Grammatical and Semantic Biases in Representation Learning from Raw Datasets

Masahiro Kaneko

Department of Information and Communication Systems

Graduate School of System Design

Tokyo Metropolitan University

March 2021

A Doctoral Dissertation

submitted to Graduate School of System Design,

Tokyo Metropolitan University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Masahiro Kaneko

Acknowledgements

まず，修士から博士まで 5 年間指導してくださった小町守先生に心より感謝いたします．自分の多くの研究に関する知識や技術は小町先生との研究を通して学んだものであり，今の自分があるのも小町先生のおかげです．そして，博士課程では自由に研究させていただけたことで，様々な場所でたくさんの研究者の方と交流することができました．また，研究室ではたくさんの共著の研究をさせていただけたことで，他の人より何倍もの研究経験を積むことができました．常に学生の味方でありオープンマインドな先生の人柄があってこそ，山あり谷ありな研究生活を乗り切ることができたと思います．

リバプール大学・Amazon の Danushka Bollegala 先生には，博士 1 年の秋ごろからリバプール大学で客員研究員として，そしてその後も今日までご指導いただき本当に感謝しております．初めて Danushka 先生に原稿を修正された際，自分が書いた箇所がほとんどなくなっていて「なるほど，これがトップ研究者と自分の差か」と衝撃を受けると同時に如何に論文をきちんと書くことが重要であるかを痛感させられました．そして，Danushka 先生と研究に関して相談する時間はたくさんのアイデアを議論できてとても楽しかったです．Danushka 先生のようなをやさしさをもったトップ研究者になれるよう頑張ります．今後ともどうぞよろしくお願いいたします．

東北大学と理研 AIP の乾健太郎先生と鈴木潤先生には，毎週の進捗報告や研究室に数カ月間滞在してご指導していただきました．データや出力を分析することの重要性やトップカンファレンスに通すための技術を学ばさせていただきました．そして，研究室単位で強い研究をするとはどういうことか経験させていただけたことも感謝しております．

修士と博士で副査を引き受けてくださった山口亨先生と高間康史先生に感謝いたします．そして，博士の副査を引き受けてくださった東京工業大学の岡崎直観先生にもお礼申し上げます．博士論文を執筆するにあたって，本質的なコメントをいただきより良いものにすることができました．

大阪大学の梶原智之さんには先輩として在学時や卒業後も研究や進路に関してたくさんのご助言をいただきました．夜中まで研究室のホワイトボードの前で研究の議論をしたことは良い思い出です．

インターンや共同研究などで一緒に研究してくださった甲南大学の永田亮先生，理研 AIP・東北大学の三田雅人さんと清野舜さん，フューチャー株式会社の水本智也さん，NICT：情報通信研究機構の Raj Dabre さん，Element AI の Wonchang Chung さんに感謝いたします．

小町研究室で一緒に過ごした皆様にもお礼申し上げます．そして，一緒に研究してくださった Aizhan Imankulova さん，鈴木由衣さん，堺澤勇也さん，松村雪桜さん，新井美桜さん，勝又智さん，甫立健悟さん，平澤寅庄さん，吉村綾馬さん，李凡さん，山下郁海さん，高橋悠進さん，蘆田真奈さん，小山碧海さん，今藤誠一郎さん，喜友名朝視顕さん，田村弘人さん，Zizheng Zhang さん，Xiaomeng Pan さん，ありがとうございました．

最後に，研究という自分が夢中になれるものを見つけることができたのは，決して立派な学生ではなかった自分のことを支え続けれくれた両親と兄のおかげだと思っています．深く感謝致します．

Publications

Journal Papers

1. 甫立健悟, 金子正弘, 勝又智, 小町守. 文法誤り訂正における訂正度を考慮した多様な訂正文の生成．自然言語処理．28 巻 2 号．2021.（採録決定）

2. 吉村綾馬, 金子正弘, 梶原智之, 小町守. 文法誤り訂正の参照文を用いない自動評価の人手評価への最適化．自然言語処理．28 巻 2 号．2021.（採録決定）

3. 三田雅人, 水本智也, 金子正弘, 永田亮, 乾健太郎．文法誤り訂正モデルの横断評価．自然言語処理．28 巻 1 号．2021.（採録決定）

4. 新井美桜, 金子正弘, 小町守. 日本語学習者向けの文法誤り検出機能付き作文用例検索システム. 人工知能学会論文誌, 35 巻 5 号, pp.1-9. 2020.

5. Masahiro Kaneko and Mamoru Komachi. Multi-Head Multi-Layer Attention to Deep Language Representations for Grammatical Error Detection. Computacion y Sistemas. Vol. 23, No. 3, pp. 883-891. 2019.

6. 金子正弘, 堺澤勇也, 小町守. 正誤情報と文法誤りパターンを考慮した単語分散表現を用いた文法誤り検出. 自然言語処理,25 巻 4 号, pp.421-439. 2018.

International Conference Papers

1. Masahiro Kaneko and Danushka Bollegala. Debiasing Pre-trained Contextualised Embeddings. The 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL, Long paper). 2021. (Accepted)

2. Masahiro Kaneko and Danushka Bollegala. Dictionary-based Debiasing of Pre-trained Word Embeddings. The 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL, Long paper). 2021. (Accepted)

3. Masahiro Kaneko and Danushka Bollegala. Autoencoding Improves Pre-trained Word Embeddings. The 28th International Conference on Computational Linguistics (COLING, Short paper). pp. 1699-1713. 2020.

4. Ikumi Yamashita, Satoru Katsumata, Masahiro Kaneko, Aizhan Imankulova and Mamoru Komachi. Cross-lingual Transfer Learning for Grammatical Error Correction. The 28th International Conference on Computational Linguistics (COLING, Long paper). pp. 4704-4715. 2020.

5. Kengo Hotate, Masahiro Kaneko and Mamoru Komachi. Generating Diverse Corrections with Local Beam Search for Grammatical Error Correction. The 28th International Conference on Computational Linguistics (COLING, Short paper). pp. 2132-2137. 2020.

6. Ryoma Yoshimura, Masahiro Kaneko, Tomoyuki Kajiwara and Mamoru Komachi. SOME: Reference-less Sub-Metrics Optimized for Manual Evaluations of Grammatical Error Correction. The 28th International Conference on Computational Linguistics (COLING, Short paper). pp. 6516-6522.

2020.

7. Aizhan Imankulova, Masahiro Kaneko, Tosho Hirasawa and Mamoru Komachi. Towards Multimodal Simultaneous Neural Machine Translation. The Fifth Conference in Machine Translation (WMT). pp. 540-549. 2020.

8. Masato Mita, Shun Kiyono, Masahiro Kaneko, Jun Suzuki and Kentaro Inui. A Self-Refinement Strategy for Noise Reduction in Grammatical Error Correction. The 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP Findings). pp. 267-280. 2020.

9. Zizheng Zhang, Tosho Hirasawa, Wei Houjing, Masahiro Kaneko and Mamoru Komachi. Translation of New Named Entities from English to Chinese. The 7th Workshop on Asian Translation (WAT). pp. 58-63. 2020.

10. Hiroto Tamura, Tosho Hirasawa, Masahiro Kaneko and Mamoru Komachi. TMU Japanese-English Multimodal Machine Translation System for WAT 2020. The 7th Workshop on Asian Translation (WAT). pp. 80-91. 2020.

11. Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki and Kentaro Inui. Encoder-Decoder Models Can Benefit from Pre-trained Masked Language Models in Grammatical Error Correction. The 58th Annual Conference of the Association for Computational Linguistics (ACL short paper). pp. 4248-4254. 2020.

12. Masahiro Kaneko, Aizhan Imankulova, Tosho Hirasawa and Mamoru Komachi. English-to-Japanese Diverse Translation by Combining Forward and Backward Outputs. The 4th Workshop on Neural Generation and Translation (WNGT): Simultaneous Translation And Paraphrase for Language Education (STAPLE) English-to-Japanese track. pp. 134-138. 2020.

13. Aizhan Imankulova, Masahiro Kaneko and Mamoru Komachi. Japanese-

Russian TMU Neural Machine Translation System using Multilingual Model for WAT 2019. The 6th Workshop on Asian Translation (WAT): News Commentary task. pp. 165-170. 2019.

