *queen* in Figure 2.1.

Inspired by our geometric analysis on the limitation of the single embedding, we improve the state-of-the-art multiple embedding solution, *mixture of softmax* (MoS) [248] by two enhancements: multiple input hidden states and multiple partitions on the vocabulary.

### 2.3.1 Method

In the section, we review the mixture of softmax (MoS) in Section 2.3.1.1, explain our first enhancement in Section 2.3.1.2, and explain our second enhancement in Section 2.3.1.3.

#### 2.3.1.1 Mixture of Softmax

Yang et al. [248] propose *mixture of softmax* (MoS) to allow a LSTM-based [86] LM to produce more linearly independent log probability distributions of the output words given different contexts. As in Figure 2.4 (c), the MoS first uses multiple linear layers $L_k^f$ to project a hidden state $\boldsymbol{h}_{c_t}$ into multiple facet embeddings $\boldsymbol{f}_{c_t,k} = L_k^f(\boldsymbol{h}_{c_t})$.[3] The multiple facets $\boldsymbol{f}_{c_t,k}$ and softmaxes would lead to multiple probability distributions, and output probability is the weighted average of the distributions:

$$P_{MoS}(x|c_t) = \sum_{k=1}^{K} \pi_{c_t,k} \frac{\exp(\boldsymbol{f}_{c_t,k}^T \boldsymbol{w}_x)}{\sum_{x'} \exp(\boldsymbol{f}_{c_t,k}^T \boldsymbol{w}_{x'})}. \tag{2.2}$$

The prior weights $\pi_{c_t,k} = \frac{\exp(L_k^\pi(\boldsymbol{h}_{c_t}))}{\sum_{k'} \exp(L_{k'}^\pi(\boldsymbol{h}_{c_t}))}$, where $L_k^\pi$ is another linear projection for dynamically generating the weights and the projection goes through a softmax to ensure $\sum_{k=1}^{K} \pi_{c_t,k} = 1$.

#### 2.3.1.2 Multiple Input Hidden States

To model the multimodal distribution, the facets (i.e., the embeddings for different softmaxes) should be able to move freely. For example, in Figure 2.1, we have three facets but only have two modes, so the two embeddings are very close to the word *king*. However, when we want to output three dissimilar top words such as the *king*, *woman*, and *knight*, one of the facets should be moved to be near to the embedding of the *knight*.

Therefore, we want our solution to satisfy two properties: a) the linear transformation matrix in $L_k^f$ should have a full rank to avoid limiting the degree of freedom in each facet, and b) the relative location of the facets should be context-dependent. MoS cannot satisfy both properties. If the first one is satisfied, the input hidden state is uniquely determined by a facet (e.g., $\boldsymbol{h}_{c_t} = (L_1^f)^{-1}(\boldsymbol{f}_{c_t,1})$). Then, there exist a global transformation between two facets (e.g., $\boldsymbol{f}_{c_t,2} = L_2^f \left((L_1^f)^{-1}(\boldsymbol{f}_{c_t,1})\right)$), which violates the second property. That is,

[3] We remove the tanh layer in the original MoS to improve its performance on GPT-2.

assuming LM can move every facet freely (i.e., the facet's degree of freedom is the same as the dimension of the hidden state), LM cannot make the first two facets close to *woman* and *king* in one context but make the two facets close to *woman* and *knight* in another context. In other words, since the facet embeddings are the projection of a single hidden state, the total degree of freedom in all facet embeddings cannot exceed the dimension of the hidden state.

Our solution to this issue is using more input hidden states to construct the facets. As the orange box in Figure 2.4, we first concatenate a $W \times H$ block of input hidden states into $\oplus_{i=0...W-1,m=0...H-1}\boldsymbol{h}_{c_{t-i}}^{M-m}$, where $M-m$ is the transformer layer index and $t-i$ is the index of the $i$th to the last word in the context. The $W \times H$ is fixed as 3×3 in this chapter. We make its dimension the same as the original hidden state $\boldsymbol{h}_{c_t}^{M}$ using a linear layer $L^h$ plus a GELU activation function [83]. Then, we concatenate it with the original hidden state to form a new input hidden state

$$\boldsymbol{q}_{c_t} = \boldsymbol{h}_{c_t}^{M} \oplus GELU\left(L^h(\oplus_{i,m}\boldsymbol{h}_{c_{t-i}}^{M-m})\right). \tag{2.3}$$

The new input hidden state is passed through the linear transformation $L_k^f$ to compute the facets $\boldsymbol{f}_{c_t,k} = L_k^f(\boldsymbol{q}_{c_t})$ and our prior weights $\pi_{c_t,k} = \frac{exp(L_k^\pi(\boldsymbol{q}_{c_t}))}{\sum_{k'} exp(L_{k'}^\pi(\boldsymbol{q}_{c_t}))}$. Since the dimension of $\boldsymbol{q}_{c_t}$ is larger than the dimension of $\boldsymbol{f}_{c_t,k}$, the inverse function $(L_k^f)^{-1}$ no longer exists.

#### 2.3.1.3 Multiple Partitions

The next word distribution could have many modes. However, using many softmaxes significantly increases our computational burden because we need to compute the dot product between each facet and all the word embeddings in our vocabulary.

Inspired by our analysis, we propose to split all the words in the vocabulary into multiple partitions and use different facets for different partitions. For example, if we can put any word from {*queen*, *man*, *woman*, *king*} into one partition and the rest of the words into another partition, we no longer have ***queen*** - ***king*** = ***woman*** - ***man*** in either of the partitions. In this method, each word only belongs to one partition, so we only need to compute one

dot product for each word. Thus, the extra computational cost only comes from the extra linear projections for preparing the facets.

In this work, we simply let the $j$th facet handle the $J \times n + j$th word (e.g., when the number of partitions $J = 4$, the first partition includes the words with indexes $0, 4, 8, ...$ ). This simple global partitioning method reduces the chance of putting all the interfering words and candidates in the same partition, while minimizing the extra computational cost in our PyTorch implementation because PyTorch supports the dilated access without copying the variable.

In many contexts $c_t$, the distribution of the next word has only a single mode and the global similarity between words may be useful. Using the multiple partitions alone might lose the similarity information between words in different partitions. Therefore, we propose to only replace the first softmax layer in MoS with the multiple partition method to learn the global similarity of words in different partitions using the other softmaxes. The architecture is illustrated in Figure 2.4 (d). Formally, we compute the probability using

$$P_{MP}(x|c_t) = \pi_{c_t,1} \frac{\exp((\boldsymbol{f}_{c_t,1}^{j_x})^T \boldsymbol{w}_x)}{\sum_{x'} \exp((\boldsymbol{f}_{c_t,1}^{j_{x'}})^T \boldsymbol{w}_{x'})} + \sum_{k=2}^{K} \pi_{c_t,k} \frac{\exp(\boldsymbol{f}_{c_t,k}^T \boldsymbol{w}_x)}{\sum_{x'} \exp(\boldsymbol{f}_{c_t,k}^T \boldsymbol{w}_{x'})}, \tag{2.4}$$

where $j_x$ is the partition index that the word $x$ belongs to and $\boldsymbol{f}_{c_t,1}^{j_x}$ is the facet for the $j_x$th partition. *Multi-facet softmax* (MFS) is equipped with multiple input hidden states and multiple partitions.

### 2.3.2 Language Modeling Experiments

We evaluate different LM architectures by comparing their capability of predicting the next word in Wikipedia 2021 and a subset of OpenWebText [177]. The size of the training, validation, and testing set are 96%, 2%, and 2% of the whole corpus, respectively. After loading the pre-trained GPT-2 models, we train the *GPT-2 Small* for 1 epoch and *GPT-2 Medium* for 0.4 epochs.

#### 2.3.2.1 Baselines

We set different numbers of softmaxes, input hidden states, and partitions in our MFS framework to construct our baselines. The configuration of different baselines could be seen in Table 2.1.

**Softmax (GPT-2)**: Using a single softmax, input hidden state, and partition as in Figure 2.4 (a) and Equation 2.1. The baseline is the same as the original GPT-2 except that we add one more linear layer that converts the hidden state $\boldsymbol{h}_{c_t}^M$ to the facet embedding $\boldsymbol{f}_{c_t,1}$ as in other methods.

**SigSoftmax [94]**: The same as *Softmax* except when predicting the next word, Kanai et al. [94] add some non-linearity into the softmax layer by multiplying the exponent and sigmoid of the logits.

**Softmax + Multi-input**: Letting *Softmax* access multiple input hidden states as in Figure 2.4 (b) and Equation 2.3. The method is similar to Tenney et al. [220], Fan et al. [58], and Tay et al. [217].

**MoS [248]**: *MoS (3)* is the *mixture of softmax* with 3 facets/softmaxes, whose probability comes from Equation 2.2. We also run the MoS with 4 softmaxes in *GPT-2 Small* and call the model *MoS (4)*.

**DOC [214]**: Similar to our enhancement using multiple input hidden states, *direct output connection* (DOC) makes each of their facets coming from a different input hidden state.

Other configurations include **Softmax + Multi-partition**, which adds four partitions into the softmax, **MFS w/o Multi-partition**, which uses only one partition in *MFS* and could also be viewed as *MoS + Multi-input*, and **MFS w/o Multi-input**, which uses only one input hidden state to generate all facets.

#### 2.3.2.2 Results

Table 2.1 shows that applying **MFS** to *GPT-2 Small* achieves more than 15% of the perplexity improvement between *GPT-2 Small* and *GPT-2 Medium*, while only increasing

| Models ↓ | Configuration | | | GPT-2 Small | | | | GPT-2 Medium | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | #S | #I | #P | Size | Time | OWT | Wiki | Size | Time | OWT | Wiki |
| Softmax (GPT-2) | 1 | 1 | 1 | 163.6M | 84ms | 18.72 | 24.06 | 407.3M | 212ms | 15.89 | 20.34 |
| SigSoftmax [94] | 1 | 1 | 1 | 163.6M | 91ms | 18.63 | 24.06 | 407.3M | 221ms | 16.07 | 20.65 |
| Softmax + Multi-input | 1 | 9 | 1 | 169.5M | 87ms | 18.50 | 23.89 | 417.8M | 219ms | 15.76 | 20.29 |
| Softmax + Multi-partition | 1 | 1 | 4 | 165.4M | 88ms | 18.77 | 24.08 | 410.5M | 218ms | 15.89 | 20.30 |
| MoS [248] (4) | 4 | 1 | 1 | 165.4M | 152ms | 18.61 | 23.77 | 410.5M | 299ms | 15.75 | 20.08 |
| MoS [248] (3) | 3 | 1 | 1 | 164.8M | 130ms | 18.63 | 23.81 | 409.4M | 270ms | 15.79 | 20.11 |
| DOC [214] | 3 | 3 | 1 | 164.8M | 130ms | 18.69 | 24.02 | 409.4M | 270ms | 15.88 | 20.34 |
| MFS w/o Multi-partition | 3 | 9 | 1 | 171.9M | 133ms | 18.37 | 23.56 | 422.0M | 276ms | 15.65 | 20.06 |
| MFS w/o Multi-input | 3 | 1 | 4 | 166.6M | 134ms | 18.60 | 23.72 | 412.6M | 275ms | 15.71 | 20.08 |
| MFS (Ours) | 3 | 9 | 4 | 175.4M | 138ms | **18.29** | **23.45** | 428.3M | 283ms | **15.64** | **20.02** |

Table 2.1: Perplexity comparison between MFS (Ours) and baselines. #S, #I, #P are the number of softmaxes (i.e., $K$), input hidden states, and partitions, respectively. The top four baselines use a single softmax. OWT and Wiki are the test set perplexity of OpenWebText and Wikipedia 2021, respectively. The standard errors of all models are smaller than 0.02 perplexity. We also compare the number of parameters and the inference time on one batch.

| Ratio in Corpus → | Non-English 14% | English 86% |
|---|---|---|
| Softmax | 13.50 (0.0%) | 19.23 (0.0%) |
| MoS [248] (3) | 13.19 (2.3%) | 19.16 (0.4%) |
| MFS w/o Multi-partition | 12.98 (3.8%) | 18.91 (1.7%) |
| MFS (Ours) | **12.83 (5.0%)** | **18.83 (2.1%)** |

Table 2.2: Perplexity of the *GPT-2 Small* in OpenWebText. The percentages of the perplexity reduction compared to Softmax are presented in the parentheses.

5% of their size differences. Except for **Softmax + Multi-partition**, adding multiple input hidden states or partitions in different configurations significantly boost the performance. In Appendix A.2.3, we further show that the improvement of **MFS** over **Softmax** could even become 3-5 times larger in the top 5-10% of the most ambiguous contexts compared to the rest of the contexts, which suggests that some improvements indeed come from successfully modeling multimodal distribution.

**MFS** usually doubles the perplexity improvements between **MoS (3)** and **Softmax** but the running time of **MFS** remains similar to **MoS (3)** because **MFS** only needs a few more linear layers, which is more efficient than adding one more softmax as in **MoS (4)**. **DOC** is worse than **MoS (3)**. This may be due to a starvation problem: the facet from the last hidden state $\boldsymbol{h}_{c_t}^M$ has the prior probability close to 1 and receives most of the gradients.

| Corpus → | OpenWebText | Wikipedia 2021 | Analogy in Templates (Appendix A.2.1) |
|---|---|---|---|
| Input Context | ... The Elastic Endpoint Security and Elastic SIEM solutions mentioned in this post are now referred to as **Elastic** | ... law and chance working together cannot generate CSI, either. Moreover, he claims that **CSI** | I went to Paris and Germany before, and I love one of the places more, which is **Germany** |
| Softmax (GPT-2) | the 0.087, E 0.043, End 0.039 | the 0.174, this 0.054, if 0.038 | Paris 0.893, France 0.045, **Germany** 0.033 |
| MFS (Ours) | **Elastic** 0.220, the 0.089, EC 0.033 | CSI 0.186, the 0.140, there 0.033 | Paris 0.544, **Germany** 0.389, France 0.064 |
| MFS Softmax 1 | end 0.051, the 0.043, security 0.023 | the 0.191, law 0.127, if 0.053 | Paris 0.979, France 0.013, **Germany** 0.007 |
| MFS Softmax 2 | **Elastic** 0.652, EC 0.080, ES 0.046 | the 0.191, there 0.049, this 0.047 | Paris 1.000 Berlin 0.000 ##Paris 0.000 |
| MFS Softmax 3 | the 0.193, E 0.040, a 0.014 | **CSI** 0.677, law 0.029, laws 0.019 | **Germany** 0.852, France 0.139, China 0.004 |

Table 2.3: Prediction visualization using a context in each dataset. We show the top three words with the highest prediction probabilities of each method. In the last three rows, we visualize the outputs of the softmax grey boxes in Figure 2.4 (d), which model different modes of the next word distribution. The prediction target is boldfaced in the context and the predictions. ## indicates there is no space before the word.

Finally, compared with **Softmax**, the mixed results in **SigSoftmax** suggest that adding non-linearity into the softmax layer without modeling the multimodal distribution might not always improve the models [168].

OpenWebText is mostly composed of English text, but some non-English text in the corpus allows us to compare the capability of different models in a multi-lingual setting. Table 2.2 shows that multiple embeddings improve the perplexity of the non-English text more than the perplexity of the English text. We hypothesize that the distribution of the next non-English word is more likely to be multimodal because GPT-2 learns the global token embeddings mostly in the English contexts, which could make the embeddings of similar tokens in non-English contexts far away.

In Table 2.3, we present three contexts from the validation set of different datasets and compare the top three predictions of **MFS** and **Softmax** on *GPT-2 Small*. In OpenWebText and Wikipedia 2021, we can see that **Softmax** misses the correct answer in its top three predictions.

## 2.4 Dynamic Partitioning

Given the context *"Choosing between **John** and **Alex**, I decided to first talk to [MASK]"*, the LMs might output neither *"John"* nor *"Alex"* as the next word. The tendency of the

LMs would make its generated story keeps introducing the new characters or even become incoherent and inconsistent [165, 198].

We suspect that the hallucination problem of LMs partially comes from the softmax bottleneck. The embeddings of character names are usually similar to each other in the word embedding space because they tend to co-occur with similar contexts. If many names are approximately linear dependent, Theorem 2 shows that LMs cannot assign high probabilities only to an arbitrary subset of the names.

In the above sections, we show that the multiple softmaxes and multiple partitions could alleviate the issue. However, computing multiple softmaxes is time-consuming, and using the multiple static random partitions has several limitations:

- If a LM outputs a facet embedding with a higher magnitude, all the words in the same partition with positive logits would have higher probabilities. Predicting the probabilities of a random set of words might result in overfitting.
- The multiple partitions break the structure of global word embeddings. Using many partitions, we lose lots of similarity information between words in different partitions. On the other hand, using too few partitions, we cannot make sure the candidates and the interfering words are in different partitions.
- In MFS, we combine the multiple partitions with the multiple softmaxes as a remedy for breaking the global word embedding structure, but using multiple softmaxes is slow and does not completely solve the problems (see appendix A.2.1 and A.2.2 for examples).

### 2.4.1 Method

To overcome the problems, we propose to partition the vocabulary in dynamic ways. One of its benefits is that we can avoid constantly breaking the global similarity information among the word embeddings without using the expensive multiple softmaxes. In this chapter, we study different ways to construct the partitions, including using the context words to form a partition and putting the most likely next words in another partition. We also propose to

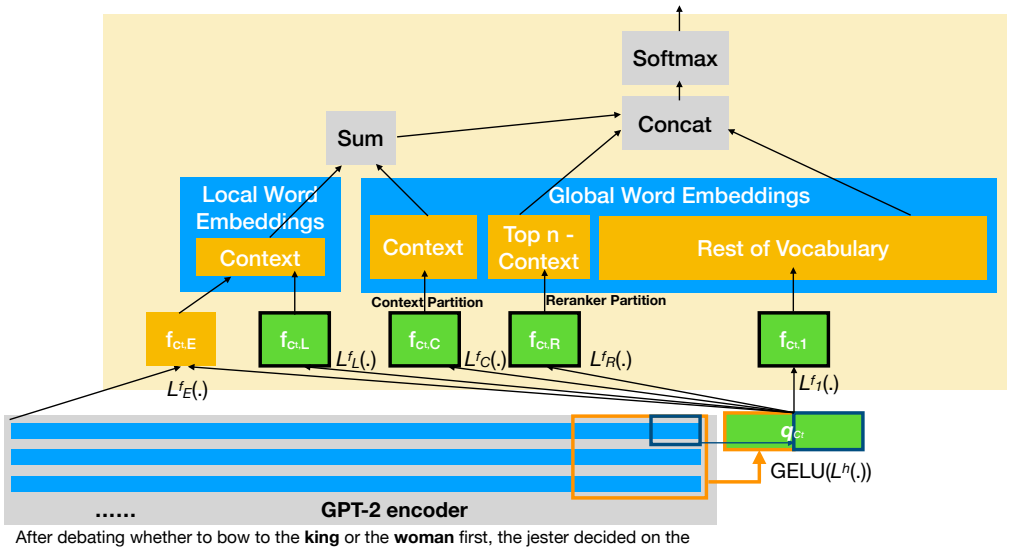



Figure 2.5: Architecture of dynamic partitioning that computes $\text{Logit}_{CLR}$ in Equation 2.11. The architecture combines the idea of the context partition, reranker partition, and local context word embedding. *Top n - Context* means the embeddings of the top n prediction words that are not in the context.

predict the embedding of the context words to further improve the performance. The results of our experiments demonstrate the effectiveness of dynamic partitioning approaches.

#### 2.4.1.1 Context Partition

In our previous examples, we notice that the desired candidates such as *woman*, *king*, *John*, and *Alex* often appear in the context and we won't see the interfering words such as *man*, *queen*, or other names in the context. We can predict a facet embedding for a partition that only contains the words in context. As a result, the magnitude of the facet corresponds to the prior probability of copying the words from the context.

Specifically, the logit of the word $x$ given the context $c_t$ is computed by

$$\text{Logit}_C(x, c_t) = \begin{cases} \boldsymbol{f}_{c_t,C}^T \boldsymbol{w}_x & \text{if } x \in c_t \\ \boldsymbol{f}_{c_t,1}^T \boldsymbol{w}_x & \text{O/W} \end{cases}, \tag{2.5}$$

where $\boldsymbol{f}_{c_t,C} = L^f_C(\boldsymbol{q}_{c_t})$ is the linear projection of the hidden state concatenation $\boldsymbol{q}_{c_t}$ in Equation 2.3. We use only one softmax to compute the final probability of predicting the word $x$:

$$P_{DP}(x|c_t) = \frac{\exp(\text{Logit}_{DP}(x, c_t))}{\sum_{x'} \exp(\text{Logit}_{DP}(x', c_t))}, \tag{2.6}$$

where DP is a dynamic partitioning approach. For example, when computing the probability using the context partition $P_C(x|c_t)$, $\text{Logit}_{DP} = \text{Logit}_C$.

Notice that although the possible next words have been mentioned in the context in many ambiguous contexts, this is not always the case. For example, in the context *My favorite actor is Ryan [MASK]*, the next word could be *Reynolds*, *Gosling*, or the last names of other *Ryan*. Hence, using only the context partition does not completely solve the multimodal distribution problem.

#### 2.4.1.2 Reranker Partition

Some candidate words that the LM wants to output might not be in the context, but their probabilities are usually higher than most of the words. Inspired by the idea of the reranker, we first retrieve the top $n$ words with the highest logits as the set $W_{c_t}(n)$ using the facet $\boldsymbol{f}_{c_t,1}$. Next, we use the $n$ words to form a partition and update the logits of the $n$ words by the dot products between their word embeddings and a new facet $\boldsymbol{f}_{c_t,R} = L^f_R(\boldsymbol{q}_{c_t})$.

Similar to the context partition, we can compute the logit of $x$ using

$$\text{Logit}_R(x, c_t) = \begin{cases} \boldsymbol{f}^T_{c_t,R}\boldsymbol{w}_x & \text{if } x \in W_{c_t}(n) \\ \boldsymbol{f}^T_{c_t,1}\boldsymbol{w}_x & \text{O/W} \end{cases}. \tag{2.7}$$

Our hypothesis is that after using the reranker partitions, LMs could output the embeddings that are more likely to be the average of all the possible answers without worrying about being accidentally close to other interfering words that are not in the top $n$ word list.

When $n$ is small, the reranker partition might not include the very likely next word. When $n$ is large, the reranker partition might not be able to separate the output candidates

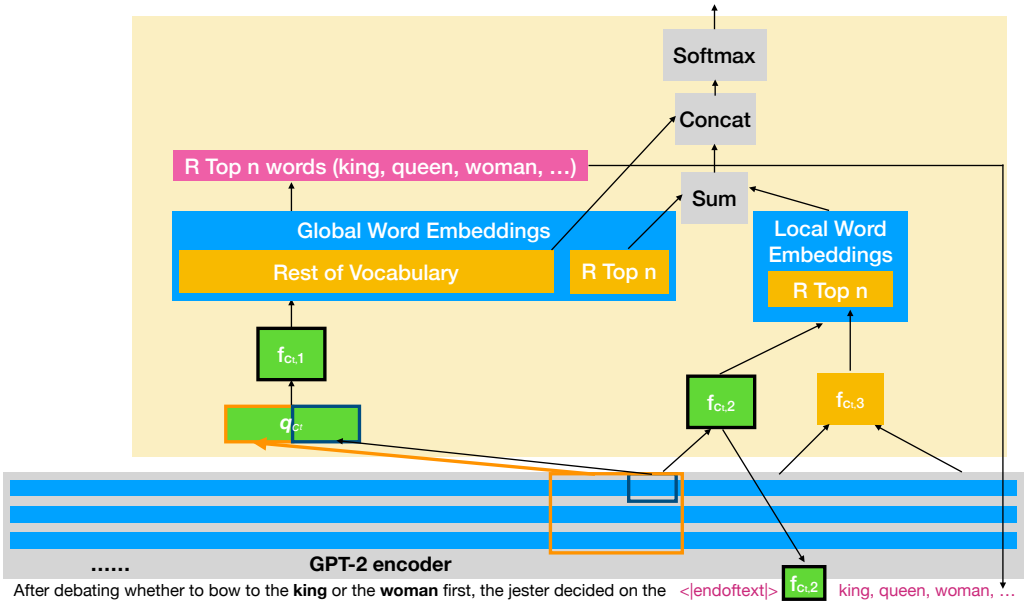



Figure 2.6: Two-stage reranker baseline architecture. This architecture performs similarly compared to the reranker partition approach but requires much more training time.

and the interfering words. To alleviate the problem, we can have multiple reranker partitions and use different facet embeddings for different partitions. Then, we can include more words in the reranker partitions while the more likely words won't be affected by the less likely words. If we use 3 reranker partitions with sizes of $n_1$, $n_2$, and $n_3$, and $n_1 < n_2 < n_3$, the logit of $x$ becomes

$$\text{Logit}_R(x, c_t) = \begin{cases} \boldsymbol{f}_{c_t,R1}^T \boldsymbol{w}_x & \text{if } x \in W_{c_t}(n_1) \\ \boldsymbol{f}_{c_t,R2}^T \boldsymbol{w}_x & \text{if } x \notin W_{c_t}(n_1) \wedge x \in W_{c_t}(n_2) \\ \boldsymbol{f}_{c_t,R3}^T \boldsymbol{w}_x & \text{if } x \notin W_{c_t}(n_1) \wedge x \notin W_{c_t}(n_2) \wedge x \in W_{c_t}(n_3) \\ \boldsymbol{f}_{c_t,1}^T \boldsymbol{w}_x & \text{O/W} \end{cases}, \tag{2.8}$$

where $W_{c_t}(n_1)$ is the set of top $n_1$ words with the highest logits, and the logits are computed by the facet embeddings for the partitions with the size larger than $n_1$ (i.e., $\boldsymbol{f}_{c_t,R2}^T \boldsymbol{w}_x$, $\boldsymbol{f}_{c_t,R3}^T \boldsymbol{w}_x$, and $\boldsymbol{f}_{c_t,1}^T \boldsymbol{w}_x$ in this case), and so on.

#### 2.4.1.3 Two-stage Reranker

To know the effectiveness of this new facet on adjusting the probabilities of the top $n$ words, we also test the traditional two-stage reranker/verifier on GPT-2. After retrieving the top $n$ words using the first facet $\boldsymbol{f}_{c_t,1}$, the two-stage reranker appends the output facet $\boldsymbol{f}_{c_t,2}$ for the context words[4] and candidates to the input context. Next, the reranker uses their hidden states to update their word embeddings as $\boldsymbol{f}_{c_t,3}$ and re-estimate the probabilities of top $n$ words as shown in Figure 2.6. In our implementation, training the two-stage reranker that reranks top 20 words in each position is at least 5 times slower than training GPT-2.

#### 2.4.1.4 Local Context Word Embedding

Although the dynamic partitioning alleviates the problem of softmax bottleneck, we still use the global word embedding. Some sentences in the context might change the similarity between words [200, 198]. For example, if the context contains *"My father is my friend"*, *"father"* and *"friend"* should become more similar. Given the context *"John and Mary are good guys. Alex and Jane are bad guys. The person who attacks me is [MASK]"*, *"Alex"* and *"Jane"* should become more similar in some directions in the word embedding space that corresponds to the bad things.

One solution is to predict the embeddings of the words in the context as in the two-stage reranker. In contrast to the two-stage reranker, this solution does not need to feed the context words into GPT-2 again. Instead, we only use another linear layer $L_E^f()$ to convert the hidden states of the context into the local context word embeddings, which minimizes the extra computational cost. We compute the local embedding of the context word $x$ by averaging all the linear transformation of hidden state concatenations $\boldsymbol{q}_{c_t^i}$ that correspond to the word $x$:

$$\boldsymbol{f}_{c_t,E,x} = \frac{\sum_{i=1}^{t} \mathbb{1}_{c_t^i=x} L_E^f(\boldsymbol{q}_{c_t^i})}{\sum_{i=1}^{t} \mathbb{1}_{c_t^i=x}}, \tag{2.9}$$

[4]The motivation is helping GPT-2 to output the local word embedding of a candidate closer to the current output facet if GPT-2 wants to increase the probability of the candidate.

where $c_t^i$ is the $i$th input words in the context $c_t$, and $\mathbb{1}_{c_t^i=x} = 1$ if $c_t^i = x$. Then, we can use the local context word embeddings to compute the logits:

$$\text{Logit}_L(x, c_t) = \begin{cases} \boldsymbol{f}_{c_t,L}^T \boldsymbol{f}_{c_t,E,x} + \boldsymbol{f}_{c_t,1}^T \boldsymbol{w}_x & \text{if } x \in c_t \\ \boldsymbol{f}_{c_t,1}^T \boldsymbol{w}_x & \text{O/W} \end{cases}, \tag{2.10}$$

where $\boldsymbol{f}_{c_t,L} = L_L^f(\boldsymbol{q}_{c_t})$.

Our local word embedding approach is similar to the pointer network [74, 146, 72, 191]. Our approach also compares the current hidden state and the previous hidden states to estimate the probability of copying the words from the context. Nevertheless, there are still several minor differences between our local embedding method and pointer networks. Hence, we implement CopyNet [74], Pointer Generator [191], and Pointer Sentinel Network [146] on top of GPT-2 and compare their performance.

#### 2.4.1.5 Hybrid Approach

We find that combining the local context embeddings, context partitions, and reranker partition leads to a good result with only modest computational cost. As shown in Figure 2.5, we add the logits from global embeddings $\boldsymbol{f}_{c_t,C}^T \boldsymbol{w}_x$ and the logits from local embedding $\boldsymbol{f}_{c_t,L}^T \boldsymbol{f}_{c_t,E,x}$ together if the word is in the context. If the word is in the top $n$ word list (assuming we only use one reranker partition) but not in the context word, we use the facet for the reranker partition $\boldsymbol{f}_{c_t,R}$ to compute the logit. Finally, the logits of the rest of the words are computed using the facet $\boldsymbol{f}_{c_t,1}$:

$$\text{Logit}_{CLR}(x, c_t) = \begin{cases} \boldsymbol{f}_{c_t,C}^T \boldsymbol{w}_x + \boldsymbol{f}_{c_t,L}^T \boldsymbol{f}_{c_t,E,x} & \text{if } x \in c_t \\ \boldsymbol{f}_{c_t,R}^T \boldsymbol{w}_x & \text{if } x \notin c_t \wedge x \in W_{c_t}(n) \\ \boldsymbol{f}_{c_t,1}^T \boldsymbol{w}_x & \text{O/W} \end{cases}. \tag{2.11}$$

To keep the modified softmax layer initially working almost the same as the original softmax layer, we initialize the linear transformation weights of $L_L^f()$ and $L_E^f()$ as $10^{-10} \cdot \mathbb{I}$. The linear

weights for the facet embeddings other than for the local word embeddings are initialized as the identity matrix $\mathbb{I}$.

We can also combine the dynamic partitioning methods with multiple softmaxes or a two-stage reranker at the cost of significantly increased computation time. When combining with multi-softmax in our experiments, we only use the dynamic partitioning in the first softmax. Besides, we use the two-stage reranker to revise the logits from the dynamic partitioning methods.

### 2.4.2 Experiments

We conduct two experiments to verify the effectiveness of our dynamic partitioning approaches in Wikipedia 2021. Both experiments train the models built on GPT-2 Small. The language modeling experiment compares the perplexity for the next word prediction. Since training the two-stage reranker for GPT-2 is very time-consuming, we only report its performance after be trained for 0.15 epoch. To check whether the context partition and multiple embeddings alleviate the new entity problem, the language generation experiment compares the ROUGE 1 F1 scores between the generated text and the context words or ground truth.

The results of the language modeling experiment are presented in Table 2.4. As we can see, combining all the dynamic partition approaches (i.e., context partition, reranker partition, and local word embedding) results in good performance. The trend after training for 0.15 epochs and 0.4 epochs is the same. If only using one method, the context partition is better than the two-stage reranker and reranker partition. After using all three dynamic partitioning methods, adding the second-stage reranker or multiple softmaxes only improves the performance a little, which suggests that the dynamic partitioning approaches can achieve similar improvements compared to much more computationally expensive softmax alternatives. Furthermore, the performance of the second-stage reranker using a sufficiently

| Training Epoch | Model Names | Configuration #S | #I | Context | Local Emb | Top $n$ | 2nd Stage | Wiki PPL |
|---|---|---|---|---|---|---|---|---|
| | Softmax | 1 | 1 | | | | | 29.53 |
| | Multi-softmax (MoS) | 3 | 1 | | | | | 29.34 |
| | CopyNet [74] | 1 | 1 | | V | | | 29.57 |
| | Pointer Generator [191] | 1 | 1 | | V | | | 29.07 |
| | Pointer Sentinel [146] | 1 | 1 | | V | | | 29.09 |
| | Softmax + Mi | 1 | 9 | | | | | 29.33 |
| | Multi-Softmax + Mi | 3 | 9 | | | | | 29.06 |
| | Softmax + R:100 + Mi | 1 | 9 | | | 100 | | 29.13 |
| 0.15 | Softmax + R:20,100,500 + Mi | 1 | 9 | | | 20,100,500 | | 29.05 |
| | Softmax + Rs2:100 + Mi | 1 | 9 | | | | V | 28.89 |
| | Softmax + C + Mi | 1 | 9 | V | | | | 28.76 |
| | Softmax + L + Mi | 1 | 9 | | V | | | 28.94 |
| | CopyNet [74] + Mi | 1 | 9 | | V | | | 29.34 |
| | Pointer Generator [191] + Mi | 1 | 9 | | V | | | 28.79 |
| | Pointer Sentinel [146] + Mi | 1 | 9 | | V | | | 28.74 |
| | Softmax + CLR:20,100 + Mi | 1 | 9 | V | V | 20,100 | | 28.46 |
| | Softmax + CLR:20,100 + Rs2:100 + Mi | 1 | 9 | V | V | 20,100 | V | 28.40 |
| | Multi-softmax + CLR:20,100 + Mi | 3 | 9 | V | V | 20,100 | | **28.38** |
| | Softmax | 1 | 1 | | | | | 28.19 |
| | Softmax + Mi | 1 | 9 | | | | | 28.05 |
| | Softmax + R:20 + Mi | 1 | 9 | | | 20 | | 27.84 |
| | Softmax + C + Mi | 1 | 9 | V | | | | 27.53 |
| 0.4 | Softmax + CL + Mi | 1 | 9 | V | V | | | 27.43 |
| | Softmax + CLR:20 | 1 | 1 | V | V | 20 | | 27.50 |
| | Softmax + CLR:20 + Mi | 1 | 9 | V | V | 20 | | 27.22 |
| | Softmax + CLR:100 + Mi | 1 | 9 | V | V | 100 | | 27.19 |
| | Multi-softmax + CLR:100 + Mi | 3 | 9 | V | V | 100 | | **27.05** |

Table 2.4: Comparison of perplexity (PPL) in a validation set of Wikipedia 2021 with 100k tokens. All models are built on GPT-2 Small, including CopyNet, Pointer Generator, and Pointer Sentinel. *R:100* means the reranker partition using $n = 100$. *Rs2:100* means the 2nd stage reranker uses $n = 100$. *R:20,100,500* means three layers of the ranker partition whose $n_1$, $n_2$, and $n_3$ values are 20,100, and 500, respectively. *C* means a context partition. *L* means a local context embedding. *Mi* means multiple input hiddent state (i.e., *#I > 1*). That is, *Softmax + Mi* refers to *Softmax + Multi-input*.

large $n$ could be viewed as an upper bound of softmax layer alternatives, and the results show that our methods is approaching the upper bound.

Although only using the reranker partition performs worse than only using the two-stage reranker, the reranker partition is much more efficient. We find that $n = 100$ is better than $n = 20$ and three reranker partitions with sizes of 20,100, and 500 perform slightly better than only using one reranker partition with a size of 100.

In addition, the results suggest that using the local context word embeddings to replace the global word embeddings in the context partition slightly degrades the language modeling performance in Wikipedia. We suspect that this demonstrates the usefulness of the global

| Model Name | All | | Uppercase | | Proper Noun | |
|---|---|---|---|---|---|---|
| | Context | Cont. | Context | Cont. | Context | Cont. |
| Softmax + Mi | 25.73 | 25.33 | 14.22 | 6.99 | 13.20 | 6.65 |
| Multi-softmax (MoS) | 25.40 | 25.16 | 14.52 | 7.28 | 13.51 | 6.73 |
| Multi-softmax + Mi | 25.82 | 25.44 | 16.78 | **8.61** | 15.65 | 7.87 |
| Softmax + C + Mi | 26.38 | **25.61** | 17.73 | 8.17 | 16.66 | 7.55 |
| Multi-softmax + C + Mi | **26.42** | 25.34 | **17.91** | 8.43 | **16.69** | **7.98** |

Table 2.5: Comparison of ROUGE 1 F1 between the generated text and the context or continuation. We present the ROUGE scores of all tokens, uppercased tokens, and proper nouns. *Multi-softmax* is the same as *MoS* and *Multi-softmax + Mi* is the same as *MFS w/o Multi-partition*.

similarity structure of word embeddings. The context partition, local word embedding, CopyNet [74], Pointer Generator [191], and Pointer Sentinel [146] bring similar improvements, which suggests that the main improvement of pointer networks on a transformer comes more from breaking the softmax bottleneck rather than comparing the hidden states at different positions.

In the language generation experiment, we generate 6000 continuations with a length of 50 given the prompts in Wikipedia and see how likely it would copy the words from the context or generate the words that also appear in the actual continuation in Wikipedia. Table 2.5 indicates that adding a context partition makes GPT-2 more likely to copy words from the context, which increases the likelihood of generating the words in the actual continuation, especially when the words are uppercased or proper nouns (i.e., entity names). For example, compared to **Softmax + Mi**, **Softmax + C + Mi** is 26% more likely to copy the proper nouns from the context and 14% more likely to generate the proper nouns in the actual continuation. Multiple softmaxes also increase the chances of copying some names from context, but the method adds much more computational resources and the ROUGE 1 differences are smaller.

In Table 2.6, we qualitatively demonstrate the advantages of the dynamic partitioning methods. The softmax layer of GPT-2 is unable to properly learn to copy or exclude the word from the input context. For example, **GPT-2**, **MoS**, and **Pointer Sentinel + Mi [146]**

| Input Context | I like tennis, baseball, golf, basketball, and | There are plates, keys, scissors, toys, and balloons in front of me, and I pick up the | Choosing between John, Alex, Mary, Kathryn, and Jack, I decided to first talk to |
|---|---|---|---|
| Softmax (GPT-2) | tennis 0.080, golf 0.053, baseball 0.051, basketball 0.049, I 0.032 0.038 | **keys** 0.059, **pieces** 0.057, phone 0.031, **balloons** 0.024, key 0.023 | the 0.121, them 0.088, **John** 0.063, my 0.039, **Alex** 0.032 |
| Multi-softmax (MoS) | tennis 0.068, golf 0.066, basketball 0.050, baseball 0.047, **other** 0.045 | **keys** 0.075, **pieces** 0.050, phone 0.026, key 0.025, **balloons** 0.015 | **John** 0.099, the 0.097, them 0.083, **Alex** 0.055, **Mary** 0.040 |
| Pointer Sentinel + Mi [146] | tennis 0.103, baseball 0.063, golf 0.062, basketball 0.052, **swimming** 0.030 | **keys** 0.078, **plates** 0.044, **scissors** 0.029, **pieces** 0.029, **balloons** 0.029 | the 0.132, them 0.053, my 0.049, **John** 0.044, **Alex** 0.043 |
| Multi-softmax + CLR:100 + Mi | **volleyball** 0.097, **football** 0.085, **soccer** 0.073, **soft** 0.036, **bad** 0.031 | **keys** 0.166, **scissors** 0.049, **toys** 0.042, **balloons** 0.042, **pieces** 0.035 | **John** 0.139, the 0.113, them 0.056, **Alex** 0.051, **Mary** 0.032 |

Table 2.6: Prediction visualization of three input contexts. We show the top five words with the highest prediction probabilities of each model. The reasonable next word predictions are boldfaced.

are very likely to output *I like tennis, baseball, golf, basketball, and tennis*, which causes a repetition problem, while **Multi-softmax + CLR:100 + Mi** can learn to exclude the sports that have been listed. On the other hand, **GPT-2** and **MoS** might output *There are plates, keys, scissors, toys, and balloons in front of me, and I pick up the phone*, which causes a hallucination problem, while **Multi-softmax + CLR:100 + Mi** can learn to copy the words that have been mentioned.

In short, the results suggest that the dynamic partition approaches improve the next word prediction and reduce the chances of keep generating new names. Nevertheless, we so far only use the automatic metric now. We leave a more comprehensive evaluation of the factuality of the generated text as future work.

## 2.5 Applications on Sequential Recommendation

The causal language modeling problem is a special case of the sequential prediction problem. If we replace the input word sequence with the input product sequence, GPT-2 could be used to recommend the next product based on the previous shopping history. Kang and McAuley [96] found that the architecture used in GPT-2 also reaches state-of-the-

art performance in sequential recommendation datasets and they call the model SASRec (self-attentive sequential recommender).

In many sequential recommendation tasks, it is important to predict if the user would buy the same product again. For example, one user might buy the same kind of snack again and again or the user is not likely to watch a movie if the user has seen the movie before. In many state-of-the-art recommendation systems, these tendencies are often considered using some post-processing rules. For example, in the grocery category, the recently-bought products should be shown to the users; while in the movie recommendation, the recently-seen movie should be excluded from the recommendation list.

Setting the business intelligence rules are tedious and might not result in an optimal performance. For example, a user usually won't buy the same book twice. However, when the user starts to buy a book twice, it might indicate that the user wants to give the book to other people as a gift. This means that we should continue recommending such book. Hence, properly learning to copy or not copy the items like our method should be a more ideal solution. The proposed dynamic partitioning approaches address this need. If we often observe the duplicated products in a user's shopping record, the encoder can learn to always output a facet embedding with high magnitude for the context partition, and vice versa.

### 2.5.1 Experiment Setup

We test our method on the four datasets: *ML-1m* [80], *Amazon-beauty* [143], *Steam* [96], and *Retailrocket*[5], where *ML-1m* is the MovieLens dataset with one million user ratings. We use Hit@10 and NDCG@10 as our metrics. Our negative samples are all the other possible items in the dataset, so the performance in each dataset highly depends on the number of unique items in the dataset. Thus, we report the geometric average rather than the arithmetic average to summarize their results.

[5] `www.kaggle.com/retailrocket/ecommerce-dataset`

RepeatNet [181] improves the performance of sequential recommendation datasets with duplicated items by modifying the CopyNet [74]. One major difference is that their probability of the history items completely comes from the copying mechanism. As a result, their method can learn to copy or exclude the history items as we do, while one drawback is that the output probabilities of the history items cannot leverage the global similarity among the items in the embedding space. Since RepeatNet uses a GRU encoder rather than a transformer, we also apply our approaches to GRU4Rec [85] in order to compare our methods with RepeatNet more fairly.

To keep the capability of using nearest neighbor search during testing time, the multi-softmax in this application uses max-pooling on the logits of every facet rather than using the mixture of softmax. When using multiple input hidden states (*Mi*), we set the H as the maximal layer of the encoder. Specifically, $H = 2$ for SASRec and $H = 1$ for GRU4Rec.

We use the evaluation framework/library called Recbole [259], which allows us to test multiple datasets and settings conveniently and fairly. We use learning rate 0.001 and set the training batch size as 128 for SASRec and as 64 for GRU4Rec and RepeatNet. All the models using their default hyperparameters except that we decrease the dropout of SASRec from 0.5 to 0.2 because we find that the change significantly improves the performance.

### 2.5.2 Results

By comparing **Softmax + C** and **Softmax** in Table 2.7, we can see that the context partition improves the NDCG@10 of SASRec by 13% and GRU4Rec by 11% on average. In *ML-1m* and *Amazon-beauty* datasets, one user does not interact with the same item twice, but the context partition still improves the performance. This shows that softmax has difficulty in learning to prevent copying products from the history, and the context partition can solve the problem.