14. Masahiro Kaneko and Danushka Bollegala. Gender-preserving Debiasing for Pre-trained Word Embeddings. The 57th Annual Conference of the Association for Computational Linguistics (ACL long paper). pp. 1641-1650. 2019.

15. Kengo Hotate, Masahiro Kaneko, Satoru Katsumata and Mamoru Komachi. Controlling Grammatical Error Correction Using Word Edit Rate. The 57th Annual Conference of the Association for Computational Linguistics: Student Research Workshop (ACL SRW). pp. 149-154. 2019.

16. Mio Arai, Masahiro Kaneko and Mamoru Komachi. Grammatical-Error-Aware Incorrect Example Retrieval System for Learners of Japanese as a Second Language. The 14th Workshop on Innovative Use of NLP for Building Educational Applications (BEA). pp. 296-305. 2019.

17. Masahiro Kaneko, Kengo Hotate, Satoru Katsumata and Mamoru Komachi. TMU Transformer System Using BERT for Re-ranking at BEA 2019 Grammatical Error Correction on Restricted Track. The 14th Workshop on Innovative Use of NLP for Building Educational Applications (BEA): Shared Task on Grammatical Error Correction. pp. 207-212. 2019.

18. Masahiro Kaneko and Mamoru Komachi. Multi-Head Multi-Layer Attention to Deep Language Representations for Grammatical Error Detection. In 20th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing). 2019.

19. Masato Mita, Tomoya Mizumoto, Masahiro Kaneko, Ryo Nagata and Ken-

taro Inui. Cross-Corpora Evaluation and Analysis of Grammatical Error Correction Models – Is Single-Corpus Evaluation Enough? In 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT short paper). pp. 1309-1314. 2019.

20. Masahiro Kaneko, Tomoyuki Kajiwara and Mamoru Komachi. TMU System for SLAM-2018. The 13th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2018): Shared Task on Second Language Acquisition Modeling. pp. 365-369. 2018.

21. Masahiro Kaneko, Yuya Sakaizawa and Mamoru Komachi. Grammatical Error Detection Using Error- and Grammaticality-Specific Word Embeddings. The 8th International Joint Conference on Natural Language Processing (IJCNLP long paper). pp. 40-48. 2017.

Domestic Conference Papers

1. 喜友名朝視顕, 吉村綾馬, 金子正弘, 小町守. 項目別マルチタスク学習による系列変換タスクの品質推定. NLP 若手の会第 15 回シンポジウム. 2020.

2. 今藤誠一郎, 甫立健悟, 平澤寅庄, 金子正弘, 小町守. 機械翻訳における非自己回帰モデルの複数言語の出力分析. NLP 若手の会第 15 回シンポジウム. 2020.

3. 小山碧海, 甫立健悟, 金子正弘, 小町守. 文法誤り訂正における複数の擬似誤り生成モデルの比較. NLP 若手の会第 15 回シンポジウム. 2020.

4. 山下郁海, 勝又智, 金子正弘, Imankulova Aizhan, 小町守. 言語間での転移学習を用いたロシア語文法誤り訂正. 言語処理学会第 26 回年

次大会, pp.1324-1327. 2020.

5. 三田雅人, 清野舜, 金子正弘, 鈴木潤, 乾健太郎. 文法誤り訂正のための自己改良戦略に基づくノイズ除去. 言語処理学会第 26 回年次大会, pp.993-996. 2020.
6. Imankulova Aizhan, 金子正弘, 平澤寅庄, 小町守. 画像を使用したマルチモーダルニューラル同時翻訳. NLP 若手の会第 14 回シンポジウム. 2019.
7. 金子正弘, 三田雅人, 鈴木潤, 乾健太郎. コロケーション・イディオム誤りを考慮した文法誤り訂正のための擬似データ生成. NLP 若手の会第 14 回シンポジウム. 2019.
8. 甫立健悟, 金子正弘, 小町守. Autoencoder を用いた頑健な文の分散表現生成の検討. NLP 若手の会第 14 回シンポジウム. 2019.
9. 山下郁海, 勝又智, 金子正弘, Imankulova Aizhan, 小町守. 英語からロシア語への転移学習を用いた文法誤り訂正. NLP 若手の会第 14 回シンポジウム. 2019.
10. 三田雅人, 水本智也, 金子正弘, 永田亮, 乾健太郎. 文法誤り訂正のコーパス横断評価: 単一コーパス評価で十分か? 言語処理学会第 25 回年次大会, pp.978-981. 2019.
11. 新井美桜, 金子正弘, 小町守. 日本語学習者向けの文法誤り検出機能付き誤用例文検索システム. 言語処理学会第 25 回年次大会, pp.1097-1100. 2019.
12. 甫立健悟, 金子正弘, 勝又智, 小町守. 文法誤り訂正における単語編集率を用いた訂正度の制御. 言語処理学会第 25 回年次大会, pp.635-638. 2019.

13. 金子正弘, 小町守. 深層言語表現モデルに対するマルチヘッド・多層アテンションによる英語文法誤り検出. 言語処理学会第 25 回年次大会, pp.446-449. 2019.

14. 甫立健悟, 松村雪桜, 勝又智, 金子正弘, 小町守. 敵対的生成ネットワークを用いた文法誤り訂正. NLP 若手の会第 13 回シンポジウム. 2018.

15. 金子正弘, Imankulova Aizhan, 小町守. 転移学習を用いてコンテキストを考慮した系列変換タスクにおけるリランキング. NLP 若手の会第 13 回シンポジウム. 2018.

16. 金子正弘, 小町守. パイプライン処理によるニューラル英語文法誤り検出と訂正. 言語処理学会第 24 回年次大会, pp.576-579. 2018.

17. 金子正弘, 堺澤勇也, 小町守. 英語学習者の文法誤りパターンと正誤情報を考慮した単語分散表現学習. 言語処理学会第 23 回年次大会, pp.729-732. 2017.

Abstract

Representation learning aims to learn grammatical and semantic information from raw datasets as vectors and models. Representation learning is the foundation of natural language processing, where deep learning is the mainstream tool because it can effectively represent textual information as dense numerical data. Representation learning allows to obtain word embeddings and language representations that model word and sentence information. Existing representation learning techniques use information such as co-occurrence of tokens from large scale raw data to learn useful representations. On the other hand, it is known that raw data contains various biases due to an imbalance of grammatical and semantic information, which adversely affects the results of representation learning. In this thesis, I work on resolving the grammatical and semantic biases of representation learning.

(1) Grammatical bias: Most raw datasets written by native speakers are grammatically correct, and there are few grammatical errors contained. Representation learning is not effective in tasks that involve grammatically incorrect text, such as grammatical error detection and grammatical error correction, due to this bias between grammatically correct and incorrect sentences.

(2) Semantic bias: Representation learning is based on co-occurrences, and the bias of co-occurrences of tokens is the cause of learned discriminatory relationships. For example, the doctor is more like “he” than “she”, and the nurse is more like “she” than “he”. This is problematic from fairness and ethical point of view.

In this study, I propose a method for learning word embeddings and language representations to solve grammatical bias, taking into account grammatical correctness and incorrectness of the data. Besides, I propose a method of calculating attentiveness for each layer to use the information in the language representation more effectively in the task of dealing with erroneous sentences. For semantic biases, I propose a debiasing method to solve gender discrimination in word embeddings.

The remainder of this thesis is organized as follows: In chapter 1, I describe the backgrounds of grammatical bias and semantic bias, and related tasks. In chapter 2, I explain the method of reducing the grammatical bias that is learned by word embeddings. In chapter 3, I describe a method for reducing the grammatical bias in language representations. chapter 4 details a method for effectively utilizing grammatical information of language representations, including grammatical bias. Then, in chapter 5, I explain the method of removing the semantic bias from word embeddings. Finally, chapter 6 summarizes this dissertation and describes the future prospects of this research.

Keywords— natural language processing, representation learning, word embeddings, language representation, grammatical error detection, grammatical error correction, discriminative bias

概要

自然言語処理における表現学習は，単語や文の文法や意味などに関する情報をデータからベクトルやモデルとして学習することである．生データは人手で情報を付与していないデータであり，容易に大規模なデータを収集できるため，表現学習の学習データとして用いられることが多い．深層学習が主流である昨今の自然言語処理において表現学習は基盤的な役割を果たしており盛んに研究されている．そして，単語の情報を表現することを単語分散表現，文の情報を表現することを言語表現と呼ぶ．既存の表現学習の多くは単語や文の共起から文法や意味に関する有益な情報を学習している．一方で，生データは単語や文の共起が偏っており，このバイアスが文法や意味の表現学習に悪影響を与えることが知られている．そのため，本研究では表現学習における文法と意味に関するバイアスの問題に取り組む．

(1) 文法バイアス：表現学習に使われているコーパスのほとんどはネイティブにより書かれており，文法的に誤ったテキストはほとんど含まれていない．そのため，コーパスで学習した単語分散表現や言語表現は文法的に誤った表現を考慮することができない．これは文法的に誤ったテキストを扱うタスクにおいて最適な単語分散表現や言語表現になっていないため問題である．文法誤りを含むテキストを扱うタスクとしては，入力文の文法的に誤った箇所を検出する文法誤り検出や入力文の文法的に誤った箇所を文法的に正しく書き換える文法誤り訂正などがある．これらは言語教育や言語学習の支援を行うことができる．

(2) 意味バイアス：表現学習では偏った単語の共起から，差別的な意味表現を学習してしまうことが知られている．例えば，コーパスでは単語 doctor は単語 she より単語 he と共起し，単語 nurse は単語 he より単語 she と共起する．共起を基に学習した単語分散表現では，共起する単語同士

は類似度が高くなるように学習される．そのため，単語 doctor は単語 he と類似度が高く，単語 nurse は単語 she と類似度が高くなる．このように単語分散表現は意味バイアスから差別的な単語の関連性を学習することがある．そして，性差別的な情報を含んだ単語分散表現を用いることで，下流タスクにも性差別的な影響を与えることが知られている．これは公平性や倫理的な観点から問題である．

本研究では，単語分散表現と言語表現の文法バイアスを，表現学習に文法誤りを考慮することで解消する．まず，単語分散表現の学習に擬似的な文法誤りと文法的に正しいテキストの両方を学習に用いる手法を提案する．そして，文法誤り検出により言語表現に事前学習された情報を保持しながら文法誤りを考慮する手法を提案する．さらに，言語表現の文法情報を効果的に活用するために，言語表現の各層に対してアテンションを計算する手法を提案する．アテンションとはどのベクトルを重要視するかを学習する仕組みのことである．意味バイアスに関しては，事前学習された単語分散表現の有益な性別情報を保持しながら性差別バイアスを除去する手法を提案する．この手法はバイアス除去に自己符号化器と回帰モデルを用いている．

文法誤り検出と文法誤り訂正の結果から，提案手法は単語分散表現と言語表現に文法誤りを効果的に考慮できることを示した．さらに，提案手法を用いることで文法誤り検出と文法誤り訂正の 2 つのタスクにおいて世界最高精度を達成した．そして，性別単語とステレオタイプ単語の単語分散表現間の類似度，事前学習された単語分散表現を用いた共参照解析モデルの性別単語とステレオタイプ単語を含む文の予測の均一さ，類似性やアナロジーに関するベンチマークデータセットの結果から，性差別情報を除去する提案手法は差別的ではない性別情報を保持しながら，既存手法と比較して最も性差別バイアスを除去できていることを示した．これらの実験から，本研究は表現学習における文法バイアスと意味バイ

アスの 2 つを解消することに成功した．

本稿では，まず第 1 章で文法バイアスと意味バイアスの背景と関連タスクについて述べる．そして，第 2 章で単語分散表現の文法バイアスを低減する手法について説明する．第 3 章で言語表現の文法バイアスを減少させる手法について述べる．さらに，第 4 章で文法バイアスを含む言語表現の文法情報を効果的に活用する手法について解説する．そして，第 5 章で単語分散表現の意味バイアスを除去する手法について説明する．最後に，第 6 章で本稿の総括とこの研究の今後の展望について述べる．

Keywords— 自然言語処理, 表現学習, 単語分散表現, 言語表現, 文法誤り検出, 文法誤り訂正, 差別バイアス

Contents

List of Figures

List of Tables

1 | Introduction

1.1 Biases in Representation Learning

In recent years, neural networks have been used in a wide range of natural language processing (NLP) tasks (Bengio, Ducharme, Vincent, & Jauvin, 2003; Socher, Pennington, Huang, Ng, & Manning, 2011; Cho et al., 2014). Neural networks in NLP differ from other fields in that it is discrete. On the other hand, neural network models are constructed with continuous values represented by vectors and matrices. Therefore, it is necessary to convert grammatical and semantic information such as words and sentences handled by NLP into vectors and matrices that can be processed by neural networks. In doing so, it is useful to acquire vectors and matrices that successfully represent the similarity and relevance of words and sentences through representation learning based on distributed representation (Hinton, 1984).

A vector that learns grammatical and semantic word information by representation learning is called a word embedding. In recent years, most of the word embeddings learning methods are based on the idea of distributional hypothesis (Harris, 1954), where information of a word is represented from the context of the word. Therefore, the model acquires information about a word by learning its co-occurrence with other words. Word2Vec (Mikolov, Chen, & Dean, 2013) learns word embeddings by predicting the central word from the context of the preceding and following words (CBoW) or by predicting the preceding and following words from the central word (skip-gram). GloVe (Pennington, Socher, & Manning, 2014) learns word embeddings by optimizing for a co-occurrence matrix built from a corpus with a penalty for high-frequency co-occurrence. It is known to improve the performance of downstream tasks on large data sets by using pre-trained word em-

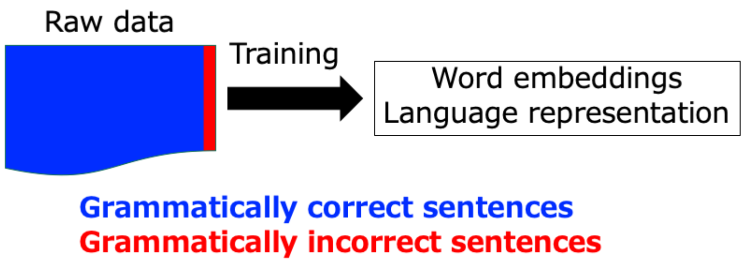

Raw data
Training
Word embeddings
Language representation
Grammatically correct sentences
Grammatically incorrect sentences

Figure 1.1: Grammatical error detection task. The correct part of the input sentence is 0 and the incorrect part is predicted to be 1.

beddings from these methods (Kenter & De Rijke, 2015).

A model that learns the information of a sentence is called a language representation. Dramatic performance improvements have been reported for many tasks by using language representations (Devlin, Chang, Lee, & Toutanova, 2019). Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) is trained by masking part of the input sentences and predicting the masked tokens as well as the adjacent sentences. A Robustly Optimized BERT Pretraining Approach (RoBERTa) (Y. Liu et al., 2019) is a language representation that tunes BERT more effectively.