Similar to Table 2.4, **RepeatNet** is slightly worse than only using the context partition **Softmax + C**, while **Softmax + CLR** usually further improves **Softmax + C**. Across the

| Model ↓ | Final Layer ↓ | ML-1m | | Amazon-beauty | | Steam | |
|---|---|---|---|---|---|---|---|
| | | Hit@10 | NDCG@10 | Hit@10 | NDCG@10 | Hit@10 | NDCG@10 |
| SASRec | Softmax | 29.65 | 16.46 | 3.64 | 2.01 | 21.81 | 16.86 |
| | Softmax + Mi | 29.92 | 16.79 | 3.50 | 1.84 | 22.80 | 17.58 |
| | Multi-Softmax (MoS) | 29.14 | 16.34 | 3.43 | 1.80 | 23.30 | 18.16 |
| | Softmax + C | 34.22 | 20.53 | 3.86 | 2.35 | 23.65 | 18.32 |
| | Softmax + C + Mi | 34.74 | 20.68 | 3.50 | 2.10 | **24.03** | **18.63** |
| | Softmax + CLR | 34.90 | **20.95** | 3.93 | 2.34 | 23.76 | 18.44 |
| | Softmax + CLR + Mi | **35.35** | **20.95** | 3.77 | 2.28 | 23.96 | 18.59 |
| | Multi-Softmax + CLR | 32.67 | 19.20 | - | - | 23.91 | 18.52 |
| GRU4Rec | Softmax | 29.29 | 15.94 | 3.35 | 1.90 | 22.71 | 17.48 |
| | Softmax + Mi | 29.55 | 16.45 | 3.85 | 2.08 | 21.83 | 16.93 |
| | Multi-Softmax (MoS) | 28.54 | 15.81 | 3.82 | 2.04 | 22.43 | 17.69 |
| | Softmax + C | 32.68 | 18.99 | 3.90 | 2.27 | 23.25 | 18.16 |
| | Softmax + C + Mi | 35.05 | 20.73 | 3.74 | 2.15 | 23.40 | 18.27 |
| | Softmax + CLR | 33.69 | 19.88 | 4.36 | 2.51 | 23.29 | 18.25 |
| | Softmax + CLR + Mi | 33.94 | 20.13 | **4.39** | **2.55** | 23.39 | 18.27 |
| | Multi-Softmax + CLR | 33.69 | 19.70 | - | - | 23.21 | 18.24 |
| | RepeatNet | 32.80 | 19.69 | 3.42 | 2.12 | 23.38 | 18.29 |

| Model ↓ | Final Layer ↓ | Retailrocket | | G mean | |
|---|---|---|---|---|---|
| | | Hit@10 | NDCG@10 | Hit@10 | NDCG@10 |
| SASRec | Softmax | 50.94 | 37.97 | 18.61 | 12.06 |
| | Softmax + Mi | 52.18 | 38.77 | 18.79 | 12.05 |
| | Multi-Softmax (MoS) | 51.47 | 38.28 | 18.61 | 11.96 |
| | Softmax + C | 52.09 | 39.65 | 20.08 | 13.68 |
| | Softmax + C + Mi | 52.24 | 39.49 | 19.77 | 13.37 |
| | Softmax + CLR | 52.49 | 40.07 | 20.34 | 13.80 |
| | Softmax + CLR + Mi | 53.10 | 40.24 | 20.29 | 13.75 |
| | Multi-Softmax + CLR | 53.21 | 40.11 | - | - |
| GRU4Rec | Softmax | 52.12 | 38.58 | 18.46 | 11.95 |
| | Softmax + Mi | 50.92 | 37.81 | 18.86 | 12.17 |
| | Multi-Softmax (MoS) | 50.94 | 37.73 | 18.79 | 12.11 |
| | Softmax + C | 52.30 | 39.90 | 19.84 | 13.29 |
| | Softmax + C + Mi | 52.55 | 40.09 | 20.04 | 13.44 |
| | Softmax + CLR | **53.48** | **40.62** | 20.68 | 13.87 |
| | Softmax + CLR + Mi | 53.03 | 40.33 | **20.73** | **13.95** |
| | Multi-Softmax + CLR | 52.56 | 40.15 | - | - |
| | RepeatNet | 52.34 | 40.54 | 19.25 | 13.26 |

Table 2.7: The validation scores of the four datasets and their geometric average (*G mean*). In all the model names with *R*, we use 3 reranker partitions with $n_1 = 20$, $n_2 = 100$, $n_3 = 500$. Due to the memory constraints of our GPU, we haven't had the results of *Multi-Softmax + CLR* at *Amazon-beauty*. Other notations are the same as the notations used in Table 2.4.

datasets, the improvements are pretty consistent no matter if we use SASRec or GRU4Rec, if we use the multiple input hidden states, and if the performance is measured by Hit@10 or NDCG@10.

We find that using multiple input hidden states (+ *Mi*) improves the performance of three out of four datasets. We suspect that the results are sometimes worse because the

models overfit the sequential recommendation datasets that are much smaller than the NLP pretraining corpus. Finally, the effectiveness of multiple softmaxes (*Multi-Softmax*) depends on the dataset and the encoder. For example, **Multi-Softmax (MoS)** significantly outperforms **Softmax** in *Steam* when we use the SASRec encoder but the improvement becomes inconsistent when we use the GRU4Rec encoder.

## 2.6 Related Work

Yang et al. [248] propose the concept of *softmax bottleneck*, which points out that the dot product in the softmax layer restricts the representation power of outputting arbitrary conditional probabilities. It also proposes *MoS* to break the softmax bottleneck in an RNN-based LM. Kanai et al. [94] and Ganea et al. [66] add nonlinearities into the softmax layer to break the bottleneck more efficiently, but the approaches gain less improvement compared to *MoS*.

A limitation of the aforementioned previous work is that they do not tell us which kinds of sentences would be affected by the bottleneck more and whether the order of the top few next words would be affected, which are main research questions of our work. Contrary to the previous belief that a large hidden state dimension would eliminate the softmax bottleneck, our theorems and empirical analyses suggest that some words in a low dimensional subspace could still make the single embedding in the softmax layer become a bottleneck of arbitrarily ranking the output words. Furthermore, our geometric analyses provide an intuitive explanation about why breaking the bottleneck using multiple embeddings leads to better performance compared to only adding the non-linearity.

Demeter et al. [45] also analyze the structural weakness of the softmax layer from a geometric perspective. They discover that the words with high prior frequencies could stop the LMs from assigning the high probabilities to rare words, which can be viewed as a special case of our theory (See Appendix A.4). For instance, our work shows that the

softmax layer could still prevent the LMs from outputting some top words even if all the possible next words have the same prior frequency.

Our theory is deeply connected to the mathematical work that counts the number of possible rankings of points in an embedding space [43, 71]. Compared to the studies, our work focuses more on analyzing the multimodal distribution in the word embedding space and its implication to language models.

An alternative to model the multimodal distribution is to use multiple embeddings to represent each output word [9, 147]. Compared to MoS or our approach that use multiple embeddings to represent each hidden state of the context, their methods require many extra parameters to store different senses of each output word. Another type of related model [196, 62] dynamically routes the signals to different experts (i.e., feed-forward networks) and Zhang et al. [258], Mittal et al. [154] use multiple embeddings in the attention layers. The methods are similar to MoS and our approach, but they add the multiple embeddings inside each layer of the transformer encoder while the proposed MFS is an alternative to the output softmax layer.

Shwartz et al. [200] find that the text generator would be affected the global attributes of characters' names. Recently, Papalampidi et al. [165] and Shuster et al. [198] report the incoherence and inconsistency of entities in language generation. Their mitigation ways include viewing entity consistency as a controllable attribute [198] and tracking the entity information in a longer context using a memory network [165]. However, the approaches assume that the locations of entities have been known and need to add significant computational overhead on top of the language generation model.

Our reranker approaches revise the generated next word, so they are related to other ways of revising the text. Examples include using GAN [241], beam search and rerank in translation [160], plug-and-play LM [44], iterative editing [11], coarse-to-fine generation [216], and non-autoregressive generation [121, 190]. Although some of the approaches can revise the text toward a more consistent text as in Shuster et al. [198], they often need

to significantly increase the inference time. Furthermore, it remains unknown whether the approaches can be used to mitigate the inconsistency problems of entities and improve the unconditional and self-supervised language modeling without prior knowledge.

## 2.7 Chapter Conclusion

We summarize our theoretical, methodological, evaluational, and analytic contributions in this chapter as follows.

- **Theory**: We show the softmax layer using a single embedding is sometimes not be able to output an appropriate rank of probabilities on a set of words with linearly dependent embeddings.
- **Method**: We propose *multi-facet softmax* (MFS) to improve MoS [248], and further propose dynamic partitioning approaches that are better than MFS while being much more efficient. All the proposed softmax alternatives can better handle ambiguous contexts without re-training the LMs from scratch.
- **Evaluation**: In addition to achieving better perplexities, we show that dynamic partitioning approaches can reduce the chance of generating new entities. Furthermore, we demonstrate that our dynamic partitioning approaches can also significantly improve sequential recommendation datasets.
- **Analysis**: Our comprehensive empirical analyses discover and explain several phenomena, such as a) using multiple embeddings is usually better than the single embedding with the non-linearity because the multiple embeddings can capture the multimodal distribution. b) the improvement is larger in ambiguous contexts or less common languages because multimodal distributions are more prevalent given those contexts. c) the pointer networks [191, 146] can significantly improve the transformer-based language model. Furthermore, we can achieve similar improvement if we replace the hidden states of the context words with their global word embedding. The findings suggest that the major

source of the improvement comes from reducing the necessity of modeling multimodal distributions.

# CHAPTER 3

# IMPROVEMENT UNSUPERVISED SIMILARITY ESTIMATION BY PREDICTING CLUSTER CENTERS

## 3.1 Introduction

In Chapter 2, we focus on breaking the softmax bottleneck to improve the next item prediction for various applications. Since most of the input sequences only appear once in the corpus, we usually can only observe one ground truth next word for each context. The chance of observing the multimodal co-occurrence distribution in the training corpus is sometimes limited.

In some co-occurrence learning paradigms, one input sequence often co-occurs with a set of items. For example, one phrase co-occurs with a set of words in the nearby context; one sentence co-occurs with words in the nearby sentences; one sentence pattern (e.g., *"$ARG1, the partner $ARG2"*) could co-occur with several entity pairs; one target paper could co-occur with the citing papers that cite the target paper. When learning to predict the co-occurrence, the common multimodal distribution makes the motivation of using multiple embedding representations even stronger.

In this chapter, our goal is to learn multi-facet embeddings where each embedding is a clustering center of the items co-occurring with the input word sequence. For example, the multi-facet representation of *real property* is illustrated in Figure 3.1. Real property can be observed in legal documents where it usually means real estate, while *real property* can also mean a true characteristic in philosophic discussions.

To highlight the advantage of predicting multiple embeddings in this kind of co-occurrence learning, we first explain the difficulties faced by the classic clustering-based

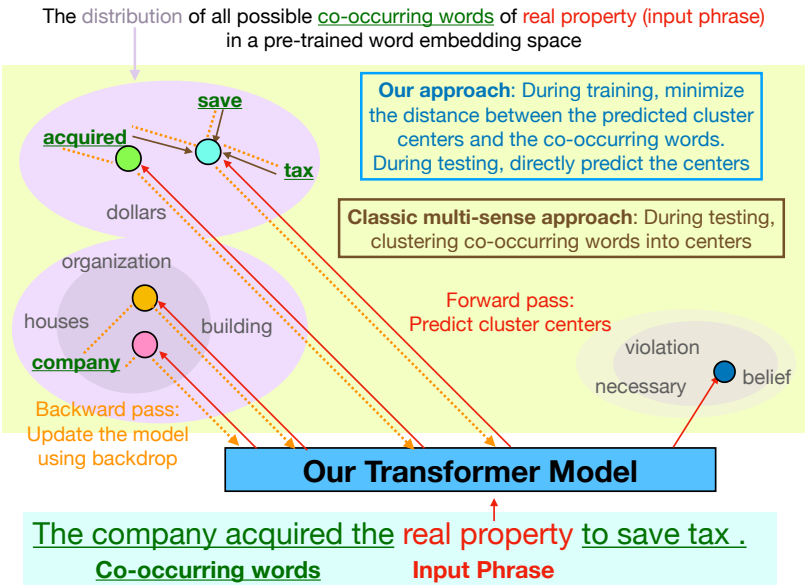



Figure 3.1: The input phrase *real property* is represented by $K = 5$ cluster centers. The previous work discovers the multiple senses by clustering the embedding of observed co-occurring words. Instead, our compositional model learns to predict the embeddings of cluster centers from the sequence of words in the input phrase so as to reconstruct the (unseen) co-occurring distribution well.

approaches. The previous unsupervised multi-sense embeddings [111, 159, 9, 202] discover those senses by clustering the observed neighboring words (e.g., *acquired*, *save*, and *tax*) and an important facet, a mode with high probability, could be represented by several close cluster centers. Notice that the approaches need to solve a distinct local clustering problem for each phrase in contrast with the topic modeling like LDA [19], which clusters all the words in the corpus into a global set of topics.

In addition to a phrase, we can also cluster the nearby words of a sentence which appears frequently in the corpus. The cluster centers usually correspond to important aspects rather than senses (see an example in Figure 3.2) because a sentence usually has multiple aspects but only one sense. However, extending the clustering-based multi-sense word embeddings to long sequences such as sentences is difficult in practice due to two efficiency challenges. First, there are usually many more unique phrases and sentences in a corpus

than there are words, while the number of parameters for clustering-based approaches is $O(|V| \times |K| \times |E|)$, where $|V|$ is the number of unique sequences, $|K|$ is the number of clusters, and $|E|$ is the embedding dimensions. Estimating and storing such a large number of parameters takes time and space. More importantly, much more unique sequences imply much fewer co-occurring words to be clustered for each sequence, especially for sentences. An effective model needs to overcome this sample efficiency challenge (i.e., sparseness in the co-occurring statistics), but clustering approaches often have too many parameters to learn the compositional meaning of each sequence without overfitting.

Nevertheless, the sentences (or phrases) sharing multiple words often lead to similar cluster centers, so we should be able to solve these local clustering problems using much fewer parameters to circumvent the challenges. As shown in Figure 3.1, instead of clustering co-occurring words beside an input sequence at test time as in previous approaches, we learn a mapping between the input sequence (e.g., phrases or sentences) and the corresponding cluster centers during training so that we can directly predict those cluster centers using a single forward pass of the neural network for an arbitrary unseen input sequence during testing.

As in Chapter 2, we use the transformer to predict the multiple embeddings given a text sequence. The main methodology differences are that we use clustering losses and negative sampling instead of softmax to save the computation cost.

When learning the representation of sentences and phrases in Section 3.2, we find that non-negative sparse coding (NNSC) clustering loss would lead to more diverse clustering centers compared to Kmeans. When learning the representation of sentence patterns in Section 3.3, we find that the Kmeans loss performs similarly compared to the NNSC loss in our preliminary experiments, so we choose to use Kmeans due to its efficiency. When predicting citation and authorship in Section 3.4, we also use Kmeans loss to allow the efficient nearest neighbor search. Finally, we find that adding an autoencoder loss could

stabilize the training process of our models for relation extraction and citation/authorship prediction.

To save computational resources and reduce optimization difficulties, all the methods in this chapter do not use pretrained language models (LMs). We demonstrate the effectiveness of our approaches by showing that multiple embeddings are significantly better than the single embedding baselines and the widely-used or state-of-the-art methods that do not use pretrained LMs.

In the following three sections, we describe three one-tower co-occurrence learning models, the intuitive reasons of why multiple embeddings are helpful in the tasks, experiments, and their connections to previous work.

## 3.2 Applications on the Representation of Sentences and Phrases

In this section, we first describe our seq2seq transformer architecture and how to learn multi-facet embedding (i.e., codebook embeddings) using the NNSC clustering loss. Then, we evaluate whether the proposed multi-facet embeddings could improve the similarity measurement between two sentences, between a sentence and a document (i.e., extractive summarization), and between phrases.

The results demonstrate multi-facet embeddings significantly outperforms the classic single embedding baselines when the input sequence is a sentence. We also demonstrate several advantages of the proposed multi-facet embeddings over the embeddings of all the words in a sequence. First, we discover that our model tends to use more embeddings to represent an important facet or important words. This tendency provides an unsupervised estimation of word importance, which improves various similarity measurements between a sentence pair. Second, our model outputs a fixed number of facets by compressing long sentences and extending short sentences. In unsupervised extractive summarization, this capability prevents the scoring function from biasing toward longer or shorter sentences. Finally, our experiments show that multiple embeddings do not improve the unsupervised

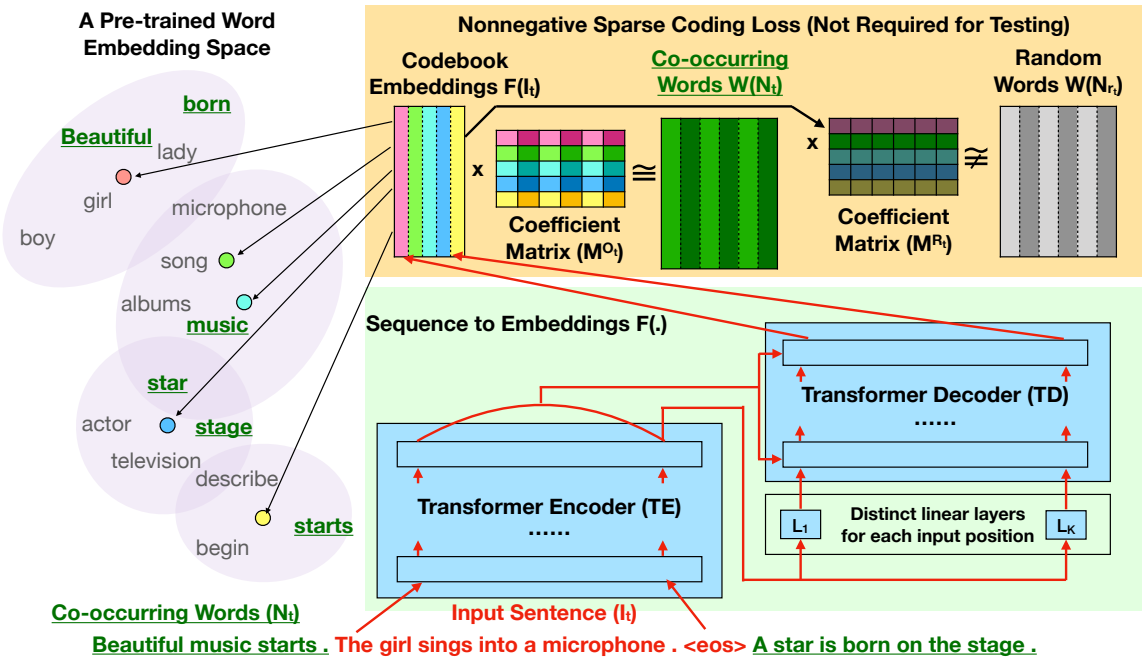



Figure 3.2: Our model for sentence representation. We represent each sentence as multiple codebook embeddings (i.e., cluster centers) predicted by our sequence to embeddings model. Our loss encourages the model to generate codebook embeddings whose linear combination can well reconstruct the embeddings of co-occurring words (e.g., *music*), while not able to reconstruct the negatively sampled words (i.e., the co-occurring words from other sentences).

phrase similarity estimation, which suggests the difficulties of modeling multi-mode co-occurrence distribution of the short input sequences.

For details of our methods and experiments, please see the appendix of Chang et al. [34].

### 3.2.1 Method

In this section, we first formalize our training setup and next describe our objective function and neural architecture. Our approach is visually summarized in Figure 3.2.

#### 3.2.1.1 Self-supervision Signal

We express $t$th sequence of words in the corpus as $I_t = w_{x_t}...w_{y_t}$<eos>, where $x_t$ and $y_t$ are the start and end position of the input sequence, respectively, and <eos> is the end of sequence symbol.

We assume neighboring words beside each input phrase or sentence are related to some facets of the sequence, so given $I_t$ as input, our training signal is to reconstruct a set of co-occurring words, $N_t = \{w_{x_t - d_1^t}, ...w_{x_t - 1}, w_{y_t + 1}, ...w_{y_t + d_2^t}\}$.[1] In our experiments, we train our multi-facet sentence embeddings by setting $N_t$ as the set of all words in the previous and the next sentence, and train multi-facet phrase embeddings by setting a fixed window size $d_1^t = d_2^t = 5$.

Since there are not many co-occurring words for a long sequence (none are observed for unseen testing sequences), the goal of our model is to predict the cluster centers of the words that could "possibly" occur beside the text sequence rather than the cluster centers of the actual occurring words in $N_t$ (e.g., the hidden co-occurring distribution instead of green and underlined words in Figure 3.2). The cluster centers of an unseen testing sequence are predictable because the model could learn from similar sequences and their co-occurring words in the training corpus.

To focus on the semantics rather than syntax, we view the co-occurring words as a set rather than a sequence as in skip-thoughts [102]. Notice that our model considers the word order information in the input sequence $I_t$, but ignores the order of the co-occurring words $N_t$.

#### 3.2.1.2 Non-negative Sparse Coding Loss

In a pre-trained word embedding space, we predict the cluster centers of the co-occurring word embeddings. The embeddings of co-occurring words $N_t$ are arranged into a matrix $\boldsymbol{W(N_t)} = [\underline{\boldsymbol{w}}_j^t]_{j=1...|N_t|}$ with size $|E| \times |N_t|$, where $|E|$ is the dimension of pre-trained word embedding, and each of its columns $\underline{\boldsymbol{w}}_j^t$ is a normalized word embedding whose 2-norm is 1. The normalization makes the cosine distance between two words become half of their squared Euclidean distance.

---

[1] The self-supervised signal is a generalization of the loss for prediction-based word embedding like Word2Vec [148]. They are the same when the input sequence length $|I_t|$ is 1.

Similarly, we denote the predicted cluster centers $\underline{\boldsymbol{c}}_k^t$ of the input sequence $I_t$ as a $|E| \times K$ matrix $\boldsymbol{F}(\boldsymbol{I_t}) = [\underline{\boldsymbol{c}}_k^t]_{k=1...K}$, where $\boldsymbol{F}$ is our neural network model and $K$ is the number of clusters. We fix the number of clusters $K$ to simplify the design of our prediction model and the unsupervised scoring functions used in the downstream tasks. When the number of modes in the (multimodal) co-occurring distribution is smaller than $K$, the model can output multiple cluster centers to represent a mode (e.g., the *music* facet in Figure 3.2 is represented by two close cluster centers). As a result, the performance in our downstream applications is not sensitive to the setting of $K$ when $K$ is larger than the number of facets in most input word sequences.

The reconstruction loss of k-means clustering in the word embedding space can be written as $||\boldsymbol{F}(\boldsymbol{I_t})\boldsymbol{M} - \boldsymbol{W}(\boldsymbol{N_t})||^2 = \sum_j ||(\sum_k \boldsymbol{M}_{k,j}\underline{\boldsymbol{c}}_k^t) - \underline{\boldsymbol{w}}_j^t||^2$, where $\boldsymbol{M}_{k,j} = 1$ if the $j$th word belongs to the $k$ cluster and 0 otherwise. That is, $\boldsymbol{M}$ is a permutation matrix which matches the cluster centers and co-occurring words and allow the cluster centers to be predicted in an arbitrary order.

Non-negative sparse coding (NNSC) [89] relaxes the constraints by allowing the coefficient $\boldsymbol{M}_{k,j}$ to be a positive value but encouraging it to be 0. We adopt NNSC in this work because we observe that the neural network trained by NNSC loss generates more diverse topics than k-means loss does. We hypothesize that it is because the loss is smoother and easier to be optimized for a neural network. Using NNSC, we define our reconstruction error as

$$
\begin{aligned}
&Er(\boldsymbol{F}(\boldsymbol{I_t}), \boldsymbol{W}(\boldsymbol{N_t})) = ||\boldsymbol{F}(\boldsymbol{I_t})\boldsymbol{M}^{\boldsymbol{O_t}} - \boldsymbol{W}(\boldsymbol{N_t})||^2 \\
&s.t., \boldsymbol{M}^{\boldsymbol{O_t}} = \arg\min_{\boldsymbol{M}} ||\boldsymbol{F}(\boldsymbol{I_t})\boldsymbol{M} - \boldsymbol{W}(\boldsymbol{N_t})||^2 + \lambda||\boldsymbol{M}||_1, \forall k, j,\ 0 \leq \boldsymbol{M}_{k,j} \leq 1, \quad (3.1)
\end{aligned}
$$

where $\lambda$ is a hyper-parameter controlling the sparsity of $\boldsymbol{M}$. We force the coefficient value $\boldsymbol{M}_{k,j} \leq 1$ to avoid the neural network learning to predict centers with small magnitudes which makes the optimal values of $\boldsymbol{M}_{k,j}$ large and unstable.

We adopt an alternating optimization strategy similar to the EM algorithm for k-means. At each iteration, our E-step estimates the permutation coefficient $\boldsymbol{M^{O_t}}$ after fixing our neural model, while our M-step treats $\boldsymbol{M^{O_t}}$ as constants to back-propagate the gradients of NNSC loss to our neural network. A pseudo-code of our training procedure could be found in Algorithm 1 in the appendix. Estimating the permutation between the prediction and ground truth words is often computationally expensive [176]. Nevertheless, optimizing the proposed loss is efficient because for each training sequence $I_t$, $\boldsymbol{M^{O_t}}$ can be efficiently estimated using convex optimization (our implementation uses RMSprop [221]). Besides, we minimize the L2 distance, $||\boldsymbol{F(I_t)M^{O_t}} - \boldsymbol{W(N_t)}||^2$, in a pre-trained embedding space as in Kumar and Tsvetkov [109] and Li et al. [117] rather than computing softmax.

To prevent the neural network from predicting the same global topics regardless of the input, our loss function for $t$th sequence is defined as

$$L_t(\boldsymbol{F}) = Er(\boldsymbol{F(I_t)}, \boldsymbol{W(N_t)}) - Er(\boldsymbol{F(I_t)}, \boldsymbol{W(N_{r_t})}), \tag{3.2}$$

where $N_{r_t}$ is a set of co-occurring words of a randomly sampled sequence $I_{r_t}$. In our experiment, we use SGD to solve $\widehat{\boldsymbol{F}} = \arg\min_{\boldsymbol{F}} \sum_t L_t(\boldsymbol{F})$. Our method could be viewed as a generalization of Word2Vec [148] that can encode the compositional meaning of the words and decode multiple embeddings.

#### 3.2.1.3 Sequence to Embeddings

Our neural network architecture is similar to transformer-based sequence to sequence (seq2seq) model [224]. We use the same encoder $TE(I_t)$, which transforms the input sequence into a contextualized embeddings

$$[\underline{\boldsymbol{e}}_{x_t} ... \underline{\boldsymbol{e}}_{y_t} \underline{\boldsymbol{e}}_{\text{<eos>}}] = TE(w_{x_t} ... w_{y_t}\text{<eos>}), \tag{3.3}$$

where the goal of the encoder is to map the similar sentences, which are likely to have similar co-occurring word distribution, to similar contextualized embeddings.

Different from the typical seq2seq model [212, 224], our decoder does not need to make discrete decisions because our outputs are a sequence of embeddings instead of words. This allows us to predict all the codebook embeddings in a single forward pass as in Lee et al. [113] without requiring an expensive softmax layer or auto-regressive decoding.[2]

To make different codebook embeddings capture different facets, we pass the embeddings of <eos>, $\underline{\boldsymbol{e}}_{\text{<eos>}}$, to different linear layers $L_k$ before becoming the input of the decoder $TD$. The decoder allows the input embeddings to attend each other to model the dependency among the facets and attend the contextualized word embeddings from the encoder, $\underline{\boldsymbol{e}}_{x_t}...\underline{\boldsymbol{e}}_{y_t}\underline{\boldsymbol{e}}_{\text{<eos>}}$, to copy the embeddings of some keywords in the word sequence as our facet embeddings more easily. Specifically, the codebook embeddings

$$\boldsymbol{F(I_t)} = TD(L_1(\underline{\boldsymbol{e}}_{\text{<eos>}})...L_K(\underline{\boldsymbol{e}}_{\text{<eos>}}), \underline{\boldsymbol{e}}_{x_t}...\underline{\boldsymbol{e}}_{y_t}\underline{\boldsymbol{e}}_{\text{<eos>}}). \tag{3.4}$$

We find that removing the attention on the $\underline{\boldsymbol{e}}_{x_t}...\underline{\boldsymbol{e}}_{y_t}\underline{\boldsymbol{e}}_{\text{<eos>}}$ significantly deteriorates our validation loss for sentence representation because there are often too many facets to be compressed into a single embedding. On the other hand, the encoder-decoder attention does not significantly change the performance of phrase representation, so we remove the connection (i.e., encoder and decoder have the same architecture) in models for phrase representation.

### 3.2.2 Experiments

Quantitatively evaluating the quality of our predicted cluster centers is difficult because the existing label data and metrics are built for global clustering. The previous multi-sense

[2]The decoder can also be viewed as another transformer encoder which attends the output of the first encoder and models the dependency between predicted cluster centers.

word embedding studies often show that multiple embeddings could improve the single word embedding in the unsupervised word similarity task to demonstrate its effectiveness. Thus, our goal of experiments is to discover when and how the multi-facet embeddings can improve the similarity measurement in various unsupervised semantic tasks upon the widely-used general-purpose representations, such as single embedding and (contextualized) word embeddings.

#### 3.2.2.1 Experiment Setup

Our models only require the raw corpus and sentence/phrase boundaries, so we will only compare them with other unsupervised alternatives that do not require any manual labels or multi-lingual resources such as PPDB [172].

Our model is trained on English Wikipedia 2016 while the stop words are removed from the set of co-occurring words. In the phrase experiments, we only consider noun phrases, and their boundaries are extracted by applying simple regular expression rules to POS tags before training. We use the cased version (840B) of GloVe embedding [174] as the pre-trained word embedding space for our sentence representation and use the uncased version (42B) for phrase representation.[3] To control the effect of embedding size, we set the hidden state size in our transformers as the GloVe embedding size (300).

#### 3.2.2.2 Qualitative Evaluation

The cluster centers predicted by our model are visualized in Table 3.1 (as using *girl* and *lady* to visualize the red cluster center in Figure 3.2).

The centers summarize the input sequence well and more codebook embeddings capture more fine-grained semantic facets of a phrase or a sentence. Furthermore, the embeddings capture the compositional meaning of words. For example, each word in the phrase *civil order* does not mean *initiatives*, *army*, or *court*, which are facets of the whole phrase. When

[3] `nlp.stanford.edu/projects/glove/`

| **Input Phrase**: civil order <eos> |
|---|
| **Output Embedding (K = 1)**: |
| e1 \| government 0.817 civil 0.762 citizens 0.748 |
| **Output Embeddings (K = 3)**: |
| e1 \| initiatives 0.736 organizations 0.725 efforts 0.725 |
| e2 \| army 0.815 troops 0.804 soldiers 0.786 |
| e3 \| court 0.758 federal 0.757 judicial 0.736 |

| **Input Sentence**: SMS messages are used in some countries as reminders of hospital appointments . <eos> |
|---|
| **Output Embedding (K = 1)**: |
| e1 \| information 0.702, use 0.701, specific 0.700 |
| **Output Embeddings (K = 3)**: |
| e1 \| can 0.769, possible 0.767, specific 0.767 |
| e2 \| hospital 0.857, medical 0.780, hospitals 0.739 |
| e3 \| SMS 0.791, Mobile 0.635, Messaging 0.631 |
| **Output Embeddings (K = 10)**: |
| e1 \| can 0.854, should 0.834, either 0.821 |
| e2 \| hospital 0.886, medical 0.771, hospitals 0.745 |
| e3 \| services 0.768, service 0.749, web 0.722 |
| e4 \| SMS 0.891, sms 0.745, messaging 0.686 |
| e5 \| messages 0.891, message 0.801, emails 0.679 |
| e6 \| systems 0.728, technologies 0.725, integrated 0.723 |
| e7 \| appointments 0.791, appointment 0.735, duties 0.613 |
| e8 \| confirmation 0.590, request 0.568, receipt 0.563 |
| e9 \| countries 0.855, nations 0.737, Europe 0.732 |
| e10 \| Implementation 0.613, Application 0.610, Programs 0.603 |

Table 3.1: Examples of the codebook embeddings predicted by our models with different $K$. The embedding in each row is visualized by the three words whose GloVe embeddings have the highest cosine similarities (also presented) with the codebook embedding.

the input is a sentence, we can see that the output embeddings are sometimes close to the embeddings of words in the input sentence, which explains why attending the contextualized word embeddings in our decoder could improve the quality of the output embeddings.

#### 3.2.2.3 Unsupervised Sentence Similarity

We propose two ways to evaluate the multi-facet embeddings using sentence similarity tasks.

**First way**: Since similar sentences should have similar word distribution in nearby sentences and thus similar codebook embeddings, the codebook embeddings of a query sentence $\widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S}_{\boldsymbol{q}}^{\boldsymbol{1}})$ should be able to well reconstruct the codebook embeddings of its similar

sentence $\widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S}_{\boldsymbol{q}}^{\mathbf{2}})$. We compute the reconstruction error of both directions and add them as a symmetric distance **SC**:

$$SC(S_q^1, S_q^2) = Er(\widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S}_{\boldsymbol{q}}^{\mathbf{1}}), \widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S}_{\boldsymbol{q}}^{\mathbf{2}})) + Er(\widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S}_{\boldsymbol{q}}^{\mathbf{2}}), \widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S}_{\boldsymbol{q}}^{\mathbf{1}})), \tag{3.5}$$

where $\widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S}_{\boldsymbol{q}}) = [\frac{\boldsymbol{c}_k^q}{||\boldsymbol{c}_k^q||}]_{k=1...K}$ is a matrix of normalized codebook embeddings and $Er$ function is defined in Equation 3.1. We use the negative distance to represent similarity.

**Second way**: One of the main challenges in unsupervised sentence similarity tasks is that we do not know which words are more important in each sentence. Intuitively, if one word in a query sentence is more important, the chance of observing related/similar words in the nearby sentences should be higher. Thus, we should pay more attention to the words in a sentence that have higher cosine similarity with its multi-facet embeddings, a summary of the co-occurring word distribution. Specifically, our importance/attention weighting for all the words in the query sentence $S_q$ is defined by

$$\underline{\boldsymbol{a}}_{\boldsymbol{q}} = \max(0, \boldsymbol{W}(\boldsymbol{S}_{\boldsymbol{q}})^T \widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S}_{\boldsymbol{q}}))\ \underline{\mathbf{1}}, \tag{3.6}$$

where $\underline{\mathbf{1}}$ is an all-one vector. We show that the attention vector (denoted as **Our a** in Table 3.2) could be combined with various scoring functions and boost their performance. As a baseline, we also report the performance of the attention weights derived from the k-means loss rather than NNSC loss and call it **Our a (k-means)**.

**Setup**: STS benchmark [27] is a widely used sentence similarity task. We compare the correlations between the predicted semantic similarity and the manually labeled similarity. We report Pearson correlation coefficient, which is strongly correlated with Spearman correlation in all our experiments. Intuitively, when two sentences are less similar to each other, humans tend to judge the similarity based on how similar their facets are. Thus, we

also compare the performance on the lower half of the datasets where their ground truth similarities are less than the median similarity in the dataset, and we call this benchmark STSB Low.

A simple but effective way to measure sentence similarity is to compute the cosine similarity between the average (contextualized) word embedding [149]. The scoring function is labeled as **Avg**. Besides, we test the sentence embedding from BERT and from skip-thought [102] (denoted as **CLS** and **Skip-thought Cosine**, respectively).

In order to deemphasize the syntax parts of the sentences, Arora et al. [7] propose to weight the word $w$ in each sentence according to $\frac{\alpha}{\alpha+p(w)}$, where $\alpha$ is a constant and $p(w)$ is the probability of seeing the word $w$ in the corpus. Following its recommendation, we set $\alpha$ to be $10^{-4}$ in this section. After the weighting, we remove the first principal component of all the sentence embeddings in the training data as suggested by Arora et al. [7] and denote the method as **SIF**. The post-processing requires an estimation of testing embedding distribution, which is not desired in some applications, so we also report the performance before removing the principal component, which is called **Prob_avg**.

We also test word mover's distance (**WMD**) [110], which explicitly matches every word in a pair of sentences. As we do in **Prob_avg**, we apply $\frac{\alpha}{\alpha+p(w)}$ to **WMD** to down-weight the importance of functional words, and call this scoring function as **Prob_WMD**. When using **Our a**, we multiple our attention vector with the weights of every word (e.g., $\frac{\alpha}{\alpha+p(w)}$ for **Prob_avg** and **Prob_WMD**).

**Results**: In Figure 3.3, we first visualize our attention weights in Equation 3.6 and our output codebook embeddings for a pair of similar sentences from STSB to intuitively explain why modeling co-occurring distribution could improve the similarity measurement.

Many similar sentences might use different word choices or using extra words to describe details, but their possible nearby words are often similar. For example, appending *in the garage* to *A man is lifting weights* does not significantly change the facets of the sentences and thus the word *garage* receives relatively a lower attention weight. This makes its

| Sentences | A man is lifting weights in a garage . | A man is lifting weights . |
|---|---|---|
| Output Embeddings | e1 \| can 0.872, even 0.851, should 0.850<br>e2 \| front 0.762, bottom 0.742, down 0.714<br>e3 \| lifting 0.866, lift 0.663, Lifting 0.621<br>e4 \| garage 0.876, garages 0.715, basement 0.707<br>e5 \| decreasing 0.677, decreases 0.655, negligible 0.649<br>e6 \| weights 0.883, Weights 0.678, weight 0.665<br>e7 \| cylindrical 0.700, plurality 0.675, axial 0.674<br>e8 \| configurations 0.620, incorporating 0.610, utilizing 0.605<br>e9 \| man 0.872, woman 0.682, men 0.672<br>e10 \| man 0.825, men 0.671, woman 0.653 | e1 \| can 0.865, either 0.843, should 0.841<br>e2 \| front 0.758, bottom 0.758, sides 0.691<br>e3 \| lifting 0.847, lift 0.635, Lifting 0.610<br>e4 \| lifting 0.837, lift 0.652, weights 0.629<br>e5 \| decreasing 0.709, decreases 0.685, increases 0.682<br>e6 \| weights 0.864, weight 0.700, Weights 0.646<br>e7 \| annular 0.738, cylindrical 0.725, circumferential 0.701<br>e8 \| methods 0.612, configurations 0.610, graphical 0.598<br>e9 \| sweating 0.498, cardiovascular 0.494, dehydration 0.485<br>e10 \| man 0.888, woman 0.690, men 0.676 |

Figure 3.3: Comparison of our attention weights and the output embeddings between two similar sentences from STSB. A darker red indicates a larger attention value in Equation 3.6 and the output embeddings are visualized using the same way in Table 3.1.

| Method | | Dev | | Test | |
|---|---|---|---|---|---|
| Score | Model | All | Low | All | Low |
| Cosine | Skip-thought | 43.2 | 28.1 | 30.4 | 21.2 |
| CLS | BERT | 9.6 | -0.4 | 4.1 | 0.2 |
| Avg | | 62.3 | 42.1 | 51.2 | 39.1 |
| SC | Our c K1 | 55.7 | 43.7 | 47.6 | 45.4 |
| | Our c K10 | 63.0 | 51.8 | 52.6 | 47.8 |
| WMD | GloVe | 58.8 | 35.3 | 40.9 | 25.4 |
| | Our a K1 | 63.1 | 43.3 | 47.5 | 34.8 |
| | Our a K10 | 66.7 | 47.4 | 52.6 | 39.8 |
| Prob_WMD | GloVe | 75.1 | 59.6 | 63.1 | 52.5 |
| | Our a K1 | 74.4 | 60.8 | 62.9 | 54.4 |
| | Our a K10 | **76.2** | **62.6** | **66.1** | 58.1 |
| Avg | GloVe | 51.7 | 32.8 | 36.6 | 30.9 |
| | Our a K1 | 54.5 | 40.2 | 44.1 | 40.6 |
| | Our a K10 | 61.7 | 47.1 | 50.0 | 46.5 |
| Prob_avg | GloVe | 70.7 | 56.6 | 59.2 | 54.8 |
| | Our a K1 | 68.5 | 56.0 | 58.1 | 55.2 |
| | Our a K10 | 72.0 | 60.5 | 61.4 | **59.3** |
| SIF† | GloVe | 75.1 | 65.7 | 63.2 | 58.1 |
| | Our a K1 | 72.5 | 64.0 | 61.7 | 58.5 |
| | Our a K10 | **75.2** | **67.6** | **64.6** | **62.2** |
| | Our a (k-means) K10 | 71.5 | 62.3 | 61.5 | 57.2 |

Table 3.2: Pearson correlation (%) in the development and test sets in the STS benchmark. The performance of all sentence pairs is indicated as All. Low means the performance on the half of sentence pairs with lower similarity (i.e., STSB Low). Our c means our codebook embeddings and Our a means our attention vectors. * indicates a supervised method. † indicates the methods which use training distribution to approximate testing distribution. The best score with and without † are highlighted.

similarity measurement from our methods, **Our c** and **Our a**, closer to the human judgment than other baselines.

In Table 3.2, **Our c SC**, which matches between two sets of facets, outperforms **WMD**, which matches between two sets of words in the sentence, and also outperforms **BERT Avg**, especially in STSB Low. The significantly worse performance from **Skip-thought Cosine** justifies our choice of ignoring the order in the co-occurring words.

All the scores in **Our * K10** are significantly better than **Our * K1**, which demonstrates the co-occurring word distribution is hard to be modeled well using a single embedding. Multiplying the proposed attention weighting consistently boosts the performance in all the scoring functions especially in STSB Low and without relying on the generalization assumption of the training distribution. Finally, using k-means loss, **Our a (k-means) K10**, significantly degrades the performance compared to **Our a K10**, which justify the proposed NNSC loss.

#### 3.2.2.4 Unsupervised Extractive Summarization

The classic representation of a sentence uses either a single embedding or the (contextualized) embeddings of all the words in the sentence. In this section, we would like to show that both options are not ideal for extracting a set of sentences as a document summary.

Table 3.1 indicates that our multiple codebook embeddings of a sentence capture its different facets well, so we represent a document summary $S$ as the union of the multi-facet embeddings of the sentences in the summary $R(S) = \cup_{t=1}^{T} \{\widehat{F}_u(S_t)\}$, where $\{\widehat{F}_u(S_t)\}$ is the set of column vectors in the matrix $\boldsymbol{\widehat{F}_u(S_t)}$ of sentence $S_t$.

A good summary should cover multiple facets that well represent all topics/concepts in the document [103]. The objective can be quantified as discovering a summary $S$ whose multiple embeddings $R(S)$ best reconstruct the distribution of normalized word embedding $\underline{\boldsymbol{w}}$ in the document $D$ [103]. That is,

$$\arg\max_{S} \sum_{\underline{\boldsymbol{w}} \in D} \frac{\alpha}{\alpha + p(w)} \max_{\underline{\boldsymbol{s}} \in R(S)} \underline{\boldsymbol{w}}^T \underline{\boldsymbol{s}}, \tag{3.7}$$

where $\frac{\alpha}{\alpha+p(w)}$ is the weights of words we used in the sentence similarity experiments [7]. We greedily select sentences to optimize Equation 3.7 as in Kobayashi et al. [103].

**Setup**: We compare our multi-facet embeddings with other alternative ways of modeling the facets of sentences. A simple way is to compute the average word embedding as a single-facet sentence embedding.[4] This baseline is labeled as **Sent Emb**. Another way is to use the (contextualized) embedding of all the words in the sentences as different facets of the sentences. Since longer sentences have more words, we normalize the gain of the reconstruction similarity by the sentence length. The method is denoted as **W Emb**. We also test the baselines of selecting random sentences (**Rnd**) and first 3 sentences (**Lead-3**) in the document.

The results on the testing set of CNN/Daily Mail [84, 191] are compared using F1 of ROUGE [123] in Table 3.3. R-1, R-2, and Len mean ROUGE-1, ROUGE-2, and average summary length, respectively. All methods choose 3 sentences by following the setting in Zheng and Lapata [260]. *Unsup, No Sent Order* means the methods do not use the sentence order information in CNN/Daily Mail.

In CNN/Daily Mail, the unsupervised methods which access sentence order information such as **Lead-3** have performance similar to supervised methods such as RL [26]. To evaluate the quality of unsupervised sentence embeddings, we focus on comparing the unsupervised methods which do not assume the first few sentences form a good summary.

**Results**: In Table 3.3, predicting 100 clusters yields the best results. Notice that our method greatly alleviates the computational and sample efficiency challenges, which allows us to set cluster numbers $K$ to be a relatively large number.

The results highlight the limitation of classic representations. The single sentence embedding cannot capture its multiple facets. On the other hand, if a sentence is represented by the embeddings of its words, it is difficult to eliminate the bias of selecting longer or

[4] Although Equation 3.7 weights each word in the document, we find that the weighting $\frac{\alpha}{\alpha+p(w)}$ does not improve the sentence representation when averaging the word embeddings.

| Setting | Method | R-1 | R-2 | Len |
|---|---|---|---|---|
| Unsupvised No Sentence Order | Random | 28.1 | 8.0 | 68.7 |
| | Textgraph (tfidf)† | 33.2 | 11.8 | - |
| | Textgraph (BERT)† | 30.8 | 9.6 | - |
| | W Emb (GloVe) | 26.6 | 8.8 | 37.0 |
| | Sent Emb (GloVe) | 32.6 | 10.7 | 78.2 |
| | W Emb (BERT) | 31.3 | 11.2 | 45.0 |
| | Sent Emb (BERT) | 32.3 | 10.6 | 91.2 |
| | Our c (K=3) | 32.2 | 10.1 | 75.4 |
| | Our c (K=10) | 34.0 | 11.6 | 81.3 |
| | Our c (K=100) | **35.0** | **12.8** | 92.9 |
| Unsupvised | Lead-3 | 40.3 | 17.6 | 87.0 |
| | PACSUM (BERT)† | **40.7** | **17.8** | - |
| Supervised | RL* | **41.7** | **19.5** | - |

Table 3.3: The ROUGE F1 scores of different methods on CNN/Daily Mail dataset. The results with † are taken from Zheng and Lapata [260]. The results with * are taken from Celikyilmaz et al. [26].

shorter sentences and a facet might be composed by multiple words (e.g., the input sentence in Table 3.1 describes a service, but there is not a single word in the sentence that means service).

#### 3.2.2.5 Unsupervised Phrase Similarity

Recently, Dubossarsky et al. [51] discovered that the multiple embeddings of each word may not improve the performance in word similarity benchmarks even if they capture more senses or facets of polysemies. Since our method can improve the sentence similarity estimation, we want to see whether multi-facet embeddings could also help the phrase similarity estimation.

In addition to **SC** in Equation 3.5, we also compute the average of the contextualized word embeddings from our transformer encoder as the phrase embedding. We find that the cosine similarity between the two phrase embeddings is a good similarity estimation, and the method is labeled as **Ours Emb**.