While word embeddings and language representations learned from the raw corpora are successful in various tasks, these word embeddings and language representations also learn undesirable bias from the raw data. For example, as shown in Figure 1.1, most of the raw data used for representation learning, such as news and Wikipedia (Mikolov, Chen, & Dean, 2013; Pennington et al., 2014; Devlin et al., 2019; Y. Liu et al., 2019), contain mostly grammatically correct sentences and little grammatically incorrect sentences. Therefore, grammatical bias is the bias learned from the distribution of raw data (Wikipedia, web news, etc.) that is biased towards grammatically correct texts so models cannot learn grammatical errors. This bias is not a problem for tasks that involve grammatically correct

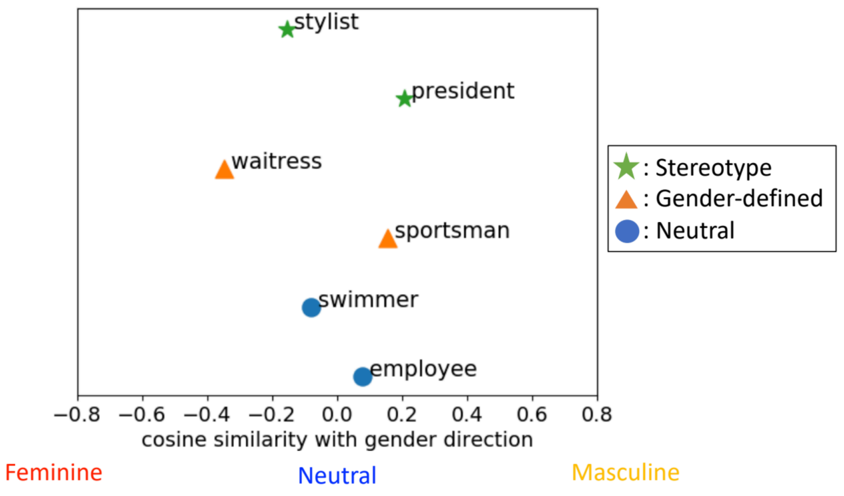


Figure 1.2: Plot of gender bias in GloVe trained on wikipedia corpus.

sentences. Still, it becomes a problem for tasks such as grammatical error detection and grammatical error correction, where grammatically incorrect sentences are given as input, as described in section 1.2 and section 1.3, since word embeddings and language representations are not learned to take grammatical errors into account. To solve this problem, I propose a method of considering grammatically incorrect sentences in word embeddings and language representations for grammatical error detection and grammatical error correction tasks. I also propose a method to handle grammatically incorrect text for language representations more effectively.

Grammatical bias is not the only bias that can be learned from raw data. Representation learning learns semantics from co-occurrences, therefore, it also learns discriminatory information from biased co-occurrences, I refer to it as semantic bias. For example, the word “stylist” frequently co-occurs with female words such as “she” and “woman”, and the word ”president” with male words such as “he” and “man”. Thus, “stylist” is more similar to a female word and “president” is more similar to a male word. It is crucial to remove semantic biases from word embed-

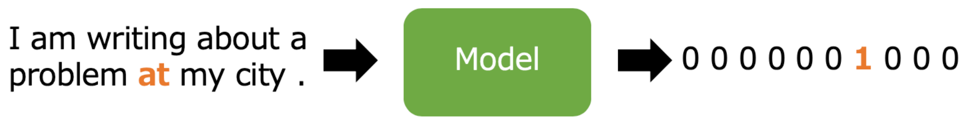


Figure 1.3: Grammatical error detection task. The correct part of the input sentence is 0 and the incorrect part is predicted to be 1.

dings because learning discriminatory expressions can have a discriminatory effect on the downstream tasks (J. Zhao et al., 2019). Figure 1.2 shows the gender bias in GloVe trained on Wikipedia corpus. Cosine similarity closer to 1 shows higher similarity to male words, closer to -1 shows higher similarity to female words, and closer to 0 shows that gender information is not retained. This figure shows that the stereotyped words contain the same amount of gender information as gender words. This indicates that the existing word embeddings contain bias and need to be debiased. On the other hand, we need to keep the non-discriminative semantic information and remove the discriminative semantic information by debiasing the word embeddings. This study proposes a debiasing method that preserves non-discriminative gender-related information while removing stereotypical discriminative gender biases from pre-trained word embeddings.

By removing these biases from the representation learning, it is possible to obtain models that are are robust to grammatical errors and models that are legally and socially acceptable. Therefore, this research will lead to the realization of artificial intelligence that utilize languages in a trusted by humans way.

1.2 Grammatical Error Detection

Grammatical error detection (GED) model can identify the location of errors, which is useful for second language learners and teachers. Figure 1.3 shows an overview of the GED task. The GED model, given grammatically incorrect text as input, predicts its grammatically incorrect parts. It can be seen as a sequence labeling task, typically solved by a supervised approach (Rei, Felice, Yuan, & Briscoe, 2017; Kasewa, Stenetorp, & Riedel, 2018).

Rei and Yannakoudakis (2016) first proposed the GED model using neural networks. Since this study showed that bidirectional long short-term memory (Bi-LSTM) (Graves & Schmidhuber, 2005) is effective in GED, various Bi-LSTM-based GED studies have been proposed (Rei et al., 2017; Kasewa et al., 2018). Bi-LSTM is a neural network model that combines forward and backward LSTM (Hochreiter & Schmidhuber, 1997), a gated recursive neural network. Because of the small size of the GED training data, pre-trained word embeddings have been used in many studies (Rei et al., 2017; Rei & Søgaard, 2018). On the other hand, these word embeddings are trained from the raw data and cannot take into account grammatical errors. To consider grammatical errors in word embeddings, I propose a method of learning word embeddings with pseudo-grammatical errors.

Recently, language representations have been used in various tasks in NLP (Devlin et al., 2019; Y. Liu et al., 2019; Yang et al., 2019; Clark, Luong, Le, & Manning, 2020). Language representations learn sentence information from a large amount of raw data. Unlike language models, these models use masks (Devlin et al., 2019; Y. Liu et al., 2019; Clark et al., 2020), and word order swapping (Yang et al., 2019) to learn text information from both directions. The pre-trained language representations are fine-tuned in the downstream tasks by replacing only the output layers. Language representations have been particularly successful in the classification tasks, and the sequence labeling tasks, therefore, are considered suitable for the GED.

On the other hand, language representations are also trained on raw corpora with few grammatical errors, and there is a gap between raw data and GED data for

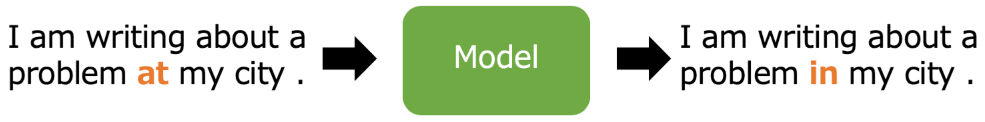

I am writing about a problem at my city .
Model
I am writing about a problem in my city .

Figure 1.4: Grammatical error correction task. The grammatically incorrect word “at” in the input sentence is corrected to the grammatically correct word “in”.

pre-training and fine-tuning. Therefore, this study aims to effectively learn grammatically incorrect texts in language representation in the GED. I propose a method to obtain the optimal hidden layer for GED by calculating the attention for each layer of language representations. The results showed that it is effective to compute the attention to the hidden layers of language representations, and it achieved state-of-the-art results in GED.

1.3 Grammatical Error Correction

GEC is a sequence generation task where a potentially grammatically incorrect text is given as input, and it is rewritten to be a grammatically correct text. Figure 1.4 illustrates the GEC process. The encoder-decoder (EncDec) (Cho et al., 2014) architecture is commonly used as a GEC model (W. Zhao, Wang, Shen, Jia, & Liu, 2019; Grundkiewicz, Junczys-Dowmunt, & Heafield, 2019; Kiyono, Suzuki, Mita, Mizumoto, & Inui, 2019). EncDec consists of encoder and decoder neural networks. The encoder encodes a sequence into a hidden representation and the decoder decodes the representation into another sequence.

The aforementioned language representations are also used in sequence generation tasks using EncDec. For example, Lample and Conneau (2019) showed that initializing the weights of EncDec with the weights of language representations improves the performance of the machine translation. Zhu et al. (2020) proposed a method to feed the output of language representations to the EncDec model as additional features and reported an improvement in machine translation performance.

However, it is not obvious whether these methods can be used in GEC as they are. For example, the distribution of the inputs to a GEC model can be considerably different (erroneous, clumsy, etc.) from that of the corpora used for training language representations; however, this issue is not addressed in the previous methods. Therefore, I investigate a method to combine language representations and the GEC EncDec model. Experiments show that the proposed method, where I first fine-tune language representations with a given GED corpus and then use the output of the fine-tuned language representations as additional features in the GEC model, maximizes the benefit of the language representations.

1.4 Main Contributions

The main contributions of this study are summarized as follows:

- I proposed a training method for word embeddings that takes grammatical errors into account and showed its effectiveness in the GED task in chapter 2. Furthermore, the state-of-the-art GED model initialized with these word embeddings achieved the state-of-the-art results of its time. The code[1] used in the experiments and the demonstration[2] are publicly available.