**Setup**: We evaluate our phrase similarity using SemEval 2013 task 5(a) English [107] and Turney 2012 [223]. The task of SemEval 2013 is to distinguish similar phrase pairs

| Method | | SemEval 2013 | | Turney (5) | Turney (10) |
|---|---|---|---|---|---|
| Model | Score | AUC | F1 | Accuracy | Accuracy |
| BERT | CLS | 54.7 | 66.7 | 29.2 | 15.5 |
| | Avg | 66.5 | 67.1 | 43.4 | 24.3 |
| GloVe | Avg | 79.5 | 73.7 | 25.9 | 12.9 |
| FCT LM† | Emb | - | 67.2 | 42.6 | 27.6 |
| Ours | SC | 80.3 | 72.8 | 45.6 | 28.8 |
| (K=10) | Emb | 85.6 | 77.1 | 49.4 | 31.8 |
| Ours | SC | 81.1 | 72.7 | 45.3 | 28.4 |
| (K=1) | Emb | **87.8** | **78.6** | **50.3** | **32.5** |

Table 3.4: Performance of phrase similarity tasks. Every model is trained on a lowercased corpus. In SemEval 2013, AUC (%) is the area under the precision-recall curve of classifying similar phrase pairs. In Turney, we report the accuracy (%) of predicting the correct similar phrase pair among 5 or 10 candidate pairs. The results with † are taken from Yu and Dredze [252].

from dissimilar phrase pairs. In Turney (5), given each query bigram, each model predicts the most similar unigram among 5 candidates, and Turney (10) adds 5 more negative phrase pairs by pairing the reverse of the query bigram with the 5 unigrams.

**Results**: The performance is presented in Table 3.4. **Ours (K=1)** is usually slightly better than **Ours (K=10)**, and the result supports the finding of Dubossarsky et al. [51]. We hypothesize that unlike sentences, most of the phrases have only one facet/sense, and thus can be modeled by a single embedding well.

Even though being slightly worse, the performance of **Ours (K=10)** remains strong compared with baselines. This implies that the similarity performance is not sensitive to the number of clusters as long as sufficiently large K is used because the model is able to output multiple nearly duplicated codebook embeddings to represent one facet (e.g., using two centers to represent the facet related to *company* in Figure 3.1). The flexibility alleviates the issues of selecting K in practice. Finally, the strong performance in Turney (10) verifies that our encoder respects the word order when composing the input sequence.

### 3.2.3 Related Work

Topic modeling [19] has been extensively studied and widely applied due to its interpretability and flexibility of incorporating different forms of input features [152]. Cao et al. [23] and Srivastava and Sutton [204] demonstrate that neural networks could be applied to discover semantically coherent topics. Instead of optimizing a global topic model, our goal is to efficiently discover different sets of topics/clusters on the words beside each (unseen) phrase or sentence.

Recently, Gupta et al. [76] and Gupta et al. [77] discover that global clustering could improve the representation of sentences and documents. In our work, we show that a local clustering could be used in several downstream applications, including word importance estimation for measuring sentence similarity. Whether combining global clustering and local clustering could lead to further improvement is an interesting future research direction.

Sparse coding on word embedding space is used to model the multiple facets of a word [61, 8], and parameterizing word embeddings using neural networks is used to test hypothesis [78] and save storage space [197]. Besides, to capture asymmetric relations such as hypernyms, words are represented as single or multiple regions in Gaussian embeddings [226, 9] rather than a single point. However, the challenges of extending these methods to longer sequences are not addressed in these studies.

One of our main challenges is to design a loss for learning to predict cluster centers while modeling the dependency among the clusters. This requires a matching step between two sets and computing the distance loss after the matching [52]. One popular loss is called Chamfer distance, which is widely adopted in the auto-encoder models for point clouds [247, 130], while more sophisticated matching loss options are also proposed [207, 13]. The goal of the previous studies focuses on measuring symmetric distances between the ground truth set and predicted set (usually with an equal size), while our loss tries to reconstruct the ground truth set using much fewer codebook embeddings.

Other ways to achieve the permutation invariant loss for neural networks include sequential decision making [236], mixture of experts [248, 234], beam search [176], predicting the permutation using a CNN [182], transformers [206, 75, 25] or reinforcement learning [237]. In contrast, our goal is to efficiently predict a set of cluster centers that can well reconstruct the set of observed instances rather than directly predicting the observed instances.

## 3.3 Applications on Relation Extraction

Distant supervision assumes that a sentence pattern expresses a relation if the sentence pattern co-occurs with an entity pair and the entity pair has the relation. For example, we assume the sentence pattern *"$ARG1, the partner of fellow $ARG2"* is likely to express the spouse relation if we observe a text clip *"... Angelina Jolie, the partner of fellow Brad Pitt ..."* in our training corpus and a knowledge base tells us that *Angelina Jolie* and *Brad Pitt* has the spouse relation. Accordingly, we can infer that another entity pair is likely to have the spouse relation if we observe the text *", the partner of fellow"* between them in a new corpus.

Universal schema [184] extends this assumption by treating every sentence pattern as a relation, which means we assume that sentence patterns or relations in a knowledge base are similar if they co-occur with the same entity pair. For example, we assume *"$ARG1, the partner of fellow $ARG2"* and *"$ARG1, the wife of fellow $ARG2"* are similar if they both co-occur with *(Kristen Bell, Dax Shepard)*. Consequently, we can infer that *"$ARG1, the wife of fellow $ARG2"* also implies spouse relation as *"$ARG1, the partner of fellow $ARG2"* even if the knowledge base does not record the spouse relation between *Kristen Bell* and *Dax Shepard*.

Compositional universal schema [225] realizes the idea by using a LSTM [86] to encode each sentence pattern into an embedding and encouraging the embedding to be similar to the embedding of the co-occurred entity pair. As in the lower part of Figure 3.4, the model makes the embeddings of two sentence patterns similar if they co-occur with the same entity

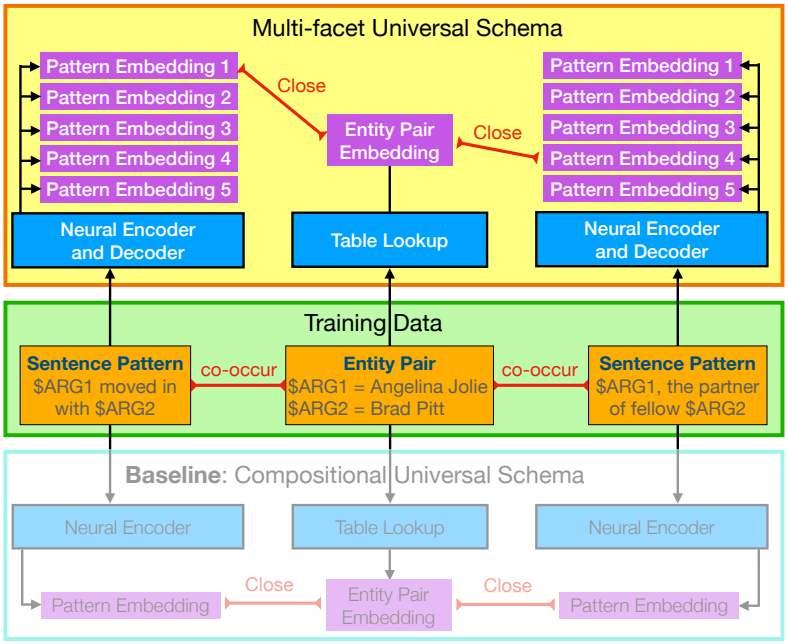



Figure 3.4: Comparison between the multi-facet and compositional universal schema. In our training loss, we encourage one of the facet embeddings from a sentence pattern to be similar to its co-occurred entity pair.

pair. Baldini Soares et al. [12] rely on a similar assumption and achieve state-of-the-art results on supervised RE tasks by replacing the LSTM with a large pre-trained language model.

Nevertheless, one sentence pattern could contain multiple facets, and each facet could imply a different relation. In Figure 3.4, *"$ARG1, the partner of fellow $ARG2"* could imply the entity pair has the spouse relation, the co-worker relation, or both. *"$ARG1 moved in with $ARG2"* could imply the spouse relation, the parent relation, ..., etc. If we squeeze the facets of a sentence pattern into a single embedding, the embedding is more likely to be affected by the irrelevant facets from other patterns co-occurred with the same entity pair (e.g., *"$ARG1 moved in with $ARG2"* might incorrectly imply the co-worker relation).

Another limitation is that single embedding representation can only provide symmetric similarity measurement between two sentence patterns. Thus, an open research challenge is

to predict the entailment direction of two sentence patterns only based on their co-occurring entity pair information.

To overcome the challenges, we propose multi-facet universal schema, where we assume that two sentence patterns share a similar facet if they co-occur with the same entity pair. As in Figure 3.4, we use a neural encoder and decoder to predict multiple facet embeddings of each sentence pattern and encourage one of the facet embeddings to be similar to the entity pair embedding. As a result, the facets that are irrelevant to the relation between the entity pairs are less likely to affect the embeddings of entity pairs and other related sentence patterns. For example, the parent facet of *"$ARG1 moved in with $ARG2"* could be excluded when updating the embeddings of *(Angelina Jolie, Brad Pitt)*.

In our experiments, we first compare the multi-facet embeddings with the single-facet embedding in distantly supervised RE tasks. The results demonstrate that multiple facet embeddings significantly improve the similarity measurement between the sentence patterns and knowledge base relations. Besides RE, we also apply multi-facet embeddings to unsupervised entailment detection tasks. In a newly collected dataset, we show that multi-facet universal schema significantly outperforms the other unsupervised baselines.

For details of our methods and experiments, please see Paul et al. [171].

### 3.3.1 Method

Our method is illustrated in Figure 3.5. In Section 3.3.1.1, we first provide our problem setup: We are given a knowledge base (KB) and a text corpus during training. Our goal is to extract relations by measuring the similarity between KB relations and an (unseen) sentence pattern or to detect entailment between two sentence patterns. In Section 3.3.1.2, we describe our objective function, which encourages the embeddings of co-occurred entity pairs to be close to the embeddings of their closest pattern facets. Finally, in Section 3.3.1.3, we provide our scoring functions for distantly supervised RE and unsupervised entailment tasks.

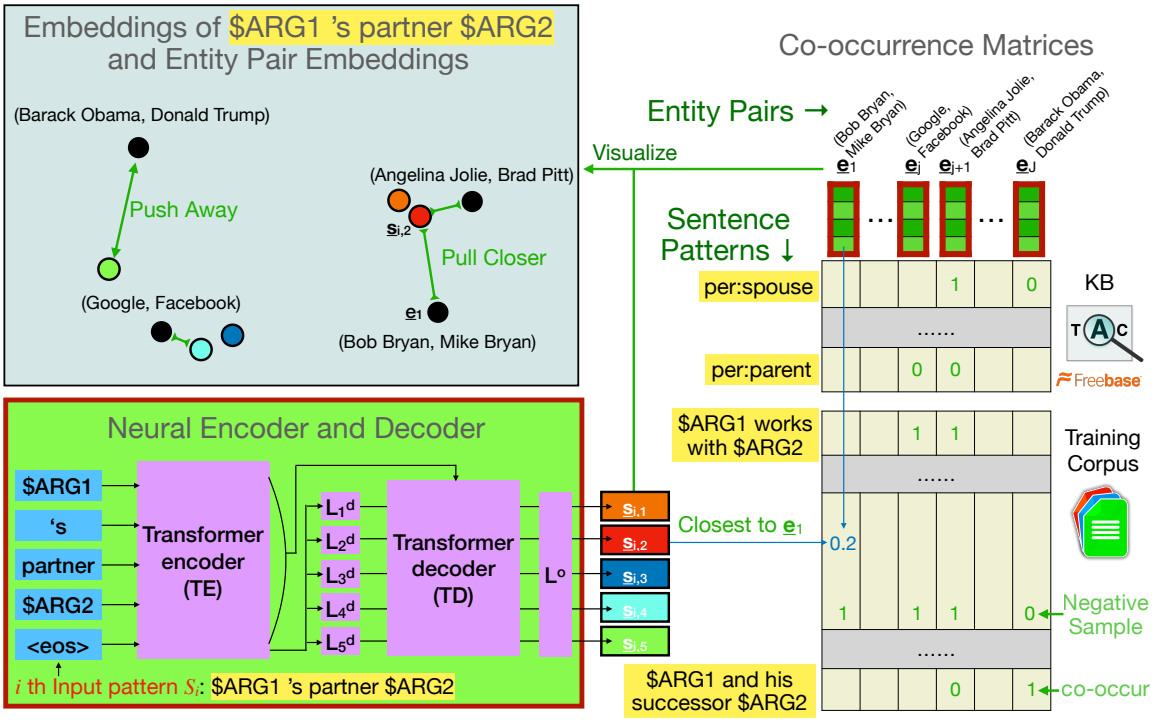



Figure 3.5: An illustration of the proposed method. The training signal comes from the co-occurrence matrices of the KB and training text corpus on the right. On the lower left, we visualize our neural encoder, which captures the compositional meaning of tokens in the sentence pattern, and our neural decoder, which models the dependency among multiple facet embeddings. When a sentence pattern co-occurs with an entity pair, the training loss minimizes the distance between the entity pair embedding and the closest facet embedding of the sentence pattern (e.g., 0.2 between $s_{i,2}$ and $e_1$). Trainable parameters in our model are highlighted using red borders. On the upper left, we visualize the embedding space to establish the connection between our method and clustering.

#### 3.3.1.1 Background and Problem Setup

Our RE problem setup is the same as compositional universal schema [225]. First, we run named entity recognition (NER) and entity linking (EL) on a raw corpus. After identifying the entity pairs in each sentence, we prepare a co-occurrence matrix as in Figure 3.5. Similarly, we represent the KB relations between entity pairs as a co-occurrence matrix and merge the matrices from the KB and the training corpus. The merged matrix has $y_{i,j} = 1$ if the $i$th sentence pair or KB relation co-occurs with the $j$th entity pair and $y_{i,j} = 0$ otherwise.

During testing, we use NER to extract an entity pair and the sentence pattern, which might not have been seen in the training corpus. Next, we extract relations by computing the similarity between the sentence pattern embeddings and the embeddings of the applicable KB relations. Besides RE, we also detect the entailment between two sentence patterns by comparing their embeddings.

#### 3.3.1.2 Objective Function

We use a seq2seq model as in Section 3.2 to compute facet embedding (i.e., sentence pattern embedding). When measuring the distance between the $j$th entity pair and the $i$th sentence pattern, we compute the Euclidean distance between the entity pair embedding $\underline{\tilde{\boldsymbol{e}}}_j$ and its closest facet embedding $\underline{\boldsymbol{s}}_{i,k}$ of the $i$th sentence pattern. The distance is defined as

$$D(\{\underline{\boldsymbol{s}}_{i,k}\}_{k=1}^{K}, \underline{\tilde{\boldsymbol{e}}}_j) = \min_{k=1}^{K} \min_{0 \leq \eta_k \leq 1} ||\underline{\tilde{\boldsymbol{e}}}_j - \eta_k \underline{\boldsymbol{s}}_{i,k}||^2, \tag{3.8}$$

where the entity pair embedding is normalized (i.e., $||\underline{\tilde{\boldsymbol{e}}}_j|| = 1$). During testing, we ignore the magnitude of facet embeddings, so we use $\eta_k$ to eliminate the magnitude of facet embeddings $\underline{\boldsymbol{s}}_{i,k}$ during training. We do not allow negative $\eta_k$ to prevent the gradient flow from pushing $\underline{\boldsymbol{s}}_{i,k}$ toward the inverse direction of $\underline{\tilde{\boldsymbol{e}}}_j$ and we ensure $\eta_k \leq 1$ to avoid the neural model from outputting $\underline{\boldsymbol{s}}_{i,k}$ with a very small magnitude.

As in Figure 3.5, we minimize the distance $D(\{\underline{\boldsymbol{s}}_{i,k}\}_{k=1}^{K}, \underline{\tilde{\boldsymbol{e}}}_j)$ in our loss function when the $i$th sentence pair co-occurs with the $j$th entity pair (i.e., $y_{i,j} = 1$). For negative samples (i.e., $y_{i,j} = 0$), we maximize the distance instead. That is, our loss function is defined as

$$\sum_{(i,j) \in R} (2 \cdot y_{i,j} - 1) r_{i,j} D(\{\underline{\boldsymbol{s}}_{i,k}\}_{k=1}^{K}, \underline{\tilde{\boldsymbol{e}}}_j) + \Omega, \tag{3.9}$$

and the regularization term $\Omega$ in the loss function will be described in Appendix B.2. $R$ is a set that includes all positive and negative samples. Positive samples are $(i, j)$ such that

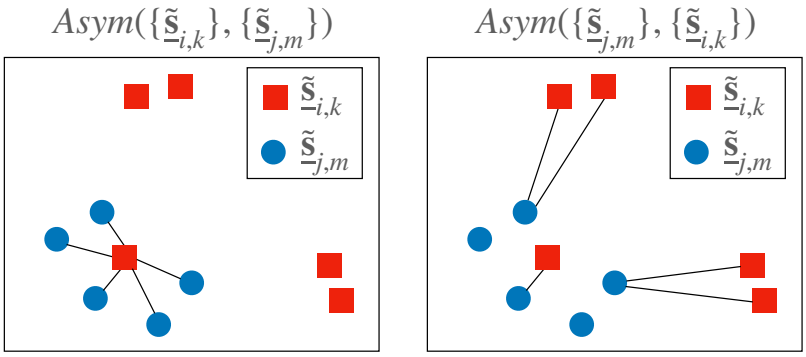



Figure 3.6: Comparison of the asymmetric similarities. $Asym(\{\tilde{\underline{\mathbf{s}}}_{i,k}\}, \{\tilde{\underline{\mathbf{s}}}_{j,m}\}) > Asym(\{\tilde{\underline{\mathbf{s}}}_{j,m}\}, \{\tilde{\underline{\mathbf{s}}}_{i,k}\})$ because the average cosine distance on the left is smaller than that on the right.

$y_{i,j} = 1$ and the negative samples are constructed by pairing a randomly selected sentence pattern with the $j$th entity pair. To balance the influence of popular entity pairs (i.e., entity pairs that co-occur with many sentence patterns) and rare entity pairs on our model, we set the weight of each pair, $r_{i,j} \propto \frac{1}{\sum_i y_{i,j}}$ and $\frac{\sum_{(i,j)\in R} r_{i,j}}{|R|} = 1$.

We generate the embeddings for KB relations in a similar way. We use a single token to represent the relation and append an <eos> (e.g., *per:spouse* <eos>) to form the input of our neural model. The KB relations usually co-occur with more entity pairs, so we set the number of facet embeddings for KB relations $K_{rel}$ to be larger than the number of facet embeddings for sentence patterns $K$.

#### 3.3.1.3 Scoring Functions

In compositional universal schema, the similarity between the $i$th and $j$th sentence patterns are measured by the symmetric cosine similarity $\tilde{\underline{\mathbf{s}}}_{i,1}^T \tilde{\underline{\mathbf{s}}}_{j,1}$, where $\tilde{\underline{\mathbf{s}}}_{i,1} = \frac{\underline{\mathbf{s}}_{i,1}}{||\underline{\mathbf{s}}_{i,1}||}$. When using multiple embeddings to represent a sentence pattern, we can compute the asymmetric similarity as

$$Asym(\{\tilde{\underline{\mathbf{s}}}_{i,k}\}, \{\tilde{\underline{\mathbf{s}}}_{j,m}\}) = \frac{\sum_{m=1}^{K} \max_{k=1}^{K} (\tilde{\underline{\mathbf{s}}}_{i,k}^T \tilde{\underline{\mathbf{s}}}_{j,m})}{K}. \tag{3.10}$$

In an example of Figure 3.6, a red square $\underline{\tilde{\boldsymbol{s}}}_{i,k}$ is close to all the blue points, which leads to a high $Asym(\{\underline{\tilde{\boldsymbol{s}}}_{i,k}\}, \{\underline{\tilde{\boldsymbol{s}}}_{j,m}\})$.

Between two sentence patterns with entailment relation, we empirically find that the embeddings of a premise (the more specific pattern) often have some facet embeddings that are far away from all the embeddings of its hypothesis (the more general pattern). Relying on the tendency, we could detect the direction of the entailment relation. For example, the $i$th sentence pattern (red squares) in Figure 3.6 is more likely to be premise if the $i$th and $j$th (blue circles) sentence patterns have an entailment relation.

We suspect the reason is that more specific patterns could contain more words that are similar to the words of other patterns expressing different relations. For example, *"$ARG1 , the wife of fellow $ARG2"* have a facet embedding for spouse relation and another facet embedding for the co-worker relation because the pattern has high word overlapping with *"$ARG1 , the wife of $ARG2"* and *"$ARG1 and her fellow $ARG2"*. Another possible reason is that the articles in our corpus tend to use more specific patterns to express the relation between a pair of entities [199].

When performing RE, we compute the symmetric similarity between $i$th sentence pattern and $j$th KB relation $Sim(\{\underline{\tilde{\boldsymbol{s}}}_{i,k}\}, \{\underline{\tilde{\boldsymbol{s}}}_{j,m}\})$ by

$$\frac{Asym(\{\underline{\tilde{\boldsymbol{s}}}_{i,k}\}, \{\underline{\tilde{\boldsymbol{s}}}_{j,m}\}) + Asym(\{\underline{\tilde{\boldsymbol{s}}}_{j,m}\}, \{\underline{\tilde{\boldsymbol{s}}}_{i,k}\})}{2}. \tag{3.11}$$

### 3.3.2 Experiments

We primarily compare our method with compositional universal schema (CUSchema) [225] because CUSchema is one of the state-of-the-art RE methods in the small model regime (without using large pre-trained language models) [31, 28]. We use distant-supervised RE tasks to evaluate our symmetric similarity measurement in Section 3.3.2.1, and detect entailment between sentence patterns to evaluate our asymmetric similarity measurement in Section 3.3.2.2.

#### 3.3.2.1 Relation Extraction

We follow the same training data and testing protocol in compositional universal schema (CUSchema) [225][5] to highlight the benefit of predicting multiple facet embeddings, and the relation extraction step in TAC KBP slot-filling tasks is used to compare the different models.

**Setup:** The training data for our RE models are prepared by distant supervision without requiring any manually labeled data. The relations in Freebase [20] are mapped to TAC relations (e.g., *org:city_of_headquarter*) and the NER tagger and entity linker are run in a raw text corpus. Then, the training data is cleaned using the methods in Roth et al. [187].

During testing, we are given a query containing the head entity and a query TAC relation in the slot-filling tasks, and the goal is to extract the tail entity from the candidate sentences. The NER tagger and query expansion are used to gather the candidate sentence patterns, and we compute the similarity scores from different models between the candidate sentence patterns and query relation. Finally, we compare the extracted second entity with the ground truth using exact string matching and report the precision, recall, and F1 scores.

Following Verga et al. [225], we use TAC 2012 as our validation set to determine the threshold score for each TAC relation. Each model's hyperparameters are tuned separately using the validation set (TAC 2012) to ensure a fair comparison.

We compare the following methods:

**Ours (Trans)**: Our method that measures the similarity between the sentence pattern $\{\underline{\tilde{\boldsymbol{s}}}_{i,k}\}$ and TAC relation $\{\underline{\tilde{\boldsymbol{s}}}_{j,m}\}$ using $Sim(\{\underline{\tilde{\boldsymbol{s}}}_{i,k}\}, \{\underline{\tilde{\boldsymbol{s}}}_{j,m}\})$ in Equation 3.11. Trans is an abbreviation of the transformer encoder. We set $K = 5$ and $K_{rel} = 11$ based on the validation set.

**Ours (LSTM)**: The same as Ours (Trans) except that we use bi-LSTM as our encoder instead.

**Ours (Single-*)**: Our methods that use single facet embedding to represent each sentence

[5] https://github.com/patverga/torch-relation-extraction

| Method | TAC 2012 (Validation) | | | TAC 2013 | | | TAC 2014 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Prec | Recall | F1 | Prec | Recall | F1 | Prec | Recall | F1 |
| USchema* | **34.8** | 23.7 | 28.2 | **42.6** | 29.4 | 34.8 | **35.5** | 24.3 | 28.8 |
| CUSchema (LSTM)* | 27.0 | **32.7** | 29.6 | 39.6 | 32.2 | 35.5 | 32.9 | 27.3 | 29.8 |
| Ours (Single-LSTM) | 25.7 | 21.7 | 23.5 | 30.4 | 26.3 | 28.2 | 22.1 | 20.5 | 21.3 |
| Ours (Single-Trans) | 26.1 | 21.6 | 23.7 | 29.5 | 25.2 | 27.2 | 19.0 | 21.2 | 20.0 |
| Ours (LSTM) | 32.0 | 28.9 | 30.3 | 41.3 | **33.9** | 37.2 | 34.1 | **29.5** | **31.6** |
| Ours (Trans) | 33.6 | 27.7 | **30.4** | 42.5 | 33.2 | **37.3** | 34.6 | 28.5 | 31.3 |
| USchema + CUSchema (LSTM)* | 29.3 | 32.8 | 30.9 | **41.9** | 34.4 | 37.7 | 29.3 | 34.1 | 31.5 |
| USchema + Ours (LSTM) | 29.2 | 33.7 | 31.3 | 38.1 | **38.9** | 38.5 | 31.5 | **34.4** | 32.9 |
| USchema + Ours (Trans) | **30.4** | **33.9** | **32.1** | 39.0 | 38.8 | **38.9** | **32.0** | 34.0 | **33.0** |

Table 3.5: Distantly supervised relation extraction using different versions of the universal schema. All numbers are %. CUSchema refers to compositional universal schema. Trans is an abbreviation of transformer. The best scores of the single models and ensemble models are highlighted. *The performance of TAC 2013 and 2014 is copied from Verga et al. [225].

pattern or KB relation. When setting $K = K_{rel} = 1$, our decoder becomes the interleaving feedforward layers and cross-attention layers attending to the output embeddings of the encoder.

**CUSchema (LSTM)**: Compositional universal schema [225]. The method is similar to Ours (Single-LSTM) but uses a different loss function, neural architecture (no decoder), and hyperparameter search procedure.

**USchema**: Universal schema [184] estimates every sentence pattern embedding by factorizing the co-occurrence matrices (i.e., replacing the bi-LSTM in CUSchema with a look-up table).

**USchema + ***: Verga et al. [225] show that taking the maximal similarity between USchema and CUSchema model improves the F1. We also apply the same merging procedure to our model.

**Results:** In Table 3.5, the proposed method **Ours (Trans)** significantly outperform **CUSchema (LSTM)** before and after combining with universal schema. As far as we know, our proposed multi-facet embedding is the first method that outperforms compositional universal schema using the same training signal in the distant-supervised RE benchmark they proposed.

| Premise (Specific Pattern) | Hypothesis (General Pattern) | Label | Ours | CUSchema | Ours Diff | Freq Diff |
|---|---|---|---|---|---|---|
| $ARG1 , the president of the $ARG2 | $ARG1 , the leader of the $ARG2 | Entailment | 0.98 | 0.94 | + | + |
| $ARG1 's chairman , $ARG2 | $ARG1 's leader , $ARG2 | Entailment | 0.95 | 0.87 | + | - |
| $ARG1 's father , $ARG2 | $ARG1 's leader , $ARG2 | Other | 0.08 | 0.52 | NA | NA |
| $ARG1 worked with $ARG2 | $ARG1 deal with $ARG2 | Entailment | 0.92 | 0.83 | + | - |
| $ARG1 had with $ARG2 | $ARG1 deal with $ARG2 | Other | 0.96 | 0.88 | NA | NA |
| $ARG1 said the $ARG2 | $ARG1 say the $ARG2 | Paraphrase | 0.93 | 0.92 | NA | NA |

Table 3.6: Example of sentence pattern pairs, its label, and our predictions in our entailment experiment. **Ours** and **Ours Diff** are the predictions from **Ours (Trans)**. **Freq Diff** is the frequency difference baseline.

Although the recall of **USchema** is low because it does not exploit the similarity between the patterns (e.g., *"$ARG1 happily married $ARG2"* is similar to *"$ARG1 married $ARG2"*), **USchema** has a high precision because it also won't be misled by the similarity (e.g., *"$ARG1, and his wife $ARG2"* expresses the *spouse* relation but *"$ARG1, his wife, and $ARG2"* does not) [225]. Thus, combining **USchema** and **Ours (Trans)** leads to the best performance.

**Ours (Trans)** and **Ours (LSTM)** perform similarly. Furthermore, **Ours (LSTM)** performs much better than **Ours (Single-LSTM)**, which demonstrates the effectiveness of using multiple embeddings. Notice that multiple facet embeddings could improve the performance even after the training data have been cleaned. This indicates that our method is complementary to the noise removal methods in Roth et al. [187].

#### 3.3.2.2 Entailment Detection

Entailment is a common and fundamental relation between two sentence patterns. Some examples could be seen in Table 3.6. Unsupervised hypernym detection (i.e., entailment at the word level) is extensively studied [199], but we are not aware of any previous work on unsupervised entailment detection at the sentence level, nor any existing entailment dataset between sentence patterns. Thus, we create one.

**Dataset Creation:** We use WordNet [150] to discover the entailment candidates of sentence pattern pairs and manually label the candidates. For each sentence pattern in the training data of Verga et al. [225], we replace one word at a time with its hypernym based

on the WordNet hierarchy. The two sentence patterns before and after replacement form an entailment candidate.

We label 1,500 pairs of the most popular sentence pattern, which co-occurs with the highest number of unique entity pairs. Each candidate could be labeled as entailment, paraphrase, or other. Finally, around 20% of the candidates are randomly chosen to form the validation set, and the rest are in the test set.

In this dataset, only 22% and 10% of candidates are labeled as entailment and paraphrase, respectively. This suggests that entailment relation between two sentence patterns is hard to be inferred by only the hypernym relation (i.e., entailment relation at the word level) in WordNet.

**Setup:** We evaluate entailment detection using the typical setup and metrics in hypernym detection [199]. Negative examples include the candidates labeled as paraphrases and others. We compare the average precision of different methods (i.e., AUC in the precision-recall curve) [81]. In addition, we predict the direction of entailment relation in each pair (i.e., which pattern is the premise) and report the accuracy. Many hypotheses have the same hypernyms such as the *leader* in Table 3.6, so we also report the macro accuracy of direction detection averaged across every hypernym in the hypotheses.

The task is challenging because all the candidates have a word-level entailment relation if their compositional meaning is ignored. Furthermore, we cannot infer the entailment direction based on the tendency that longer sentence patterns tend to be more specific because most of the candidate pairs in this dataset have the same length.

As described in Section 3.3.1.3, our models detect the direction by computing **Ours Diff** as $Asym(\{\underline{\tilde{\boldsymbol{s}}}_{i,k}\}, \{\underline{\tilde{\boldsymbol{s}}}_{j,m}\}) - Asym(\{\underline{\tilde{\boldsymbol{s}}}_{j,m}\}, \{\underline{\tilde{\boldsymbol{s}}}_{i,k}\})$ and predict the $i$th sentence pattern to be premise if **Ours Diff** $> 0$. When performing entailment classification, we use as the asymmetric similarity scores $Asym(\{\underline{\tilde{\boldsymbol{s}}}_{i,k}\}, \{\underline{\tilde{\boldsymbol{s}}}_{j,m}\})$. We report the performance of **Ours (Trans)**, which is the same best model in the RE experiment.

| Method | Classification | Direction Detection | |
|---|---|---|---|
| | AP@all | Micro Acc | Macro Acc |
| Random | 21.9 | 50.0 | 50.0 |
| Freq Diff | 21.4 | 54.5 | 47.3 |
| CUSchema | 31.2 | 50.0 | 50.0 |
| Ours (Single) | 23.6 | 50.0 | 50.0 |
| Ours | **37.8** | **63.1** | **55.4** |

Table 3.7: Comparison of entailment detection methods. AP and Acc are average precision and accuracy, respectively. All numbers are %. Our methods use a transformer as their encoder.

In entailment classification, we compare the results with cosine similarity from **Ours (Single-Trans)** and **CUschema**. We also test the frequency difference, which is a strong baseline in hypernym direction detection [33]. **Freq Diff** = Freq($S_j$) - Freq($S_i$) where Freq($S_i$) is the number of unique entity pairs co-occurred with the $i$th sentence pattern. The baseline predicts $S_i$ to be premise if **Freq Diff** $> 0$ because more general sentence patterns should co-occur with more entity pairs. As a reference, we also report the performance of random scores.

**Results:** The quantitative and qualitative comparison are presented in Table 3.7 and Table 3.6, respectively. Our model that uses multi-facet embeddings significantly outperforms the other baselines. We hypothesize that a major reason is that the sentence patterns with an entailment relation are often similar in some but not all of the facets, and our asymmetric similarity measurement is better at capturing the facet overlapping.

### 3.3.3 Related Work

Relation extraction (RE) is widely studied. Han et al. [79] summarize the trend of recent studies and point out one of the major challenges is the cost of collecting the labels. Distant supervision [153] and its follow-up work enable us to collect a large amount of training data at a low cost, but the violation of its assumptions often introduces substantial noise into the supervision signal. Our goal is to alleviate the noise issue by representing every sentence pattern using multiple embeddings.

Other noise reduction methods have also been proposed [187]. For instance, we can adopt multi-instance learning techniques [250, 211, 4], global topic model [2], or both [186]. We can also reduce the noise by counting the number of shared entity pairs between a sentence pattern and a KB relation [213, 208]. Nevertheless, the studies focus on mitigating the noise caused by assuming similarity between the sentence patterns and KB relations that co-occur with the same entity pairs, while our method can also reduce the noise from two sentence patterns sharing the same entity pair. Besides, our method is complementary to popular noise reduction methods because our improvement is shown in the training data that have been cleaned [225].

Our method is conceptually related to some studies for lexical semantics. For example, word sense induction or unsupervised hypernymy detection can be addressed by clustering the co-occurring words [159, 9, 33]. However, the clustering-based methods do not apply to RE because the co-occurring matrix for RE is much sparser.

## 3.4 Applications on Citation and Authorship Prediction

In Section 3.2, we find that the longer input sequence lengths often contain more facets and might attract more interactions from diverse items. In this section, we want to test the effectiveness of multiple embeddings on representing a short document, the title and abstract of a paper. We choose two types of interactions/co-occurrence signals: citation and authorship because a paper might be written by different types of authors or cited by different types of papers. For instance, a multidisciplinary paper written by machine learning (ML) researchers and biologists is about applying ML to a biomedical application. To represent the paper as a single embedding, the existing systems are forced to average embeddings of ML researchers and biologists, and the resulting paper embeddings might be far away from both types of author embeddings.

In this section, we propose a content-based neural recommendation system that predicts cluster centers from only the title and abstract of a paper and each cluster center is an

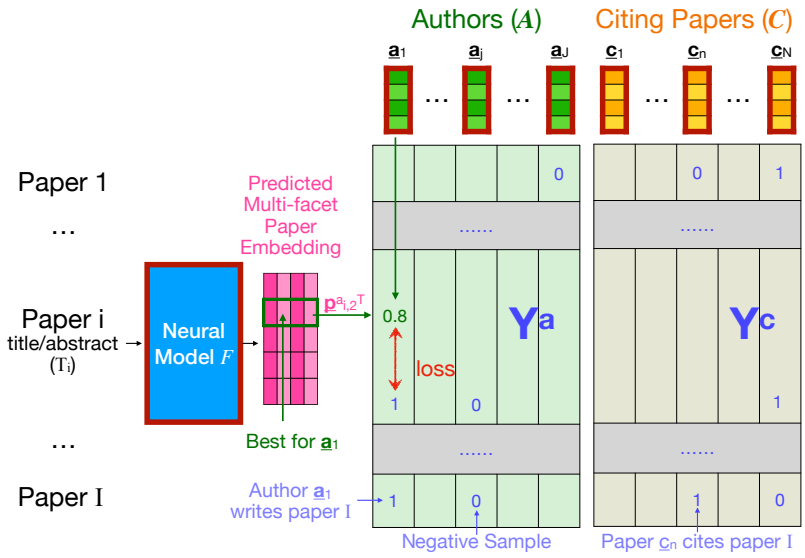



Figure 3.7: The proposed learning framework. We train the components with red bolder (i.e., $F$, $\boldsymbol{A}$, and $\boldsymbol{C}$) by minimizing the difference between our predictions and the interaction matrices, $\boldsymbol{Y^a}$ and $\boldsymbol{Y^c}$. During testing, we encode an unseen paper using our neural model into $K$ embeddings ($K = 5$ in this case) and predict the relevancy scores by choosing the best one among the $K$ embeddings.

embedding corresponding to a facet of the paper. To achieve the goal, we encourage the multi-facet embeddings of papers to be similar during training if they are written by the same author(s) or cited by the same paper(s). Our experiment results demonstrate that the proposed multi-facet embeddings improve our authorship/citation prediction model.

### 3.4.1 Method

#### 3.4.1.1 Background and Problem Formulation

As in Figure 3.7, the inputs of our authorship prediction problem are a training paper set $T = \cup_{i=1}^{I}\{T_i\}$ and a $I \times J$ authorship matrix $\boldsymbol{Y^a} = [y^a_{i,j}]_{i,j}$, where $T_i$ is the concatenation of the title and abstract of the $i$th paper and $y^a_{i,j} = 1$ if the $j$th author writes the $i$th paper and $y^a_{i,j} = 0$ otherwise. Our goal is to predict the likelihood that the $j$th author writes an unseen paper based on its text (i.e., fill each value in an empty row).

Similarly, in our citation prediction problem, we are given $T$ and a citation matrix $\boldsymbol{Y^c} = [y^c_{i,n}]_{i,n}$, where $y^c_{i,n} = 1$ if the $n$th paper cites the $i$th paper, and we want to compute

how likely we can properly add a new citation into the $n$th paper based on the existing citations of the $n$th paper.

#### 3.4.1.2 Authorship and Citation Prediction

To simplify the explanation, we first describe our loss for author prediction and extend the loss to the citation prediction.

To predict the authorship of an unseen paper, we learn a neural encoder $F^a$ to map the text of the paper to multiple embeddings, and train the model by encouraging the embedding of its author $\underline{\boldsymbol{a}}_j$ and one of the paper embeddings to be close to each other.

We define author prediction loss $L^S(F^a, \boldsymbol{A}, T, \boldsymbol{Y^a})$ as

$$\sum_{(i,j)\in R^a} r^a_{i,j} \left(S(F^a(T_i), \underline{\boldsymbol{a}}_j) - y^a_{i,j}\right)^2, \tag{3.12}$$

where $S(F^a(T_i), \underline{\boldsymbol{a}}_j)$ is the predicted relevancy score that represent how likely the $j$th author writes the $i$th paper. $F^a(T_i) = \{\underline{\boldsymbol{p}}^a_{i,k}\}_{k=1}^K$ is a set of $K$ embeddings of the $i$th paper and $\underline{\boldsymbol{p}}^a_{i,k}$ is the $k$th embedding predicted by our neural model. $\underline{\boldsymbol{a}}_j$ is the embedding of the $j$th author and the trainable author embedding matrix $\boldsymbol{A} = [\underline{\boldsymbol{a}}_j]_{j=1}^J$.

For each positive sample $(i, j)$ such that $y^a_{i,j} = 1$, we create a negative sample by replacing the $i$th paper with a randomly selected paper. $R^a$ is a set that includes all positive and negative samples. To avoid the model biased toward the authors who write many papers, we set the weight of each pair $r^a_{i,j}$ to be proportional to the inverse of the paper number written by the $j$th author, $\frac{1}{\sum_i y^a_{i,j}}$, as in Equation 3.9, and normalize the weights to make the average of $r^a_{i,j}$ to be 1.

In the existing work like Bansal et al. [14], each paper is represented as a single embedding $F^a(T_i) = \{\underline{\boldsymbol{p}}^a_{i,1}\}$ and $S(F^a(T_i), \underline{\boldsymbol{a}}_j) = \underline{\boldsymbol{a}}_j^T \underline{\boldsymbol{p}}^a_{i,1}$. Then, to increase relevancy scores between each paper and its authors, Equation 3.12 encourages the paper embedding to be the average of its author embeddings and encourages its author embeddings close to each

other. When the authors have diverse backgrounds, forcing the embeddings of the papaer and its authors to be similar results in the loss of information.

To address the limitation, we represent each paper using $K$ different embeddings. As in Figure 3.7, we use a neural model to encode the title and abstract of an input paper and output $K$ embeddings. For each author, we select the output paper embedding that produces the largest dot product to predict how likely the author writes the paper. Formally, the predicted paper embedding for the $j$th author

$$S(F^a(T_i), \underline{\boldsymbol{a}}_j) = \max(0, \max_{k=1}^{K} \underline{\boldsymbol{a}}_j^T \underline{\boldsymbol{p}}_{i,k}^a). \tag{3.13}$$

We truncate $S(F^a(T_i), \underline{\boldsymbol{a}}_j)$ to 0 when $\underline{\boldsymbol{a}}_j^T \underline{\boldsymbol{p}}_{i,k}^a < 0$ to ensure the relevancy scores are positive and let a larger magnitude of $\underline{\boldsymbol{p}}_{i,k}^a$ yield larger scores.

Using $K > 1$ embeddings, we allow the embeddings of authors with different backgrounds to be more diverse. For example, assuming a biomedical researcher and a ML researchers co-author a ML paper with a biomedical application, the embedding of the biomedical author could be close to a paper embedding and the ML author embedding could be close to another paper embedding.

The same loss can be applied to our citation prediction problem $L^S(F^c, \boldsymbol{C}, T, \boldsymbol{Y^c})$ except that we train the citing paper embedding matrix $\boldsymbol{C}$ and the neural model $F^c$ by factorizing the citation interaction matrix $\boldsymbol{Y^c}$. Multiple paper embeddings are also helpful in this problem because a multidisciplinary paper could be cited by other papers in different domains and with very different citing paper embeddings.

We perform multi-task learning by sharing the most of the parameters in $F^a$ and $F^c$ and jointly completes the values of the authorship matrix and citation matrix by minimizing

$$\alpha \cdot L^S(F^a, \boldsymbol{A}, T, \boldsymbol{Y^a}) + \beta \cdot L^S(F^c, \boldsymbol{C}, T, \boldsymbol{Y^c}) + \Omega^S(F, \boldsymbol{H}, T) \tag{3.14}$$

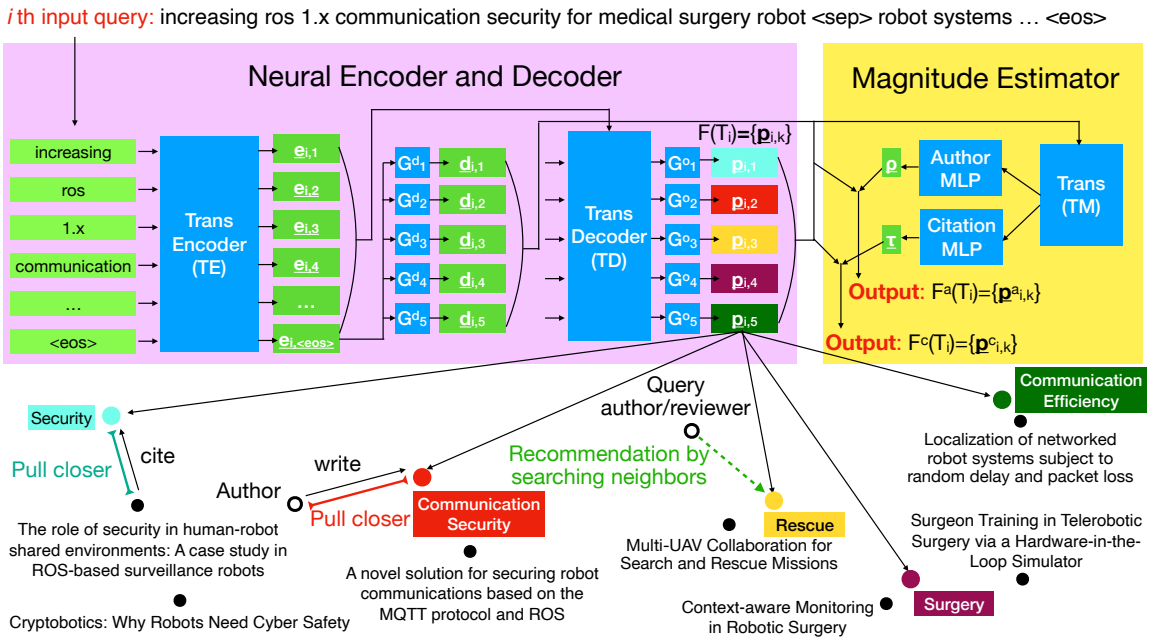



Figure 3.8: The architecture of our neural model and its predicted $K = 5$ paper embeddings when its input is a lowercased robotics paper [49]. All the embeddings of input papers (colored dots), authors (white dots), and citing papers (black dots) are mapped to the same vector space. The top-left purple box outputs paper embedding $\underline{\boldsymbol{p}}_{i,k}$ and the top-right yellow box outputs paper embedding $\underline{\boldsymbol{p}}^a_{i,k}$ for authorship prediction and $\underline{\boldsymbol{p}}^c_{i,k}$ for citation prediction. We manually tag each paper embedding (e.g., security and surgery) according to the citing papers closest to them for the visualization purpose.

where we set $\alpha : \beta = 5 : 1$ and $\Omega^S(F, \boldsymbol{H}, T)$ is an autoencoder regularization term as we did in Equation 3.9 for the relation extraction model. Please see Appendix B.3 for the definition of $\Omega^S(F, \boldsymbol{H}, T)$. The loss encourages some embeddings of two papers to be similar (i.e., the papers share some facets) if the papers are likely to be written by the same author or cited by the same paper.

In the lower part of Figure 3.8, we visualize the predicted embeddings of the input query paper and the nearby embeddings of citing papers. When a paper cites the query, the embedding of the citing paper is pulled closer to the closest query paper embedding during training. Having multiple embeddings lets the citing paper embedding be updated by relevant facets of the query and ignore the irrelevant ones.