- I proposed an effective method for removing grammatical bias from pre-trained language representations in chapter 3. The experiments show that debiased language representations obtained by fine-tuning on GED and combining them with the GEC EncDec model is effective. Furthermore, this

[1]`https://github.com/kanekomasahiro/grammatical-error-detection`
[2]`http://kanekomasahiro.sakura.ne.jp/revision_support/`

model achieved state-of-the-art in GEC. The code[3] used in the experiments is publicly available.

- I proposed a method to learn the most appropriate grammatical information from the language representations by calculating the attention of each layer of the language representations to reduce the influence of grammatical bias in chapter 4. And the analysis revealed that it is effective to use the information of all layers of language representations for GED. Furthermore, this method achieved state-of-the-art in four GED data sets.
- I proposed a method for pre-trained word embeddings that retains useful information and removes discriminatory gender information in chapter 5. Experimental results showed that the proposed method was the most debiasing while retaining the pre-trained information such as word similarity and analogy. The code and debiased word embeddings[4] used in the experiments are publicly available.

[3]https://github.com/kanekomasahiro/bert-gec
[4]https://github.com/kanekomasahiro/gp_debias

2 | Error- and Grammaticality-Specific Word Embeddings

The aim of this chapter is to reduce grammatical bias in GED by learning word embeddings that take into account grammaticality and error patterns. Most GED studies using neural networks initialize the model with pre-trained word embeddings to take advantage of large scale data. For example, Rei and Yannakoudakis (2016) achieved the state-of-the-art accuracy in English grammatical error detection using a Bi-LSTM. Their approach uses word embeddings learned from a large-scale native corpus to address the data sparseness problem of learner corpora.

However, most of the word embeddings, including the one used by Rei and Yannakoudakis (2016), model only the context of the words from a raw corpus written by native speakers, and do not consider specific grammatical errors of language learners. This leads to the problem wherein the word embeddings of correct and incorrect expressions tend to be similar so that the classifier must decide grammaticality of a word from contextual information with a similar input vector.

To address this problem, I introduce two methods: 1) grammaticality-specific word embeddings (GWE), which employ grammatical error patterns, that is to say the word pairs that learners tend to easily confuse; 2) error-specific word embeddings (EWE), which consider grammatical correctness of n-grams. In this thesis, I use the term grammaticality to refer to the correct or incorrect label of the target word given its surrounding context. I also combine these methods, which I will refer to as error-and grammaticality-specific word embeddings (E&GWE).

Furthermore, I conducted experiments using the large-scale Lang-8[1] English learner

[1] http://lang-8.com/

corpus. The results demonstrated that representation learning is crucial for exploiting a noisy learner corpus for GED.

2.1 Related Works

Many studies on GED try to address specific types of grammatical errors (Tetreault & Chodorow, 2008; Han, Chodorow, & Leacock, 2006; Kochmar & Briscoe, 2014). In contrast, Rei and Yannakoudakis (2016) target all errors using a Bi-LSTM, whose embedding layer is initialized with word2vec. I also address unrestricted GED; however, I focus on learning word embeddings that consider a learner's error pattern and grammaticality of the target word. In this dissertation, subsequently, the word embeddings give statistically significant improvements over their method using exactly the same training data.

Several studies considering grammatical error patterns in language learning have been performed. For example, Sawai, Komachi, and Matsumoto (2013) suggest correction candidates for verbs using the learner error pattern, and X. Liu, Han, Li, Stiller, and Zhou (2010) automatically correct verb selection errors in English essays written by Chinese students learning English, based on the error patterns created from a synonym dictionary and an English-Chinese bilingual dictionary. The main difference between these previous studies and mine is that the previous studies focused only on verb selection errors.

As an example of research on learning word embeddings that consider grammaticality, Alikaniotis, Yannakoudakis, and Rei (2016) proposed a model for constructing word embeddings by considering the importance of each word in predicting a quality score for an English learner's essay. Their approach learns word embedding from a document-level score using the mean square error whereas I learn word embeddings from a word-level binary error information using the hinge loss.

The use of a large-scale learner corpus on GEC is described in works such as Z. Xie, Avati, Arivazhagan, Jurafsky, and Ng (2016) and Chollampatt, Hoang, and Ng (2016); Chollampatt, Taghipour, and Ng (2016). These studies used the Lang-8 corpus as training data for phrase-based machine translation (Z. Xie et al., 2016) and neural network joint models (Chollampatt, Hoang, & Ng, 2016; Chollampatt, Taghipour, & Ng, 2016). In this study, Lang-8 was used to extract error patterns that were then utilized to learn word embeddings. The experiments show that Lang-8 cannot be used as a reliable annotation for LSTM-based classifiers. Instead, I need to extract useful information as error patterns to improve the performance of error detection.

2.2 C&W Embeddings

These models extend an existing word embedding learning algorithm called C&W Embeddings (Collobert & Weston, 2008) and learn word embeddings that consider grammatical error patterns and grammaticality of the target word. I first describe the well-known C&W embeddings, and then explain the extensions.

Collobert and Weston (Collobert & Weston, 2008; Collobert et al., 2011) proposed a window-based neural network model that learns distributed representations of target words based on the local context. Here, target word $\boldsymbol{w}_t$ is the central word in the window sized sequence of words $\mathcal{S} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t, \ldots, \boldsymbol{w}_n)$. The representation of the target word $\boldsymbol{w}_t$ is compared with the representations of other words that appear in the same sequence $(\forall \boldsymbol{w}_i \in \mathcal{S} | \boldsymbol{w}_i \neq \boldsymbol{w}_t)$. A negative sample $\mathcal{S}' = (\boldsymbol{w}_1, ..., \boldsymbol{w}_c, ..., \boldsymbol{w}_n | \boldsymbol{w}_c \sim \mathcal{V})$ is created by replacing the target word $\boldsymbol{w}_t$, where $\boldsymbol{w}_t \neq \boldsymbol{w}_c$, with a randomly selected word from the vocabulary $\mathcal{V}$ to distinguish between the negative sample $\mathcal{S}'$ and the original word sequence $\mathcal{S}$. In their method, the word sequence $\mathcal{S}$ and the negative sample $\mathcal{S}'$ are converted into vectors in the embedding layer, which are fed as embeddings. They concatenate each converted vector and treat it as input vector $\boldsymbol{x}$. The input vector $\boldsymbol{x}$ is then subjected to a linear transformation (Eq. 2.1) to calculate the vector $\boldsymbol{i}$ of the hidden layer. Then, the resulting vector is subjected to another linear transformation (Eq. 2.2) to obtain the output $f(\boldsymbol{x})$.

$$\boldsymbol{i} = \sigma(\mathbf{W}_{hx}\boldsymbol{x} + \boldsymbol{b}_h) \quad (2.1)$$

$$f(\boldsymbol{x}) = \mathbf{W}_{oh}\boldsymbol{i} + \boldsymbol{b}_o \quad (2.2)$$

Here, $\mathbf{W}_{hx}$ is the weight matrix between the input vector and the hidden layer, $\mathbf{W}_{oh}$ is the weight matrix between the hidden layer and the output layer, $\boldsymbol{b}_o$ and $\boldsymbol{b}_h$ are biases, and σ is an element-wise nonlinear function tanh.

This model for word representation learns distributed representations by making the ranking of the original word sequence S higher than that of the negative samples S', which includes noise due to replaced words. The difference between the original word sequence and the word sequence including noise is optimized to be at least 1.

$$\text{loss}_c(\mathcal{S}, \mathcal{S}') = \max(0, 1 - f(\boldsymbol{x}) + f(\boldsymbol{x}')) \quad (2.3)$$

Here, $\boldsymbol{x}'$ is an embedding of the word $\boldsymbol{w}_c$.

The proposed models learn distributed representations using the same hinge loss (Equation 2.3) so the model could distinguish between correct and incorrect phrase pairs.