### 3.4.2 Neural Architecture

We modify the seq2seq architecture in Section 3.2 for our paper recommendation applications. The main difference is that we let the output of transformer decoder $\underline{\boldsymbol{u}}_{i,k}$ passes through $K$ different linear layers $G_k^o$: $\underline{\boldsymbol{p}}_{i,k} = G_k^o(\underline{\boldsymbol{u}}_{i,k})$. We discover that this extra layer is crucial to prevent multi-facet embeddings collapsing to a single embedding in recommendation tasks.

As in Figure 3.8, we use a transformer to encode the input sequence and model the compositional meaning of words. Then, we use another transformer as our decoder to reduce the number of embeddings to $K$ and also model the dependency among the output embeddings. Notice that the transformers could be replaced by any encoder and we also try bi-GRU [39] as in Bansal et al. [14].

In Equation 3.13, the higher magnitude of paper embedding $\underline{\boldsymbol{p}}^a_{i,k}$ leads to higher relevancy scores to authors, so predicting the magnitude of each paper corresponds to predicting its prior interaction probability (i.e., how many authors who write the paper).

We concatenate $\underline{\boldsymbol{p}}_{i,k}$ with the input embedding of our decoder $\underline{\boldsymbol{d}}_{i,k}$, and feed these embeddings to another 2-layer transformer $TM$. The output embedding of $TM$ is passed to two feed-forward networks with 3 layers to convert the embedding to two vectors, $[\rho_{i,k}]_{k=1}^K$ and $[\tau_{i,k}]_{k=1}^K$, as the magnitudes of paper embeddings for author prediction $\underline{\boldsymbol{p}}^a_{i,k}$ and citation prediction $\underline{\boldsymbol{p}}^c_{i,k}$, respectively. That is,

$$\forall 1 \leq k \leq K \;\; \underline{\boldsymbol{p}}^a_{i,k} = \rho_{i,k}\underline{\boldsymbol{p}}_{i,k}, \text{ and } \; \underline{\boldsymbol{p}}^c_{i,k} = \tau_{i,k}\underline{\boldsymbol{p}}_{i,k}, \tag{3.15}$$

where each paper embedding has its own predicted magnitude, because the importance of facets might be different.

### 3.4.3 Experiments

In this section, we evaluate the ability of our model to predict how likely each author writes an unseen paper and each citing paper cites an unseen paper.

#### 3.4.3.1 Evaluation Setup

To train our model, we collect 260,857 papers from S2ORC [132] that are in CS domain and cited by at least one ML/AI related paper on ArXiv. After we remove the authors who only write one paper and the citing papers that cite less than 5 papers in our paper set, the dataset contains 681,714 authorship and 9,313,128 citation interactions.

As in Bansal et al. [14], we randomly choose 10% of papers that have at least 5 authors as our test set and take the 10% of the rest of the papers as our validation set. The test and validation set consist of 3,033 and 26,018 papers, respectively. The performance on the validation and test set follow the same trend, so we only report the results on the test set.

The hidden size of our encoder and decoder are 200 dimensions. The hyperparameters (except the $K$ value) are determined by optimizing the performance of the $K = 1$ model in the validation set.

#### 3.4.3.2 Comparing Methods

The first type of methods we test is the variants of our models. The second type is based on widely-used cosine similarity between document embeddings. The third type is the combination of the above two types.

**K=1**: We compute $S(F^a(T_i), \underline{\boldsymbol{a}}_j)$ in Equation 3.13 as our authorship relevancy. This baseline extends the cold-start recommendation model in Bansal et al. [14] by training on authorship and citation data, adding regularization loss $\Omega^S(F, T)$ in Equation 3.14, and replacing the GRU encoder with transformers. When $K = 1$, our transformer decoder falls back to 3 layers of attention to the output of our encoder plus feed-forward layers.

Similar to the authorship prediction, we compute the relevancy score for citation prediction using $S(F^c(T_i), \underline{\boldsymbol{c}}_n)$, where $F^c(T_i) = \{\underline{\boldsymbol{p}}^c_{i,k}\}_{k=1}^K$ is a set of embeddings of $i$th paper for

citation prediction, $\underline{\boldsymbol{c}}_n$ is the embedding for $n$th citing paper, and the scoring function $S(.)$ is defined in Equation 3.13.

**K=1 (no auto)**: We do not use the autoencoder regularization during training by setting the $\gamma = 0$ in Equation 3.14. Notice that the method is closer to the model in Bansal et al. [14] than K=1.

**K=5**: Our method using $5$ embeddings to represent each paper. We try $K = 3$ and $K = 5$ and find that both models outperform $K = 1$ with a similar margin, so we focus on comparing $K = 5$ and $K = 1$.

**K=1 (GRU) and K=5 (GRU)**: Our model that uses 2-layers of bidirectional GRU (bi-GRU) instead of 3 layers of transformer as the text encoder.

**CBOW Avg**: We first downweight the words with high frequency (e.g., stop words) [7] and compute the weighted average of CBOW (continuous bag of words) embedding [148] of each paper as our paper embedding. Then, the relevancy score between a testing paper and an author is computed by the average of cosine similarities between the testing paper embedding and the embeddings of all papers written by the author. Similarly, we average cosine similarities for citation prediction. We adopt CBOW from Bansal et al. [14], which is trained on ACM papers.

**CBOW Max**: Same as CBOW (Uniform) Max except that we compute the paper embedding using Equation B.4 [7] as in CBOW Avg.

**K=* + CBOW ***: We combine the relevancy scores from different cold-start recommendation variants ($s_{cold}$) and those from CBOW Avg/Max ($s_{CBOW}$) using $\lambda s_{cold} + (1-\lambda)s_{CBOW}$. The $\lambda = 0.2$ is determined by optimizing the MAP of $K = 1$ model in the authorship prediction.

#### 3.4.3.3 Metrics

Given a query author, models predict which papers the query author writes in the test set. Similarly, given a citing paper, we predict which papers it cites in the test set. In other

| Method | Authorship | | | | | Citation | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MAP | AUC | NDCG | R@50 | R@200 | MAP | AUC | NDCG | R@50 | R@200 |
| K=1 (no auto) | 10.15 | **90.71** | 25.14 | 47.68 | 72.84 | 18.50 | 96.48 | 33.83 | 71.29 | 88.80 |
| K=1 | 12.36 | 90.44 | 26.87 | 46.88 | 70.85 | 18.81 | 96.27 | 33.89 | 69.22 | 88.13 |
| K=3 | 15.13 | 89.68 | 29.61 | 52.89 | 73.98 | 22.86 | 96.39 | 37.70 | 75.19 | 89.96 |
| K=5 | 15.51 | 90.49 | 29.91 | 52.74 | 73.76 | 24.06 | 96.28 | 38.71 | 75.15 | 89.76 |
| K=1 (GRU) | 13.80 | 90.47 | 28.08 | 47.42 | 70.43 | 21.24 | 96.58 | 36.12 | 71.34 | 89.11 |
| K=5 (GRU) | **15.53** | 90.63 | **30.03** | **53.37** | **74.33** | **25.90** | **97.05** | **40.39** | **76.81** | **90.68** |
| CBOW Avg | 16.67 | 87.42 | 29.96 | 44.06 | 63.02 | 16.33 | 91.10 | 30.11 | 50.20 | 71.88 |
| CBOW Max | 20.37 | 88.29 | 33.45 | 46.97 | 65.58 | 15.46 | 91.26 | 29.34 | 49.34 | 72.79 |
| K=1 + CBOW Avg | 18.70 | 90.54 | 32.43 | 51.02 | 71.77 | 21.76 | 94.80 | 35.92 | 64.82 | 83.06 |
| K=5 + CBOW Avg | 19.66 | 90.35 | 33.34 | 53.07 | **72.47** | 23.66 | 94.76 | 37.71 | 67.91 | 84.08 |
| K=1 + CBOW Max | 22.61 | 90.89 | 36.00 | 54.32 | 73.11 | 22.17 | 94.62 | 36.27 | 64.73 | 83.72 |
| K=5 + CBOW Max | 23.23 | 90.63 | 36.57 | 55.79 | 73.44 | 24.13 | 94.54 | 38.10 | 67.94 | 84.48 |
| K=1 (GRU) + CBOW Max | 22.96 | **90.93** | 36.29 | 54.80 | 73.13 | 22.95 | 94.82 | 37.00 | 66.09 | 84.25 |
| K=5 (GRU) + CBOW Max | **23.69** | 90.82 | **37.05** | **56.49** | 74.12 | **24.42** | **94.85** | **38.42** | **68.81** | **85.06** |

Table 3.8: Results for cold-start authorship and citation prediction in our test set. AUC is the area under the ROC curve. We highlight the best values of cold-start recommendation methods with and without using weighted average of CBOW. K=1 (no auto) is an extension from Bansal et al. [14].

words, we evaluate the relevancy scores in each column of the matrix using the following classification/retrieval metrics and report the average across all columns.

**MAP**: Mean average precision (i.e., the area under the precision-recall curve) [261].

**AUC**: Average ROC-AUC (the area under the ROC curve) [81] across all columns.

**NDCG**: Average of normalized discounted cumulative gain [145] across all columns.

**R@50 and R@200**: Recall@N computes the recall of positive labels in the top $N$ papers with the highest relevancy scores. Recall@50 is a common metric for paper recommendation [14].

#### 3.4.3.4 Results and Discussion

Table 3.8 presents the performance of different methods. The results justify the motivation of using multi-facet embedding because **K=5** usually significantly outperforms **K=1**, especially in the citation prediction task.

Cold-start recommendation methods work the best in the citation prediction, and document similarity estimation (e.g., **CBOW Avg**) performs better in authorship prediction. The

hybrid methods consistently outperform only using document similarity estimation, which suggests our methods capture signals missed by the document similarity estimation.

Multi-facet embedding improves citation prediction more than authorship prediction. We hypothesize that this is due to the different nature of the interaction. When an author writes an input paper, the input paper tends to be similar to the other papers this author writes, so the document similarity estimation performs well in this task. On the other hand, when a citing paper cites an input paper, the input paper tends to be related but not similar to the other papers this citing paper cites [65] because the citation might be determined by one of the facets in the input paper.

### 3.4.4 Related Work

Due to its practicality, scientific paper recommendation has a long history and rich literature [15]. In addition to the classic methods based on collaborative filtering, recent work also exploits the citation of the testing papers [95] and the manually defined tags/topics [254] as additional information sources. Instead, we focus on a cold-start setting where the recommendation is made only based on the title and abstract of the testing papers as in Bansal et al. [14].

Citation recommendation is also studied widely [136, 59]. Collaborative filtering could effectively recommend the new citations based on existing citations of the paper [144, 127], but the method cannot recommend papers only based on its text. Bhagavatula et al. [17] designs a content-based and scalable citation recommendation system by representing each paper as a single embedding and performing efficient retrieval. However, the approach cannot capture multiple facets of the paper, which degrades performance significantly in our experiments.

A related type of recommendation systems clusters user and/or item embeddings globally [189, 93, 137]. However, global clustering causes information loss because the embeddings often need to be clustered differently when representing different items.

Multiple embeddings are also used to represent a graph node for node classification and link prediction [246, 128, 53] or an interaction history for sequence-aware recommendation [108, 234].

## 3.5 Chapter Conclusion

In this chapter, we find that multiple embeddings are particularly helpful when the input sequence is long enough to contain multiple facets/aspects and different facets would attract different types of co-occurred items. If the input sequence is too short (e.g., phrase) or each input sequence often co-occurs with the similar items (e.g., authorship), multiple embeddings might bring smaller improvement on co-occurrence prediction or might not improve the similarity estimation at all.

Furthermore, we show that multiple embeddings provide other benefits besides the similarity estimation. By modeling the distribution better, we can use the clustering center to estimate the importance of words, reduce the bias of selecting the long sentences in some unsupervised summarization approaches, improve the distantly-supervised relation extraction performance of compositional universal schema, detect entailment relation between two sentence patterns without any labels, and improve the cold-start prediction of citation and authorship.

# CHAPTER 4

# INTERACTIVE LANGUAGE GENERATION

In Chapter 3, we design our loss function such that the multi-facet embeddings become the clustering centers of the co-occurred item embeddings and we use the multiple embeddings to improve the similarity estimation. In this chapter, we show that the predicted clustering centers could be viewed as the possible future topics that might be mentioned after a prompt context and we build a novel interactive writing framework on the basis.

## 4.1 Introduction

Interactive writing assistants have wide applications in creative writing [185, 40, 1], education [135], and gaming [227]. Nevertheless, the existing systems' options usually do not provide fine-grained control and/or require substantial human labor. To address these limitations, we propose a framework that provides a set of future topics and guides the text generation by the user-chosen topics.

The topic options are generated dynamically based on the input prompt to provide fine-grained control, and our models are self-supervised without the need to define the attributes or collect annotations. As depicted in Figure 4.1, a user can peek at the most probable $K$ topics (shown as bags of words) appearing after the input prompt and control the generation by choosing the topics.

In Figure 4.2, we compare multiple generated sentences conditioned on different chosen topic(s) or specified word(s). For example, if the user chooses a topic about *humanity*, *life*, and *spirituality*, our system continues the input prompt "*Barack Obama writes a new book*" with "*on spirituality and the roles of religion in society*". Then, we can use the generated

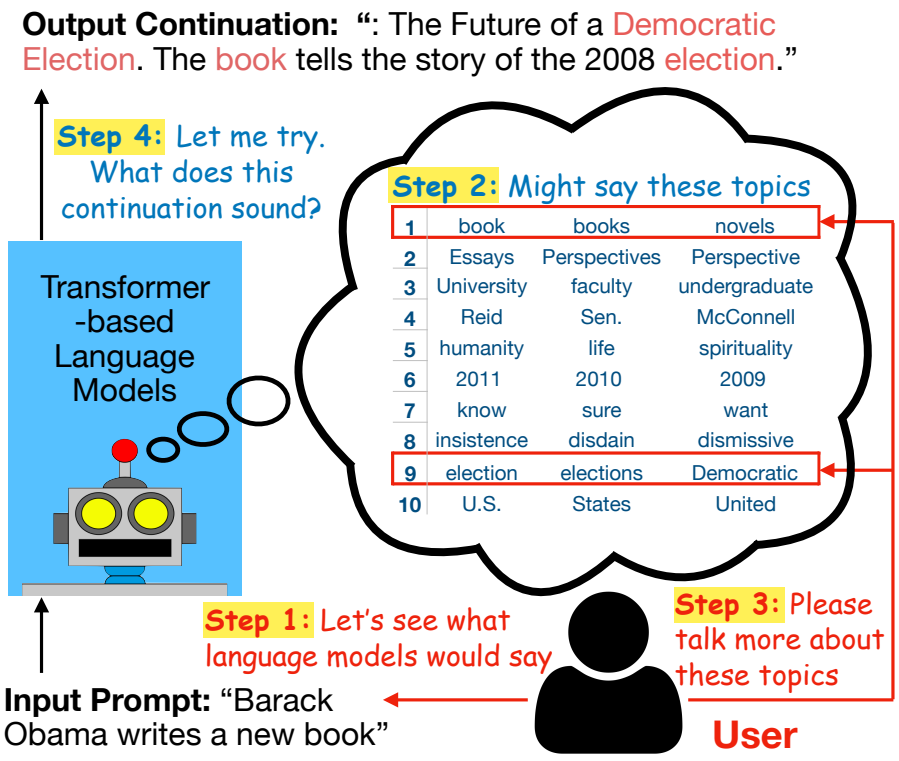



Figure 4.1: Given an input prompt, the transformer-based language model (LM) provides $K = 10$ topics that might be mentioned next and each topic is represented by $M = 3$ words. The user could guide the generation process by choosing a subset of topics.

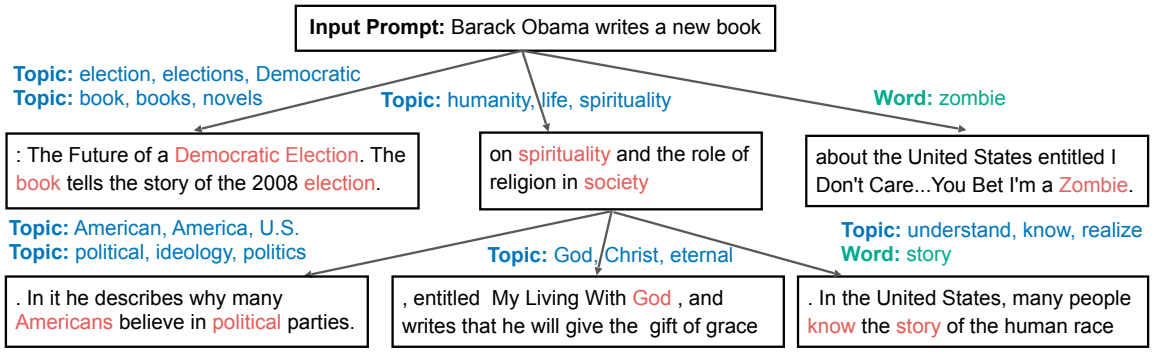



Figure 4.2: Examples of our generated options and continuations. We highlight the words in the continuation that are related to the chosen topics or to the specified word.

text as the new input prompt and update the set of topics to include other more relevant topics such as *God*, *Christ*, and *eternal*. The process can be repeated to create a plot tree.

A user can also control the generation by specifying word(s) if the user wants to see the words that are not in the topic list or seeks a transition to a word that is not directly related to the input prompt. For example, a user can ask our system to generate a sentence about *zombie*. Consequently, the continuation of *"Barack Obama writes a new book"* becomes *"about the United States entitled I Don't Care...You Bet I'm a Zombie"*.

The system is realized by two components: an option generator and a conditional text generator. The option generator is very similar to the models we developed in Chapter 3. Given a prompt, the option generator suggests a set of $K$ topics. After a user chooses a subset of the topics and specifies some words, the embedding of every word or topic will guide the conditional text generator to produce the continuation that is both consistent with the existing prompt and relevant to the chosen topics and words.

Both components are self-supervised and use pretrained GPT-2 models [177] to encode the input prompt. During training, the option generator predicts the cluster centers of future words, which are in the continuation of the prompt, based on the contextualized embeddings from GPT-2. The conditional text generator fine-tunes GPT-2 to predict the next words given the prompt and a few subsequent words. Since both components' input and output only come from the prompt and its continuation, training the system only requires a raw corpus, word tokenizers, and a list of stop words. This makes the proposed method suitable for open-domain story generation and easily being fine-tuned for a specific domain.

In experiments, we demonstrate that our system recommends high-quality topics and often generate sentences that follow the chosen topics. We compare our option generator with global topic models such as LDA [19] or local topic models such as clustering the words in the input prompt. The results show that the proposed method generates significantly more topics that are plausible and promote the narrative. Moreover, we compare our conditional text generator with PPLM (Plug and Play Language Models) [44] and demonstrate that our generation is more fluent and relevant to the chosen topics. Our code is available at `https://github.com/iesl/interactive_LM`.

## 4.2 Method

The proposed framework consists of two components: option generator and conditional text generator. In Figure 4.3, we illustrate the two components and their interaction. First, given the prompt $x_1, ..., x_I$ inputted by a user, the option generator at the bottom of the figure outputs $K$ topics. After the user chooses two topics about *book* and *election* and specifies one extra word *story*, the topics and word are passed to our text generator as the generation guidance. Accordingly, the generator continues to write the next token $\widehat{y}_1$.

In the following sections, we introduce our model designs and the way to train each component.

### 4.2.1 Option Generator

When we do not have labeled attributes in a corpus, we can create options by clustering all the words in a corpus into topics [222]. The clustering could be done by topic modeling approaches such as LDA [19]. The resulting topics are static (i.e., the clustering is performed globally without considering the prompt). However, the prompt might have a narrow focus and the related words of interest are all clustered into a single topic.

A simple remedy is to cluster only the words in the prompt rather than all the words in the corpus. The topics are created dynamically and locally given a prompt and can capture more fine-grained aspects in the continuations. However, the topics derived from the prompt might provide less inspiration because the users have seen the prompt. Another major drawback of the approach is that the generated topics might encourage the LM to generate repetitive sentences or make a narrative circle inside a loop.

Motivated by the challenges, we propose an option generator that predicts the cluster centers based on the prompt instead of clustering the words in the prompt during testing.

#### 4.2.1.1 Model Prediction

The goal of our option generator is to predict the $K$ cluster centers of words in the possible continuations and use the cluster centers as the topics user could choose from. The

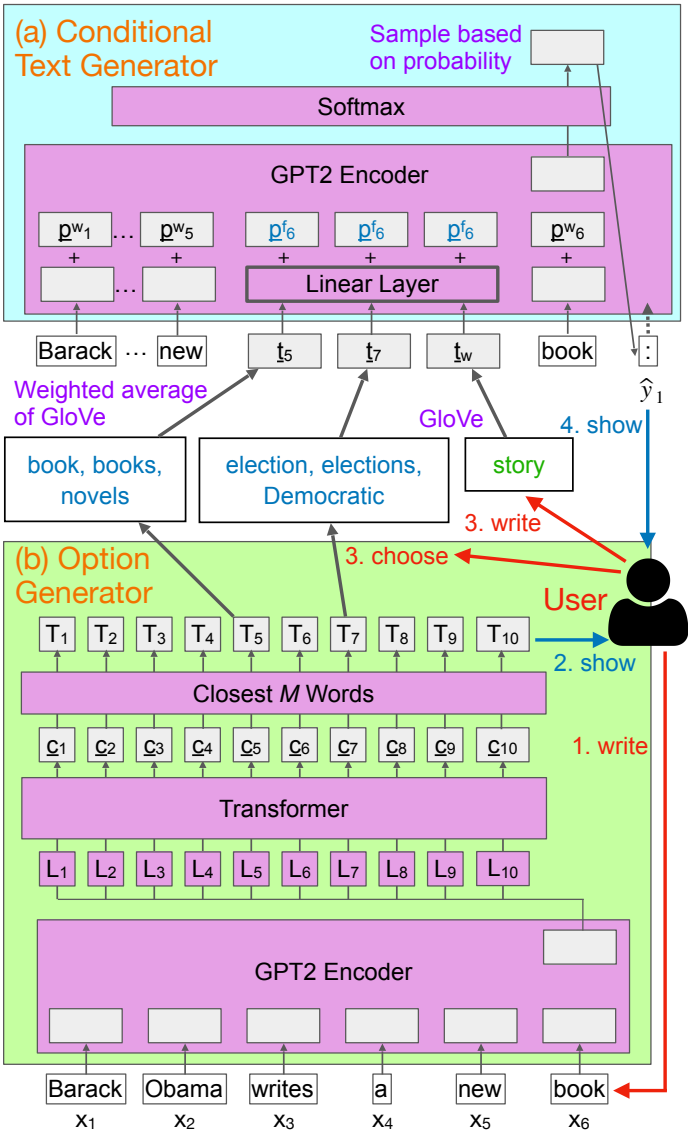



Figure 4.3: Our model architectures for (a) conditional text generator and (b) option generator. During testing, the information flows from the bottom to the top.

goal is similar to Section 2.3 except that our model predicts multiple embeddings that are close to a set of future words rather than close to the immediate next word.

As in Figure 4.3 (b), the option generator uses GPT-2 to encode the input prompt $x_1, ..., x_I$ and passes the output embedding to $K$ different linear layers $L_1, ..., L_K$. To model the dependency of clusters, a transformer [224] takes the $K$ embeddings as input and predicts the cluster centers $\underline{c}_1, ...\underline{c}_K$ in GloVe [174] space. During testing, each predicted cluster

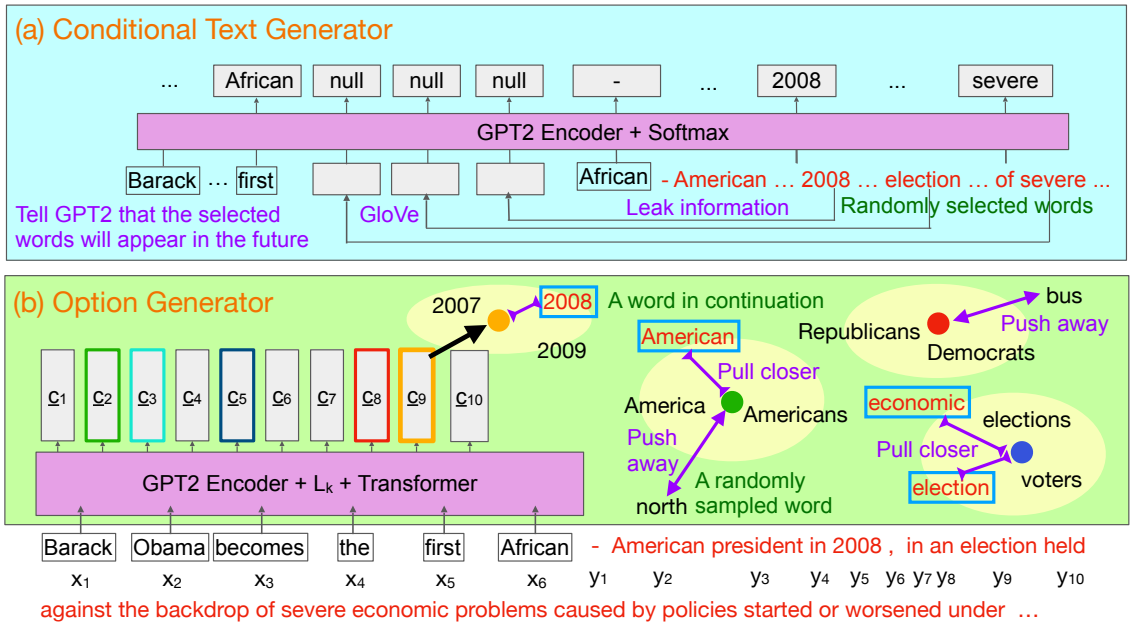



Figure 4.4: Training our two components using the same sentence. (a) We randomly pick $n = 3$ words in the actual continuation as our conditions for the text generator, and the null labels mean their predicted probabilities are ignored in our loss. (b) We visualize 5 out of $K = 10$ generated topics in a normalized GloVe space. Red words are the ones that appear in the continuation and pull the nearby cluster centers closer during training.

center is normalized by its L2 norm, and we use the $M$ closest words in the normalized GloVe space to represent the topic $T_i$, which users can choose.

We choose to learn the cluster centers in GloVe space rather than GPT-2 or BERT [47] space because the non-contextualized word embeddings are easier to visualize. Users can easily understand the meaning of a cluster center by seeing nearby words. We normalize GloVe space in this work to make the squared L2 distance equal to twice the cosine distance between two embeddings.

Our architecture is similar to the one in Chapter 3, but we use a pretrained GPT-2 encoder rather than train a BERT-like transformer from scratch. Another difference is that we ignore the connection between the second transformer and the output of GPT-2 to save GPU memory for handling a longer input prompt.

#### 4.2.1.2 Model Training

In Figure 4.4 (b), we visualize our training procedure. For each input prompt in the training corpus, we run a forward pass through the transformers and get predicted cluster centers $\underline{c}_1, ...\underline{c}_K$. Next, we collect 50 words in the continuation (except stop words) as positive examples and match the words with cluster centers as in the E-step of the EM algorithm [46]. By using the NNSC clustering loss in Equation 3.2, we minimize the distances between the centers and their nearby positive examples by backpropagating the gradients through the matching and updating our transformer models. We also randomly sample some words as negative examples and maximize the distances between the cluster centers and their nearby embeddings from negative examples.

Using Figure 4.4 (b) as an example, the orange cluster center is pulled closer toward the embedding of *2008*, which appears in the continuation. The green cluster center is pushed away from the embedding of *north*, a randomly sampled word. Since each output embedding $\underline{c}_k$ is pulled by only the nearby embeddings of words in the continuation, the output embedding will naturally become the cluster center of the nearby continuation word embeddings. Notice that the related topics like *Democrats* and *Republicans* are not observed in the prompt and continuation, but our model can predict a red cluster center close to them because the model can learn from other similar input prompts whose continuation mentions words like *Democrats*.

### 4.2.2 Conditional Text Generator

After the user chooses topic(s) or specifies word(s), each topic or word is converted to a GloVe embedding. The component aims to generate the text given the input prompt and the GloVe embeddings of the topics or words we prefer to see in the continuation.

Users only see the $M$ words closest to the $k$th predicted cluster center $\underline{c}_k$ from our option generator, so we compute the $k$th topic embedding as

$$\underline{t}_k = \frac{\sum_{m=1}^{M} \cos(\underline{e}_m^w, \underline{c}_k)\underline{e}_m^w}{||\sum_{m=1}^{M} \cos(\underline{e}_m^w, \underline{c}_k)\underline{e}_m^w||}, \tag{4.1}$$

where $\underline{e}_m^w$ is the normalized GloVe embedding of the $m$th closet word and $\cos(\underline{e}_m^w, \underline{c}_k)$ is the cosine similarities between the $m$th word embedding and the embedding $\underline{c}_k$.

#### 4.2.2.1 Model Prediction

During testing, the topic embeddings $\underline{t}_k$ or embedding of the specified words are inserted into GPT-2 encoder before $x_I$, the last word piece in the prompt. The inserted embeddings nudge GPT-2 to generate the sentences containing the desired words with a higher probability.

As Figure 4.3 (a) shows, the GloVe embeddings are first passed through a linear layer to make their dimension become the same as the hidden state size of GPT-2. Then, the transformed embeddings are added with special positional embeddings $\underline{p}_I^f$, which are different from those for the prompt $\underline{p}_i^w$. The special positional embedding tells GPT-2 that the inserted embeddings have a different meaning and where the conditional generation starts.

The GPT-2 encoder's output goes through a softmax layer, which computes the probability of each token being observed as the first word piece in the continuation $y_1$. We adopt top-k sampling [56], which reduces the chance of sampling words with low probability, to pick the next word, and autoregressively sample one token $\widehat{y}_o$ at a time to generate the continuation $\widehat{y}_1, ..., \widehat{y}_O$.

#### 4.2.2.2 Model Training

We train the generator using the continuation of a prompt and some randomly selected non-stop words in the continuation as its generation conditions. Since the continuation contains the randomly-selected words, the generator would be heavily penalized if it ignores the conditions by assigning low probabilities to the selected words in all the continuation positions.

An example is illustrated in Figure 4.4 (a). Given an input prompt in the training set, we randomly pick a number $n$ from 0 to $K$ and sample $n$ words from the next $O = 25$ words (except stop words). Next, the normalized GloVe embeddings of $n$ words are inserted to the GPT-2 encoder before the last word piece in the prompt, and we ignore the output probabilities corresponding to the inserted positions during training. To speed up the training, we conduct the future word insertion in multiple positions of each training text sequence.

We insert the future words just before the text that might contain the words rather than at the beginning as in the classic seq2seq model, because we do not want the model to learn to generate the continuation based on the future topics that have not yet be specified by the users (e.g., GPT-2 should not know that it will see *election* in the future when it learns to generate *Barack Obama ...* during training).

By allowing the LM to see the upcoming words earlier, we leak partial label information to the LM input. Consequently, GPT-2 learns to utilize the information and generate the sentence containing the desired words to achieve a lower perplexity loss. Notice that the training method allows us to specify our topical preference without significantly scarifying generation efficiency and fluency, but it cannot guarantee to generate all the desired topics, especially when we specify multiple ones.

One concern of the method is that the LM cannot see all possible sets of topics or words users might specify during training. Besides, each GloVe embedding used to supervise LM comes from a single word, but we ask the LM to condition on average GloVe embedding of the top $M$ words during testing. Nevertheless, we observe that the LM is often able to generalize well in our experiments because similar words have similar GloVe embeddings, lots of training instances could be easily prepared by the self-supervised method, and our option generator usually provides the topics mentioned in the continuation in our training corpus.

## 4.3 Experiments

We evaluate two components separately, and both evaluations include automated metrics and human judgment. Throughout the evaluation, the number of topics $K = 10$ and the length of generations is 50 word pieces. We find that fixing $K = 10$ works well in our experiments. If the possible continuations cover more than 10 topics, our option generator tends to output the important topics. If they cover fewer topics, our option generator tends to output the related topics that are not explicitly mentioned in the prompt or the duplicated topics.

### 4.3.1 Datasets

We use 90% of English Wikipedia 2016 as our training set for both components, 5% as our validation set to determine the hyperparameters such as the number of epochs, and the remaining 5% as our test set to perform the automated evaluation.

For human evaluation, we collect labels from Amazon Mechanical Turk (MTurk). We randomly sample sentences from the training set of STS benchmark (STSb) [27] as our input prompts. Compared with Wikipedia, the sentences from STSb are easier to understand for annotators because a large portion of sentences in Wikipedia involves terminologies, depends on a longer context, or might even just be a list of names.

In STSb, we sample 24 sentences as our prompts, and each method generates one continuation for each input prompt. Each generated continuation or topics will be scored by three different workers.

### 4.3.2 Option Generator Evaluation

We evaluate the topics from different option generators by judging whether the topics will appear in the continuation and whether the topics would promote the narrative. The goal is to have topics that are relevant and provide new information. The topics that are too similar to the prompt words might be redundant and not helpful because the users have already seen the prompt.

#### 4.3.2.1 Automatic Evaluation Metrics

- **Sim**: If the generated topics $T$ can help users to write the continuation, the embedding of every non-stop word in the actual continuation should be similar to the embeddings of a generated topic. Thus, we compute

$$\text{Sim}(\bar{Y}, T) = \sum_{o=1}^{O'} \max_{k=1}^{K} (\underline{t}_k)^T \underline{e}_o^{\bar{y}}, \tag{4.2}$$

  where $\bar{Y} = \{\bar{y}_o\}_{o=1}^{O'}$ is a set of non-stop words in the continuation and $O' = 25$. $\underline{t}_k$ is the normalized embedding of $k$th topic in $T$ from Equation 4.1 and $\underline{e}_o^{\bar{y}}$ is the $o$th word in $\bar{Y}$.

- **Sim Short**: When computing Sim, we use the input prompts containing around 180 words on average. To examine the topic quality at the start of writing, where the authors might need assistance the most, we also report Sim$(\bar{Y}, T)$ on short input prompts (with 35 words on average).

- **Sim Diff**: The options that are helpful to users should be sufficiently different from the words in the input prompt to promote the narrative and avoid generating repeated content. Thereby, we also evaluate methods using Sim Diff = Sim$(\bar{Y}, T)$ - Sim$(\bar{X}, T)$, where $\bar{X} = \{\bar{x}_i\}_{i=1}^{I'}$ are the non-stop words in the input prompt.

#### 4.3.2.2 Human Evaluation

Our questionnaire shows the prompt and asks which generated topics are likely to appear in a reasonable continuation and which topics promote the narrative. For each method, we report the average number of its topics that are likely to appear (L), promote the topic (TP), and both (L&TP). For example, an MTurk worker is shown three topics generated by a method given a prompt: $ABC$. The worker thinks $A$ is likely to appear in the continuation and $AB$ promote the topic. Then, L=$|\{A\}|$=1, TP=$|\{AB\}|$=2, and L&TP=$|\{A\} \cap \{AB\}|$=$|\{A\}|$=1 for this prompt.

#### 4.3.2.3 Option Generator Baselines

We compare our generator with two types of methods.[1] The first type performs the clustering globally and selects the most relevant topics to the input prompt from the static set of clusters. We cluster all the words into $J = 150$ topics by LDA [19] (**LDA-global**) and into $J = 1000$ topics by Kmeans on the normalized GloVe embedding space [222] (**Kmeans-global**). We also randomly sample $K$ words from the whole vocabulary as our cluster centers (**Sample-global**).

Similar to Equation 4.1, we find the $M$ words with the closest embeddings to each cluster center to represent the topic and compute the topic embedding $\underline{t}_j$ as the weighted average embedding of $M$ words in the $j$th topic. Among all $J$ cluster centers, we pick the $K$ topics with the closest $\underline{t}_j$ to the prompt embedding, where the prompt embedding is the average embedding of all words in the input prompt.

The second type of methods discovers the $K$ topics from the input prompt. We cluster non-stop words in the prompt using non-negative sparse coding [89] (**NNSC-local**) and Kmeans (**Kmeans-local**). We also sample $K$ non-stop words from the prompt and call it **Sample-local**. Similar to Equation 4.1, we represent each topic using $M$ words and compute the weighted average of their embeddings $\underline{t}_k$ as the input of our text generator. Notice that the locally clustering methods produce similar results when the prompts come from STSb due to their short lengths, so we only test **Kmeans-local** in our human evaluation.

#### 4.3.2.4 Results

In Table 4.1, we show that local methods generate the options more relevant to the input prompt than the global methods due to significantly higher Sim and Sim Short. Our method performs better compared to other local methods, especially in Sim Diff, which highlights the high novelty of our generated topics. The improvement on Sim Short is larger than that

[1] Another alternative is to generate many continuations and cluster the words in the generation. However, the method takes time, which might be prohibited by limited computational resources and the real-time interaction requirement.

| Scope | Method | Sim | Sim Short | Sim Diff |
|---|---|---|---|---|
| Global | Sample | 14.63 | 14.42 | **0.16** |
| | LDA | 36.86 | 36.02 | -2.82 |
| | Kmeans | **40.65** | **39.91** | -3.40 |
| Local | Sample | 41.50 | 41.23 | -12.51 |
| | NNSC | 46.70 | 42.80 | -15.94 |
| | Kmeans | 47.94 | 43.89 | -16.12 |
| | Ours | **48.38** | **46.29** | **0.45** |

Table 4.1: Comparison of the option generators using automatic metrics. The best numbers within each scope are highlighted.

| Scope | Method | L | TP | L&TP |
|---|---|---|---|---|
| Global | LDA | $5.76 \pm 0.50$ | $6.24 \pm 0.33$ | $5.26 \pm 0.31$ |
| | Kmeans | $6.94 \pm 0.36$ | $6.13 \pm 0.30$ | $5.96 \pm 0.31$ |
| Local | Kmeans | $\mathbf{8.65} \pm 0.16$ | $5.31 \pm 0.50$ | $5.14 \pm 0.50$ |
| | Ours | $7.85 \pm 0.25$ | $\mathbf{6.96} \pm 0.26$ | $\mathbf{6.75} \pm 0.28$ |

Table 4.2: Comparison of option generators using human judgment (mean $\pm$ standard error). L and TP refer to likelihood and topic promotion, respectively.

| Input Prompt | The study also found that skin cancer nearly tripled in Norway and Sweden since the 1950s. | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| LDA-global | | | | Kmeans-local | | | | Ours | | | |
| 1 | population, households | 6 | company, companies | 1 | Norway, Sweden | 6 | also, however | 1 | research, scientific | 6 | 1980s, 1970s |
| 2 | patients, treatment | 7 | Norwegian, Norway | 2 | tripled, doubled | 7 | since, Since | 2 | tissues, tissue | 7 | even, though |
| 3 | psychology, research | 8 | story, book | 3 | nearly, almost | 8 | Sweden, Finland | 3 | patients, diagnosis | 8 | susceptibility, pathogenic |
| 4 | police, prison | 9 | hospital, Hospital | 4 | cancer, skin | 9 | study, studies | 4 | DNA, gene | 9 | decreased, increased |
| 5 | chemical, carbon | 10 | Icelandic, Iceland | 5 | 1950s, 1940s | 10 | found, discovered | 5 | orange, purple | 10 | Sweden, Norway |

Table 4.3: Comparison of all $K$ topics for the input prompt using $M = 2$ words closest to each topic.

| Input Prompt | | The study also found that skin cancer nearly tripled in Norway and Sweden since the 1950s. |
|---|---|---|
| Generator | | Generated Text |
| Option | Text | |
| LDA-global | Ours | A study of the Norwegian police has confirmed the cancer case. The law in Norway was the subject of the |
| Kmeans-local | Ours | The study also found that skin cancer nearly tripled in Norway and Sweden since the 1950s. As well, skin |
| Ours | PPLM | In this study, a study was conducted conducted in Italy and in Finland. From the 1990s to the 1970s, there |
| None | GPT-2 | The study also revealed that only 20% of the deaths in Norway were caused by a sudden cardiac response |
| Ours | Ours | Recent studies have shown that melanin causes a decrease in genetic susceptibility in people in Norway, |

Table 4.4: The continuations that are generated by conditioning on all of $K$ topics from different option generators. The input prompt comes from STSb.

on Sim because our method could suggest the related topics that are not explicitly mentioned in the short prompt (e.g., *U.S.* in Figure 4.1).

The human evaluation results are presented in Table 4.2. Our method wins in terms of generating relevant topics that promote the narrative. The **Kmeans-local** performs better

in L because most of the words in the input prompts could be mentioned again in the next sentence. However, it often leads to the redundant topics that are too similar to the prompt.

Table 4.3 compares the options generated by different methods while Table 4.4 compares the text generated using different option generators and text generators. In Table 4.3, we can see that most topics in **Kmeans-local** do not promote the narrative, which makes the generated continuation become a copy of the input prompt in Table 4.4. Notice that the high redundancy problem is hard to be solved by a conditional text generator because the relatedness between the prompt and the generated text is hard to be controlled [192].

### 4.3.3 Conditional Text Generator Evaluation

To demonstrate our text generator's effectiveness, we use our option generator to prepare the topic embeddings and randomly select $n$ topics as our conditions to simulate the user's choice, where $n$ is a random number from 1 to $K$. The sentences generated by different methods are compared.

#### 4.3.3.1 Automatic Evaluation Metrics

We match the union of $M \times K$ top words in the chosen topics with the words in the generated continuations and count the number of tokens that are matched exactly (token), the number of matched word types (word), and the number of topics that contain at least one matched word (topic) to measure the relevancy between the continuations and the chosen topics. Notice that the scores are underestimated because the generation might mention words in different morphological variations or other words related to the topics.

The fluency of the generated text is measured using the perplexity [195] of the original GPT-2 (with 345M parameters) without being fine-tuned on Wikipedia. Dist-n [116] is the ratio between the number of unique n-grams and the number of all n-grams in the continuations, where n=1 or 2. Higher Dist-n implies more diverse generations. The average inference time per input prompt is also presented.

| Text Generation Method | Automatic Metrics | | | | | | Inference Time | Human Judgement | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Relevancy Hit | | | Quality | | | | Relevancy | | Fluency |
| | Token | Word | Topic | PPL (↓) | Dist-1 | Dist-2 | s (↓) | Recall | Precision | Score |
| PPLM | 1.48 | 0.99 | 0.77 | 18.49 | **40.29** | **80.83** | 17.74 | $30.56 \pm 2.96$ | $56.01 \pm 4.41$ | $3.83 \pm 0.13$ |
| Ours | **2.36** | **1.79** | **1.40** | **16.39** | 37.98 | 79.65 | **1.02** | **41.46** $\pm 3.47$ | **56.41** $\pm 4.41$ | **4.07** $\pm 0.10$ |
| GPT-2 | 1.27 | 0.84 | 0.64 | 14.24 | 39.80 | 80.22 | 1.00 | $24.49 \pm 2.77$ | $48.69 \pm 4.61$ | $4.15 \pm 0.11$ |

Table 4.5: Comparison of conditional text generators. The numbers in Dist-1, Dist-2, Recall, and Precision are percentages. Lower perplexity (PPL) and inference time are better. The better performance between PPLM and our method is highlighted. In human evaluation, we report the mean $\pm$ standard error of each method.

#### 4.3.3.2 Human Evaluation

We present the prompt and the generated continuation and ask the worker to score the generation's fluency from 1 (not fluent at all) to 5 (very fluent). Next, we show $K$ topics and ask which topics are mentioned in the generation. Treating the worker's choices as prediction and the topics our model conditions on as ground truth, we report the average precision and recall of the prediction.

#### 4.3.3.3 Conditional Text Generator Baselines

We compare our method with **PPLM** (Plug and Play Language Models) [44] due to its strong performance against the weighted decoding approach from Ghazvininejad et al. [69] when the condition is a bag of words.

The condition for **PPLM** is the union of the top $M$ words in the chosen topics and each word's weight is neglected. We use our generation model without conditioning on any word (i.e., $n = 0$) during testing[2] as the base model of **PPLM**. We also present the performance of the base model itself as a reference to know the significance of our improvement (denoted as **GPT-2**).

[2]We find the model performs similarly compared with GPT-2 with no condition during training.

#### 4.3.3.4 Results

Table 4.5 indicates that our model outperforms **PPLM** in all metrics except in Dist-1 and Dist-2. We suspect that our model generates slightly less diverse sentences in order to make the generation more relevant to the given topics.

The generation might mention a topic even if it is not chosen as a condition, so we achieve similar precision compared to **PPLM** in human evaluation. The recall of **PPLM** means that only around 30% of given topics are mentioned. The low recall indicates the difficulty of mentioning multiple randomly selected topics in the next 50 word pieces while keeping the sentence fluent. By contrast, achieving 40% on recall demonstrates the effectiveness of our conditional text generator.

Compared with **PPLM**, our model requires an additional training step but achieves low inference time and high relevancy to the given topics/words once the training is finished. The benefits make it preferable in our interactive writing application.

### 4.3.4 Option Generator Comparison Using Generated Continuations

To see whether the proposed option generator improves the quality of the continuations, we use all of $K$ topics from different methods to guide our conditional text generator and compare their generated continuations. In addition to all the methods we described in Section 4.3.2.3, we also present the results of our text generator without conditioning on any topics (i.e., $n = 0$) as a reference and call the method **None**.

#### 4.3.4.1 Automatic Evaluation Metrics

- **BLEU**: For each generated text guided by the set of $K$ topics, we report BLEU-2 [166] between the generation and the actual continuation containing $O = 25$ words. We adopt the smoothing method 3 in Chen and Cherry [36] because there is sometimes no bigram overlapping between the predicted continuation and the actual continuation.

| Scope | Method | BLEU | BLEU Diff | Word Hit | Self-BLEU (↓) | Dist-1 | Dist-2 |
|---|---|---|---|---|---|---|---|
| Global | Sample | **7.39** | **5.66** | 0.34 | **9.45** | **47.60** | **86.79** |
| | LDA | 7.19 | 4.87 | **2.01** | 13.06 | 36.02 | 78.73 |
| | Kmeans | 7.12 | 4.65 | 1.30 | 12.23 | 36.62 | 81.49 |
| Local | Sample | 8.38 | 2.71 | 2.93 | 18.03 | 35.76 | 77.00 |
| | NNSC | **8.44** | 3.24 | 2.94 | 17.20 | 35.43 | 76.71 |
| | Kmeans | 8.32 | 3.06 | 2.96 | 16.97 | 35.39 | 77.10 |
| | Ours | 8.38 | **5.55** | **3.02** | **15.97** | **36.18** | **78.71** |
| NA | None | 8.50 | 5.59 | - | 13.17 | 39.69 | 80.17 |

Table 4.6: Comparison of the continuations generated by different option generators using automatic metrics. The values are percentages except in Word Hit. Higher numbers are better except in Self-BLEU. The best numbers within each scope are highlighted.

- **BLEU Diff**: Similar to Sim Diff, BLEU Diff is the BLEU score between the generation and the continuation minus the BLEU score between the generation and the input prompt.
- **Word Hit**: If the generated topics are not relevant to the input prompt, our conditional text generator might have difficulty in mentioning the related words in the continuation. We report how many unique word types representing $K$ topics are mentioned in the generated continuation.
- **Self-BLEU**: The metric computes the average pairwise BLEU scores of 3 generations [263]. Lower Self-BLEU implies the options encourage more diverse generations.

#### 4.3.4.2 Human Evaluation

We show the continuation guided by all topics and ask how fluent the sentence is (F), how helpful the sentence can promote the narrative (NP), and the overall quality of the generation (A). The worker can choose from 5 options, and 5 means very fluent, very helpful, and excellent, respectively.

| Scope | Method | F | NP | A |
|---|---|---|---|---|
| Global | LDA | $3.07 \pm 0.17$ | $2.82 \pm 0.16$ | $3.06 \pm 0.13$ |
| | Kmeans | $3.65 \pm 0.13$ | $3.42 \pm 0.14$ | $3.42 \pm 0.12$ |
| Local | Kmeans | $3.71 \pm 0.13$ | $3.56 \pm 0.15$ | $3.39 \pm 0.13$ |
| | Ours | $\mathbf{3.85} \pm 0.14$ | $\mathbf{3.64} \pm 0.15$ | $\mathbf{3.67} \pm 0.14$ |

Table 4.7: Comparison of the continuations generated by different option generators using human judgment (mean $\pm$ standard error). F, NP, and A refer to fluency, narrative promotion, and overall, respectively.

#### 4.3.4.3 Results

The automatic evaluation results are presented in Table 4.6. As expected, the options generated by the local methods lead to the continuations that are more similar to the actual continuation (i.e., higher BLEU score) compared to that generated by the global methods. Global topics encourage the generated text to be unrelated to the input prompt, so leading to more diverse sentences (i.e., lower Self-BLEU and higher Dist-1 and Dist-2).

Our method performs better in most metrics than the other local methods, especially in BLEU Diff, while achieving comparable BLEU, which means our generated options often result in the relevant and diverse continuations that are sufficiently different from the prompt. Furthermore, the human evaluation results in Table 4.7 show that our method outperforms other baselines in all metrics.

## 4.4 Related Work

Different interactive writing assistants provide different forms of options to let users express their preferences. The options could be manually defined classes (e.g., sentiment) [99, 44], semantic frames [222], or event structures such as (subject, verb, object, modifier) [141, 215, 5]. The forms of options allow users to control the attributes of the generated text but require labels or classifiers that map the text to the attributes/options.

The options could also be a single query word at the beginning [10], the article title [245], politeness [161] or specificity [192] of the text, or the length of the generated sentence [222].

However, the options cannot provide fine-grained control on topical directions of the generated contents.

A related research direction is the multi-stage story generation. To make a long story more coherent, recent work proposes to generate a skeleton and then generate the full text guided by the skeleton. The skeleton could be a sequence of SRL frames [57], a sequence of event structure (subject, verb, object, preposition, modifier) [5], a story premise [56], or a story summary [38]. Users can revise the skeleton to control the generated text, but the approaches assume the existence of the skeleton extractor or labels in the training corpus. Besides, the systems cannot suggest options given the partial text, which is one of the main focuses of our interactive writing assistant.

The skeleton could also be multiple keyphrases. The keyphrases are extracted based on word frequency [90, 242, 216], an off-the-shelf keyword extraction method [173, 70, 251, 180, 256], a sentence compression dataset and reinforcement learning [243], or image caption datasets and ConceptNet [122]. Most of the studies focus on modeling the long-term dependency among the keyphrases and/or forcing the generation to contain the keyphrases. Instead, we focus on allowing users to determine the topical directions of the generation. Compared with conditioning on keyphrases, our interactive writing assistant is especially helpful when users do not know the exact phrases they want to see or when the given keyphrase extractor does not detect the desired topics.

## 4.5 Chapter Conclusion

In this chapter, we propose an interactive writing assistant that generates topic options given an input prompt and generates the continuation of the prompt given the topics chosen by a user. We decompose the framework into two components and propose a novel model for each component. The automated evaluation and human evaluation indicate that our system generates many topics that are related to but different from the prompt, and generates the sentences that are fluent and relevant to the chosen topics.

# CHAPTER 5

# CONTRASTIVE LEARNING ON TWO-TOWER MODELS

## 5.1 Introduction

The improvements of multiple embeddings in Chapter 3 suggest that the text sequence co-occurred with various words in the nearby sentences, entity pairs, or citing papers. They also suggest multiple embeddings can represent the multimodal co-occurred distribution well. Nevertheless, we only studied the models that do not use a pre-trained language model (LM) for the one-tower co-occurrence learning, but the state-of-the-art models in the sentence representation and citation prediction all use two-tower contrastive learning and pre-trained LM. In this chapter, we want to study how to use multiple embeddings to improve the state-of-the-art models based on the BERT encoder in these applications.

In Section 5.2, we propose Multi-CLS BERT, a novel ensembling method for CLS-based prediction tasks that is almost as efficient as a single BERT model. Multi-CLS BERT uses multiple CLS tokens with a parameterization and objective that encourages their diversity. Thus instead of fine-tuning each BERT model in an ensemble (and running them all at test time), we need only fine-tune our single Multi-CLS BERT model (and run the one model at test time, ensembling just the multiple final CLS embeddings).

To test its effectiveness, we build Multi-CLS BERT on top of a state-of-the-art pretraining method for BERT [6]. In experiments on GLUE and SuperGLUE we show that our Multi-CLS BERT reliably improves both overall accuracy and confidence estimation. When only 100 training samples are available in GLUE, the Multi-CLS $\text{BERT}_{\text{Base}}$ model can even outperform the corresponding $\text{BERT}_{\text{Large}}$ model. We analyze the behavior of our Multi-CLS

BERT, showing that it has many of the same characteristics and behavior as a typical BERT 5-way ensemble, but with nearly 4-times less computation and memory.

In Section 5.3, we find that a scientific paper encoder with multiple CLS tokens is able to better specialize to multiple domains. We present Multi$^2$SPE, a simplified variant of the proposed Multi-CLS BERT, which encourages each of multiple CLS tokens to learn diverse ways of aggregating token embeddings, then summing them together to create a single vector representation. We further propose a new multi-domain benchmark, Multi-SciDocs, to test vector encoders for scientific papers. We show that our encoder reduces the error by up to 25% in multi-domain citation prediction, while adding only negligible computation to the forward pass of a classic BERT encoder.

## 5.2 Applications on Natural Language Understanding Benchmarks

BERT (Bidirectional Encoder Representations from Transformers) [47] is one of the most widely-used language model (LM) architectures for natural language understanding (NLU) tasks. We often fine-tune the pretrained BERT or its variants such as RoBERTa [131] so that the LMs learn to aggregate all the contextualized word embeddings into a single CLS embedding for a downstream text classification task.

During fine-tuning, different initializations and different training data orders significantly affect BERT's generalization performance, especially with a small training dataset [50, 255, 155]. One simple and popular solution to the issue is to fine-tune BERT model multiple times using different random seeds and ensemble their predictions to improve its accuracy and confidence estimation. Although very effective, the memory and computational cost of ensembling a large LM is often prohibitive [244, 120]. Naturally, we would like to ask, "Is it possible to ensemble BERT models at no extra cost?"

To answer the question, we propose Multi-CLS BERT, which enjoys the benefits of ensembling without sacrificing efficiency. Specifically, we input the multiple CLS tokens to BERT and encourage the different CLS embeddings to aggregate the information from

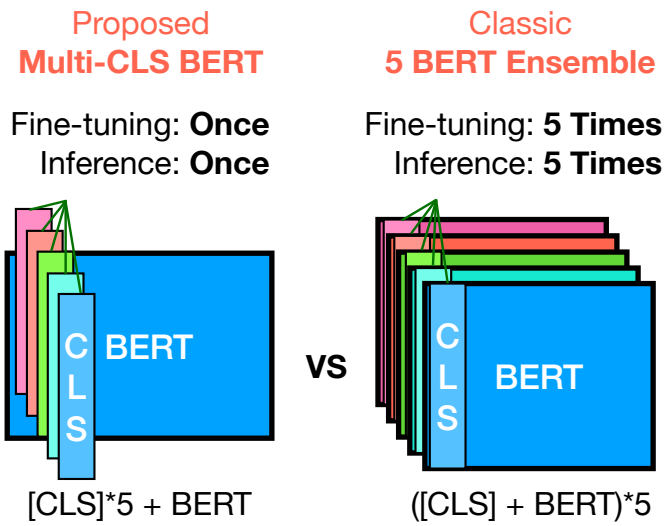



Figure 5.1: Comparison of Multi-CLS BERT and the classic BERT ensemble. Multi-CLS BERT only ensembles the multiple CLS embeddings in one BERT encoder rather than ensemble multiple BERT encoders with different parameter weights.

different aspects of the input text. As shown in Figure 5.1, the proposed Multi-CLS BERT shares all the hidden states of the input text and only ensembles different ways of aggregating the hidden states. Since the input text is usually much longer than the number of inputted CLS embeddings, Multi-CLS BERT is almost as efficient as the original BERT.

Allen-Zhu and Li [3] discovered that the key of an effective ensembling model is the diversity of individual models and the models trained using different random seeds have more diverse predictions compared to simply using dropout [205, 64] or averaging the weights of the models during training [63]. To ensure the diversity of CLS embeddings without fine-tuning Multi-CLS BERT using multiple seeds, we propose several novel diversification techniques. For example, we insert different linear layers into the transformer encoder for different CLS tokens. Furthermore, we propose a novel re-parametrization trick to prevent the linear layers from learning the same weights during fine-tuning.

We test the effectiveness of these techniques by modifying the multi-task pretraining method proposed by Aroca-Ouellette and Rudzicz [6], which combines four self-supervised losses. In our experiments, we demonstrate that the resulting Multi-CLS BERT can signifi-

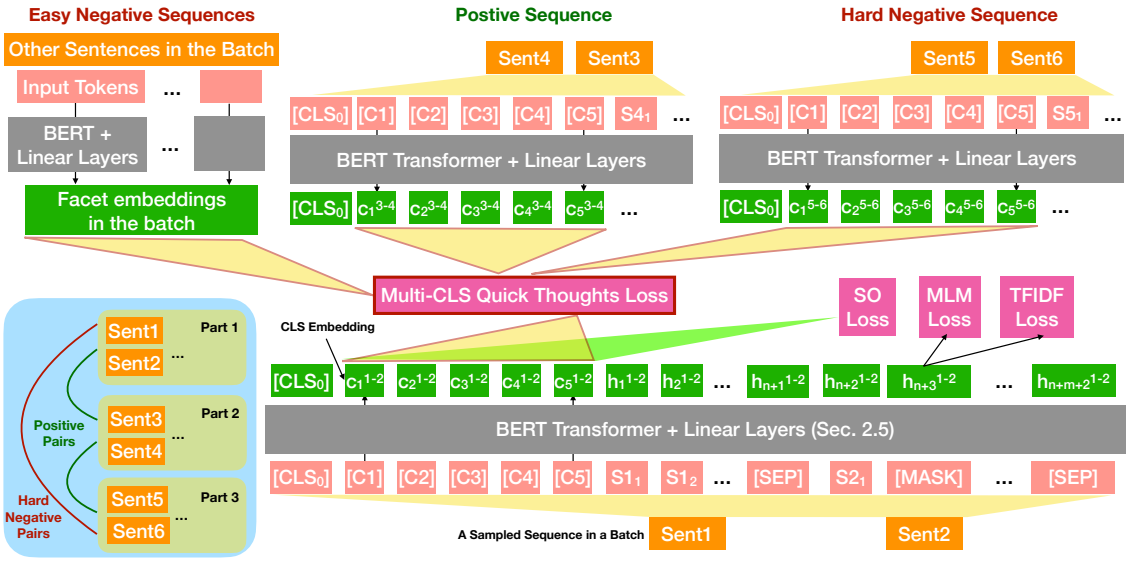



Figure 5.2: Our MCQT, SO, MLM, and TFIDF loss, which are a modification of multi-task pretraining proposed in Aroca-Ouellette and Rudzicz [6]. The multi-CLS quick thought (MCQT) loss maximizes the CLS similarities between a sequence (sentences 1 and 2) and the next sequence (sentences 3 and 4) while minimizing the CLS similarities to other random sequences and the sequence after the next one (sentences 5 and 6). Notice that sentence 4 is inputted before sentence 3 because the sentence order is swapped for the SO loss.

cantly improve the accuracy on GLUE [229] and SuperGLUE [228], especially when the training sizes are small. Similar to the BERT ensemble model, we further show that multiple CLS embeddings significantly reduce the expected calibration error, which measures the quality of prediction confidence, on the GLUE benchmark.

### 5.2.1 Method

In sections 5.2.1.1 and 5.2.1.2, we first review its state-of-the-art pretraining method from Aroca-Ouellette and Rudzicz [6]. In Section 5.2.1.3, we modify one of its losses, quick thoughts (QT), to pretrain our multiple embedding representation. In Section 5.2.1.4, we encourage the CLS embeddings to capture the fine-grained semantic meaning of the input sequence by adding hard negatives during the pretraining. To diversify the CLS embeddings, we modify the transformer encoder in Section 5.2.1.5 and propose a new reparametrization method during the fine-tuning in Section 5.2.1.6.

#### 5.2.1.1 Multi-task Pretraining

After testing many self-supervised losses, Aroca-Ouellette and Rudzicz [6] find that combining the masked language modeling (MLM) loss, TFIDF loss, sentence ordering (SO) loss [210], and quick thoughts (QT) loss [133] could lead to the best performance.

The MLM loss is to predict the masked words and the TFIDF loss is to predict the importance of the words in the document. Each input text sequence consists of multiple sentences. They swap the sentence orders in some input sentences and use the CLS embedding to predict whether the order is swapped in the SO loss. Finally, QT loss is used to encourage the CLS embeddings of the consecutive sequences to be similar.

To improve the state-of-the-art pretraining method, we modify the multi-task pretraining method by using multiple CLS embeddings to represent the input sequence and using non-immediate consecutive sentences as the hard negative. Our training method is illustrated in Figure 5.2.

#### 5.2.1.2 Quick Thoughts Loss

Two similar sentences tend to have the same label in a downstream application, so pretraining should aim to pull the CLS embeddings of these similar sentences closer. The QT loss achieves this goal by assuming consecutive text sequences are similar and encouraging their CLS embeddings to be similar.

Aroca-Ouellette and Rudzicz [6] propose an efficient way of computing QT loss in a batch by evenly splitting each batch with size $B$ into two parts. The first part contains $B/2$ text sequences randomly sampled from the pretrained corpus, and the second part contains each of the $B/2$ sentences that are immediately subsequent to those in the first part. Then, for each sequence in the first part, they use the consecutive sequence in the second part as the positive example and the other $B/2 - 1$ sequences as the negative examples. We can write the QT loss for the sequences containing sentences 1, 2, 3, and 4 as

$$L_{QT}(s^{1-2}, s^{3-4}) = -\log(\frac{\exp(\text{Logit}^{QT}_{s^{1-2},s^{3-4}})}{\sum_s \exp(\text{Logit}^{QT}_{s^{1-2},s})}), \tag{5.1}$$

where $s$ is the sentences in the second part of the batch, $\text{Logit}^{QT}_{s^{1-2},s^{3-4}} = (\frac{\boldsymbol{c}^{1-2}}{||\boldsymbol{c}^{1-2}||})^T \frac{\boldsymbol{c}^{3-4}}{||\boldsymbol{c}^{3-4}||}$ is the score for classifying sequence $s^{3-4}$ as the positive example, $\frac{\boldsymbol{c}^{1-2}}{||\boldsymbol{c}^{1-2}||}$ is the L2-normalized CLS embedding for sentences 1 and 2. The normalization is intended to stabilize the pretraining by limiting the gradients' magnitudes.

#### 5.2.1.3 Multiple CLS Embeddings

A text sequence could have multiple facets; two sequences could be similar in some facets but dissimilar in others, especially when the text sequences are long. The QT loss squeezes all facets of a sequence into a single embedding and encourages all facets of two consecutive sequences to be similar, potentially causing information loss.

Some facets might better align with the goal of a downstream application. For example, the facets that contain more sentiment information would be more useful for sentiment analysis. To preserve the diverse facet information during pretraining, we propose multi-CLS quick thoughts loss (MCQT). The loss integrates two ways of computing the similarity of two sequences. The first way computes the cosine similarity between the most similar facets, and the second computes the cosine similarity between the summations of all facets. We linearly combine the two methods as the logit of the two input sequences:

$$\text{Logit}^{MC}_{s^{1-2},s^{3-4}} = \lambda \max_{i,j}(\frac{\boldsymbol{c}^{1-2}_i}{||\boldsymbol{c}^{1-2}_i||})^T \frac{\boldsymbol{c}^{3-4}_j}{||\boldsymbol{c}^{3-4}_j||} + (1-\lambda)(\frac{\sum_i \boldsymbol{c}^{1-2}_i}{||\sum_i \boldsymbol{c}^{1-2}_i||})^T \frac{\sum_j \boldsymbol{c}^{3-4}_j}{||\sum_j \boldsymbol{c}^{3-4}_j||}. \tag{5.2}$$

where $\lambda$ is a constant hyperparameters; $\boldsymbol{c}^{1-2}_k$ and $\boldsymbol{c}^{3-4}_k$ are the CLS embeddings of sentences 1-2 and sentences 3-4, respectively.

The first term only considers the most similar facets to allow some facets to be dissimilar. Furthermore, the term implicitly diversifies CLS embeddings by considering each CLS embedding independently. In contrast, the second term encourages the CLS embeddings to

work collaboratively, as in a typical ensemble model, and also let every CLS embedding receive gradients more evenly. Notice that we sum the CLS embeddings before the normalization so that the encoder could predict the magnitude of each CLS embedding as its weight in the summation.

To show that the proposed method can improve the state-of-the-art pretraining methods, we keep the MLM loss and TFIDF loss unchanged. For the sentence ordering (SO) loss, we project the $K$ hidden states $\boldsymbol{h}_k^c$ into the embedding $\boldsymbol{h}^{SO}$ with the hidden state size $D$ for predicting the sentence order: $\boldsymbol{h}^{SO} = L^{SO}(\oplus_k \boldsymbol{h}_k^c)$, where $\oplus_k \boldsymbol{h}_k^c$ is the concatenation of $K$ hidden states with size $K \times D$.

#### 5.2.1.4 Hard Negative

For a large transformer-based LM, distinguishing the next sequence from random sequences could be easy. The LM can achieve low QT loss by outputting nearly identical CLS embeddings for the sentences with the same topic while ignoring the fine-grained semantic information [167]. In this case, using multiple CLS embeddings might become underutilized.

The hard negative is a common method of adjusting the difficulties of the contrastive learning [12, 42]. Our way of collecting hard examples is illustrated in the bottom-left block of Figure 5.2. To efficiently add the hard negatives in the pretraining, we split the batch into three parts. For each sequence in the first part, we would use its immediate next sequence in the second part as the positive example, use the sequence after the next one in the third part as the hard negative, and use all the other sequences in the second or the third part as the easy negatives. We select such sequence after the next one as our hard negatives because the sequence usually share the same topic with the input sequence but is more likely to have different fine-grained semantic facets compared to the immediate next sequence.

After adding the hard negative, the modified QT loss of the three consecutive sequences becomes

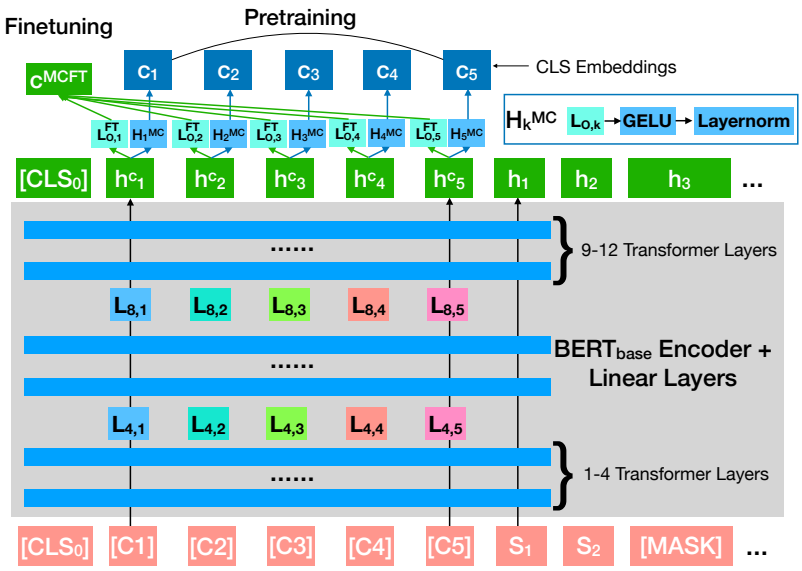



Figure 5.3: The architecture of Multi-CLS BERT encoder that is built on $\text{BERT}_{\text{Base}}$ model. The different linear layers are applied to the hidden states corresponding to different CLS tokens to increase the diversity of the resulting CLS embeddings.

$$L_{MCQT}(s^{1-2}, s^{3-4}, s^{5-6}) = \\ -\log\left(\frac{\exp(\text{Logit}^{MC}_{s^{1-2},s^{3-4}})}{\sum_{s\in\{s^{3-4},\ldots,s^{5-6},\ldots\}}\exp(\text{Logit}^{MC}_{s^{1-2},s})}\right) - \log\left(\frac{\exp(\text{Logit}^{MC}_{s^{5-6},s^{3-4}})}{\sum_{s\in\{s^{3-4},\ldots,s^{1-2},\ldots\}}\exp(\text{Logit}^{MC}_{s^{5-6},s})}\right), \tag{5.3}$$

where MCQT refer to multi-CLS quick thoughts, $\{s^{3-4}, ..., s^{5-6}, ...\}$ are all the sequences in the second and the third part, and $\{s^{3-4}, ..., s^{1-2}, ...\}$ are all the sequences in the first and the second part.

#### 5.2.1.5 Architecture-based Diversification

Initially, we simply input multiple special CLS tokens ([C1], ..., [CK]) after the original CLS token, $[\text{CLS}_0]$, and take the corresponding hidden states as the CLS embeddings, but we found that the CLS embeddings quickly become almost identical during the pretraining.

Subsequently, instead of using the same final transformation head $H^{QT}$ for all CLS hidden states, we use a different linear layer $L_{O,k}$ in the final head $H^{MC}_k$ to transform the

hidden state $\boldsymbol{h}_k^c$ for the $k$th CLS. We set the bias term in $L_{O,k}$ to be the constant $\mathbf{0}$ because we want the differences between the CLS to be dynamic and context-dependent.

Nevertheless, even though the resulting CLS embeddings $\boldsymbol{c}_k = H_k^{MC}(\boldsymbol{h}_k^c)$ are differentiated, the hidden states $\boldsymbol{h}_k^c$ before the transformation head usually still collapse into almost identical embeddings.

To solve the collapsing problem, we insert multiple linear layers $L_{l,k}$ into the transformer encoder. In Figure 5.3, we illustrate our encoder architecture built on the $\text{BERT}_{\text{Base}}$ model. After the $4$th transformer layer, we insert the layers $L_{4,k}$ to transform the hidden states before inputting them to the $5$th layer. Similarly, we insert $L_{8,k}$ between the $8$th transformer layer and $9$th transformer layer. For $\text{BERT}_{\text{Large}}$, we insert $L_{l,k}(.)$ after layer 8 and layer 16. Notice that although the architecture looks similar to the adapter [88] or prefix-tuning [118], our purpose is to diversify the CLS embeddings rather than freezing parameters to save computational time.

#### 5.2.1.6 Fine-Tuning

To avoid overfitting and increasing computational overhead, we pool multiple CLS hidden states into the single CLS embedding for downstream task fine-tuning. As a result, we can use the same classifier architecture on top of Multi-CLS BERT and BERT, which also simplifies their comparison.

We discover that simply summing all the CLS hidden states still usually makes the hidden states and the inserted linear layers (e.g., $L_{O,k}$) almost identical after fine-tuning. To avoid collapsing, we aggregate the CLS hidden states by proposing a novel re-parameterization trick:

$$\boldsymbol{c}^{MCFT} = \sum_k \left(L_{O,k}^{FT}(\boldsymbol{h}_k^c)\right), \tag{5.4}$$

where $L_{O,k}^{FT}(\boldsymbol{h}_k^c) = (\mathbf{W}_{O,k} - \frac{1}{K}\sum_{k'} \mathbf{W}_{O,k'})\boldsymbol{h}_k^c$, and $\mathbf{W}_{O,k}$ is the linear weights of $L_{O,k}$. Then, if all the $L_{O,k}^{FT}$ become identical (i.e., $\forall k, \mathbf{W}_{O,k} = \frac{1}{K}\sum_{k'} \mathbf{W}_{O,k'}$), $L_{O,k}^{FT}(\boldsymbol{h}_k^c) = \mathbf{0} = \boldsymbol{c}^{MCFT}$.

However, gradient descent would not allow the model to constantly output the zero vector, so $L_{O,k}^{FT}$ remains different during the fine-tuning.

### 5.2.2 Experiments

The parameters of neural networks are more restricted as more training samples are available [139] and the improvement of deep ensemble models comes from the diversity of individual models [63], so the benefits of ensembling are usually more obvious when the training set size is smaller. Therefore, in addition to using the full training dataset, we also test the settings where the models are trained by 1k samples [255] or 100 samples from each task in GLUE [229] or SuperGLUE [228]. Another benefit of the 1k- and 100-sampling settings is that the average scores would be significantly influenced by most datasets rather than by only a subset of relatively small datasets [24].

#### 5.2.2.1 Experiment Setup

To accelerate the pretraining experiments, we initialize the weights using the pretrained BERT models [47] and continue the pretraining using different loss functions on Wikipedia 2021 and BookCorpus [262].

All of the methods are based on uncased BERT as in Aroca-Ouellette and Rudzicz [6]. We compare the following methods:

- **Pretrained**: The pretrained BERT model released from Devlin et al. [47].
- **MTL**: Pretraining using the four losses selected in Aroca-Ouellette and Rudzicz [6]: MLM, QT, SO, and TFIDF. We remove the continue learning procedure used in ERNIE [210] because we find that simply summing all the losses leads to better performance (see our ablation study in Section 5.2.2.3).
- **Ours (K=5, $\lambda$)**: The proposed Multi-CLS BERT method using 5 CLS tokens. We show the results of setting $\lambda = \{0, 0.1, 0.5, 1\}$ in Equation 5.7. We reduce the maximal sentence length by 5 to accommodate the extra 5 CLS tokens.

| Configuration ↓ | Model Name ↓ | Model Size ↓ | GLUE | | | SuperGLUE | | |
|---|---|---|---|---|---|---|---|---|
| | | | 100 | 1k | Full | 100* | 1k* | Full |
| BERT Base | Pretrained | 109.5M | 55.71 ± 0.62 | 71.67 ± 0.15 | 82.05 ± 0.08 | 57.18 ± 0.43 | 61.55 ± 0.37 | 65.04 ± 0.36 |
| | MTL | 109.5M | 59.29 ± 0.27 | 73.26 ± 0.13 | 83.30† ± 0.07 | 57.50 ± 0.41 | 62.94 ± 0.36 | 66.33 ± 0.33 |
| | Ours (K=1) | 111.3M | 57.84 ± 0.32 | 73.28 ± 0.13 | 83.40 ± 0.07 | 57.31 ± 0.35 | 63.35 ± 0.18 | 66.29 ± 0.18 |
| | Ours (K=5, $\lambda = 0$) | 118.4M | 61.54 ± 0.32 | **74.14** ± 0.12 | 83.41 ± 0.07 | 58.29 ± 0.33 | 63.71 ± 0.26 | **66.80** ± 0.25 |
| | Ours (K=5, $\lambda = 0.1$) | 118.4M | **61.80** ± 0.35 | 74.10 ± 0.13 | **83.47** ± 0.05 | 58.20 ± 0.31 | 63.61 ± 0.27 | 66.74 ± 0.26 |
| | Ours (K=5, $\lambda = 0.5$) | 118.4M | 60.49 ± 0.35 | 74.02 ± 0.12 | **83.47** ± 0.08 | **58.41** ± 0.38 | **63.78** ± 0.25 | **66.80** ± 0.24 |
| | Ours (K=5, $\lambda = 1$) | 118.4M | 59.86 ± 0.34 | 73.75 ± 0.14 | 83.43 ± 0.07 | 57.84 ± 0.40 | 63.56 ± 0.22 | 66.39 ± 0.22 |
| BERT Large | MTL | 335.2M | 61.39 ± 0.37 | 75.30 ± 0.27 | 84.13 ± 0.11 | 59.03 ± 0.54 | 65.21 ± 0.38 | 69.16 ± 0.37 |
| | Ours (K=1) | 338.3M | 59.19 ± 0.43 | 75.35 ± 0.21 | 84.59 ± 0.07 | 57.35 ± 0.42 | 64.67 ± 0.43 | 69.24 ± 0.41 |
| | Ours (K=5, $\lambda = 0$) | 350.9M | 63.19 ± 0.49 | 75.73 ± 0.26 | 84.51 ± 0.05 | 59.46 ± 0.44 | 65.43 ± 0.38 | 69.56 ± 0.31 |
| | Ours (K=5, $\lambda = 0.1$) | 350.9M | **64.24** ± 0.40 | **76.27** ± 0.12 | **84.61** ± 0.08 | **59.88** ± 0.43 | 65.58 ± 0.26 | **70.03** ± 0.25 |
| | Ours (K=5, $\lambda = 0.5$) | 350.9M | 63.02 ± 0.42 | 75.95 ± 0.10 | 84.49 ± 0.08 | 59.42 ± 0.34 | **65.84** ± 0.25 | 69.79 ± 0.25 |
| | Ours (K=5, $\lambda = 1$) | 350.9M | 62.07 ± 0.45 | 75.85 ± 0.17 | **84.61** ± 0.07 | 58.74 ± 0.50 | 65.00 ± 0.29 | 69.04 ± 0.27 |

Table 5.1: The macro average scores on the development set. All numbers are percentages. The standard errors are shown as the confidence intervals. We make the best scores of the model built on $\text{BERT}_{\text{Base}}$ boldface and similar for the models built on $\text{BERT}_{\text{Large}}$. †The number is much higher than $81.4$, the GLUE score reported by Aroca-Ouellette and Rudzicz [6] because we continue training from the pretrained BERT and we use better fine-tuning hyperparameters. *The scores do not contain ReCoRD in SuperGLUE.[1]

- **Ours (K=1)**: We set $K = 1$ in our method to verify the effectiveness of using multiple embeddings. During fine-tuning, the CLS embedding is a linear transformation of the single facet CLS $= L_{O,1}(\boldsymbol{h}_1^f)$.

The GLUE and SuperGLUE scores are significantly influenced by the pretraining random seeds [193] and fine-tuning random seeds [50, 255, 155]. To stably evaluate the performance of different pretraining methods, we pretrain models using four random seeds and fine-tune each pretrained model using four random seeds, and report the average performance on the development set across all 16 random seeds. To further stabilize the fine-tuning process and reach better performance, we follow the fine-tuning suggestions from Zhang et al. [255] and Mosbach et al. [155], including training longer, limiting the gradient norm, and using Adam [101] with bias term and warmup.

#### 5.2.2.2 Main Results

Our results are presented in Table 5.1. We can see that **Ours (K=5)** is consistently better than other baselines and that the improvement is larger in datasets with fewer training samples. For example, in GLUE 100, it achieves $61.80$ on average using $\text{BERT}_{\text{Base}}$ with 118.4M parameters, which outperforms **MTL** using $\text{BERT}_{\text{Large}}$ with 335.2M parameters ($61.39$). **MTL** significantly improves the scores of original BERT model (**Pretrained**), confirming the effectness of the QT, SO, and TFIDF losses. Compared to **MTL**, **Ours (K=1)** is slightly better in GLUE 1k and GLUE Full, but worse in GLUE 100.

We observe that $\lambda = 0.1$ usually performs well, which justifies the inclusion of both the highest logit and average logit in Equation 5.7. The $\lambda = 0$ model has significantly worse performance only in $\text{BERT}_{\text{Large}}$ model. This suggests that the benefits of Multi-CLS BERT depend on our pretraining method and maximizing the highest logit stabilizes the pretraining of a larger model.

#### 5.2.2.3 Ablation Study

In our ablation studies, we would like to test the effectiveness of the design choices in our baseline **MTL** and our best model, **Ours (K=5, $\lambda = 0.1$)**. The model variants we test include:

- **MLM only**: Removing the QT, SO, and TFIDF losses in **MTL**. That is, we simply continue training **Pretrained** using only the MLM loss.
- **CMTL+**: The best pretrained method reported in Aroca-Ouellette and Rudzicz [6]. It uses the continual learning method [210] to weight each loss in **MTL**.
- **MLM+SO+TFIDF**: **MTL** without the QT loss.
- **No Inserted Layers**: Removing the $L_{l,k}(.)$ in the transformer encoder from our method.

[1] In SuperGLUE 100 and 1k, we exclude the ReCoRD dataset because the performance of all models is much worse than the most frequent class baseline.

- **No Hard Negative**: Removing the hard negatives described in Section 5.2.1.4 from our method.
- **Sum Aggregation**: Simply summing the facets (i.e., using $L_{O,k}$ to replace $L^{FT}_{O,k}$ in Equation 5.5).
- **Default**: **Ours (K=k, $\lambda = 0.1$)**, where $k = \{1, 3, 5, 10\}$.
- **SWA**: Stochastic weight averaging [188, 92] averages the weights along the optimization trajectory.
- **Ensemble on Dropouts**: Running the forward pass of **Ours (K=1)** with dropout using 5 different seeds and averaging their prediction probabilities for each class in each task.
- **Ensemble on FT Seeds**: Fine-tuning **Ours (K=1)** or **Ours (K=5, $\lambda = 0.1$)** using 5 different seeds and averaging their prediction probabilities.

Our results are presented in Table 5.2. We can see that continuing training using **MLM only** loss degrades the performance, which indicates that our improvement does not come from training BERT longer. Removing QT loss results in mixed results. The better performance of **MTL** compared to **CMTL+** suggests that the continual training technique used in Aroca-Ouellette and Rudzicz [6] is harmful with our evaluation settings.

Removing the inserted layers (**No Inserted Layers**) or removing the re-parametrization trick (**Sum Aggregation**) makes the performance of **Ours (K=5, $\lambda = 0.1$)** close to the **Ours (K=1)** baseline. This result highlight the importance of diversity of CLS embeddings. The performance of **Ours (K=3)** and **Ours (K=10)** is usually better than **Ours (K=1)**, but are worse than **Ours (K=5)**. In both $\text{BERT}_{\text{Base}}$ and $\text{BERT}_{\text{Large}}$ models, removing hard negatives degrades the GLUE scores but slightly increases the SuperGLUE scores.

In GLUE 100 and 1k, we do not get good results by using other efficient ensembling methods such as **SWA** and **Ensemble on Dropouts**. This suggests that the gradient descent trajectory and different dropout maps might not produce prediction diversity sufficient for an effective BERT ensemble model [63].

| Model ↓ | Model Description ↓ | K ↓ | GLUE 100 | GLUE 1k | SuperGLUE* 100 | SuperGLUE* 1k |
|---|---|---|---|---|---|---|
| Baselines (BERT Base) | Pretrained | 1 | 56.85 | 71.68 | 57.90 | 62.14 |
| | MLM only | 1 | 55.38 | 70.74 | 57.39 | 61.77 |
| | CMTL+ | 1 | 58.65 | 72.57 | 56.88 | 62.63 |
| | MLM + SO + TFIDF | 1 | 60.35 | 72.65 | 57.88 | 62.60 |
| | MTL | 1 | 59.53 | 73.12 | 57.51 | 62.95 |
| Ours (BERT Base) | No Inserted Layers | 1 | 58.06 | 73.18 | 57.97 | 63.34 |
| | | 5 | 60.12 | 73.35 | 56.46 | 62.00 |
| | No Hard Negative | 1 | 58.44 | 73.30 | 57.19 | 63.33 |
| | | 5 | 61.77 | 74.18 | **58.89** | **63.86** |
| | Sum Aggregation | 5 | 58.87 | 73.94 | 57.41 | 63.82 |
| | Default | 1 | 57.76 | 73.30 | 57.53 | 63.22 |
| | | 3 | 61.09 | 73.95 | 57.85 | 63.31 |
| | | 5 | **62.62** | **74.49** | 58.82 | **63.86** |
| | | 10 | 60.99 | 73.59 | 58.25 | 62.82 |
| | SWA | 1 | 57.31 | 72.91 | - | - |
| | Ensemble on Dropouts | 1 | 58.45 | 72.86 | - | - |
| | Ensemble on FT Seeds | 1 | 60.07 | 75.20 | - | - |
| | | 5 | 63.34 | 75.35 | - | - |
| Ours (BERT Large) | No Hard Negative | 1 | 60.36 | 75.69 | 58.47 | 65.04 |
| | | 5 | 63.23 | 75.77 | **60.33** | **65.75** |
| | Default | 1 | 60.01 | 76.03 | 57.38 | 65.10 |
| | | 5 | **64.33** | **76.38** | 59.99 | 65.51 |

Table 5.2: The macro average scores on the development set for our ablation study. We highlight the best performance after excluding the ensemble baselines, which require much more computation. The scores are different in Table 5.1 because we use two pretraining random seeds instead of four in the ablation study. SWA refers to Stochastic weight averaging [92]. *SuperGLUE score does not contain ReCoRD.

On the other hand, ensembling the models that are fine-tuned using different random seeds indeed boosts the performance at the expense of high computational costs. The ensembled Multi-CLS BERT (**Ensemble on FT Seeds K=5**) still outperforms the ensembled **K=1** baseline, but ensembling makes their performance differences smaller. These results imply that the improvements of Multi-CLS BERT overlap with the improvements of a BERT ensemble model.

#### 5.2.2.4 Ensembling Analysis

We compare the inference time and expected calibration error (ECE) [157] of using multiple CLS embeddings, using a single CLS embedding, and ensembling BERT models with different fine-tuning seeds in Table 5.3. A lower ECE means a better class probability

| | Inference Time (s) | GLUE* (ECE) 100 | 1k |
|---|---|---|---|
| Ours (K=1) | 0.2918 ± 0.0002 | 25.22 ± 1.99 | 19.32 ± 1.64 |
| Ours (K=5, $\lambda = 0.1$) | 0.3119 ± 0.0004 | **15.46** ± 1.79 | **17.01** ± 1.64 |
| Ensemble of Ours (K=1) | 1.4590 ± 0.0012 | 13.85 ± 0.97 | 10.80 ± 0.88 |

Table 5.3: The comparison of inference time and expected calibration error (ECE). The confidence intervals are standard errors. *Only includes the classification tasks (i.e., excludes STS-b).

| | GLUE* 100 | GLUE* 1k |
|---|---|---|
| Multi-CLS vs ENS | 32.57 | 41.35 |
| Dropout vs ENS | 37.17 | 45.53 |
| Least vs ENS | 39.57 | 48.85 |
| ENS vs ENS | 38.67 | 50.14 |

Table 5.4: The overlapping ratio of the top 20% most uncertain examples using different uncertainty estimation methods. ENS is ensemble of Ours (K=5, $\lambda = 0.1$) with different fine-tuning seeds. *Only includes the classification tasks (i.e., excludes STS-b).

estimation. For example, if a model outputs class 1 with 0.9 probability for 100 samples, ECE = 0 means that 90 samples among them are indeed class 1.

Table 5.3 shows that **Ours (K=5)** is much faster than the BERT ensemble and almost as efficient as **Ours (K=1)**, because a BERT ensemble needs to run for multiple forward passes and we reduce the maximal sentence length by 5 in **Ours (K=5)**. Additionally, the ECE of **Ours (K=5)** is lower than **Ours (K=1)** but not as low as the ECE from ensembling BERT models with different fine-tuning seeds. That is, without significantly increasing inference time, ensembling multiple CLS embeddings improves the output confidence, even though not as much as ensembling BERT models.

Next, we analyze the correlation of uncertainty estimation from different methods in Table 5.4. When ensembling BERT models with different dropout maps (**Dropout**) or different fine-tuning seeds (**ENS**), we can estimate the prediction uncertainty by the variance of the prediction probability from each individual BERT model. We can also use one minus prediction probability as the uncertainty (**Least**). In **Multi-CLS**, we measure the

disagreement among the CLS embeddings as the uncertainty[2] and would like to see how many top-20% most uncertain samples from the disagreement of CLS embeddings are also the top-20% most uncertain samples for a BERT ensemble model.

Table 5.4 reports the ratio of the number of the overlapping uncertain samples from two estimation methods to the number of 20% samples in the development set. We can see that the ratio from Multi-CLS BERT and the BERT ensemble model (**Multi-CLS vs ENS**) is close to the ratios from other uncertainty estimations and the BERT ensemble model (**Dropout vs ENS**, **Least vs ENS**, and **ENS vs ENS**). This shows that different CLS embeddings can classify the uncertain samples differently, as is the case for the different BERT models in a BERT ensemble model.

In short, we find that a) ensembling the original BERT leads to greater improvement than ensembling the Multi-CLS BERT and b) the disagreement of different CLS embeddings highly correlates with the disagreement of the BERT models from different fine-tuning seeds. Both findings support our perspective that Multi-CLS BERT is an efficient ensembling method.

### 5.2.3 Related Work

Due to its effectiveness, ensembling BERT in a better or more efficient way has recently attracted researchers' attention. Nevertheless, the existing approaches often need to rely on distillation [244, 142, 264] or still require significant extra computational cost during training and testing [104, 120].

Some recent vision models can also achieve ensembling almost without extra computational cost by sharing the weights [238], partitioning the model into subnetworks [82, 257], or partitioning the embeddings [112]. However, it is unknown if the approaches are applicable to the pretraining and fine-tuning of language models.

---

[2]See Appendix D.1.3 for details

Similar to Multi-CLS BERT, mixture of softmax (MoS) [248] also uses multiple embeddings to improve the pretraining loss. Recently, Narang et al. [158], Tay et al. [218] have found that MoS is one of the few modifications that can improve on the original BERT architecture on the NLU benchmarks. Nevertheless, Narang et al. [158] also point out that MoS requires significant extra training cost to compute the multiplication between each hidden state and all the word embeddings in the vocabulary.

Another approach represents a document using sentence embeddings [91, 156] or contextualized word embeddings [100, 134] for information retrieval applications. However, the goal of this approach is to improve the representation of a relatively long document and it is unknown if its benefits could be extended to the GLUE tasks that require fine-tuning and often involve only one or two sentences.

## 5.3 Applications on Scientific Paper Representation Benchmarks

With an ever-increasing amount of research publications, it has become virtually essential to develop NLP methods that would allow researchers to efficiently process the wealth of scientific knowledge. Leveraging pretrained language models and citation graphs, SPECTER [42] brings sizeable improvement over the previously state-of-the-art paper encoders and similarity estimation models such as SciBERT [16] and Citeomatic [17]. Recently, SciNCL [163] has introduced more sophisticated positive and negative sampling strategies to improve SPECTER further.

Despite all the progress made so far, what is yet missing from literature is examining whether existing encoders can effectively represent the scientific papers across diverse subject areas.

In previous work [17, 16, 42], the training and evaluation data primarily consist of scientific papers from specific subject areas such as computer science and medicine. While these choices might be due to non-technical reasons such as the lack of open access articles [175] or insufficient number of users from certain domains, it naturally makes us wonder

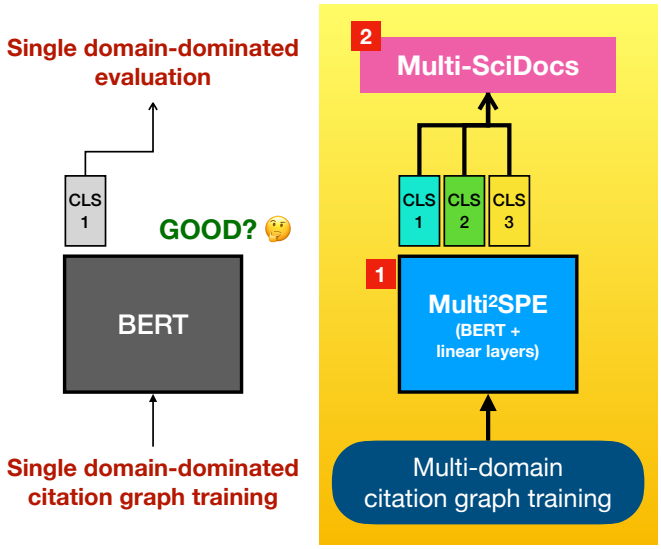



Figure 5.4: An overview of our two-parted solution. 1) Multi$^2$SPE is our modified Multi-CLS BERT model that better utilizes multi-domain citation data through multiple diversified CLS embeddings. 2) Multi-SciDocs is our new benchmark for testing embeddings of scientific papers under multi-domain settings.

whether we can represent papers from more diverse scientific domains using a single encoder and whether we could improve state-of-the-art models under the multi-domain settings.