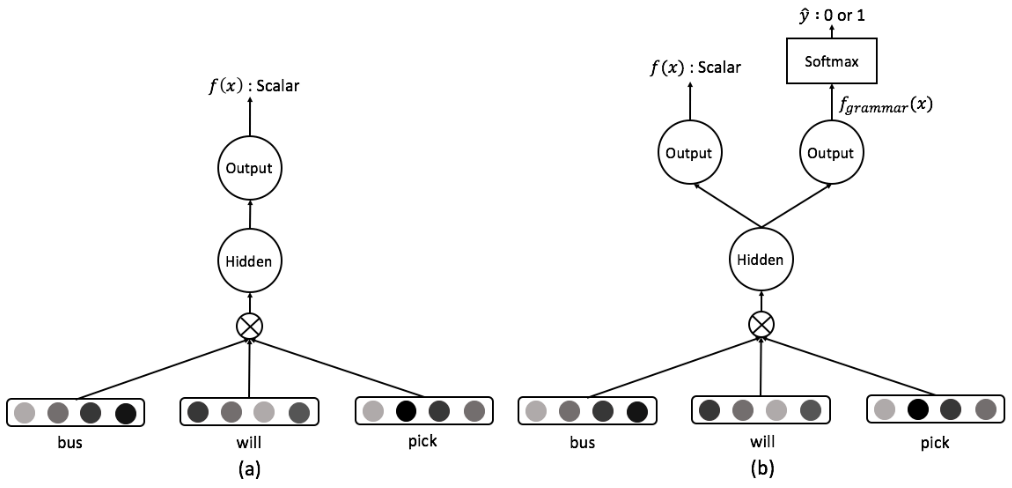


Figure 2.1: Architecture of the learning methods for word embeddings (a) EWE and (b) GWE. Both models concatenate the word vectors of a sequence for window size and feed them into the hidden layer. Then, EWE outputs a scalar value, and GWE outputs a prediction of the scalar value and the label of the word in the middle of the sequence.

2.3 Grammaticality-Specific Word Embeddings

GWE learns word embeddings using the same model as C&W embeddings. However, rather than creating negative samples randomly, I created them by replacing the target word $\boldsymbol{w}_t$ with words $\boldsymbol{w}_c$ that learners tend to easily confuse with the target word $\boldsymbol{w}_t$. In such a case, $\boldsymbol{w}_c$ is sampled by the conditional probability:

$$P(\boldsymbol{w}_c|\boldsymbol{w}_t) = \frac{\text{count}(\boldsymbol{w}_c, \boldsymbol{w}_t)}{\sum_{\boldsymbol{w}'_c} \text{count}(\boldsymbol{w}'_c, \boldsymbol{w}_t)} \tag{2.4}$$

where, $\boldsymbol{w}_t$ is a target word, $\boldsymbol{w}'_c$ is a set of $\boldsymbol{w}_c$ regarding $\boldsymbol{w}_t$, count function calculates the frequency of their co-occurrence.

This model learns to distinguish between a correct and an incorrect word by considering grammatical errors of learners. Replacement candidates, treated as error patterns, are extracted from a learner corpus annotated with correction. Figure 2.1

(a) represents architecture of GWE.

*The bus will pick you up right at your hotel **entery/*entrance**.*

The above sentence is a simple example from the test data of FCE-public corpus. In this sentence, the word “entery” is incorrect and the “entrance” is the correct word. In this case, $\boldsymbol{w}_t$ is “entrance” and $\boldsymbol{w}_c$ is “entery”. Note that I use only one-to-one (substitution) error patterns.

Due to the data sparseness problem, the context of infrequent words cannot be properly learned. This problem is solved by using a large corpus to pre-train word2vec. By fine-tuning vectors whose contexts have already been learned, it is possible to learn word embeddings with no or few replacement candidates in a learner corpus.

2.4 Error-Specific Word Embeddings

Similar to the approach of Alikaniotis et al. (2016) for essay score prediction, I extend C&W embeddings to distinguish between correct words and incorrect words by considering errors in distributed representations (Figure 2.1 (b)). For that purpose, I add an additional output layer to predict correct score of word sequences, and extend Equation 2.3 to calculate following two error functions.

$$f_{\text{grammar}}(\boldsymbol{x}) = \mathbf{W}_{gh}\boldsymbol{i} + \boldsymbol{b}_g \tag{2.5}$$

$$\hat{\boldsymbol{y}} = \mathit{softmax}(f_{\text{grammar}}(\boldsymbol{x})) \tag{2.6}$$

$$\text{loss}_p(\mathcal{S}) = -\sum \boldsymbol{y} \cdot \log(\hat{\boldsymbol{y}}) \tag{2.7}$$

$$\text{loss}(\mathcal{S}, \mathcal{S}') = \alpha \cdot \text{loss}_c(\mathcal{S}, \mathcal{S}') + (1-\alpha) \cdot \text{loss}_p(\mathcal{S}) \tag{2.8}$$

In Equation 2.5, f_{grammar} is the predicted label of the original word sequence S. $\mathbf{W}_{gh}$ is the weight matrix and $\boldsymbol{b}_g$ is the bias. In Equation 2.6, the prediction probability $\hat{\boldsymbol{y}}$ is computed using the softmax function for $f_{grammar}$. The error loss_p is computed using the cross-entropy function using the gold label's vector $\boldsymbol{y}$ of the target word (Equation 2.7). Finally, two errors are combined to calculate loss (Equation 2.8). Here, α is a hyperparameter that determines the weight of the two error functions.

I use the original tag label (0/1) of the FCE-public data as the correct score of word sequences for learning. Note that I do not use label information from Lang-8, because the error annotation of Lang-8 error annotations are too noisy to train an error detection model directly from the corpus. Negative examples of EWE are created randomly, that are similar to the case with C&W.

2.5 Error- and Grammaticality-Specific Word Embeddings

E&GWE is a model that combines EWE and GWE. In particular, E&GWE model creates negative examples using an grammatical error as in GWE and outputs score and predicts correct score as in EWE.

2.6 Bidirectional LSTM

I use bidirectional LSTM (Bi-LSTM) (Graves & Schmidhuber, 2005) as a classifier for all these experiments for English grammatical error detection, because Bi-LSTM demonstrates the state-of-the-art accuracy for this task compared to other architectures such as CRF and CNNs (Rei & Yannakoudakis, 2016).

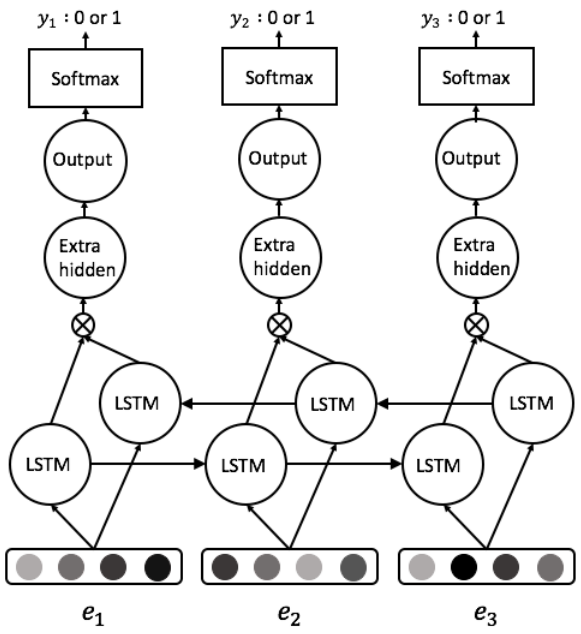


Figure 2.2: A bidirectional LSTM network. The word vectors e_i enter the hidden layer to predict the labels of each word.

The LSTM calculation is expressed as follows:

$$\boldsymbol{i}_t = \sigma(\mathbf{W}_{ie}e_t + \mathbf{W}_{ih}\boldsymbol{h}_{t-1} + \mathbf{W}_{ic}\boldsymbol{c}_{t-1} + \boldsymbol{b}_i) \tag{2.9}$$

$$\boldsymbol{f}_t = \sigma(\mathbf{W}_{fe}\boldsymbol{e}_t + \mathbf{W}_{fh}\boldsymbol{h}_{t-1} + \mathbf{W}_{fc}\boldsymbol{c}_{t-1} + \boldsymbol{b}_f) \tag{2.10}$$

$$\boldsymbol{c}_t = \boldsymbol{i}_t \odot g(\mathbf{W}_{ce}\boldsymbol{e}_t + \mathbf{W}_{ch}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} \tag{2.11}$$

$$\boldsymbol{o}_t = \sigma(\mathbf{W}_{oe}\boldsymbol{e}_t + \mathbf{W}_{oh}\boldsymbol{h}_{t-1} + \mathbf{W}_{oc}\boldsymbol{c}_t + \boldsymbol{b}_o) \tag{2.12}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \boldsymbol{h}(c_t) \tag{2.13}$$

Here, $\boldsymbol{e}_t$ is the word embedding of word $\boldsymbol{w}_t$, and $\mathbf{W}_{ie}$, $\mathbf{W}_{fe}$, $\mathbf{W}_{ce}$ and $\mathbf{W}_{oe}$ are weight matrices. Each $\boldsymbol{b}_i$, $\boldsymbol{b}_f$, $\boldsymbol{b}_c$ and $\boldsymbol{b}_o$ are biases. An LSTM cell block has an input gate $\boldsymbol{i}_t$, a memory cell $\boldsymbol{c}_t$, a forget gate $\boldsymbol{f}_t$ and an output gate $\boldsymbol{o}_t$ to control information flow. In addition, g and h are the sigmoid function and σ is the tanh. $\odot$ is the pointwise multiplication.

I apply a bidirectional extension of LSTM, as shown in Figure 2.2, to encode the word embedding $\boldsymbol{e}_i$ from both left-to-right and right-to-left directions.

$$\boldsymbol{y}_t = \mathbf{W}_{yh}(\boldsymbol{h}_t^L \otimes \boldsymbol{h}_t^R) + \boldsymbol{b}_y \tag{2.14}$$

The Bi-LSTM model maps each word $\boldsymbol{w}_t$ to a pair of hidden vectors $\boldsymbol{h}_t^L$ and $\boldsymbol{h}_t^R$, i.e., the hidden vector of the left-to-right LSTM and right-to-left LSTM, respectively. $\otimes$ is the concatenation. $\mathbf{W}_{yh}$ is a weight matrix and $\boldsymbol{b}_y$ is a bias. I also added an extra hidden layer for linear transformation between each of the composition function and the output layer, as discussed in the previous study.

2.7 Experiments

2.7.1 Settings

I used the FCE-public dataset and the Lang-8 English learner corpus to train classifiers and word embeddings. For this evaluation, I used the test set from the FCE-public dataset (Yannakoudakis, Briscoe, & Medlock, 2011) for all experiments.

FCE-public dataset. First, I compared the proposed methods (EWE, GWE, and E&GWE) to previous methods (W2V and C&W) relative to training word embeddings (see Table 2.1). For this purpose, I trained the word embeddings and a classifier, which were initialized using pre-trained word embeddings, with the training set from the FCE-public dataset.

This dataset is one of the most famous English learner corpus in grammatical error correction. It contains essays written by English learners. It is annotated with grammatical errors along with error classification. I followed the official split of the data: $30,953$ sentences as a training set, $2,720$ sentences as a test set, and

$2,222$ sentences as a development set. In the FCE-public dataset, the number of target words of error patterns is 4,184, the number of tokens of the replacement candidates is 9,834, and the number of types (unique words) is 6,420. All manually labeled words in the FCE-public dataset were set as the gold target to train the EWE. For a missing word error, an error label is assigned to the word immediately after the missing word (see Table 2.4 (c)). To prevent overfitting, singleton words in the training data were taken as unknown words.

Lang-8 corpus. Furthermore, I added the large-scale Lang-8 English learner corpus to the FCE-public dataset to train word embeddings (FCE+GWE-L8 and FCE+E&GWE-L8) to explore the effect of a large data on the proposed methods. I used a classifier trained using only the FCE-public dataset whose word embeddings were initialized with the large-scale pre-trained word embeddings to compare the results with those of a classifier trained directly using a noisy large-scale data whose word embeddings were initialized using word2vec (FCE&L8+W2V, see Table 2.2).

Lang-8 learner corpus has over 1 million manually annotated English sentences written by ESL learners. Extraction of error patterns from Lang-8 in the process of creating negative samples to train word embeddings was performed as follows:

1. Extract word pairs using the dynamic programming from a correct sentence and an incorrect sentence.
2. If the learner's word of the extracted word pair is included in the vocabulary created from FCE-public, include it to the error patterns.

In the Lang-8 dataset the number of types of target words of the replacement candidates is 10,372, the number of tokens of the replacement candidates is 272,561, and the number of types is 61,950.

The experiments on FCE+GWE-L8 and FCE+E&GWE-L8 were conducted by combining error patterns from all of Lang-8 corpus and the training part of FCE-public corpus to train word embeddings. However, since the number of error patterns of Lang-8 is larger than that of FCE-public, I normalized each frequency so that the ratio was 1:1.

I use $F_{0.5}$ as the main evaluation measure, following a previous study (Rei & Yannakoudakis, 2016). This measure was also adopted in the CoNLL-14 shared task on error correction task (Ng et al., 2014). It combines both precision and recall, while assigning twice as much weight to precision because accurate feedback is often more important than coverage in error detection applications (Nagata & Nakatani, 2010). Nagata and Nakatani (2010) presented a precision-oriented error detection system for articles and numbers that demonstrated precision of 0.72 and a recall of 0.25 and achieved a learning effect that is comparable to that of a human tutor.

2.7.2 Word Embeddings

I set parameters for word embeddings according to the previous study (Rei & Yannakoudakis, 2016). The dimension of the embedding layer of C&W, GWE, EWE and E&GWE is 300 and the dimension of the hidden layer is 200. I used a publicly released word2vec vectors (Chelba et al., 2013) trained on the News crawl from Google news[2] as pre-trained word embeddings. I set other parameters in this model by running a preliminary experiment in which the window size is 3, the number of negative samples is 600, the linear interpolation α is 0.03, and the optimizer is the Adam algorithm (Kingma & Ba, 2015) with the initial learning rate of 0.001. GWE is initialized randomly and EWE is initialized using pre-trained word2vec

[2]https://github.com/mmihaltz/word2vec-GoogleNews-vectors

Bi-LSTM + embeddings	**P**	**R**	**$F_{0.5}$**
FCE + W2V (R&Y, 2016)	46.1	28.5	41.1
FCE + W2V (our reimplementation of (R&Y, 2016))	45.8±0.1	27.8±0.4	40.5±0.3
FCE + C&W	45.1±0.3	26.7±0.4	39.6±0.3
FCE + GWE	46.1±0.1⋆	28.0±0.1⋆	40.8±0.1⋆
FCE + EWE	46.5±0.1⋆	28.3±0.4⋆	41.2±0.2⋆
FCE + E&GWE	**46.7**±0.1⋆	**28.6**±0.1⋆	**41.4**±0.1⋆

Table 2.1: LSTM and word embeddings are trained only using FCE-public.

Bi-LSTM + embeddings	**P**	**R**	**$F_{0.5}$**
FCE&L8 + W2V	12.3±2.6	32.8±2.2	14.0±2.6
FCE + GWE-L8	50.5±3.4⋆	**30.1**±1.2⋆	44.4±2.7⋆
FCE + E&GWE-L8	**50.8**±3.6⋆	30.0±1.2⋆	**44.6**±2.8⋆

Table 2.2: Either FCE-public and a large-scale Lang-8 corpus are used to train LSTM or word embeddings.

2.7.3 Classifier

I use EWE, GWE, and E&GWE word embeddings to initialize the Bi-LSTM neural network, and predict the correctness of the target word in the input sentence. I update initialized weights of embedding layer while training classifiers, since it showed better results. The parameters and settings of the network are the same as in a previous study (Rei & Yannakoudakis, 2016). Specifically, in Bi-LSTM the dimensions of the embedding layer, the first hidden layer, and the second hidden layer are 300, 200, and 50, respectively. The Bi-LSTM model was optimized using the Adam algorithm (Kingma & Ba, 2015) with an initial learning rate of 0.001, and a batch size of 64 sentences.