In this section, we lay out our two-parted solution to overcome this limitation: the first part is our scientific paper encoder, Multi$^2$SPE, a varient of Multi-CLS BERT. This is built upon the intuition that extracting embeddings through just one CLS token is limiting, when more ideal ways of mixing contextualized word embeddings could be different for each subject area. For the second part, we introduce the Multi-SciDocs benchmark, to better understand the capabilities of scientific document representations in handling multi-domain settings.

Comparing Multi$^2$SPE and the single-CLS baselines on Multi-SciDocs suggests that training Multi$^2$SPE on single domain-dominated citation graphs already boosts the scores on multi-domain tasks; with more balanced multi-domain training, Multi$^2$SPE provides even bigger improvements.

### 5.3.1 Multi$^2$SPE: Multi-Domain $\times$ Multi-CLS Scientific Paper Encoder

The improvements in Section 3.4 suggest that the citing paper distribution is sometimes multi-mode and the multiple embedding is helpful in citation recommendation. However, our performance is not state-of-the-art because we use one-tower co-occurrence learning and our encoder does not start from a pre-trained LM. In this section, we would like to improve the BERT-based model using multiple CLS embeddings.

Since one CLS embedding corresponds to merely a single scheme of aggregating word embeddings, it might be sufficient for the documents from one domain but may be far from ideal for other domains. We address this observation by prepending multiple CLS tokens to input documents and introducing small architectural additions that encourage each CLS embedding to learn a distinctive way of mixing word embeddings together for the final document representation.

#### 5.3.1.1 Multiple CLS Encoder

With multiple CLS tokens (`[CLS_1], ..., [CLS_K]`), we insert linear layers $L_{l,k}$ at the sequence positions of each CLS embeddings as shown in Figure 5.5, to encourage the CLS embeddings to pay attention to different contextualized word embeddings. Similarly to Multi-CLS BERT, we insert additional linear transformations after the 4th, 8th, and 12th BERT layers.

We also adopt the re-parameterization trick used during the fine-tuning stage of Multi-CLS BERT to ensure that all the added linear transformations at each BERT layer are different and not similar to each other:

$$L_{l,k}(\boldsymbol{h}^c_{l,k}) = (\mathbf{W}_{l,k} - \frac{1}{K}\sum_{k'}\mathbf{W}_{l,k'})\boldsymbol{h}^c_{l,k} + \boldsymbol{b}_{l,k}, \tag{5.5}$$

$L_{l,k}$ is the linear transformation for $k$th CLS token at the layer $l$, $\mathbf{W}_{l,k} - \frac{1}{K}\sum_{k'}\mathbf{W}_{l,k'}$ is the linear projection weights and $\boldsymbol{b}_{l,k}$ is the bias term. To prevent $\mathbf{W}_{l,k} - \frac{1}{K}\sum_{k'}\mathbf{W}_{l,k'} = \mathbf{0}$, the gradient descent tend to learn different $\mathbf{W}_{l,k}$ for different $k$.

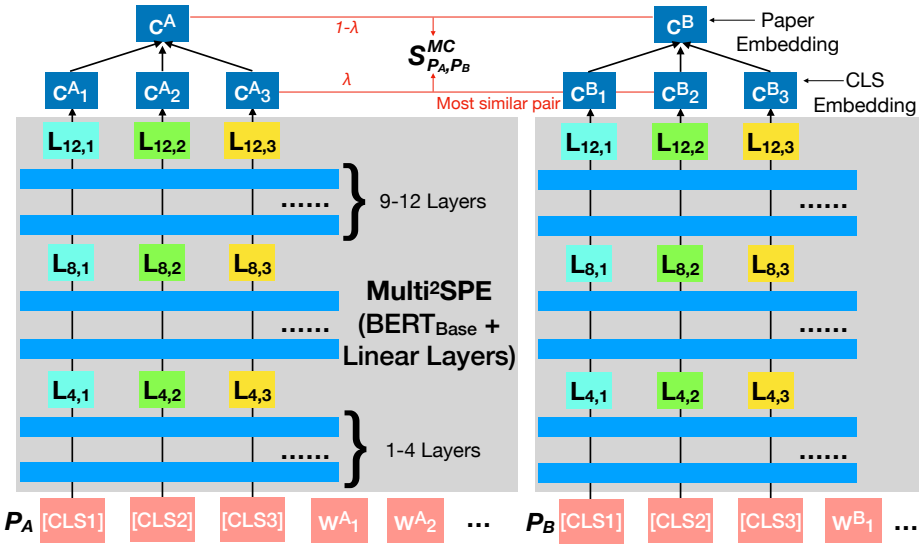



Figure 5.5: The architecture of Multi$^2$SPE and its similarity measurement during training $S^{MC}_{\mathcal{P}^A,\mathcal{P}^B}$.

#### 5.3.1.2 Contrastive Citation Prediction Loss

Existing state-of-the-art scientific paper encoders such as SPECTER [42] and SciNCL [163] use training signals coming from a contrastive citation prediction task: their objective function is to encourage the embedding of each query paper to be close to those of the paper cited by them, and be far away the papers they did not cite, $\mathcal{P}^-$.

Similarly, we minimize the cross entropy loss of a given query paper $\mathcal{P}^Q$, a cited paper $\mathcal{P}^+$, and a paper not cited, $\mathcal{P}^-$:

$$L_{\mathcal{P}^Q,\mathcal{P}^+,\mathcal{P}^-} = -\log\left(\frac{\exp(S^{MC}_{\mathcal{P}^Q,\mathcal{P}^+})}{\sum_{\mathcal{P}\in\{\mathcal{P}^+,\mathcal{P}^-\}}\exp(S^{MC}_{\mathcal{P}^Q,\mathcal{P}})}\right), \tag{5.6}$$

where $S^{MC}_{\mathcal{P}^Q,\mathcal{P}^+}$ is the similarity between the query paper $\mathcal{P}^Q$ and the cited paper $\mathcal{P}^+$ from the multiple CLS encoder. It is also the logit score for predicting the paper $\mathcal{P}^+$ as the cited paper.

#### 5.3.1.3 Measuring Document Similarity with Multiple Embeddings

One typical use of document embeddings is to perform a nearest neighbor search for retrieving candidates similar to the query document. While it would be possible to use each of CLS embeddings separately as in Section 3.4, or concatenate them together to encode each document, we would significantly increase the computational costs of the retrieval process. Instead, during inference, we simply take the summation of CLS embeddings from paper $A$ to be its final paper representation $\boldsymbol{c}^A = \sum_k \boldsymbol{c}^A_k$ and $\boldsymbol{c}^A_k = L_{12,k}(\boldsymbol{h}^{c,A}_{12,k})$.

During the contrastive training (Section 5.3.1.2), we compute the similarity between two papers $\mathrm{S}^{MC}_{\mathcal{P}^A,\mathcal{P}^B}$ using dot products between their paper embeddings $(\boldsymbol{c}^A)^T(\boldsymbol{c}^B)$ and the most similar CLS embeddings $\max_{i,j}(\boldsymbol{c}^A_i)^T\boldsymbol{c}^B_j$:

$$\mathrm{S}^{MC}_{\mathcal{P}^A,\mathcal{P}^B} = \lambda \max_{i,j}(\boldsymbol{c}^A_i)^T\boldsymbol{c}^B_j + (1-\lambda)(\boldsymbol{c}^A)^T(\boldsymbol{c}^B), \tag{5.7}$$

where $\boldsymbol{c}^A = \sum_k \boldsymbol{c}^A_k$ and $\lambda$ is the hyperparameter for controlling the dependency between the CLS embeddings. Smaller $\lambda$ makes similarity measurement in training and testing more consistent and encourages the CLS embeddings to collaborate with each other. Larger $\lambda$ encourages each of the CLS embeddings to become more meaningful paper embeddings on their own. The NLU experiments suggests that setting $\lambda > 0$ can greatly stabilize the $\mathrm{BERT}_{\mathrm{Large}}$ model while slightly improving $\mathrm{BERT}_{\mathrm{Base}}$.

### 5.3.2 Multi-SciDocs

Cohan et al. [42] proposed SciDocs as a comprehensive benchmark for evaluating scientific paper embeddings. SciDocs introduces 12 metrics from 7 tasks, but we have discovered that the domain distributions of 5 tasks are heavily biased toward computer science (CS) papers.[3] The only exceptions are MeSH (Medical Subject Headings) [125],

[3]Please see Appendix D.2.3 for detailed statistics.

| | MAG | multi. cite | | multi. co-cite | | Avg |
|---|---|---|---|---|---|---|
| | F1 | MAP | nDCG | MAP | nDCG | |
| SPECTER | 78.90 | 78.14 | 88.06 | 65.97 | 73.46 | 76.90 |
| Multi$^2$SPE (3 CLS, $\lambda = 0.1$) | 80.24 $\pm$ 0.18<br>6.35% | 81.57 $\pm$ 0.13<br>15.71% | 90.12 $\pm$ 0.07<br>17.29% | 69.12 $\pm$ 0.08<br>9.25% | 75.97 $\pm$ 0.07<br>9.45% | 79.40 $\pm$ 0.04<br>10.82% |
| SciNCL | 79.59 | 82.45 | 90.56 | 69.94 | 76.62 | 79.83 |
| Multi$^2$SPE (3 CLS, $\lambda = 0.1$) | 80.73 $\pm$ 0.27<br>5.58% | 83.25 $\pm$ 0.21<br>4.57% | 91.05 $\pm$ 0.12<br>5.16% | 71.10 $\pm$ 0.32<br>3.86% | 77.51 $\pm$ 0.24<br>3.80% | 80.73 $\pm$ 0.19<br>4.44% |

(a) Single domain (CS) training

| | MAG | multi. cite | | multi. co-cite | | Avg |
|---|---|---|---|---|---|---|
| | F1 | MAP | nDCG | MAP | nDCG | |
| SPECTER | 79.99 | 79.30 | 88.73 | 68.59 | 75.60 | 77.97 |
| Multi$^2$SPE (3 CLS, $\lambda = 0.1$) | 81.36 $\pm$ 0.29<br>6.85% | 84.08 $\pm$ 0.10<br>23.08% | 91.54 $\pm$ 0.06<br>24.96% | 71.79 $\pm$ 0.22<br>10.18% | 78.15 $\pm$ 0.17<br>10.45% | 81.10 $\pm$ 0.06<br>14.20% |
| SciNCL | 80.27 | 85.18 | 92.15 | 73.02 | 79.08 | 81.94 |
| Multi$^2$SPE (3 CLS, $\lambda = 0.1$) | 81.04 $\pm$ 0.05<br>3.90% | 85.73 $\pm$ 0.29<br>3.73% | 92.46 $\pm$ 0.17<br>3.98% | 74.05 $\pm$ 0.25<br>3.82% | 79.85 $\pm$ 0.19<br>3.69% | 82.63 $\pm$ 0.18<br>3.81% |

(b) Multiple domain training

Table 5.5: Results of our methods and baselines on Multi-SciDocs. All scores are averaged over four random seeds. We show standard errors as their confidence interval. Percentages indicate relative error reduction over the baselines (SPECTER or SciNCL).

which covers the papers from the biomedical domain and MAG (Microsoft Academic Graph) [203], which is a document classification task into 19 subject areas.

Thus, for a better measurement of multi-domain performance, we have created the multi-domain (co-)citation prediction tasks. We refer to the collection of 3 multi-domain tasks, multi. cite, multi. co-cite, and MAG as Multi-SciDocs.

For multi. (co-)cite datasets, we randomly sample the query papers from S2ORC [132], avoiding a certain domain from being the majority of query papers. For each query, we collect 500 negative papers and up to 5 positive papers. The task is to assign higher similarity scores to the positive papers and lower scores to the negative papers. In both datasets, the negative samples come from randomly sampled papers. In the multi. cite dataset, the positive samples are the papers cited by the query paper. In the multi. co-cite dataset, the positive samples and the query paper are both cited by another paper.

### 5.3.3 Experiments and Analyses

In the experiments, we evaluate Multi$^2$SPE and the corresponding baselines with Multi-SciDocs. SPECTER and SciNCL are our single [CLS] token baselines: both use identical neural architectures and loss functions, but differ in sampling methods used to create their contrastive triples. Since we found training datasets in previous literature to be potentially limiting in handling papers from various scientific domains, we build our own multi-domain

training datasets that follow the same sampling methods of SPECTER and SciNCL, but are more balanced in terms of the domain distribution.[4]

#### 5.3.3.1 Results

Our main results are shown in Table 5.5. We can see that Multi$^2$SPE has consistently outperformed the baselines in all training cases. The scores from MAG show that Multi$^2$SPE is better capable of classifying the texts into diverse subject domains. In multi-domain citation prediction, its error reductions are up to 25%. We hypothesize that the large improvement partially comes from the prevalent cross-domain citations in both our training and evaluation data. We note that the overall gains are smaller for SciNCL. We suspect that SciNCL's sampling method reduces the number of cross-domain citations in the dataset, which would have helped increase the diversity in CLS embeddings.

#### 5.3.3.2 Ablation Studies

In Table 5.6, we start by examining the effect of $\lambda$, the hyperparameter for controlling dependencies between CLS embeddings. While the differences are relatively small for $\lambda = 0.0$, we observe noticeable performance drops as we increase $\lambda$ to 0.5 and 1.0. Our intuition is that it is generally more beneficial to encourage all embeddings to become a meaningful whole together, rather than directing each of them to stand on their own.

In the second set of our ablation studies, we quantify the performance benefits of each architectural changes we introduced in Section 5.3.1.1. We can see that multiple CLS tokens are crucial, as having just one CLS leads to clear performance drop. Increasing the number of CLS tokens from 3 to 5 leads to mixed results. Their overall similar performance suggests that the quality of Multi$^2$SPE is not sensitive to the number of CLS tokens. Lastly, we

[4]Please see Appendix D.2.3 for the comparison of domain distribution of SPECTER/SciNCL single-domain datasets and our multi-domain datasets.

| | MAG F1 | cite MAP | co-cite MAP |
|---|---|---|---|
| Multi$^2$SPE (3 CLS, $\lambda = 0.1$) | 81.36 | 84.08 | 71.79 |
| **CLS Embedding Independence** | | | |
| $\rightarrow \lambda = 0.0$ | 81.06<br>-1.64% | 84.09<br>0.11% | 71.90<br>0.39% |
| $\rightarrow \lambda = 0.5$ | 80.70<br>-3.53% | 83.05<br>-6.45% | 71.46<br>-1.18% |
| $\rightarrow \lambda = 1.0$ | 79.78<br>-8.49% | 83.69<br>-2.40% | 71.18<br>-2.15% |
| **Architectural Changes** | | | |
| $\rightarrow$ 1 CLS token | 80.32<br>-5.57% | 83.32<br>-4.76% | 70.30<br>-5.30% |
| $\rightarrow$ 5 CLS tokens | 80.83<br>-2.84% | 84.03<br>-0.30% | 72.59<br>2.84% |
| $\rightarrow$ No linear layer injection in BERT | 80.82<br>-2.88% | 83.54<br>-3.38% | 71.12<br>-2.39% |
| $\rightarrow$ No re-parameterization trick | 80.89<br>-2.55% | 83.79<br>-1.82% | 71.11<br>-2.42% |

Table 5.6: Ablation studies conducted on SPECTER and multiple domain training data. All scores are averaged over four random seeds. Percentages indicate relative error reduction over the baseline (3 CLS, $\lambda = 0.1$).

observe that both the linear layer injection to BERT and the re-parameterization trick have clear contributions to our models' better performance.

### 5.3.4 Related Work

Many studies focus only specific scientific NLP tasks such as citation recommendation [17, 59, 60, 136] and paper recommendation [15, 254]. Instead, our goal is improving a general-purpose scientific paper encoder such as SPECTER [42] and SciNCL [163].

One common type of approaches for building scientific document encoders uses global topic distribution to model different *facets* of each paper [151, 98, 231, 129]. However, the classic approach lacks the ability to capture the compositional meaning of words compared to the neural based approaches and thus results in suboptimal performance [14].

Another line of efforts relies on pre-defined facets [29, 30, 162] or topics [254] for specific domains of interest, and measure paper similarities based on those facets/topics. Recently, Mysore et al. [156] suggests encoding a paper into multiple sentence embeddings to allow the users to search similar papers using partially constructed query papers. In

contrast, Multi-CLS BERT automatically learn to identify the facets that are helpful to the citation prediction task and combine all the facets into a single embedding to improve the similarity measurement and nearest neighbor search over the single CLS baseline while maintaining similar level of computational costs.

## 5.4 Chapter Conclusion

In this chapter, we propose representing the input text using $K$ CLS embeddings rather than using the single CLS embedding in BERT. Compared to BERT, Multi-CLS BERT significantly increases the GLUE and SuperGLUE scores and reduces the expected calibration error in GLUE. Compared to SPECTER, Multi$^2$SPE significantly improves the scores of Multi-SciDocs, which is proposed to measure the embeddings quality of the papers from multiple domains. Moreover, their only added cost is to reduce the maximal text length by $K$ and add a little extra time for computing the inserted linear transformations. Therefore, we recommend the wide use of multiple CLS embeddings for the almost free performance gain.

# CHAPTER 6

# CONCLUSION AND FUTURE WORK

"Life is a matter of choices, and every choice you make makes you." –John C. Maxwell

Language models (LMs) often encounter diverse choices when predicting the next word. This thesis first theoretically shows that the choices sometimes constitute a multimodal distribution, which cannot be modeled by a single hidden state embedding. Inspired by the theory, we propose several alternatives to softmax that uses multiple embeddings to capture the modes in the next word distribution or to estimate the probabilities of words in different partitions. The diverse embeddings improve the qualities of the generated model and allow a user to peek the future topics in a novel interactive language generation framework.

By assuming similar input text sequences would induce similar choice distribution, we use the choice clusters predicted by LMs to represent the input text sequence and improve the supervised downstream fine-tuning and unsupervised sentence similarity estimation. The results demonstrate that in many co-occurrence learning tasks, the single embedding representation limits the choices of LMs and we can design efficient multiple embedding representation to overcome the limitation.

## 6.1 Take Home Messages

- For both casual language models such as GPT-2 and bidirectional language models such as BERT, we propose architectures and loss functions that overcome the bottleneck of single embedding. Our proposal is effective in various applications and metrics, and efficient in terms of both computational cost and model size.

- We theoretically identify limitations of output softmax layer and empirically show that the multi-mode distribution is not rare in GPT-2 Small.
- Our analyses demonstrate that using multiple CLS embeddings in BERT can be viewed as an efficient way to ensemble multiple BERT encoders.
- In sequential recommendation, learning to copy or exclude the items in the input history can significantly improve not only the datasets with duplicated items but also the datasets without duplicated items.
- When using multiple facet embeddings to represent a text sequence, we find that the predicted embeddings are easy to collapse to identical embeddings. In different model architectures, we often need different modifications of losses and architectures to diversify the outputted embeddings.
- The learned facet embeddings could be useful in many different ways. In addition to predicting the co-occurred item (e.g., a next word or cited paper), the multiple embeddings could provide asymmetric similarity, estimate the word importance without supervision, reduce issues caused by variable-length sentences, predict the future topics, improve the performance of BERT especially in a few shot setting, and reduce the expected calibration error of a BERT model.
- We consistently observe performance improvement by adopting multiple embeddings on various kinds of models if a) the number of possible co-occurred items such as vocabulary size is large, b) the multimodal distribution is common in our training signal, and c) the facet embeddings are properly diversified.

## 6.2 Limitations and Future Work

Multi-facet embedding representation can be viewed as a general tool to improve the deep learning models trained by self-supervised co-occurrence learning. In this thesis, we develop several variants of the multi-facet embeddings and demonstrate their effec-

tiveness in important applications. Nevertheless, as most of the other tools that improve deep learning models, our understanding to multi-facet embedding is still limited and the challenges/concerns we mentioned in Section 1.1 are not completely solved. Here, we list some open problems in each concern for the future work of the thesis.

- **Optimization Difficulties**: In each of the applications, we usually must develop different tricks to diversify the facets and improve the performance. For example, we aggregate the facet embeddings using mixture of softmax in Section 2.3, NNSC or Kmeans loss in Chapter 3 and Chapter 4, and substracting the average facets during fine-tuning in Chapter 5.

  We still do not know why some methods are only applicable in some applications. For example, when pretraining BERT model in Chapter 5, we find that aggregation by mixing average and maximal outperforms mixture of softmax, NNSC loss, and even the aggregation way we used in the fine-tuning stage.

- **Insufficient Theoretical Support**: Although we have developed theory that highlights the limitation of single embedding and motivates our multiple embedding alternatives, our theory does not explain all fundamental weaknesses in the softmax layer. There are several alternatives to single embedding representation in Euclidean space, such as box embeddings [170], hyperbolic embeddings [37], energy networks [114], and our multiple embedding representation. There are also other theoretical findings such as Bhattacharjee and Dasgupta [18], which studies the minimal number of dimensions of an embedding space that can robustly reconstruct any co-occurrence relation. Nevertheless, I am not aware of a systematic study on the relations of these alternatives and a general theory for the limitations of single embeddings that can guide us to choose these alternatives in a new dataset. Furthermore, most of the multiple embedding approaches (and probably other softmax alternatives) require some tricks, which are usually developed by trial and error. Given a new problem, we do not know exactly which existing tricks might perform better in the task of interest and if there exists a better trick to be discovered.

In addition, we do not have an accurate estimation of how often multimodal distributions happen in the next word prediction, how many multimodal distributions can be modeled by our current approaches, and how much our proposed approaches can solve the issues caused by the global and static word similarity. We do not know why we need to use modeling multimodal distribution in NLG applications to explain the performance gain, while some of our NLU approaches are more like an efficient ensembling method. We are not exactly sure why our improvement in GLUE and SuperGLUE tasks are smaller compared to classic ensembling approaches when more training data is available. Finally, we empirically observe that the training methods that lead to more diverse facets seem to also make the facet embeddings less depend on the input. There might be some theories that can explain this observation.

- **Unknown Effectiveness and Applicability**:

  In some applications, after trying several tricks, we still cannot make the performance of multiple embeddings surpass the single embedding baseline. Examples of the applications include the phrase similarity estimation in Section 3.2.2.5, paper reviewer affinity estimation[1], and cold-start recommendation given a small training dataset such as CiteUlike [232]. We suspect that the possible reasons are as follows. a) Our encoder cannot learn to output diverse facet embeddings. This might be due to the sparsity of the training signal and/or the limited diversity of the co-occurred items (i.e., the distributions of the co-occurred text are mostly single-modal). b) There are some distribution shifts between training and testing, and the shift makes the model overfit the training distribution or context-dependent word similarity that is less generalizable compared to global word similarity. At testing time, some minor modes in the distribution are actually outliers we should ignore. c) There may be still some other tricks we haven't discovered to improve the quality and diversity

[1]Our single embedding baseline in this problem is a part of the state-of-the-art affinity estimation used by various machine learning top conferences and OpenReview.

of the multiple embedding representation. Thus, the next question is how can we know when the multiple embedding representation will benefits which applications.

Due to time limitation, we haven't fine-tuned the hyperparameters for our sequential recommendation experiment in Section 2.5 and we also do not know if the improvement persists after being combined with the second stage reranker, which is commonly used in industry. We haven't tested if Multi-CLS BERT can improve the generalization ability of the BERT under a distribution shift as the classic ensemble models [164, 219].

We would like to know whether multiple embeddings, especially the dynamic partitioning, could improve the consistency of generated text while reduce its repetition in summarization and dialogue generation. We are curious about if our approaches can sample more diverse next items from the predicted multimodal distribution for sequential recommendation, code generation, and math question answering. Furthermore, we plan to apply our approach to information retrieval models to discover the facets in a query.

Finally, our experiments focus on the relatively small language models such as GPT-2 Small and BERT base. In future, it would also be interesting to investigate if our proposed softmax alternatives can also improve very large LM such as GPT-3 or the extreme classification models that process the data other than the natural language.

# BIBLIOGRAPHY


[1] N. Akoury, S. Wang, J. Whiting, S. Hood, N. Peng, and M. Iyyer. STORIUM: A Dataset and Evaluation Platform for Machine-in-the-Loop Story Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6470–6484, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.525. URL https://aclanthology.org/2020.emnlp-main.525. 14, 90

[2] E. Alfonseca, K. Filippova, J.-Y. Delort, and G. Garrido. Pattern learning for relation extraction with a hierarchical topic model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 54–59, Jeju Island, Korea, 2012. Association for Computational Linguistics. URL https://aclanthology.org/P12-2011. 79

[3] Z. Allen-Zhu and Y. Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *ArXiv preprint*, abs/2012.09816, 2020. URL https://arxiv.org/abs/2012.09816. 111

[4] S. Amin, K. A. Dunfield, A. Vechkaeva, and G. Neumann. A data-driven approach for noise reduction in distantly supervised biomedical relation extraction. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, pages 187–194, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.bionlp-1.20. URL https://aclanthology.org/2020.bionlp-1.20. 79

[5] P. Ammanabrolu, E. Tien, W. Cheung, Z. Luo, W. Ma, L. J. Martin, and M. O. Riedl. Story realization: Expanding plot events into sentences. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7375–7382. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6232. 107, 108

[6] S. Aroca-Ouellette and F. Rudzicz. On Losses for Modern Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4970–4981, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.403. URL https://aclanthology.org/2020.emnlp-main.403. xvii, xxiii, 8, 11, 109, 111, 112, 113, 118, 119, 120, 121, 187, 188

[7] S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SyK00v5xx. 8, 60, 63, 86, 181, 182, 183

[8] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018. doi: 10.1162/tacl_a_00034. URL https://aclanthology.org/Q18-1034. 66

[9] B. Athiwaratkun and A. Wilson. Multimodal word distributions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1645–1656, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1151. URL https://aclanthology.org/P17-1151. 3, 45, 49, 66, 79

[10] J. Austin. The book of endless history: Authorial use of GPT2 for interactive storytelling. In R. E. Cardona-Rivera, A. Sullivan, and R. M. Young, editors, *Interactive Storytelling - 12th International Conference on Interactive Digital Storytelling, ICIDS 2019, Little Cottonwood Canyon, UT, USA, November 19-22, 2019, Proceedings*, volume 11869, pages 429–432. Springer, 2019. doi: 10.1007/978-3-030-33894-7_47. URL https://doi.org/10.1007/978-3-030-33894-7_47. 107

[11] A. Awasthi, S. Sarawagi, R. Goyal, S. Ghosh, and V. Piratla. Parallel iterative edit models for local sequence transduction. 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)*, pages 4260–4270, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1435. URL https://aclanthology.org/D19-1435. 45

[12] L. Baldini Soares, N. FitzGerald, J. Ling, and T. Kwiatkowski. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1279. URL https://aclanthology.org/P19-1279. 68, 115

[13] L. Balles and T. Fischbacher. Holographic and other point set distances for machine learning, 2019. URL https://openreview.net/forum?id=rJlpUiAcYX. 66

[14] T. Bansal, D. Belanger, and A. McCallum. Ask the GRU: Multi-task learning for deep text recommendations. In *RecSys*, 2016. xvi, 8, 81, 84, 85, 86, 87, 88, 132, 184

[15] J. Beel, B. Gipp, S. Langer, and C. Breitinger. paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, 2016. 88, 132

[16] I. Beltagy, A. Cohan, and K. Lo. SciBERT: Pretrained contextualized embeddings for scientific text. In *EMNLP*, 2019. 125

[17] C. Bhagavatula, S. Feldman, R. Power, and W. Ammar. Content-based citation recommendation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 238–251, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1022. URL https://aclanthology.org/N18-1022. 88, 125, 132

[18] R. Bhattacharjee and S. Dasgupta. What relations are reliably embeddable in euclidean space? In *Algorithmic Learning Theory*, pages 174–195. PMLR, 2020. 136

[19] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 601–608. MIT Press, 2001. URL https://proceedings.neurips.cc/paper/2001/hash/296472c9542ad4d4788d543508116cbc-Abstract.html. 49, 66, 92, 93, 101

[20] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In J. T. Wang, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250. ACM, 2008. doi: 10.1145/1376616.1376746. URL https://doi.org/10.1145/1376616.1376746. 74

[21] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html. 14, 22, 180

[22] X. Cai, J. Huang, Y. Bian, and K. Church. Isotropy in the contextual embedding space: Clusters and manifolds. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=xYGNO86OWDH. 180

[23] Z. Cao, S. Li, Y. Liu, W. Li, and H. Ji. A novel neural topic model and its supervised extension. In B. Bonet and S. Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2210–2216. AAAI Press, 2015. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9303. 66

[24] D. Card, P. Henderson, U. Khandelwal, R. Jia, K. Mahowald, and D. Jurafsky. With little power comes great responsibility. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9263–9274, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.745. URL https://aclanthology.org/2020.emnlp-main.745. 118

[25] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. *ArXiv preprint*, abs/2005.12872, 2020. URL https://arxiv.org/abs/2005.12872. 3, 67

[26] A. Celikyilmaz, A. Bosselut, X. He, and Y. Choi. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1150. URL https://aclanthology.org/N18-1150. xvi, 63, 64

[27] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001. URL https://aclanthology.org/S17-2001. 59, 99

[28] A. Chaganty, A. Paranjape, P. Liang, and C. D. Manning. Importance sampling for unbiased on-demand evaluation of knowledge base population. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1038–1048, Copenhagen, Denmark, 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1109. URL https://aclanthology.org/D17-1109. 73

[29] T. Chakraborty, A. Krishna, M. Singh, N. Ganguly, P. Goyal, and A. Mukherjee. Ferosa: A faceted recommendation system for scientific articles. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2016. 132

[30] J. Chan, J. C. Chang, T. Hope, D. Shahaf, and A. Kittur. Solvent: A mixed initiative system for finding analogies between research papers. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–21, 2018. 132

[31] H. Chang, A. Munir, A. Liu, J. T. Wei, A. Traylor, A. Nagesh, N. Monath, P. Verga, E. Strubell, and A. McCallum. Extracting multilingual relations under limited resources: TAC 2016 cold-start KB construction and slot-filling using compositional universal schema. In *Proceedings of the 2016 Text Analysis Conference, TAC 2016, Gaithersburg, Maryland, USA, November 14-15, 2016*. NIST, 2016. URL https://tac.nist.gov/publications/2016/participant.papers/TAC2016.UMass_IESL.proceedings.pdf. 73

[32] H.-S. Chang and A. McCallum. Softmax bottleneck makes language models unable to represent multi-mode word distributions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8048–8073, 2022. 9, 11, 16

[33] H.-S. Chang, Z. Wang, L. Vilnis, and A. McCallum. Distributional inclusion vector embedding for unsupervised hypernymy detection. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 485–495, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1045. URL https://aclanthology.org/N18-1045. 78, 79

[34] H.-S. Chang, A. Agrawal, and A. McCallum. Extending multi-sense word embedding to phrases and sentences for unsupervised semantic applications. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2021. 10, 11, 52

[35] H.-S. Chang, J. Yuan, M. Iyyer, and A. McCallum. Changing the mind of transformers for topically-controllable language generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2601–2611, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.223. URL https://aclanthology.org/2021.eacl-main.223. 10, 11

[36] B. Chen and C. Cherry. A systematic comparison of smoothing techniques for sentence-level BLEU. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367, Baltimore, Maryland, USA, 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-3346. URL https://aclanthology.org/W14-3346. 105

[37] B. Chen, X. Huang, L. Xiao, Z. Cai, and L. Jing. Hyperbolic interaction model for hierarchical multi-label classification. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7496–7503. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6247. 136

[38] G. Chen, Y. Liu, H. Luan, M. Zhang, Q. Liu, and M. Sun. Learning to predict explainable plots for neural story generation. *ArXiv preprint*, abs/1912.02395, 2019. URL https://arxiv.org/abs/1912.02395. 108

[39] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179. 84

[40] E. Clark, A. S. Ross, C. Tan, Y. Ji, and N. A. Smith. Creative writing with a machine in the loop: Case studies on slogans and stories. In *23rd International Conference on Intelligent User Interfaces*, 2018. 90

[41] K. Cobbe, V. Kosaraju, M. Bavarian, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *ArXiv preprint*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168. 14

[42] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. Weld. SPECTER: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.207. URL https://aclanthology.org/2020.acl-main.207. 7, 8, 11, 115, 125, 128, 129, 132, 190

[43] T. M. Cover. The number of linearly inducible orderings of points in d-space. *SIAM Journal on Applied Mathematics*, 15(2):434–439, 1967. 45

[44] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, and R. Liu. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1edEyBKDS. 45, 92, 104, 107

[45] D. Demeter, G. Kimmel, and D. Downey. Stolen probability: A structural weakness of neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2191–2197, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.198. URL https://aclanthology.org/2020.acl-main.198. xxiii, 44, 172, 178, 179, 180

[46] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. 96

[47] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423. 95, 110, 118

[48] N. Ding, X. Wang, Y. Fu, G. Xu, R. Wang, P. Xie, Y. Shen, F. Huang, H. Zheng, and R. Zhang. Prototypical representation learning for relation extraction. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=aCgLmfhIy_f. 3

[49] R. Dóczi, F. Kis, B. Sütő, V. Póser, G. Kronreif, E. Jósvai, and M. Kozlovszky. Increasing ros 1.x communication security for medical surgery robot. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2016. xxii, 83

[50] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *ArXiv preprint*, abs/2002.06305, 2020. URL https://arxiv.org/abs/2002.06305. 110, 119

[51] H. Dubossarsky, E. Grossman, and D. Weinshall. Coming to your senses: on controls and evaluation sets in polysemy research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1732–1740, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1200. URL https://aclanthology.org/D18-1200. 4, 64, 65

[52] T. Eiter and H. Mannila. Distance measures for point sets and their computation. *Acta Informatica*, 34(2):109–133, 1997. 66

[53] A. Epasto and B. Perozzi. Is a single embedding enough? learning node representations that capture multiple social contexts. In L. Liu, R. W. White, A. Mantrach, F. Silvestri, J. J. McAuley, R. Baeza-Yates, and L. Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 394–404. ACM, 2019. doi: 10.1145/3308558.3313660. URL https://doi.org/10.1145/3308558.3313660. 3, 89

[54] K. Ethayarajh, D. Duvenaud, and G. Hirst. Towards understanding linear word analogies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1315. URL https://aclanthology.org/P19-1315. 15, 19

[55] W. Falcon, J. Borovec, A. Wälchli, N. Eggert, J. Schock, J. Jordan, N. Skafte, Ir1dXD, V. Bereznyuk, E. Harris, T. Murrell, P. Yu, S. Præsius, T. Addair, J. Zhong, D. Lipin, S. Uchida, S. Bapat, H. Schröter, B. Dayma, A. Karnachev, A. Kulkarni, S. Komatsu, Martin.B, J.-B. SCHIRATTI, H. Mary, D. Byrne, C. Eyzaguirre, cinjon, and A. Bakhtin. Pytorchlightning/pytorch-lightning: 0.7.6 release. 2020. doi: 10.5281/zenodo.3828935. URL https://doi.org/10.5281/zenodo.3828935. 191

[56] A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL https://aclanthology.org/P18-1082. 97, 108, 185

[57] A. Fan, M. Lewis, and Y. Dauphin. Strategies for structuring story generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1254. URL https://aclanthology.org/P19-1254. 108

[58] A. Fan, T. Lavril, E. Grave, A. Joulin, and S. Sukhbaatar. Addressing some limitations of transformers with feedback memory. *ArXiv preprint*, abs/2002.09402, 2020. URL https://arxiv.org/abs/2002.09402. 28

[59] M. Färber and A. Jatowt. Citation recommendation: Approaches and datasets. *ArXiv preprint*, abs/2002.06961, 2020. URL https://arxiv.org/abs/2002.06961. 88, 132

[60] M. Färber and A. Sampath. Hybridcite: A hybrid model for context-aware citation recommendation. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, pages 117–126, 2020. 132

[61] M. Faruqui, Y. Tsvetkov, D. Yogatama, C. Dyer, and N. A. Smith. Sparse overcomplete word vector representations. 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)*, pages 1491–1500, Beijing, China, 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1144. URL https://aclanthology.org/P15-1144. 66

[62] W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *ArXiv preprint*, abs/2101.03961, 2021. URL https://arxiv.org/abs/2101.03961. 45

[63] S. Fort, H. Hu, and B. Lakshminarayanan. Deep ensembles: A loss landscape perspective. *ArXiv preprint*, abs/1912.02757, 2019. URL https://arxiv.org/abs/1912.02757. 111, 118, 121

[64] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1050–1059. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/gal16.html. 111

[65] L. Galke, F. Mai, I. Vagliano, and A. Scherp. Multi-modal adversarial autoencoders for recommendations of citations and subject labels. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization (UMAP)*, 2018. 88

[66] O. Ganea, S. Gelly, G. Bécigneul, and A. Severyn. Breaking the softmax bottleneck via learnable monotonic pointwise non-linearities. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2073–2082. PMLR, 2019. URL http://proceedings.mlr.press/v97/ganea19a.html. 4, 44

[67] J. Gao, D. He, X. Tan, T. Qin, L. Wang, and T. Liu. Representation degeneration problem in training natural language generation models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SkEYojRqtm. 180

[68] Y. Gao, C. Herold, W. Wang, and H. Ney. Exploring kernel functions in the softmax layer for contextual word classification. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong, 2019. Association for Computational Linguistics. URL https://aclanthology.org/2019.iwslt-1.24. 3, 180

[69] M. Ghazvininejad, X. Shi, J. Priyadarshi, and K. Knight. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada, 2017. Association for Computational Linguistics. URL https://aclanthology.org/P17-4008. 104

[70] S. Goldfarb-Tarrant, H. Feng, and N. Peng. Plan, write, and revise: an interactive system for open-domain story generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 89–97, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4016. URL https://aclanthology.org/N19-4016. 108

[71] I. Good and T. Tideman. Stirling numbers and a geometric, structure from voting theory. *Journal of Combinatorial Theory, Series A*, 23(1):34–45, 1977. 45

[72] E. Grave, A. Joulin, and N. Usunier. Improving neural language models with a continuous cache. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=B184E5qee. 9, 36

[73] A. Grivas, N. Bogoychev, and A. Lopez. Low-rank softmax can have unargmaxable classes in theory but rarely in practice. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6738–6758, 2022. 179

[74] J. Gu, Z. Lu, H. Li, and V. O. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1154. URL https://aclanthology.org/P16-1154. 9, 36, 38, 39, 42

[75] J. Gu, Q. Liu, and K. Cho. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7: 661–676, 2019. doi: 10.1162/tacl_a_00292. URL https://aclanthology.org/Q19-1042. 67

[76] V. Gupta, A. Saw, P. Nokhiz, H. Gupta, and P. Talukdar. Improving document classification with multi-sense embeddings. In *ECAI*, 2020. 66

[77] V. Gupta, A. Saw, P. Nokhiz, P. Netrapalli, P. Rai, and P. P. Talukdar. P-SIF: document embeddings using partition averaging. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7863–7870. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6292. 3, 66

[78] R. Han, M. Gill, A. Spirling, and K. Cho. Conditional word embedding and hypothesis testing via Bayes-by-backprop. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4890–4895, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1527. URL https://aclanthology.org/D18-1527. 66

[79] X. Han, T. Gao, Y. Lin, H. Peng, Y. Yang, C. Xiao, Z. Liu, P. Li, J. Zhou, and M. Sun. More data, more relations, more context and more openness: A review and outlook for relation extraction. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 745–758, Suzhou, China, 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.aacl-main.75. 78

[80] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015. 41

[81] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009. ISBN 9780387848570. doi: 10.1007/978-0-387-84858-7. URL https://doi.org/10.1007/978-0-387-84858-7. 77, 87

[82] M. Havasi, R. Jenatton, S. Fort, J. Z. Liu, J. Snoek, B. Lakshminarayanan, A. M. Dai, and D. Tran. Training independent subnetworks for robust prediction. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=OGg9XnKxFAH. 124

[83] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *ArXiv preprint*, abs/1606.08415, 2016. URL https://arxiv.org/abs/1606.08415. 26

[84] K. M. Hermann, T. Kociský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/afdec7005cc9f14302cd0474fd0f3c96-Abstract.html. 63

[85] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. In Y. Bengio and Y. LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06939. 9, 17, 42

[86] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997. 25, 67

[87] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rygGQyrFvH. 14

[88] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019. URL http://proceedings.mlr.press/v97/houlsby19a.html. 117

[89] P. O. Hoyer. Non-negative sparse coding. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, 2002. 5, 54, 101

[90] D. Ippolito, D. Grangier, C. Callison-Burch, and D. Eck. Unsupervised hierarchical story infilling. In *Proceedings of the First Workshop on Narrative Understanding*, pages 37–43, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-2405. URL https://aclanthology.org/W19-2405. 108

[91] D. Iter, K. Guu, L. Lansing, and D. Jurafsky. Pretraining with contrastive sentence objectives improves discourse performance of language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4859–4870, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.439. URL https://aclanthology.org/2020.acl-main.439. 125

[92] P. Izmailov, D. Podoprikhin, T. Garipov, D. P. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. In A. Globerson and R. Silva, editors, *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 876–885. AUAI Press, 2018. URL http://auai.org/uai2018/proceedings/papers/313.pdf. xvii, 121, 122

[93] K. Ji, R. Sun, X. Li, and W. Shu. Improving matrix approximation for recommendation via a clustering-based reconstructive method. *Neurocomputing*, 173:912–920, 2016. 88

[94] S. Kanai, Y. Fujiwara, Y. Yamanaka, and S. Adachi. Sigsoftmax: Reanalysis of the softmax bottleneck. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 284–294, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/9dcb88e0137649590b755372b040afad-Abstract.html. 4, 28, 29, 44

[95] A. Kanakia, Z. Shen, D. Eide, and K. Wang. A scalable hybrid research paper recommender system for microsoft academic. In L. Liu, R. W. White, A. Mantrach, F. Silvestri, J. J. McAuley, R. Baeza-Yates, and L. Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2893–2899. ACM, 2019. doi: 10.1145/3308558.3313700. URL https://doi.org/10.1145/3308558.3313700. 88

[96] W.-C. Kang and J. McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206. IEEE, 2018. 8, 9, 17, 40, 41

[97] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *ArXiv preprint*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361. 14

[98] M. Karimzadehgan, C. Zhai, and G. Belford. Multi-aspect expertise matching for review assignment. In *Proceedings of the 17th ACM conference on Information and knowledge management*, 2008. 132

[99] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher. CTRL: A conditional transformer language model for controllable generation. *ArXiv preprint*, abs/1909.05858, 2019. URL https://arxiv.org/abs/1909.05858. 107

[100] O. Khattab and M. Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In J. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen, and Y. Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48. ACM, 2020. doi: 10.1145/3397271.3401075. URL https://doi.org/10.1145/3397271.3401075. 3, 125

[101] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980. 119, 184

[102] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3294–3302, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/f442d33fa06832082290ad8544a8da27-Abstract.html. 53, 60

[103] H. Kobayashi, M. Noguchi, and T. Yatsuka. Summarization based on embedding distributions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1984–1989, Lisbon, Portugal, 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1232. URL https://aclanthology.org/D15-1232. 62, 63

[104] S. Kobayashi, S. Kiyono, J. Suzuki, and K. Inui. Diverse lottery tickets boost ensemble from a single pretrained model. In *Challenges & Perspectives in Creating Large Language Models*, 2022. URL https://openreview.net/forum?id=rCzgE3zHL-q. 124

[105] L. Kong, C. de Masson d'Autume, L. Yu, W. Ling, Z. Dai, and D. Yogatama. A mutual information maximization perspective of language representation learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Syx79eBKwr. 1, 18, 180

[106] W. Kong, S. Khadanga, C. Li, S. Gupta, M. Zhang, W. Xu, and M. Bendersky. Multi-aspect dense retrieval. 2022. 3

[107] I. Korkontzelos, T. Zesch, F. M. Zanzotto, and C. Biemann. SemEval-2013 task 5: Evaluating phrasal semantics. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 39–47, Atlanta, Georgia, USA, 2013. Association for Computational Linguistics. URL https://aclanthology.org/S13-2007. 64

[108] M. Kula. Mixture-of-tastes models for representing users with diverse interests. *ArXiv preprint*, abs/1711.08379, 2017. URL https://arxiv.org/abs/1711.08379. 3, 89

[109] S. Kumar and Y. Tsvetkov. Von mises-fisher loss for training sequence to sequence models with continuous outputs. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJlDnoA5Y7. 55

[110] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 957–966. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/kusnerb15.html. 60

[111] J. H. Lau, P. Cook, D. McCarthy, D. Newman, and T. Baldwin. Word sense induction for novel sense detection. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 591–601, Avignon, France, 2012. Association for Computational Linguistics. URL https://aclanthology.org/E12-1060. 49

[112] S. Lavoie, C. Tsirigotis, M. Schwarzer, K. Kawaguchi, A. Vani, and A. Courville. Simplicial embeddings in self-supervised learning and downstream classification. *ArXiv preprint*, abs/2204.00616, 2022. URL https://arxiv.org/abs/2204.00616. 124

[113] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR, 2019. URL http://proceedings.mlr.press/v97/lee19d.html. 56

[114] J.-Y. Lee, D. Patel, P. Goyal, and A. McCallum. Structured energy network as a dynamic loss function. case study. a case study with multi-label classification. 2021. 136

[115] B. Li, H. Zhou, J. He, M. Wang, Y. Yang, and L. Li. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.733. URL https://aclanthology.org/2020.emnlp-main.733. 18

[116] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California, 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1014. URL https://aclanthology.org/N16-1014. 103

[117] L. H. Li, P. H. Chen, C.-J. Hsieh, and K.-W. Chang. Efficient contextual representation learning with continuous outputs. *Transactions of the Association for Computational Linguistics*, 7:611–624, 2019. doi: 10.1162/tacl_a_00289. URL https://aclanthology.org/Q19-1039. 55

[118] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353. 117

[119] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. Dal Lago, T. Hubert, P. Choy, C. de Masson d'Autume, I. Babuschkin, X. Chen, P.-S. Huang, J. Welbl, S. Gowal, A. Cherepanov, J. Molloy, D. Mankowitz, E. Sutherland Robson, P. Kohli, N. de Freitas, K. Kavukcuoglu, and O. Vinyals. Competition-level code generation with alphacode. *ArXiv preprint*, abs/2203.07814, 2022. URL https://arxiv.org/abs/2203.07814. 14

[120] C. Liang, P. He, Y. Shen, W. Chen, and T. Zhao. Camero: Consistency regularized ensemble of perturbed language models with weight sharing. *ArXiv preprint*, abs/2204.06625, 2022. URL https://arxiv.org/abs/2204.06625. 110, 124

[121] Y. Liao, X. Jiang, and Q. Liu. Probabilistically masked language model capable of autoregressive generation in arbitrary word order. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 263–274, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.24. URL https://aclanthology.org/2020.acl-main.24. 45

[122] B. Y. Lin, W. Zhou, M. Shen, P. Zhou, C. Bhagavatula, Y. Choi, and X. Ren. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.165. URL https://aclanthology.org/2020.findings-emnlp.165. 108

[123] C.-Y. Lin and E. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157, 2003. URL https://aclanthology.org/N03-1020. 63

[124] Y.-C. Lin. Breaking the softmax bottleneck for sequential recommender systems with dropout and decoupling. *ArXiv preprint*, abs/2110.05409, 2021. URL https://arxiv.org/abs/2110.05409. 3

[125] C. E. Lipscomb. Medical subject headings (MeSH). *Bulletin of the Medical Library Association*, 88(3):265, 2000. 129

[126] A. Liu, S. Swayamdipta, N. A. Smith, and Y. Choi. Wanli: Worker and ai collaboration for natural language inference dataset creation. *ArXiv preprint*, abs/2201.05955, 2022. URL https://arxiv.org/abs/2201.05955. 14

[127] H. Liu, X. Kong, X. Bai, W. Wang, T. M. Bekele, and F. Xia. Context-based collaborative filtering for citation recommendation. *IEEE Access*, 3:1695–1703, 2015. 88

[128] N. Liu, Q. Tan, Y. Li, H. Yang, J. Zhou, and X. Hu. Is a single vector enough?: Exploring node polysemy for network embedding. In A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 932–940. ACM, 2019. doi: 10.1145/3292500.3330967. URL https://doi.org/10.1145/3292500.3330967. 3, 89

[129] X. Liu, T. Suel, and N. D. Memon. A robust model for paper reviewer assignment. In A. Kobsa, M. X. Zhou, M. Ester, and Y. Koren, editors, *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 25–32. ACM, 2014. doi: 10.1145/2645710.2645749. URL https://doi.org/10.1145/2645710.2645749. 132

[130] X. Liu, Z. Han, X. Wen, Y. Liu, and M. Zwicker. L2G auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention. In *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019*, pages 989–997, 2019. doi: 10.1145/3343031.3350960. URL https://doi.org/10.1145/3343031.3350960. 66

[131] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692, 2019. URL https://arxiv.org/abs/1907.11692. 110

[132] K. Lo, L. L. Wang, M. Neumann, R. Kinney, and D. Weld. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.447. URL https://aclanthology.org/2020.acl-main.447. 85, 130, 183, 190

[133] L. Logeswaran and H. Lee. An efficient framework for learning sentence representations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJvJXZb0W. 113

[134] Y. Luan, J. Eisenstein, K. Toutanova, and M. Collins. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345, 2021. doi: 10.1162/tacl_a_00369. URL https://aclanthology.org/2021.tacl-1.20. 3, 125

[135] L. Luo, W. Cai, S. Zhou, M. Lees, and H. Yin. A review of interactive narrative systems and technologies: a training perspective. *Simulation*, 91(2):126–147, 2015. doi: 10.1177/0037549714566722. URL https://doi.org/10.1177/0037549714566722. 90

[136] S. Ma, C. Zhang, and X. Liu. A review of citation recommendation: from textual content to enriched context. *Scientometrics*, pages 1–28, 2020. 88, 132

[137] X. Ma, H. Lu, Z. Gan, and Q. Zhao. An exploration of improving prediction accuracy by constructing a multi-type clustering based recommendation framework. *Neurocomputing*, 191:388–397, 2016. 88

[138] X. Ma, Y. Jiang, N. Bach, T. Wang, Z. Huang, F. Huang, and W. Lu. MuVER: Improving first-stage entity retrieval with multi-view entity representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2617–2624, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.205. URL https://aclanthology.org/2021.emnlp-main.205. 3

[139] D. J. MacKay. Probable networks and plausible predictions-a review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 6(3):469, 1995. 118

[140] F. Mai and J. Henderson. Bag-of-vectors autoencoders for unsupervised conditional text generation. *ArXiv preprint*, abs/2110.07002, 2021. URL https://arxiv.org/abs/2110.07002. 3

[141] L. J. Martin, P. Ammanabrolu, X. Wang, W. Hancock, S. Singh, B. Harrison, and M. O. Riedl. Event representations for automated story generation with deep neural nets. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 868–875. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17046. 107

[142] Y. Matsubara, L. Soldaini, E. Lind, and A. Moschitti. Ensemble transformer for efficient and accurate ranking tasks: an application to question answering systems. *ArXiv preprint*, abs/2201.05767, 2022. URL https://arxiv.org/abs/2201.05767. 124

[143] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In R. Baeza-Yates, M. Lalmas, A. Moffat, and B. A. Ribeiro-Neto, editors, *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 43–52. ACM, 2015. doi: 10.1145/2766462.2767755. URL https://doi.org/10.1145/2766462.2767755. 41

[144] S. M. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S. K. Lam, A. M. Rashid, J. A. Konstan, and J. Riedl. On the recommending of citations for research papers. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, 2002. 88

[145] F. McSherry and M. Najork. Computing information retrieval performance measures efficiently in the presence of tied scores. In *ECIR*, 2008. 87

[146] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Byj72udxe. 9, 36, 38, 39, 40, 46

[147] N. Miao, H. Zhou, C. Zhao, W. Shi, and L. Li. Kernelized bayesian softmax for text generation. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12487–12497, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/967c2ae04b169f07e7fa8fdfd110551e-Abstract.html. 2, 3, 45

[148] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html. 15, 18, 53, 55, 86, 172

[149] D. Milajevs, D. Kartsaklis, M. Sadrzadeh, and M. Purver. Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–719, Doha, Qatar, 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1079. URL https://aclanthology.org/D14-1079. 60

[150] G. A. Miller. *WordNet: An electronic lexical database*. MIT press, 1998. 76

[151] D. Mimno and A. McCallum. Expertise modeling for matching papers with reviewers. In *SIGKDD*, 2007. 132

[152] D. M. Mimno and A. McCallum. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In *UAI*, 2008. 66

[153] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, 2009. Association for Computational Linguistics. URL https://aclanthology.org/P09-1113. 78

[154] S. Mittal, S. C. Raparthy, I. Rish, Y. Bengio, and G. Lajoie. Compositional attention: Disentangling search and retrieval. In *International Conference on Learning Representations, ICLR*, 2022. 45

[155] M. Mosbach, M. Andriushchenko, and D. Klakow. On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=nzpLWnVAyah. 110, 119, 188

[156] S. Mysore, A. Cohan, and T. Hope. Multi-vector models with textual guidance for fine-grained scientific document similarity. In *NAACL*, 2022. 125, 132

[157] M. P. Naeini, G. F. Cooper, and M. Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In B. Bonet and S. Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2901–2907. AAAI Press, 2015. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9667. 122, 189

[158] S. Narang, H. W. Chung, Y. Tay, L. Fedus, T. Fevry, M. Matena, K. Malkan, N. Fiedel, N. Shazeer, Z. Lan, Y. Zhou, W. Li, N. Ding, J. Marcus, A. Roberts, and C. Raffel. Do transformer modifications transfer across implementations and applications? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5758–5773, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.465. URL https://aclanthology.org/2021.emnlp-main.465. 3, 4, 16, 125

[159] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar, 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1113. URL https://aclanthology.org/D14-1113. 3, 49, 79

[160] N. Ng, K. Yee, A. Baevski, M. Ott, M. Auli, and S. Edunov. Facebook FAIR's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 314–319, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5333. URL https://aclanthology.org/W19-5333. 45

[161] T. Niu and M. Bansal. Polite dialogue generation without parallel data. *Transactions of the Association for Computational Linguistics*, 6:373–389, 2018. doi: 10.1162/tacl_a_00027. URL https://aclanthology.org/Q18-1027. 107

[162] M. Ostendorff, T. Ruas, T. Blume, B. Gipp, and G. Rehm. Aspect-based document similarity for research papers. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6194–6206, Barcelona, Spain (Online), 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.545. URL https://aclanthology.org/2020.coling-main.545. 132

[163] M. Ostendorff, N. Rethmeier, I. Augenstein, B. Gipp, and G. Rehm. Neighborhood contrastive learning for scientific document representations with citation embeddings. *ArXiv preprint*, abs/2202.06671, 2022. URL https://arxiv.org/abs/2202.06671. 125, 128, 132

[164] D. Pace, A. Russo, and M. Shanahan. Learning diverse representations for fast adaptation to distribution shift. *ArXiv preprint*, abs/2006.07119, 2020. URL https://arxiv.org/abs/2006.07119. 138

[165] P. Papalampidi, K. Cao, and T. Kocisky. Towards coherent and consistent use of entities in narrative generation. *ArXiv preprint*, abs/2202.01709, 2022. URL https://arxiv.org/abs/2202.01709. 31, 45

[166] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040. 105

[167] V. Papyan, X. Han, and D. L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020. 115

[168] D. G. Parthiban, Y. Mao, and D. Inkpen. On the softmax bottleneck of recurrent language models. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13640–13647. AAAI Press, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17608. 4, 16, 30

[169] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html. 191

[170] D. Patel, P. Dangati, J.-Y. Lee, M. Boratko, and A. McCallum. Modeling label space interactions in multi-label classification using box embeddings. In *International Conference on Learning Representations*, 2022. 136

[171] R. Paul, H.-S. Chang, and A. McCallum. Multi-facet universal schema. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 909–919, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.77. URL https://aclanthology.org/2021.eacl-main.77. 10, 11, 69

[172] E. Pavlick, P. Rastogi, J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style 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 2: Short Papers)*, pages 425–430, Beijing, China, 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-2070. URL https://aclanthology.org/P15-2070. 57

[173] N. Peng, M. Ghazvininejad, J. May, and K. Knight. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-1505. URL https://aclanthology.org/W18-1505. 108

[174] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162. 10, 57, 94, 182

[175] H. Piwowar, J. Priem, V. Larivière, J. P. Alperin, L. Matthias, B. Norlander, A. Farley, J. West, and S. Haustein. The state of oa: a large-scale analysis of the prevalence and impact of open access articles. *PeerJ*, 6:e4375, 2018. 125

[176] K. Qin, C. Li, V. Pavlu, and J. Aslam. Adapting RNN sequence prediction model to multi-label set prediction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3181–3190, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1321. URL https://aclanthology.org/N19-1321. 55, 67

[177] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019. 1, 14, 16, 27, 92

[178] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html. 21

[179] S. Rajaee and M. T. Pilehvar. A cluster-based approach for improving isotropy in contextual embedding space. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 575–584, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.73. URL https://aclanthology.org/2021.acl-short.73. 180

[180] H. Rashkin, A. Celikyilmaz, Y. Choi, and J. Gao. PlotMachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.349. URL https://aclanthology.org/2020.emnlp-main.349. 108

[181] P. Ren, Z. Chen, J. Li, Z. Ren, J. Ma, and M. de Rijke. Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4806–4813. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33014806. URL https://doi.org/10.1609/aaai.v33i01.33014806. 17, 42

[182] S. H. Rezatofighi, R. Kaskman, F. T. Motlagh, Q. Shi, D. Cremers, L. Leal-Taixé, and I. Reid. Deep perm-set net: learn to predict sets with unknown permutation and cardinality using deep neural networks. *ArXiv preprint*, abs/1805.00613, 2018. URL https://arxiv.org/abs/1805.00613. 67

[183] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.442. URL https://aclanthology.org/2020.acl-main.442. 172

[184] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia, 2013. Association for Computational Linguistics. URL https://aclanthology.org/N13-1008. 67, 75

[185] M. Roemmele and A. S. Gordon. Creative help: A story writing assistant. In H. Schoenau-Fog, L. E. Bruni, S. Louchart, and S. Baceviciute, editors, *Interactive Storytelling - 8th International Conference on Interactive Digital Storytelling, ICIDS 2015, Copenhagen, Denmark, November 30 - December 4, 2015, Proceedings*, volume 9445, pages 81–92. Springer, 2015. doi: 10.1007/978-3-319-27036-4_8. URL https://doi.org/10.1007/978-3-319-27036-4_8. 90

[186] B. Roth and D. Klakow. Feature-based models for improving the quality of noisy training data for relation extraction. In Q. He, A. Iyengar, W. Nejdl, J. Pei, and R. Rastogi, editors, *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 1181–1184. ACM, 2013. doi: 10.1145/2505515.2507850. URL https://doi.org/10.1145/2505515.2507850. 79

[187] B. Roth, T. Barth, M. Wiegand, and D. Klakow. A survey of noise reduction methods for distant supervision. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, 2013. 74, 76, 79

[188] D. Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988. 121

[189] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology*, 2002. 88

[190] N. Savinov, J. Chung, M. Binkowski, E. Elsen, and A. van den Oord. Step-unrolled denoising autoencoders for text generation, 2021. 45

[191] A. See, P. J. Liu, and C. D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://aclanthology.org/P17-1099. 9, 36, 38, 39, 46, 63

[192] A. See, S. Roller, D. Kiela, and J. Weston. What makes a good conversation? how controllable attributes affect human judgments. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1702–1723, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1170. URL https://aclanthology.org/N19-1170. 103, 107

[193] T. Sellam, S. Yadlowsky, J. Wei, N. Saphra, A. D'Amour, T. Linzen, J. Bastings, I. Turc, J. Eisenstein, D. Das, I. Tenney, and E. Pavlick. The multiberts: BERT reproductions for robustness analysis. *ArXiv preprint*, abs/2106.16163, 2021. URL https://arxiv.org/abs/2106.16163. 119

[194] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162. 185

[195] I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In D. Schuurmans and M. P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 3776–3784. AAAI Press, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11957. 103

[196] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=B1ckMDqlg. 45

[197] R. Shu and H. Nakayama. Compressing word embeddings via deep compositional code learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=BJRZzFlRb. 66

[198] K. Shuster, J. Urbanek, A. Szlam, and J. Weston. Am i me or you? state-of-the-art dialogue models cannot maintain an identity, 2021. 31, 35, 45

[199] V. Shwartz, E. Santus, and D. Schlechtweg. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 65–75, Valencia, Spain, 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-1007. 73, 76, 77

[200] V. Shwartz, R. Rudinger, and O. Tafjord. "you are grounded!": Latent name artifacts in pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6850–6861, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.556. URL https://aclanthology.org/2020.emnlp-main.556. 35, 45

[201] M. Sigman and G. A. Cecchi. Global organization of the wordnet lexicon. *Proceedings of the National Academy of Sciences*, 99(3):1742–1747, 2002. 172

[202] S. P. Singh, A. Hug, A. Dieuleveut, and M. Jaggi. Context mover's distance & barycenters: Optimal transport of contexts for building representations. In S. Chiappa and R. Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 3437–3449. PMLR, 2020. URL http://proceedings.mlr.press/v108/singh20a.html. 49

[203] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. Hsu, and K. Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246, 2015. 130

[204] A. Srivastava and C. Sutton. Autoencoding variational inference for topic models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=BybtVK9lg. 66

[205] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 111

[206] M. Stern, W. Chan, J. Kiros, and J. Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5976–5985. PMLR, 2019. URL http://proceedings.mlr.press/v97/stern19a.html. 67

[207] R. Stewart, M. Andriluka, and A. Y. Ng. End-to-end people detection in crowded scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2325–2333. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.255. URL https://doi.org/10.1109/CVPR.2016.255. 66

[208] Y. Su, H. Liu, S. Yavuz, I. Gür, H. Sun, and X. Yan. Global relation embedding for relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 820–830, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1075. URL https://aclanthology.org/N18-1075. 79

[209] Y. Su, T. Lan, Y. Wang, D. Yogatama, L. Kong, and N. Collier. A contrastive framework for neural text generation. *ArXiv preprint*, abs/2202.06417, 2022. URL https://arxiv.org/abs/2202.06417. 180

[210] Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, and H. Wang. ERNIE 2.0: A continual pre-training framework for language understanding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8968–8975. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6428. 113, 118, 120

[211] M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465, Jeju Island, Korea, 2012. Association for Computational Linguistics. URL https://aclanthology.org/D12-1042. 79

[212] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html. 56

[213] S. Takamatsu, I. Sato, and H. Nakagawa. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 721–729, Jeju Island, Korea, 2012. Association for Computational Linguistics. URL https://aclanthology.org/P12-1076. 79

[214] S. Takase, J. Suzuki, and M. Nagata. Direct output connection for a high-rank language model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4599–4609, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1489. URL https://aclanthology.org/D18-1489. 28, 29

[215] P. Tambwekar, M. Dhuliawala, L. J. Martin, A. Mehta, B. Harrison, and M. O. Riedl. Controllable neural story plot generation via reward shaping. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5982–5988. ijcai.org, 2019. doi: 10.24963/ijcai.2019/829. URL https://doi.org/10.24963/ijcai.2019/829. 107

[216] B. Tan, Z. Yang, M. Al-Shedivat, E. Xing, and Z. Hu. Progressive generation of long text with pretrained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4313–4324, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.341. URL https://aclanthology.org/2021.naacl-main.341. 45, 108

[217] Y. Tay, M. Dehghani, V. Aribandi, J. P. Gupta, P. Pham, Z. Qin, D. Bahri, D. Juan, and D. Metzler. Omninet: Omnidirectional representations from transformers. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 10193–10202. PMLR, 2021. URL http://proceedings.mlr.press/v139/tay21b.html. 28

[218] Y. Tay, M. Dehghani, S. Abnar, H. W. Chung, W. Fedus, J. Rao, S. Narang, V. Q. Tran, D. Yogatama, and D. Metzler. Scaling laws vs model architectures: How does inductive bias influence scaling?, 2022. URL https://arxiv.org/abs/2207.10551. 3, 4, 16, 125

[219] D. Teney, E. Abbasnejad, S. Lucey, and A. van den Hengel. Evading the simplicity bias: Training a diverse set of models discovers solutions with superior ood generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16761–16772, 2022. 138

[220] I. Tenney, P. Xia, B. Chen, A. Wang, A. Poliak, R. T. McCoy, N. Kim, B. V. Durme, S. R. Bowman, D. Das, and E. Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SJzSgnRcKX. 28

[221] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012. 55, 185

[222] L. Tu, X. Ding, D. Yu, and K. Gimpel. Generating diverse story continuations with controllable semantics. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 44–58, Hong Kong, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5605. URL https://aclanthology.org/D19-5605. 93, 101, 107

[223] P. D. Turney. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 2012. 64

[224] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html. 55, 56, 94

[225] P. Verga, D. Belanger, E. Strubell, B. Roth, and A. McCallum. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 886–896, San Diego, California, 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1103. URL https://aclanthology.org/N16-1103. xvi, 8, 67, 70, 73, 74, 75, 76, 79

[226] L. Vilnis and A. McCallum. Word representations via gaussian embedding. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6623. 66

[227] N. Walton. AI dungeon, 2020. URL https://play.aidungeon.io/. 90

[228] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/4496bf24afe7fab6f046bf4923da8de6-Abstract.html. 11, 112, 118

[229] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7. 11, 112, 118

[230] B. Wang and A. Komatsuzaki. Gpt-j-6b: A 6 billion parameter autoregressive language model, 2021. 22

[231] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In C. Apté, J. Ghosh, and P. Smyth, editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 448–456. ACM, 2011. doi: 10.1145/2020408.2020480. URL https://doi.org/10.1145/2020408.2020480. 132

[232] H. Wang, B. Chen, and W.-J. Li. Collaborative topic regression with social regularization for tag recommendation. In *IJCAI*, 2013. 137

[233] S. Wang, L. Hu, Y. Wang, Q. Z. Sheng, M. A. Orgun, and L. Cao. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 3771–3777. ijcai.org, 2019. doi: 10.24963/ijcai.2019/523. URL https://doi.org/10.24963/ijcai.2019/523. 3

[234] T. Wang, K. Cho, and M. Wen. Attention-based mixture density recurrent networks for history-based recommendation. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, 2019. 3, 67, 89

[235] Y. Wang, L. Cui, and Y. Zhang. How can bert help lexical semantics tasks? *ArXiv preprint*, abs/1911.02929, 2019. URL https://arxiv.org/abs/1911.02929. 15, 19

[236] S. Welleck, Z. Yao, Y. Gai, J. Mao, Z. Zhang, and K. Cho. Loss functions for multiset prediction. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5788–5797, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/fb3f76858cb38e5b7fd113e0bc1c0721-Abstract.html. 67

[237] S. Welleck, K. Brantley, H. D. III, and K. Cho. Non-monotonic sequential text generation. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6716–6726. PMLR, 2019. URL http://proceedings.mlr.press/v97/welleck19a.html. 67

[238] Y. Wen, D. Tran, and J. Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Sklf1yrYDr. 124

[239] P. West, C. Bhagavatula, J. Hessel, J. D. Hwang, L. Jiang, R. L. Bras, X. Lu, S. Welleck, and Y. Choi. Symbolic knowledge distillation: from general language models to commonsense models. *ArXiv preprint*, abs/2110.07178, 2021. URL https://arxiv.org/abs/2110.07178. 14

[240] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6. 191

[241] Q. Wu, L. Li, and Z. Yu. Textgail: Generative adversarial imitation learning for text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14067–14075, 2021. 45

[242] Z. Wu, M. Galley, C. Brockett, Y. Zhang, X. Gao, C. Quirk, R. Koncel-Kedziorski, J. Gao, H. Hajishirzi, M. Ostendorf, et al. A controllable model of grounded response generation. *ArXiv preprint*, abs/2005.00613, 2020. URL https://arxiv.org/abs/2005.00613. 108

[243] J. Xu, X. Ren, Y. Zhang, Q. Zeng, X. Cai, and X. Sun. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4306–4315, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1462. URL https://aclanthology.org/D18-1462. 108

[244] Y. Xu, X. Qiu, L. Zhou, and X. Huang. Improving bert fine-tuning via self-ensemble and self-distillation. *ArXiv preprint*, abs/2002.10345, 2020. URL https://arxiv.org/abs/2002.10345. 110, 124

[245] R. Yan. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In S. Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2238–2244. IJCAI/AAAI Press, 2016. URL http://www.ijcai.org/Abstract/16/319. 107

[246] L. Yang, Y. Guo, and X. Cao. Multi-facet network embedding: Beyond the general solution of detection and representation. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 499–506. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16326. 3, 89

[247] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 206–215. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00029. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Yang_FoldingNet_Point_Cloud_CVPR_2018_paper.html. 66

[248] Z. Yang, Z. Dai, R. Salakhutdinov, and W. W. Cohen. Breaking the softmax bottleneck: A high-rank RNN language model. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=HkwZSG-CZ. 3, 4, 5, 9, 16, 18, 24, 25, 28, 29, 44, 46, 67, 125, 173, 179

[249] Z. Yang, T. Luong, R. Salakhutdinov, and Q. V. Le. Mixtape: Breaking the softmax bottleneck efficiently. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15922–15930, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/512fc3c5227f637e41437c999a2d3169-Abstract.html. 4, 180

[250] L. Yao, S. Riedel, and A. McCallum. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1013–1023, Cambridge, MA, 2010. Association for Computational Linguistics. URL https://aclanthology.org/D10-1099. 79

[251] L. Yao, N. Peng, R. M. Weischedel, K. Knight, D. Zhao, and R. Yan. Plan-and-write: Towards better automatic storytelling. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7378–7385. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33017378. URL https://doi.org/10.1609/aaai.v33i01.33017378. 108

[252] M. Yu and M. Dredze. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242, 2015. doi: 10.1162/tacl_a_00135. URL https://aclanthology.org/Q15-1017. xvi, 65

[253] A. Zagoury, E. Minkov, I. Szpektor, and W. W. Cohen. What's the best place for an AI conference, vancouver or _____: Why completing comparative questions is difficult. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 14292–14300. AAAI Press, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17681. 21, 180

[254] D. Zhang, S. Zhao, Z. Duan, J. Chen, Y. Zhang, and J. Tang. A multi-label classification method using a hierarchical and transparent representation for paper-reviewer recommendation. *ACM Transactions on Information Systems (TOIS)*, 38(1):1–20, 2020. 88, 132

[255] T. Zhang, F. Wu, A. Katiyar, K. Q. Weinberger, and Y. Artzi. Revisiting few-sample BERT fine-tuning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=cO1IH43yUF. 110, 118, 119, 188

[256] Y. Zhang, G. Wang, C. Li, Z. Gan, C. Brockett, and B. Dolan. POINTER: Constrained progressive text generation via insertion-based generative pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8649–8670, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.698. URL https://aclanthology.org/2020.emnlp-main.698. 108

[257] Z. Zhang, V. R. Gao, and M. R. Sabuncu. Ex uno plures: Splitting one model into an ensemble of subnetworks. *ArXiv preprint*, abs/2106.04767, 2021. URL https://arxiv.org/abs/2106.04767. 124

[258] Z. Zhang, N. Shao, C. Gao, R. Miao, Q. Yang, and J. Shao. Mixhead: Breaking the low-rank bottleneck in multi-head attention language models. *Knowledge-Based Systems*, page 108075, 2022. 45

[259] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, Y. Chen, X. Pan, K. Li, Y. Lu, H. Wang, C. Tian, et al. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 4653–4664, 2021. 42

[260] H. Zheng and M. Lapata. Sentence centrality revisited for unsupervised summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6236–6247, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1628. URL https://aclanthology.org/P19-1628. xvi, 63, 64

[261] M. Zhu. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2:30, 2004. 87

[262] Y. Zhu, R. Kiros, R. S. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.11. URL https://doi.org/10.1109/ICCV.2015.11. 118

[263] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu. Texygen: A benchmarking platform for text generation models. In K. Collins-Thompson, Q. Mei, B. D. Davison, Y. Liu, and E. Yilmaz, editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1097–1100. ACM, 2018. doi: 10.1145/3209978.3210080. URL https://doi.org/10.1145/3209978.3210080. 106

[264] S. Zuo, Q. Zhang, C. Liang, P. He, T. Zhao, and W. Chen. Moebert: from bert to mixture-of-experts via importance-guided adaptation. *ArXiv preprint*, abs/2204.07675, 2022. URL https://arxiv.org/abs/2204.07675. 124

# APPENDIX A

# APPENDIX FOR CHAPTER 2

## A.1 Appendix Overview

To demonstrate the wide applicability of our approaches, we conduct more experiments in Appendix A.2. Next, we provide the proof of our theorems in Appendix A.3, and show that the softmax weakness studied by Demeter et al. [45] is a special case of our theory in Appendix A.4. Finally, we list some future work of Chapter 2 in Appendix A.5.

## A.2 More Experiments

We conduct the following synthetic experiments to analyze the pros and cons of multi-facet softmax and confirm the source of the improvement comes from modeling multimodal distribution.

### A.2.1 Evaluation on Ambiguous Templates

We synthesize a dataset using templates [183] to verify whether the softmax layer in the original GPT-2 really has difficulty in learning to output the bimodal distribution in Figure 2.1 and whether the multiple embedding methods could overcome the problem. First, we collect the four words with semantic analogy relations in Google analogy dataset [148]. Next, we insert two out of the four words into our manually written templates to form the contexts. For example, given the context "*I went to Paris and Germany before, and I love one of the places more, which is*", the GPT-2 learns to predict either *Paris* or *Germany*. The prediction of **MFS** and the **softmax** baseline for this example is compared in the last column of Table 2.3.

The two words can be either the diagonal words (e.g., *king* and *woman*) or the edge word (e.g., *king* and *queen*) in the parallelogram. Finally, we create a dataset with 122k training contexts, 250k validation contexts, and 122k testing contexts, where the word pairs in the testing set are unseen in the training set to see whether the model could learn to output the bimodal distribution in a general way.[1]

[1]The setting is realistic because any related words could become the next word in some ambiguous contexts and all the words are related in a certain way [201]. We cannot expect the training corpora to contain the ambiguous contexts with so many possible next words.

Table A.1: Perplexity comparison of different *GPT-2 Small* models on the words with different types of analogy relations. The validation set (valid) includes all four types of relations.

| Analogy Relation Types → Models ↓ | Diagonal (e.g., *king* or *woman*) | | | | | Edge (e.g., *king* or *queen*) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | valid | capital-common | capital-world | city-in-state | family | valid | capital-common | capital-world | city-in-state | family |
| Softmax (GPT-2) | 2.30 | 3.30 | 2.00 | 2.25 | 2.95 | 2.11 | 2.42 | 1.91 | 2.26 | 2.38 |
| MoS [248] (3) | 1.75 | 2.18 | 1.60 | 1.85 | 2.82 | 1.87 | 2.26 | **1.70** | 2.04 | 2.27 |
| MFS w/o Multi-partition | **1.72** | **2.13** | **1.59** | **1.82** | **2.52** | **1.84** | **2.23** | 1.72 | **1.96** | **2.16** |
| MFS (Ours) | 1.74 | 2.15 | **1.59** | **1.82** | 2.63 | 1.92 | 2.28 | 1.78 | 2.00 | 2.24 |

We load the models pre-trained on OpenWebText and continue fine-tuning the models on the last word of each sentence for 10 epochs. We report the testing performance of the best model selected by the validation loss. Since the sets of the word pairs in the training and testing set are disjoint, updating the output word embedding would make GPT-2 solve the task by memorizing/overfitting the training set quickly and lead to much worse testing performance. Thus, we freeze the output word embedding during the training.

Table A.1 indicates that when the possible next words are the diagonal words, the **Softmax** model performs much worse compared to other multiple embedding alternatives. In the edge word dataset, the multiple embedding solutions are still better but have a much smaller gap. **MFS w/o Multi-partition** slightly improves **MoS**. We hypothesize the reason is that multiple input hidden states could help the facets to be moved more freely. Finally, multiple partitions seem to cause slight overfitting in this bimodal distribution prediction task.

### A.2.2 Adversarial Template Analysis

To test whether the proposed methods still can effectively utilize the information from the global word embeddings, we design an adversarial template to create the contexts that can only be completed by averaging the global word embeddings. For example, *"**Miami** is not in **Wisconsin** but is in [MASK]=**Florida**"*.

In this task, the validation perplexity of **Softmax**, **MoS**, **MFS w/o Multi-partition**, and **MFS** are 2.50, 2.59, 2.54, and 2.88, respectively. Since multiple embeddings are not required, it is not surprising that **Softmax** performs the best. Nevertheless, the differences are smaller than the differences in Table A.1. We believe that the similar losses are due to the fact that multiple embeddings are a generalization of the single embedding, so GPT-2 could learn to generate the same embedding for all facets to mimic the behavior of single embedding if required.

The significantly worse performance of **MFS** here is caused by the multiple partition technique. This result supports our motivation of combining multiple partitions with multiple softmaxes and shows that multiple partitions handle ambiguous contexts better (as shown in Table A.3) by sacrificing some global word embedding structures. Nevertheless, a corpus usually has more ambiguous contexts than the adversarial context tested here, so using multiple embeddings and multiple partitions performs better in Wikipedia and OpenWebText overall.

Table A.2: Prediction visualization using a context in each dataset. Each row visualizes a model as in Table 2.3. The models are built on *GPT-2 Medium* in OpenWebText and Wikipedia and on *GPT-2 Small* in the synthesized dataset. *MFS Avg* shows the words that are closest to the average facet embedding in *MFS*. See the details in Appendix A.2.3. We underline the words that appear in the top predictions of both *MFS* and *MFS Avg*.

| Corpus → | OpenWebText | Wikipedia 2021 | Similar Nouns in Templates |
|---|---|---|---|
| Input Context | ... "Part of the Clinton inevitability strategy was to lock down the usual suspects in left-liberal policy," said Dan Nexon, a Georgetown professor who served as one of those informal Sanders advisers. **Nex** | ... The projective line over the dual numbers was described by Josef Grünwald in 1906. This ring includes a nonzero nilpotent "n" satisfying. The plane of dual numbers has a **project** | There are the militia and the enemy in front of a woman, and she decides to pursue the **militia** |
| Softmax (GPT-2) | He 0.014, But 0.011, The 0.007 | finite 0.062, hom 0.059, **project** 0.034 | enemy 0.860, **militia** 0.111, Militia 0.005 |
| MFS (Ours) | **Nex** 0.013, He 0.012, But 0.011 | **project** 0.096, hom 0.049, dual 0.046 | enemy 0.535, **militia** 0.433, enemies 0.029 |
| MFS Avg | ", He, But, The, In, And, (, It | hom, dual, finite, non, ", complex, unit | **militia**, enemy, Militia, enemies, militias |
| MFS Softmax 1 | But 0.005, He 0.004, The 0.002 | **project** 0.201, dual 0.075, finite 0.030 | enemy 0.772, **militia** 0.189, Militia 0.017 |
| MFS Softmax 2 | **Nex** 0.260, " 0.028, He 0.023 | hom 0.093, unit 0.040, non 0.037 | **militia** 0.938, Militia 0.062, militias 0.000 |
| MFS Softmax 3 | He 0.025, But 0.022, The 0.014 | finite 0.065, map 0.041, plane 0.030 | enemy 1.000, enemies 0.000, foe 0.003 |

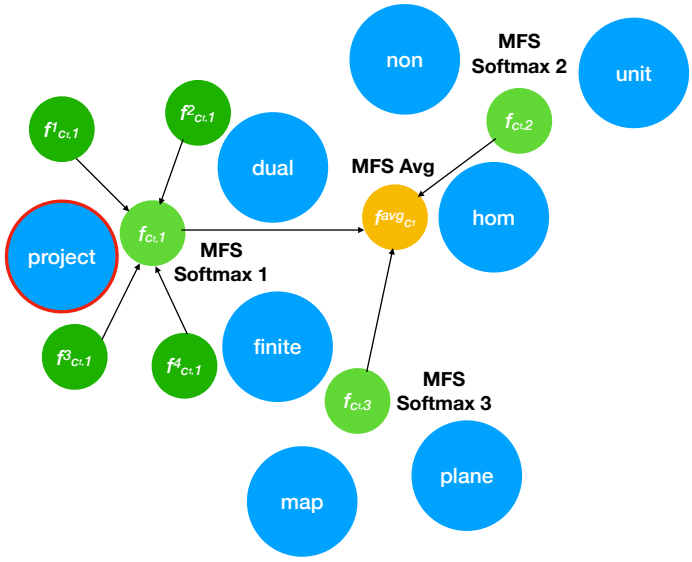



Figure A.1: Illustration of the MFS predictions given the Wikipedia context in the second column of Table A.2. The green circles mean the facet embeddings from MFS. The orange circle is the average of the facet embeddings (**MFS Avg**). The blue circles are the word embeddings that are close to the facet embeddings and **MFS Avg**. The word *project* is highlighted because it is the next word in our ground truth.

Table A.3: The loss improvement comparison between the *Improvement Models* and *Reference Models*. The models are named using their number of softmaxes, input hidden states, and partitions. Thus, S3I9P4 is *MFS*, S3I9P1 is *MFS w/o Multi-partition*, S1I9P1 is *Softmax + Multi-input*, S3I1P1 is *MoS (3)*, and S1I1P1 is *Softmax*. *Multi-mode Percentage* is the percentage of the contexts where the *Improvement Models* output multimodal distribution. *Multi-mode Loss Improvement* refers to the average improvement when *Improvement Models* outputs multimodal distribution and *Other Loss Improvement* refers to the improvement of the contexts where the facets of *Improvement Models* are close to each other. *Improvement Ratio* divides *Multi-mode Loss Improvement* by *Other Loss Improvement*.

| Corpus → | OpenWebText | | | | | Wikipedia 2021 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Improvement Model | S3I9P4 | S3I9P4 | S3I9P4 | S3I9P1 | S3I1P1 | S3I9P4 | S3I9P4 | S3I9P4 | S3I9P1 | S3I1P1 |
| Reference Model | S3I9P1 | S3I1P1 | S1I1P1 | S1I9P1 | S1I1P1 | S3I9P1 | S3I1P1 | S1I1P1 | S1I9P1 | S1I1P1 |
| Multi-mode Percentage (%) | 10.03 | 10.03 | 10.03 | 4.81 | 3.24 | 5.85 | 5.85 | 5.85 | 2.66 | 3.05 |
| Multi-mode Loss Improvement | 0.0248 | 0.0474 | 0.0649 | 0.0203 | 0.0110 | 0.0282 | 0.0644 | 0.1000 | 0.0472 | 0.0295 |
| Other Loss Improvement | 0.0035 | 0.0158 | 0.0211 | 0.0086 | 0.0064 | 0.0033 | 0.0128 | 0.0219 | 0.0136 | 0.0100 |
| Improvement Ratio | 7.01 | 3.00 | 3.08 | 2.34 | 1.71 | 8.63 | 5.04 | 4.57 | 3.47 | 2.94 |

### A.2.3 Analysis of Improvement on Multimodal Distribution

To confirm that the perplexity improvements actually come from modeling the multimodal distribution, we define a metric to measure how multi-mode a distribution is, and then we can compare the perplexity improvement from multimodal distributions and the improvement from the distributions that are close to a single-mode distribution.

For the method with multiple embeddings, we first compute the weighted average of all the facets $\boldsymbol{f}_{c_t}^{avg} = \sum_{k=1}^{K} \pi_{c_t,k} \boldsymbol{f}_{c_t,k}$, where we lower the influence of $k$th facet embedding $\boldsymbol{f}_{c_t,k}$ with lower prior weight $\pi_{c_t,k}$ and $\boldsymbol{f}_{c_t,1} = \frac{1}{J} \sum_{j=1}^{J} \boldsymbol{f}_{c_t,1}^{j}$ if $J$ partitions are used. Figure A.1 illustrates $\boldsymbol{f}_{c_t}^{avg}$ and $\boldsymbol{f}_{c_t,k}$ using the example in the second column of Table A.2.

We visualize the new average facet using the words that are closest to the $\boldsymbol{f}_{c_t}^{avg}$ in the **MFS Avg** row of Table A.2. We can see that the prediction of **MFS Avg** is different from **MFS** but similar to **Softmax**. This means there are indeed some other words between the actual next word and the other possibilities, which makes the prediction of **MFS** multi-mode.

Next, to quantify the difference between **MFS** and **MFS Avg**, we define *multi-mode ratio* as $\frac{\sum_{b=1}^{T} P_M(y_b|c_t)}{\sum_{b=1}^{T} P_M(x_b|c_t)}$, where $P_M$ could be either $P_{MoS}$ from equation 2.2 or $P_{MP}$ from equation 2.4. $\{y_1, ..., y_T\}$ is the set of words with embeddings closest to $\boldsymbol{f}_{c_t}^{avg}$ and $\{x_1, ..., x_T\}$ is the set of words with highest $P_M(x_b|c_t)$. Using the Wikipedia context in Table A.2 as an example, the word *project* is retrieved by **MFS** but not by **MFS Avg**, so its *multi-mode ratio* for $T = 2$ is $\frac{P_{MFS}(hom|c_t)+P_{MFS}(dual|c_t)}{P_{MFS}(project|c_t)+P_{MFS}(hom|c_t)} = \frac{0.049+0.046}{0.096+0.049} \approx 0.66$. Figure A.1 illustrates the relation between the **MFS Softmax k** and **MFS Avg**.

When the ratio is closer to 1, the context is less ambiguous and the prediction is closer to a single-mode distribution. We set $T = 20$ and call the prediction with *multi-mode ratio* smaller than $0.9$ multimodal distribution and in Table A.3,[2] we compare the loss (i.e., log of the perplexity) improvements in the multimodal distributions and the improvements in the nearly single-mode distributions.

Table A.3 shows that all the multiple embedding approaches have larger loss improvements when outputting multimodal distributions. The table shows the results based on *GPT-2 Small* and the same analysis using *GPT-2 Medium* also show the same trend. As we use multiple input hidden states and partitions, the differences would be enlarged. Especially when we compare **MFS** and **MFS w/o Multi-partition**, the loss improvements of highly ambiguous context is 7 or 8 times larger than the other loss improvements, which means a large portion of the overall improvement lies on a small percentage of ambiguous contexts. For the multimodal distribution in Wikipedia, the loss improvement between **MFS** and **Softmax** could reach 0.10, which is close to the improvement between GPT-2 Small and Medium (0.16). Thus, we expect that if the corpus has more ambiguous contexts, **MFS** could achieve larger overall loss improvement.

## A.3 Proof of Theorems

To prove Theorem 1, we first introduce a lemma. Assuming in the word embedding of GPT-2, ***woman*** + ***king*** = ***queen*** + ***man***, we want to show that GPT-2 cannot output *woman* and *king* as the top two words in this lemma. This means we cannot find a hidden state $\boldsymbol{h}$ and a threshold $\tau > 0$ such that $\boldsymbol{h}^T\boldsymbol{w}_{\textit{woman}} \geq \tau$ and $\boldsymbol{h}^T\boldsymbol{w}_{\textit{king}} \geq \tau$ but $\boldsymbol{h}^T\boldsymbol{w}_{\textit{queen}} < \tau$ and $\boldsymbol{h}^T\boldsymbol{w}_{\textit{man}} < \tau$. This example could be generalized into the following Lemma and Theorems. We can generalize the example as follows:

**Lemma 1.** *Let the output word embeddings in the set* $W = \{\boldsymbol{w}_{l_j} \neq \boldsymbol{0} | j = 1...L\} \cup \{\boldsymbol{w}_{r_j} \neq \boldsymbol{0} | j = 1...R\}$ *satisfy* $-a_{l_1}\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{w}_{l_L} = a_{r_1}\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{w}_{r_R}$*, where their coefficient* $-a_{l_1}, ..., -a_{l_L}, a_{r_1}, ..., a_{r_R}$ *are all positive constants and* $-a_{l_1} - ... - a_{l_L} \geq a_{r_1} + ... + a_{r_R}$*. Then, there is no hidden state* $\boldsymbol{h}$ *and a threshold* $\tau > 0$ *that make* $\min_{\boldsymbol{w}_g \in G} \boldsymbol{h}^T\boldsymbol{w}_g \geq \tau$ *and* $\max_{\boldsymbol{w}_s \in S} \boldsymbol{h}^T\boldsymbol{w}_s < \tau$*, where* $G = \{\boldsymbol{w}_{l_j} | j = 1...L\}$ *and* $S = \{\boldsymbol{w}_{r_j} | j = 1...R\}$.

*Proof.* To prove by contradiction, we assume there is a $\boldsymbol{h}$ such that $\forall \boldsymbol{w}_{l_j} \in G, \boldsymbol{h}^T\boldsymbol{w}_{l_j} \geq \tau$ and $\forall \boldsymbol{w}_{r_j} \in S, \boldsymbol{h}^T\boldsymbol{w}_{r_j} < \tau$. Thus, we can get $-a_{l_1}\boldsymbol{h}^T\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{h}^T\boldsymbol{w}_{l_L} \geq -a_{l_1}\tau - ... - a_{l_L}\tau \geq (a_{r_1} + ... + a_{r_R})\tau > a_{r_1}\boldsymbol{h}^T\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{h}^T\boldsymbol{w}_{r_R}$, which contradicts to $-a_{l_1}\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{w}_{l_L} = a_{r_1}\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{w}_{r_R}$. □

We can rephrase the condition and the conclusion to have our Theorem 1.

[2] We also tried T=5 or 10 and the trends are similar. If we set the threshold smaller than $0.9$, the improvement ratios (e.g., MFS over MoS) would increase but the multi-mode percentages would decrease.

**Theorem 3.** *If the nonzero output embeddings of $N$ words in a set $W$ are linearly dependent and on one side of a plane through the origin, the single embedding representation cannot produce positive logits to a subset of the word in $W$ that are higher than all the logits of the other words in $W$.*

*Proof.* Let the set $W = \{\boldsymbol{w}_i \neq \mathbf{0} | i = 1...N\}$ contain the embeddings of the $N$ words. Based on the premise, we can write $\mathbf{0} = a_1\boldsymbol{w}_1 + ... + a_N\boldsymbol{w}_N$ and $\min_{\boldsymbol{w}_i \in W} \boldsymbol{h}_0^T \boldsymbol{w}_i > 0$, where $\boldsymbol{h}_0$ is a normal vector of the plane. At least one of the $a_i$ is negative. Otherwise, we will get the contradiction $0 = \boldsymbol{h}_0^T\mathbf{0} = a_1\boldsymbol{h}_0^T\boldsymbol{w}_1 + ... + a_N\boldsymbol{h}_0^T\boldsymbol{w}_N \geq (a_1 + ... + a_N)\min_{\boldsymbol{w}_i \in W} \boldsymbol{h}_0^T\boldsymbol{w}_i > 0$. Similarly, at least one of $a_i$ is positive. We can move all the terms in $\mathbf{0} = a_1\boldsymbol{w}_1 + ... + a_N\boldsymbol{w}_N$ with negative $a_i$ to the left as $-a_{l_1}\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{w}_{l_L} = a_{r_1}\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{w}_{r_R}$. If $-a_{l_1} - ... - a_{l_L} \geq a_{r_1} + ... + a_{r_R}$, we choose $G = \{\boldsymbol{w}_{l_j} | j = 1...L\}$. Otherwise, we choose $G = \{\boldsymbol{w}_{r_j} | j = 1...R\}$

If we can have a hidden state such that the positive logits of words in $G$ are always larger than the logits of the other words in $W$ (let's call the complementary set $S$), there must exist $\tau > 0$ that can make $\min_{\boldsymbol{w}_g \in G} \boldsymbol{h}^T\boldsymbol{w}_g \geq \tau$ and $\max_{\boldsymbol{w}_s \in S} \boldsymbol{h}^T\boldsymbol{w}_s < \tau$, which violates our Lemma 1. □

Notice that Theorem 3 does not cover the situations where the target top words have the negative logits (i.e., some logits in $G$ are negative). In the single softmax model, we believe the situations rarely happen in the LMs empirically. If some logits of the target top words are still positive, the words that are somehow similar to those words are very likely to also be positive, which would be ranked higher than the target top words with the negative logits. If the logits of all the target top words are negative, the logits of all the words would be negative. Then, the word embeddings with smaller magnitudes tend to have the logits closer to 0, so having the larger logits than the other negative logits. This means the prior probability of the words would be inversed when the hidden states sometimes produce the all negative logits. If LM always use negative logits to compute probability, Lemma 1 and Theorem 3 still hold if we set $\tau < 0$ and switch the choices of $G$ and $S$.

Next, we would like to generalize our Theorem 1 by using a more practical condition where the word embeddings are almost linearly dependent. Notice that the theorem needs to assume the magnitude of the hidden state is limited. Otherwise, the margin could be arbitrarily magnified. In practice, the magnitude is not arbitrarily large in GPT-2 and BERT because a too large magnitude of hidden state could magnify the gradients too much to have a stable training process.

**Theorem 4.** *Let the output word embeddings in the set $W = \{\boldsymbol{w}_i \neq \mathbf{0} | i = 1...N\}$ satisfy $\boldsymbol{w}_1 = a_2\boldsymbol{w}_2 + ... + a_N\boldsymbol{w}_N + \boldsymbol{\varepsilon}$, where the constant $a_2, ..., a_N$ are neither all zero nor all negative and $||\boldsymbol{\varepsilon}|| < \epsilon$. Then, there must be a non-trivial partition $P = \{G, S\}$ of $W$ such that there is no hidden state $||\boldsymbol{h}|| \leq r$ and a threshold $\tau \geq r\epsilon$ that make $\min_{\boldsymbol{w}_g \in G} \boldsymbol{h}^T\boldsymbol{w}_g \geq (1 + \delta)\tau$ and $\max_{\boldsymbol{w}_s \in S} \boldsymbol{h}^T\boldsymbol{w}_s < \tau$, where $\delta = \frac{2}{1 + \sum_{i=2...N} |a_i|}$.*

*Proof.* We can first move all the terms with negative $a_i$ to the left as $\boldsymbol{w}_1 - a_{l_1}\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{w}_{l_L} = a_{r_1}\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{w}_{r_R} + \boldsymbol{\varepsilon}$. We perform proof by contradiction, so we assume the logits of the words in $G$ can always be larger than $(1 + \delta)\tau$ and the logits of the words in $S$ can always be smaller than $\tau$.

Case 1: $1 - a_{l_1} - ... - a_{l_L} \geq a_{r_1} + ... + a_{r_R}$, so $1 - a_{l_1} - ... - a_{l_L} \geq \frac{1+\sum_{i=2...N}|a_i|}{2}$. We choose $G = \{\boldsymbol{w}_1, \boldsymbol{w}_{l_1}, ..., \boldsymbol{w}_{l_L}\}$ and $S = \{\boldsymbol{w}_{r_1}, ..., \boldsymbol{w}_{r_R}\}$. Thus, we can get $\boldsymbol{h}^T\boldsymbol{\varepsilon} \leq ||\boldsymbol{h}||||\boldsymbol{\varepsilon}|| \leq r\epsilon \leq \tau$ and

$$\begin{aligned}
&\boldsymbol{h}^T\boldsymbol{w}_1 - a_{l_1}\boldsymbol{h}^T\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{h}^T\boldsymbol{w}_{l_L} && \text{(A.1)}\\
\geq&(1 - a_{l_1} - ... - a_{l_L})(1+\delta)\tau && \text{(A.2)}\\
=&(1 - a_{l_1} - ... - a_{l_L})(1 + \frac{2}{1+\sum_{i=2...N}|a_i|})\tau && \text{(A.3)}\\
\geq&(1 - a_{l_1} - ... - a_{l_L})(1 + \frac{1}{1 - a_{l_1} - ... - a_{l_L}})\tau && \text{(A.4)}\\
=&(1 - a_{l_1} - ... - a_{l_L} + 1)\tau && \text{(A.5)}\\
\geq&(a_{r_1} + ... + a_{r_R} + 1)\tau && \text{(A.6)}\\
>&a_{r_1}\boldsymbol{h}^T\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{h}^T\boldsymbol{w}_{r_R} + \boldsymbol{h}^T\boldsymbol{\varepsilon}, && \text{(A.7)}
\end{aligned}$$

which contradict with $\boldsymbol{w}_1 - a_{l_1}\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{w}_{l_L} = a_{r_1}\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{w}_{r_R} + \boldsymbol{\varepsilon}$.

Case 2: $1 - a_{l_1} - ... - a_{l_L} < a_{r_1} + ... + a_{r_R}$. We choose $G = \{\boldsymbol{w}_{r_1}, ..., \boldsymbol{w}_{r_R}\}$ and $S = \{\boldsymbol{w}_1, \boldsymbol{w}_{l_1}, ..., \boldsymbol{w}_{l_L}\}$. Therefore,

$$\begin{aligned}
&a_{r_1}\boldsymbol{h}^T\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{h}^T\boldsymbol{w}_{r_R} && \text{(A.8)}\\
\geq&(a_{r_1} + ... + a_{r_R})(1 + \frac{2}{1+\sum_{i=2...N}|a_i|})\tau && \text{(A.9)}\\
>&(a_{r_1} + ... + a_{r_R})(1 + \frac{1}{a_{r_1} + ... + a_{r_R}})\tau && \text{(A.10)}\\
=&(a_{r_1} + ... + a_{r_R} + 1)\tau && \text{(A.11)}\\
>&(1 - a_{l_1} - ... - a_{l_L} + 1)\tau && \text{(A.12)}\\
>&\boldsymbol{h}^T\boldsymbol{w}_1 - a_{l_1}\boldsymbol{h}^T\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{h}^T\boldsymbol{w}_{l_L} - \boldsymbol{h}^T\boldsymbol{\varepsilon}. && \text{(A.13)}
\end{aligned}$$

□

## A.4 Connection to Demeter et al. [45]

The theory in Demeter et al. [45] is as follows: "*Let $C$ be the convex hull of the embeddings $\{\boldsymbol{x}_i\}$ of a vocabulary $V$. If an embedding $\boldsymbol{x}_i$ for word $\boldsymbol{w}_i \in V$ is interior to $C$, then the maximum probability $P(\boldsymbol{w}_i)$ assigned to $\boldsymbol{w}_i$ using a dot-product softmax is bounded by the probability assigned to at least one word $\boldsymbol{w}_i$ whose embedding is on the convex hull*"

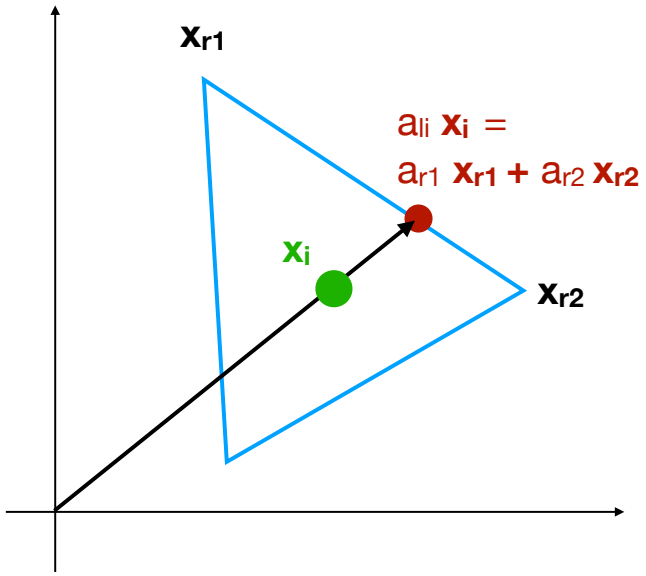



Figure A.2: An example for explaining the connection between our Theorem 1 and the theorem from Demeter et al. [45].

The theory is a special case of our Lemma 1 if we only consider the hidden states that would lead to the positive logit of the interior word $\boldsymbol{w}_i$. To see that, we first find a constant $a_{l_i} > 1$ such that $a_{l_i}\boldsymbol{x}_i$ intersects with one supporting hyperplane of the convex hull. This intersection point could be expressed by $\sum_j a_{r_j}\boldsymbol{x}_{r_j}$, where the word embeddings $\boldsymbol{x}_{r_j}$ are vertexes of $C$ and $\sum_j a_{r_j} = 1$. As a result, we satisfy the condition of our Lemma 1: $a_{l_i}\boldsymbol{x}_i = \sum_j a_{r_j}\boldsymbol{x}_{r_j}$ and $a_{l_i} > \sum_j a_{r_j}$. Please see an illustration in Figure A.2 for an example. Then, Lemma 1 suggests that the logit $\boldsymbol{h}^T\boldsymbol{x}_i$ cannot be larger than the logits of all the word embeddings $\boldsymbol{h}^T\boldsymbol{x}_{r_j}$. This means at least one of the $\boldsymbol{h}^T\boldsymbol{x}_{r_j}$ on the convex hull would lead a larger prediction probability, which is also the conclusion of the theory in Demeter et al. [45].

Grivas et al. [73] discover that Transformer-based language models usually do not have any word embedding inside a convex hull formed by other word embeddings. Nevertheless, our empirical analyses in Section 2.2.3 and Appendix A.2.1 suggest that the Transformer-based langugae models, even a very large one, indeed suffer from the softmax bottleneck characterized by our theories, a generalized theory of Demeter et al. [45] and Yang et al. [248].

## A.5 Future Work

This chapter studies a general limitation of LMs and proposes solutions. The proposed theory can help us to understand that some types of mistakes or biases of LMs could come from *softmax bottleneck* and their incapability of modeling the correct distribution. For example, there are 60% of male characters and 40% of female characters in our training corpus. The language generation model might be forced to assign more than 60% probability to male characters as being much more likely to output *king* than *woman* in Figure 2.1. We want to more systematically investigate whether modeling multimodal distribution could help LMs to reduce the undesired bias and to better distinguish similar words [253]. Generally speaking, our study deepens our understanding of the weaknesses of modern LMs and we believe the knowledge can help us to design a better LM that increases the positive impacts and reduces the negative impacts in the future.

The hidden state size of GPT-3 175B [21] is huge (12,288). An interesting question is whether some sets of output word embeddings in GPT-3 are still in a low-dimensional subspace and whether the *softmax bottleneck* is still a prominent problem on the road of pursuing general intelligence when such a large hidden state dimension is used. We also would like to know if models using multiple facets could reach similar performance by a smaller hidden state size.

Recently, Gao et al. [67], Rajaee and Pilehvar [179], Cai et al. [22], Su et al. [209] point out the structure in the contextual embedding space prevents it from having an isotropic property. Our study and Demeter et al. [45] show that the structure in the word embedding space only models the global similarity between words and prevents the LM from outputting arbitrary context-dependent word distributions. We would like to know if we can discover a new LM architecture with a better contextual/word embedding space that could better model context-dependent word similarities and balance it with the global word similarities. In addition, our finding might be one of the reasons that we can improve the language generation quality by encouraging hidden states to be more isotropic [209].

Gao et al. [68] show that a mixture of kernel functions outperform MoS. *Mixtape* [249] is another efficient solution to the *softmax bottleneck*, whose hidden state for each word is the weighted average of the facets where the weights are dynamically predicted. If only using one softmax (i.e., $K = 1$), our multiple partition method could be viewed as a special case of *Mixtape* that uses a global and binarized weight to prevent complications of predicting weights of each word. Our results indicate that multiple partitions need to be combined with multiple softmax layers in order to gain consistent performance improvement. A potential future direction is to compare *MFS* with *mixture of kernel functions* and *Mixtape* on the transformer-based LMs or combine *MFS* with *mixture of kernel functions* and *Mixtape* to gain further improvements.

The results in Kong et al. [105] suggest that predicting n-gram could be better than predicting individual words in BERT in some applications. The total number of possible n-gram is several orders of magnitude higher than the number of individual tokens in the vocabulary. In addition, the linear dependency among n-gram might be common. For example, the embedding of the ***brown color*** + ***a dog*** may be similar to the embedding of the ***brown dog***.

# APPENDIX B

# APPENDIX FOR CHAPTER 3

## B.1 Training Algorithms

We summarize training procedure for our sentence representation model in algorithm 1, and our relation extraction model in algorithm 2.

## B.2 Regularization by Autoencoder for Relation Extraction

In this section, we describe the regularization term $\Omega$ defined in Equation 3.9.

The co-occurrence matrix between the sentence patterns and entity pairs is very sparse because most of the sentence patterns only co-occur with a few entity pairs. The sparsity might make the training process of multi-facet embeddings sensitive to the hyperparameters.

We discover that adding a simple autoencoder regularization is an effective way to stabilize the training. This regularization term aims to make the average of our facet embeddings of a sentence pattern similar to the weighted average of our word embeddings in that sentence pattern. The regularization is a kind of autoencoder because it reconstructs the weighted average embeddings of words in the input sentence pattern using the output facet embeddings. The regularization term $\Omega$ is defined as

$$\gamma \sum_{(i,j)\in R^{auto}} (2 \cdot \mathbb{1}_{i=j} - 1)||\underline{\boldsymbol{s}}_j^{aw'} - \mu(S_i)||^2, \tag{B.1}$$

where $\gamma$ is a weight for the regularization term, $R^{auto} = \cup_{i=1}^{I}\{(i,i),(i,q)\}$ is the set of all positive and negative training pairs, $I$ is the number of sentence patterns, and $q$ is a randomly selected index of sentence patterns which serves as our negative example, $\mathbb{1}_{i=j} = 1$ if $i = j$ and 0 otherwise, $\mu(S_i) = \frac{\sum_{k=1}^{K} \underline{\boldsymbol{s}}_{i,k}}{K}$ is the average of facet embeddings of the sentence pattern $S_i$. $\underline{\boldsymbol{s}}_j^{aw'}$ is a weighted average embedding of words in the $j$th sentence pattern that passes through a linear transformation $\boldsymbol{H}$. Weighting each word embedding by a smoothed inverse frequency provides a better text similarity measurement [7] because the frequently occurring words often do not contribute much to the semantic meaning (e.g., stop words). Similarly , we compute

$$\underline{\boldsymbol{s}}_j^{aw'} = \boldsymbol{H}\underline{\boldsymbol{s}}_j^{aw} = \boldsymbol{H}\sum_{w\in S_j} \frac{\nu}{\nu + p(w)}\underline{\boldsymbol{w}}, \tag{B.2}$$

**Algorithm 1:** Training using NNSC loss

**Input :** Training corpus, sequence boundaries, and pre-trained word embedding.
**Output :** $F$
Initialize $F$
**foreach** $I_t, \boldsymbol{W}(\boldsymbol{N_t}), \boldsymbol{W}(\boldsymbol{N_{r_t}})$ *in training corpus* **do**
  Run forward pass on encoder and decoder to compute $\boldsymbol{F}(\boldsymbol{I_t})$
  Compute
  $\boldsymbol{M^{O_t}} = \arg\min_{\boldsymbol{M}} ||\boldsymbol{F}(\boldsymbol{I_t})\boldsymbol{M} - \boldsymbol{W}(\boldsymbol{N_t})||^2 + \lambda||\boldsymbol{M}||_1 \forall k, j,\ 0 \leq \boldsymbol{M}_{k,j} \leq 1,$
  Compute
  $\boldsymbol{M^{R_t}} = \arg\min_{\boldsymbol{M}} ||\boldsymbol{F}(\boldsymbol{I_t})\boldsymbol{M} - \boldsymbol{W}(\boldsymbol{N_{r_t}})||^2 + \lambda||\boldsymbol{M}||_1 \forall k, j,\ 0 \leq \boldsymbol{M}_{k,j} \leq 1,$
  Run forward pass to compute $L_t$ in equation 3.2
  Treat $\boldsymbol{M^{O_t}}$ and $\boldsymbol{M^{R_t}}$ as constants, update $F$ by backpropagation
**end**

**Algorithm 2:** Training procedure (using batch size = 1)

**Input :** Sentence patterns and KB relations $S$, co-occurrence matrix $\{y_{i,j}\}$, entity pair embeddings from USchema, and pre-trained word embeddings.
**Output :** Our neural encoder and decoder
Initialize the entity pair embeddings using the embeddings learned by USchema.
Initialize the word embeddings of our neural encoder using pre-trained word embeddings and randomly initialize other parameters
**foreach** $S_i$ *in training corpus* **do**
  Run forward pass on the neural encoder and decoder to compute facet embeddings $\{\underline{\boldsymbol{s}}_{i,k}\}_{k=1}^{K}$
  Collect positive examples (i.e., $\{j|y_{i,j} = 1\}$) and negative examples for the $i$th sentence pattern or KB relation
  **foreach** *Positive and negative examples* $(i, j)$ **do**
    Compute $\underline{\tilde{\boldsymbol{e}}}_j = \frac{\underline{\boldsymbol{e}}_j}{||\underline{\boldsymbol{e}}_j||}$
    Compute $\eta_k = \min(1, \max(0, \frac{\underline{\tilde{\boldsymbol{e}}}_j^T \underline{\boldsymbol{s}}_{i,k}}{||\underline{\boldsymbol{s}}_{i,k}||^2})), \forall 1 \leq k \leq K$
    Select $k_{best}$th facet embedding by $k_{best} = \arg\min_k ||\underline{\tilde{\boldsymbol{e}}}_j - \eta_k \underline{\boldsymbol{s}}_{i,k}||^2$
    Add $(2 \cdot y_{i,j} - 1) r_{i,j} ||\underline{\tilde{\boldsymbol{e}}}_j - \eta_{k_{best}} \underline{\boldsymbol{s}}_{i,k_{best}}||^2$ to the loss
  **end**
  Add the autoencoder loss $\Omega$ in equation B.1 using pre-trained word embeddings
  Update neural encoder and decoder, entity pair embeddings, and $\boldsymbol{H}$ by backpropagation
**end**

where $\boldsymbol{H}$ linearly transforms the word embedding into the entity pair embedding space. $\nu = 10^{-4}$ is a constant set suggested in Arora et al. [7], $p(w)$ is the frequency of the word $w$ divided by the number of words in the corpus, and $\underline{\boldsymbol{w}}$ is the pretrained embedding of word $w$. We use 840B GloVe [174] as our word embedding in this work.

## B.3 Regularization by Autoencoder for Authorship/Citation Prediction

Most of the papers are only written by a few authors and cited by a few papers, so the interaction matrix is still very sparse even after jointly considering the authorship and citation information. If an author writes two papers and both papers are only cited by another paper, the papers are forced to have the same embedding even though the two papers have very different keywords in their titles and abstracts. This sparsity might encourage the embeddings of papers and authors in a small research subdomain to be collapsed into almost identical vectors.

To learn more diverse paper embeddings (especially within small subdomains), we compute a weighted average of word embeddings in each paper, project the average into the same space of author and citation embeddings, and encourage the average of our paper embeddings predicted by our neural model to be close to the average word embedding.

Since the words in the paper are both input and output of our neural model, the loss term $\Omega^S(F, \boldsymbol{H}, T)$ in Equation 3.14 is called autoencoder regularization. The autoencoder regularization is defined as

$$\gamma \sum_{(i,j)\in R^{auto}} \left( (\underline{\boldsymbol{p}}_j^{aw'})^T \mu(F(T_i)) - \mathbb{1}_{i=j} \right)^2, \tag{B.3}$$

where $R^{auto} = \cup_{i=1}^{I} \{(i,i), (i,j_i)\}$ and $j_i$ is a randomly selected number between 1 and number of papers $I$. $\mathbb{1}_{i=j} = 1$ if $i = j$ and 0 otherwise. $\mu(F(T_i)) = \frac{\sum_{k=1}^{K} \underline{\boldsymbol{p}}_{i,k}}{K}$ is the average of multi-facet embeddings, $\gamma$ is a hyperparameters, and we set $\alpha : \beta : \gamma = 5 : 1 : 10$ in our experiments. $\underline{\boldsymbol{p}}_j^{aw'}$ is the weighted average embedding of words in the $j$th paper, which will be defined in the next paragraph.

As in Appendix B.2, we use inverse frequency to weight the word embedding Arora et al. [7]

$$\underline{\boldsymbol{p}}_j^{aw'} = \boldsymbol{H} \underline{\boldsymbol{p}}_j^{aw} = \boldsymbol{H} \sum_{w\in T_j} \frac{\nu}{\nu + p(w)} \underline{\boldsymbol{w}}, \tag{B.4}$$

where $\boldsymbol{H}$ is a trainable matrix that linearly transforms the word embedding into the author and citation embedding space. $\nu$ is a constant set to be $10^{-4}$ as suggested in Arora et al. [7], $p(w)$ is the probability of seeing the word $w$ in the corpus, and $\underline{\boldsymbol{w}}$ is the pretrained embedding of word $w$.

## B.4 Experiment Setup Details for Authorship/Citation Prediction

We test the methods on ML/AI related domains, so we only include papers from S2ORC [132] that are in the CS domain and cited by at least one ML/AI related paper on ArXiv. ML/AI related papers on ArXiv refer to the papers from the following ArXiv categories: Artificial Intelligence (cs.AI), Computation and Language (cs.CL), Computer Vision and Pattern Recognition (cs.CV), Information Retrieval (cs.IR), Information Theory (cs.IT), Learning (cs.LG), Multiagent Systems (cs.MA), Neural and Evolutionary Computing (cs.NE), Robotics (cs.RO), and Machine Learning (stats.ML).

In training and the authorship prediction evaluation, we assume the author with the same name in S2ORC is the same person. We hypothesize that the resulting noise won't bias toward a particular method.

## B.5 Implementation Details for Authorship/Citation Prediction

We initialize the token embedding of text encoder using the pretrained word embedding before training but allow the word embedding inside the text encoder to be updated during training. In the experiments, we use the same 200-dimensional CBOW embeddings as Bansal et al. [14] and the baseline CBOW Avg.

The pretrained word embedding $\underline{\boldsymbol{w}}$ in equation B.4 is fixed during training and only the linear transformation matrix $\boldsymbol{H}$ is updated. To stabilizing training, we apply L2 regularization with weights $10^{-6}$ to the author embeddings, citing paper embeddings, and all parameters of neural models. We use Adam [101] and one 1080ti GPU to optimize our neural model in a week.

In our preprocessing step, text in the titles and abstracts are tokenized into words using Spacy[1]. The words that appear less than 5 times are mapped to <unk> and the length of the input to the text encoder is truncated to 512 words. When the abstract is not available in our training or testing corpus, we only use the paper title as our input. Before passing the input to the Transformer encoder, we add the word embeddings, position embeddings, and type embeddings together, where we give different types to the words in the title and the words in the abstract to help the Transformer to distinguish the title and abstract.

[1] https://spacy.io/

# APPENDIX C

# APPENDIX FOR CHAPTER 4

## C.1 Implementation Details

The training algorithm for our option generator could be seen in Algorithm 3. During training, the input prompt is tokenized into word pieces, and the actual continuation is tokenized into words. We run the byte pair encoding [194] to get word pieces required by GPT2 and run Spacy tokenizer[1] to get words required by GloVe. The two tokenization results are aligned to collect the training examples.

## C.2 Experiment Details

We truncate the probabilities after the top 40 in top-k sampling [56]. In all the experiments, we set $M = 5$ words to represent each topic, although the figures and tables use $M = 2$ or $M = 3$ due to the space limit. We set $K = 10$. Our Transformer decoder for option generation has 5 layers.

In the following subsections, we describe the details about our baselines, the automatic evaluation, and human evaluation.

### C.2.1 Baselines

We adopt the default hyper-parameters of LDA in gensim[2]. The cluster centers of Kmeans are optimized using random initialization and EM algorithm for at most 300 iterations.[3] We use RMSprop [221] to optimize NNSC for 2,000 iterations.

**PPLM** uses the default hyperparameters for conditioning on a bag of words in its GitHub repository[4]. We try several different hyperparameters in PPLM and also try to apply **PPLM** to the original GPT2 with 117M parameters and to the GPT2 that is fine-tuned on Wikipedia. They produce similar relevancy and perplexity, which are significantly worse than ours in automated evaluation.

---

[1] `spacy.io/`

[2] `https://radimrehurek.com/gensim/models/ldamulticore.html`

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`

[4] `https://github.com/uber-research/PPLM`

**Algorithm 3:** Training procedure for our option generator (using batch size = 1)

**Input :** Training corpus, stop word list, pretrained GPT2 encoder, and pre-trained word embeddings.

**Output :** Neural option generator

Initialize our encoder using a pretrained GPT2 model and randomly initialize the other parameters

**foreach** $x_1, ..., x_I$ *in training corpus* **do**

  Run forward pass of our model given $x_1, ..., x_I$ to compute the cluster centers $\underline{c}_1, ...\underline{c}_K$

  Collect the positive examples $\bar{y}_1, ..., \bar{y}_O$ (i.e., non-stop words after $x_I$) and their word embeddings $\underline{e}_o^{\bar{y}}$

  Collect the negative examples $\bar{y}'_1, ..., \bar{y}'_O$ (i.e., a randomly sampled continuation without stop words) and their word embeddings $\underline{e}_o^{\bar{y}'}$

  $L = 0$

  **foreach** $\bar{y}_o$ *in the positive example* **do**

    Estimate $\hat{a}_1, ..., \hat{a}_K = \underset{0 \leq a_1, ..., a_K \leq 1}{\arg\min} || \sum_{k=1}^{K} a_k \underline{c}_k - \underline{e}_o^{\bar{y}} ||^2 + \lambda \sum_{k=1}^{K} a_k$ using RMSprop

    $L = L + || \sum_{k=1}^{K} \hat{a}_k \underline{c}_k - \underline{e}_o^{\bar{y}} ||^2$

  **end**

  **foreach** $\bar{y}'_o$ *in the negative example* **do**

    Estimate $\hat{b}_1, ..., \hat{b}_K = \underset{0 \leq b_1, ..., b_K \leq 1}{\arg\min} || \sum_{k=1}^{K} b_k \underline{c}_k - \underline{e}_o^{\bar{y}'} ||^2 + \lambda \sum_{k=1}^{K} b_k$ using RMSprop

    $L = L - || \sum_{k=1}^{K} \hat{b}_k \underline{c}_k - \underline{e}_o^{\bar{y}'} ||^2$

  **end**

  Update our neural model by backpropagation through cluster centers $\underline{c}_1, ...\underline{c}_K$ to minimize $L$

**end**

The code of **PPLM** can only condition on a single word piece, so we need to remove the rare words that contain multiple word pieces. We filter out the input prompt in the test set if **PPLM** cannot condition on any word in the randomly sampled topics.

# APPENDIX D

# APPENDIX FOR CHAPTER 5

## D.1 Experiment Details for Multi-CLS BERT

We first describe the architecture details and pretraining details of our methods and baselines in Appendix D.1.1. Then, we list the hyperparameter setup in the fine-tuning in Appendix D.1.2. Finally, we explain the details of the ensemble baselines and their related analyses in Appendix D.1.3.

### D.1.1 Our Models and Baselines

The models built on $\text{BERT}_{\text{Base}}$ are pretrained using two billion tokens and each batch contains 30 sequences. The models built on $\text{BERT}_{\text{Large}}$ are pretrained using one billion tokens and each batch contains 48 sequences. The learning rate is $2 \cdot 10^{-5}$ and the warmup ratio is $0.001$ for the pretraining stage.

We implement Multi-CLS BERT by modifying the code of Aroca-Ouellette and Rudzicz [6][1]. We use [unused0] – [unused(K-1)] tokens in the original BERT tokenizer as our input CLS tokens [C1] – [CK]. We still keep the original CLS tokens to increase the comparability with the **MTL** baseline.

We use NVIDIA GeForce RTX 2080, 1080, and TITAN X, M40 GPUs for the $\text{BERT}_{\text{Base}}$ experiments and use GeForce RTX 8000 and Tesla M40 for the $\text{BERT}_{\text{Large}}$ experiments. In Table 5.1, the model size excludes the top classifier parameters used in each task.

We test **CMTL+** using the default hyperparameters of Aroca-Ouellette and Rudzicz [6] and we do not try different hyperparameters or different schedules of pretraining losses. **No Inserted Layers** only removes the $L_{l,k}(.)$ while still using different $H_k^{MC}$ on top during pretraining. **SWA** averages the weights of every model checkpoint that is evaluated using the validation dataset.

[1] `https://github.com/StephAO/olfmlm`

### D.1.2 Fine-tuning

We start from the default evaluation hyperparameters used in Aroca-Ouellette and Rudzicz [6] and modify the settings based on the suggestions from Zhang et al. [255] and Mosbach et al. [155]. We find that the best hyperparameters depend on the training size. For example, batch size 16 works well in GLUE Full but is much worse than batch size 4 in GLUE 100. Furthermore, the performance of the default hyperparameters on some tasks is suboptimal or unstable even after averaging the performance from 16 trials. Therefore, we coarsely tune the hyperparameters to maximize and stabilize the performance of the **Ours (K=1)** baseline under the memory and computational time constraints in our GPUs. The preliminary results suggest that the hyperparameters also maximize the performance of **MTL**.

Next, we list fine-tuning hyperparameters for all the tasks[2]. Our fine-tuning stops after 20 epochs, 60k batches, or consecutive 10k batches without a validation improvement (whichever comes first). We use the first 5k validation samples to select the best fine-tuned model checkpoints for the evaluation. The maximal gradient norm is 1. The maximal length for sentences and CLS tokens is 128 for GLUE and 256 for SuperGLUE.

For each task, we select the best learning rate from $c \cdot 10^{-5}$ and $c = 1, 2, 3, 4, 5, 7$. When running large datasets in GLUE Full and SuperGLUE Full (MNLI, QQP, QNLI, SST-2, BoolQ, MultiRC, and WiC) using $\text{BERT}_{\text{Large}}$, we use learning rates $c = 2, 4, 6, 8, 10, 14$ to accelerate the training. The batch sizes for GLUE 100, 1k, Full are 4, 8, 16, respectively. The batch size for SuperGLUE is 4 except that the $\text{BERT}_{\text{Large}}$ models use 8 in SuperGLUE 1k and Full. For $\text{BERT}_{\text{Base}}$, the warmup ratio is 0.1. For $\text{BERT}_{\text{Large}}$, the warmup ratio is 0.2 and the weight decay is $10^{-6}$.

For each fine-tuning random seed, we randomly select a different subset in the settings where only 100 or 1k training samples are available. For the datasets with less than 500 training samples in SuperGLUE and SuperGLUE 1k (i.e., CB and COPA), we repeat the experiments 32 times to further stabilize the scores. For the pre-trained BERT baseline, we use 16 fine-tuning random seeds. To reduce the computational cost, we use two pretraining random seeds and four fine-tuning random seeds in our ablation study in Table 5.2.

Compared to other tasks, ReCoRD needs to be trained much longer than other tasks in SuperGLUE, so we only use one fine-tuning seed for each of the four pretrained models with different seeds. Our fine-tuning stops after 600k batches ($\text{BERT}_{\text{Base}}$) / 300k batches ($\text{BERT}_{\text{Large}}$) or consecutive 160k batches without a validation improvement (whichever comes first).

To stabilize the performance of each model on ReCoRD, we use the first 40k validation samples to select the best fine-tuned model checkpoints. We set batch size as 8 and learning rate as $1 \cdot 10^{-5}$ for $\text{BERT}_{\text{Base}}$. For $\text{BERT}_{\text{Large}}$, we set batch size as 32 and learning rate as $2 \cdot 10^{-5}$.

---

[2]We use different values for some hyperparameters in ReCoRD. See the details below.

### D.1.3 Ensemble Models

**Ensemble on FT Seeds (K=1)** in Table 5.2 is the same as **Ensemble of Ours (K=1)** in Table 5.3. **Ensemble on FT Seeds (K=5)** in Table 5.2 is the same as **ENS** in Table 5.4. **Ensemble on Dropouts** in Table 5.2 is the same as **Dropout** in Table 5.4. All results are the average of four models that use four different pretrained models and the best learning rate among $c \cdot 10^{-5}$ ($c = 1, 2, 3, 4, 5, 7$) in the fine-tuning stage.

In Table 5.3, we compute the expected calibration error (ECE) [157] by

$$\sum_{j=1}^{10} \frac{|B_j|}{N} |\text{acc}(j) - \text{conf}(j)|, \tag{D.1}$$

where $\text{acc}(j)$ is the model accuracy in the $j$th bin $B_j$, $N$ is the number of validation samples, and $\text{conf}(j) = \frac{1}{|B_j|} \sum_{x \in B_j} \max_y P(y|x)$ is the average of the highest prediction probability $P(y|x)$ in the $j$th bin. We put the samples into 10 equal-size bins according to their highest prediction probability $\max_y P(y|x)$.

In Table 5.3, we use Tesla M40 to measure the inference time of the models built on $\text{BERT}_{\text{Base}}$. We set batch size 16 and run 1000 batches to get the average inference time of one batch in every GLUE task. We repeat the experiments five times and report their average and standard error. For the ensemble model, we assume the time of averaging multiple prediction probabilities is negligible and directly multiply the inference time of **Ours (K=1)** by 5.

In Table 5.4, we would like to see if CLS embeddings disagree with each other as other ensemble baselines did. In **Multi-CLS**, we compute the uncertainty of each sample $x$ as the average variance of prediction probability of each CLS embedding $\text{mean}_l\left(\text{var}_k P(y = l|x, k)\right)$ and estimate the prediction probability of the $k$th CLS embedding by

$$P(y = l|x, k) = \frac{\exp\left(\boldsymbol{q}_{l,k}^T L_{O,k}^{FT}(\boldsymbol{h}_k^c(x, y_{gt}))\right)}{\sum_i \exp\left(\boldsymbol{q}_{i,k}^T L_{O,k}^{FT}(\boldsymbol{h}_k^c(x, y_{gt}))\right)}, \tag{D.2}$$

where $L_{O,k}^{FT}(\boldsymbol{h}_k^c(x, y_{gt}))$ is the CLS embedding of the input $x$ after fine-tuning, and $\boldsymbol{q}_{i,k} = \frac{1}{N_i} \sum_{y_{gt}=i} L_{O,k}^{FT}(\boldsymbol{h}_k^c(x, y_{gt}))$ is the $i$th class embedding for the $k$th CLS embedding, which is computed by averaging the $k$th CLS embeddings of the input $x$ with the $i$th class label.

In Table 5.4, the two ensemble models for **ENS vs ENS** use the same set of 5 fine-tuning seeds and the two **Ours (K=5, $\lambda = 0.1$)** pretrained with different random seeds. Both uncertainty estimation models for **Multi-CLS vs ENS**, **Dropout vs ENS**, and **Least vs ENS** are based on the same pretrained **Ours (K=5, $\lambda = 0.1$)** model.

## D.2 Training and Testing Dataset Information for Multi$^2$SPE

### D.2.1 Creating training data

**SPECTER, single domain (CS)**: We use the original dataset files provided by the SPECTER authors.[3]

[3] https://github.com/allenai/specter/issues/2#issuecomment-625428992

**SPECTER, multi domain**: We use the papers in shard 11 of the 20200705 version of S2ORC [132] as query papers. Since we tried to be unbiased as possible towards different subject areas during the paper selection, we filter out any papers without MAG information in S2ORC. Then we scan the entirety of S2ORC to fetch all direct and indirect citations, as defined in Cohan et al. [42], for the chosen query papers. We feed the acquired citation information into the sampling code provided by the SPECTER authors.
**SPECTER, single domain (Medicine)**: We follow the same steps as SPECTER multi-domain, except for the use of shard 22, 33, and 44 and limiting all query papers to belong to the 'Medicine' MAG field.
**SciNCL**: For SciNCL single domain, we use the 'SciNCL w/ leakage' dataset.[4] For SciNCL multi-domain, we use the 'SciNCL w/o leakage' dataset.[5] We examined the 'SciNCL w/o leakage' dataset and found it to be fairly balanced in terms of subject field distribution of query papers, as they also randomly choose query papers from S2ORC.

### D.2.2 Creating Multi-SciDocs

For a better measurement of multi-domain performance, we have created the multi-domain (co-)citation prediction tasks. We refer to the collection of 3 multi-domain tasks, multi. cite, multi. co-cite, and MAG as Multi-SciDocs.

For multi. cite and multi. co-cite datasets, we use shard 7 of the 20200705 version of S2ORC [132] as query papers, avoiding certain domain from being the majority of query papers. For each query, we collect 500 negative papers and up to 5 positive papers. This is the same setting used in the original SciDocs [42]. The task is to assign higher similarity scores to the positive papers and lower scores to the negative papers. In both datasets, the negative samples come from randomly sampled papers. In the multi. cite dataset, the positive samples are the papers cited by the query paper. In the multi. co-cite dataset, the positive samples and the query paper are both cited by another paper. We made minimal modification to the SciDocs execution code[6] to allow testing with our custom dataset files.

For MAG, we use the same test set included in the original SciDocs.

### D.2.3 MAG subject field distribution of training and testing datasets

Please refer to Table D.1.

---

[4] https://github.com/malteos/scincl/releases/tag/0.1

[5] https://github.com/malteos/scincl/releases/tag/0.1-wol

[6] https://github.com/allenai/scidocs

## D.3 Reproducibility Information for Multi$^2$SPE

**Implementation of architectural changes**: For multiple CLS tokens, we use `[unused]` tokens available in SciBERT vocabulary except for the first CLS token, where we make use of the existing `[CLS]` token. We note that our current best model, Multi-CLS BERT (3 CLS, $\lambda = 0.1$) have 114.8M parameters. which is relatively a small increase from the 109M parameters of $\text{BERT}_{\text{Base}}$ with sequence classification head.

**Training hyperparameters**: Since we compare the performance of our models with the SPECTER/SciNCL baselines, we replicate the training setup originally used in SPECTER as much as possible. We initialize the BERT layer with the pretrained SciBERT[7] We use the Adam optimizer with a learning rate of 2e-5. A linear learning rate scheduler with warm up fraction of 0.1 is used. Since SPECTER uses an effective batch size of 32 with gradient accumulation, we achieve the same effective batch size by doing gradient accumulation at every 16 steps with the real batch size of 2. We train all the models for 2 epochs.

**Random seeds**: All experiments are ran with 4 different seeds, `1783, 1918, 1945, 1991`. All the reported metrics in this paper are the average scores from the 4 seeds, unless otherwise noted.

**Hardwares and softwares used**: To fully utilize all the GPU resources available to us, We trained all our models using multiple NVIDIA RTX 2080 Ti, GTX 1080 Ti and GTX TITAN X GPUs. To achieve better reproducibility, each random seed was always ran with the same GPU model (`1783` $\rightarrow$ TITAN X, `1918` $\rightarrow$ 2080 Ti, `1945` $\rightarrow$ 1080 Ti, `1991` $\rightarrow$ 2080 Ti.)

We use the version 4.9.2 of HuggingFace Transformers library [240] for BERT implementation, and PyTorch Lightning version 1.4.2 [55] alongside PyTorch 1.8.2 [169] for training logic organization.

We also make use of the source codes released by the authors of SPECTER[8] and SciNCL[9] to perform dataset creation and loading.

[7] `https://huggingface.co/allenai/scibert_scivocab_uncased`

[8] `https://github.com/allenai/specter`

[9] `https://github.com/malteos/scincl`

| MAG field | *Training* | | | | *Testing - SciDocs* | | *Testing* - Multi-SciDocs | | |
|---|---|---|---|---|---|---|---|---|---|
| | **SPECTER** | **SciNCL** | **SPECTER$_{multi}$** | **SciNCL$_{multi}$** | **cite** | **cocite** | **MAG** | **multi. cite** | **multi. cocite** |
| | Percentage (Count) | Percentage (Count) | Percentage (Count) | Percentage (Count) | Percentage (Count) | Percentage (Count) | Percentage (Count) | Percentage (Count) | Percentage (Count) |
| Art | 0.03% (53) | 0.04% (62) | 0.02% (12) | 0.06% (108) | 0% (0) | 0% (0) | 4.67% (175) | 0.03% (13) | 0% (0) |
| Biology | 1.03% (1748) | 1.31% (2257) | 19.55% (14561) | 13.77% (23904) | 0.49% (6) | 0.33% (4) | 6.05% (227) | 19.66% (9813) | 25.14% (785) |
| Business | 0.68% (1156) | 0.67% (1148) | 0.58% (433) | 0.92% (1591) | 0.58% (7) | 0.41% (5) | 5.23% (196) | 0.58% (289) | 0.29% (9) |
| Chemistry | 0.17% (282) | 0.29% (495) | 3.27% (2433) | 3.65% (6336) | 0.16% (2) | 0.08% (1) | 5.25% (197) | 3.54% (1767) | 2.59% (81) |
| Computer Science | **61.00%** (103869) | **61.78%** (106268) | 12.03% (8962) | 15.20% (26391) | **65.16%** (791) | **62.78%** (769) | 5.07% (190) | 11.61% (5795) | 10.63% (332) |
| Economics | 0.63% (1078) | 0.67% (1159) | 1.82% (1355) | 2.06% (3578) | 0.66% (8) | 0.49% (6) | 5.76% (216) | 1.82% (907) | 1.19% (37) |
| Engineering | 7.17% (12212) | 7.34% (12623) | 2.13% (1586) | 4.79% (8315) | 6.59% (80) | 6.29% (77) | 5.55% (208) | 2.56% (1277) | 1.31% (41) |
| Environmental Science | 0.05% (85) | 0.06% (102) | 0.33% (243) | 0.47% (821) | 0% (0) | 0% (0) | 5.76% (216) | 0.37% (183) | 0.10% (3) |
| Geography | 0.24% (405) | 0.25% (430) | 0.47% (350) | 0.76% (1322) | 0.41% (5) | 0.24% (3) | 4.59% (172) | 0.49% (247) | 0.22% (7) |
| Geology | 0.11% (183) | 0.13% (216) | 1.32% (982) | 1.38% (2395) | 0.08% (1) | 0.16% (2) | 5.65% (212) | 1.16% (577) | 0.64% (20) |
| History | 0.01% (20) | 0.02% (26) | 0.02% (15) | 0.13% (223) | 0% (0) | 0% (0) | 4.99% (187) | 0.05% (24) | 0.10% (3) |
| Materials Science | 0.44% (749) | 0.50% (867) | 1.31% (976) | 2.21% (3844) | 0.33% (4) | 0.82% (10) | 5.76% (216) | 1.65% (825) | 0.70% (22) |
| Mathematics | 7.53% (12828) | 7.84% (13490) | 7.95% (5923) | 5.75% (9988) | 7.50% (91) | 9.06% (111) | 5.20% (195) | 8.01% (3997) | 6.47% (202) |
| Medicine | 8.59% (14629) | 9.41% (16187) | 40.72% (30325) | 31.79% (55178) | 6.10% (74) | 7.67% (94) | 4.88% (183) | 39.37% (19649) | 43.27% (1351) |
| Philosophy | 0.02% (41) | 0.03% (52) | 0.03% (19) | 0.10% (168) | 0% (0) | 0% (0) | 4.67% (175) | 0.05% (26) | 0% (0) |
| Physics | 1.41% (2399) | 1.51% (2605) | 3.93% (2930) | 3.08% (5353) | 1.32% (16) | 2.69% (33) | 5.25% (197) | 4.49% (2242) | 2.75% (86) |
| Political Science | 0.13% (219) | 0.14% (240) | 0.22% (165) | 0.61% (1054) | 0.16% (2) | 0.08% (1) | 5.36% (201) | 0.25% (123) | 0.03% (1) |
| Psychology | 4.18% (7125) | 4.27% (7340) | 3.79% (2819) | 3.57% (6197) | 3.13% (38) | 3.27% (40) | 5.41% (203) | 3.77% (1884) | 4.42% (138) |
| Sociology | 0.34% (576) | 0.37% (641) | 0.52% (389) | 0.94% (1638) | 0.25% (3) | 0.16% (2) | 4.93% (185) | 0.55% (276) | 0.13% (4) |
| Unknown | 6.23% (10611) | 3.38% (5811) | 0.00% (0) | 8.74% (15169) | 7.08% (86) | 5.47% (67) | 0% (0) | 0% (0) | 0% (0) |
| Total | 100.00% (170268) | 100.00% (172019) | 100.00% (74478) | 100.00% (173573) | 100.00% (1214) | 100.00% (1225) | 100.00% (3751) | 100.00% (49914) | 100.00% (3122) |

Table D.1: Training and testing dataset statistics. Note: Since some papers are categorized under multiple MAG fields in S2ORC, they are counted more than once in this table. Unknown refers to the papers without MAG information in S2ORC.