2.7.4 Results

Table 2.1 shows experimental results comparing Bi-LSTM models trained on FCE-public dataset initialized with two baselines (FCE+W2V and FCE+C&W) and the proposed word embeddings (FCE+GWE, FCE+EWE and FCE+E&GWE) in the

	Error type	Verb	Missing-article	Noun	Noun type
(a)	FCE + W2V	56	**48**	26	9
	FCE + C&W	53	46	24	7
(b)	FCE + GWE	60	37	29	12
	FCE + EWE	62	43	29	11
	FCE + E&GWE	64	40	31	14
(c)	FCE + GWE-L8	66	36	37	**19**
	FCE + E&GWE-L8	**67**	40	**39**	18
	Total number of errors	131	112	77	32

Table 2.3: Numbers of correct instances for typical error types.

	Bi-LSTM + embeddings	Detection result
(a)	Gold	The bus will pick you up right at your hotel *entrance*.
	FCE + W2V	The bus will pick you up right at your hotel entery.
	FCE + E&GWE-L8	The bus will pick you up right at your hotel **entery**.
(b)	Gold	There are shops which *sell clothes*, *food*, and books $\cdots$
	FCE + W2V	There are shops which sales cloths, foods, and books $\cdots$
	FCE + E&GWE-L8	There are shops which sales cloths, **foods**, and books $\cdots$
(c)	Gold	All the buses and *the MTR* have air-condition.
	FCE + W2V	All the buses and **MTR** have air-condition.
	FCE + E&GWE-L8	All the buses and MTR have air-condition.

Table 2.4: Examples of error detection by FCE+W2V and FCE+E&GWE-L8. Gold corrections in *italic*, and detected errors in **bold**.

error detection task. I used two models for FCE+W2V: FCE+W2V (R&Y 2016) is the experimental result reported in a previous study (Rei & Yannakoudakis, 2016), and FCE+W2V (my reimplementation of (R&Y, 2016)) is the experimental result of my reimplementation of Rei and Yannakoudakis (2016). FCE+E&GWE is a model combining FCE+GWE and FCE+EWE. I conducted Wilcoxon signed rank test ($p \leq 0.05$) 5 times.

Table 2.2 shows the result of using additional large-scale Lang-8 corpus. Compared to FCE&L8+W2V, FCE+GWE-L8 has better results within the three evaluation metrics. From this result, it can be seen that it is better to extract and use error patterns than simply using Lang-8 corpus as a training data to train a classifier, as it contains noise in the correct sentences. Furthermore, by combining with EWE method, accuracy was improved as in the above experiment.

In terms of precision, recall, and $F_{0.5}$, the methods in this study were ranked as FCE+E&GWE-L8 $>$ FCE+GWE-L8 $>$ FCE+E&GWE $>$ FCE+EWE $>$ FCE+GWE $>$ FCE+W2V $>$ FCE+C&W. Error patterns and grammaticality consistently improved the accuracy of grammatical error detection, showing that the proposed methods are effective. Also, the proposed method has a statistically significant difference compared with previous research even without using large-scale Lang-8 corpus. It outperformed the preceding state-of-the-art (Rei & Yannakoudakis, 2016) in all evaluation metrics.

2.8 Discussion

Table 2.3 shows the number of correct answers of each model for some typical errors. Error types are taken from the gold label of the FCE-public dataset.

First, I analyze verb errors and missing articles, which have the largest differences between the numbers of correct answers of baselines and the proposed methods (see Table 2.3 (a) and (b)). The proposed methods gave more correct answers for verb errors, whereas FCE+W2V and FCE+C&W gave more correct answers for missing article errors. A possible explanation is that unigram-based error patterns are too powerful for word embeddings to learn other errors that can be learned from the contextual clues.

Second, I examine the difference made by adding the error patterns extracted from Lang-8 (see Table 2.3 (b) and (c)): FCE+GWE and FCE+GWE-L8 have the greatest difference in the number of correct answers in noun and noun type errors. FCE+GWE-L8 has more correct answers for noun errors such as *suggestion* and *advice* and noun type errors such as *time* and *times*. The reason is that Lang-8 includes a wide variety of lexical choice errors of nouns while FCE-public covers only a limited number of error variations.

Table 2.4 demonstrates the examples of error detection of the baseline FCE+W2V and the best proposed method FCE+E&GWE-L8 on the test data. Table 2.4 (a) shows an example of a noun error, and as it can be seen, FCE+E&GWE-L8 detected the error in contrast to FCE+W2V. Noun type errors are presented in Table 2.4 (b). Here, FCE+W2V did not detect any error, while FCE+E&GWE-L8 could detect the mass noun error, frequently found in a learner corpus. Detection of "sale" and "cloths" was failed in both models, but they are hard to detect since the former requires syntactic information and the latter involves common knowledge. In Table 2.4 (c), FCE+W2V succeeded in detection of a missing article error, but FCE+E&GWE-L8 did not. Even though proposed word embeddings learn substitution errors effectively, they cannot properly learn insertion and deletion errors. It is a future work to extend word embeddings to include these types of errors and focus on contextual errors that are difficult to deal with the model, for example, missing articles.

3 | Error and Grammaticality-Specific Language Representations

In this chapter, we propose a method to reduce the grammatical bias of language representations in GEC by fine-tuning. Numerous studies on GEC have successfully used EncDec based models, and in fact, most current state-of-the-art neural GEC models employ this architecture (W. Zhao et al., 2019; Grundkiewicz et al., 2019; Kiyono et al., 2019). Language representations are generally used in downstream tasks (Y. Liu, 2019; H. Zhang et al., 2019). For example, in neural machine translatoion, Zhu et al. (2020) demonstrated that it is more effective to provide the output of the final layer of a language representation to the EncDec model as features.

In light of this trend, one natural, intriguing question is whether neural EncDec GEC models can benefit from the recent advances of language representations since language representations such as BERT (Devlin et al., 2019) have been shown to yield substantial improvements in a variety of NLP tasks (Qiu et al., 2020). BERT, for example, builds on the Transformer architecture (Vaswani et al., 2017) and is trained on large raw corpora to learn general representations of linguistic components (e.g., words and sentences) in context, which have been shown useful for various tasks. In recent years, language representations have been used not only for classification and sequence labeling tasks but also for language generation, where combining a language representation with EncDec models of a downstream task makes a noticeable improvement (Lample & Conneau, 2019). However, grammatical biases are also learned in language representations, indicating that they are not robust to grammatical errors (Yin, Long, Meng, & Chang, 2020). Therefore, it is necessary to reduce the grammatical bias of language representations and combine them with the EncDec model.

To combine the EncDec model with the language representations, I use the fusion (fuse) method, which has shown the best performance in machine translation (Zhu et al., 2020). In the fuse method, pre-trained representations of a language representation are used as additional features during the training of a task-specific model. When applying this method for GEC, what the language representations has learned in pre-training will be preserved; however, the language representations will not be adapted to either the GEC task or the task-specific distribution of inputs (i.e., erroneous sentences in a learner corpus), which also preserve grammatical bias. It may hinder the GEC model from effectively exploiting the potential of the language representations. Given these drawbacks, it is not as straightforward to gain the advantages of language representations in GEC as one might expect. Therefore, I propose a method to reduce the grammatical bias of language representations, and aim to improve the performance of the GEC model.

In this investigation, I employ BERT, which is a widely used language representation (Qiu et al., 2020), and evaluate the following three methods: (a) initialize an EncDec GEC model using pre-trained BERT as in Lample and Conneau (2019) (BERT-init), which is initialized with the parameters of a pre-trained language representations and then is trained over a task-specific training set (Lample & Conneau, 2019; Rothe, Narayan, & Severyn, 2019), as a baseline, (b) pass the output of pre-trained BERT into the EncDec GEC model as additional features (BERT-fuse) (Zhu et al., 2020), and (c) combine the best parts of (a) and (b).

In this new method (c), I first fine-tune BERT with the GEC corpus and then use the output of the fine-tuned BERT model as additional features in the GEC model. To implement this, I further consider two options: (c1) additionally train pre-trained BERT with GEC corpora (BERT-fuse mask) for removing grammatical bias, and (c2) fine-tune pre-trained BERT by way of the grammatical error detection (GED) task (BERT-fuse GED). In (c2), I expect that the GEC model will be trained so that it can leverage both the representations learned from large general corpora (pre-trained BERT) and the debiased task-specific information useful for GEC induced from the GEC training data.

These experiments show that using the output of the fine-tuned BERT model with removed grammatical bias as additional features in the GEC model (method (c)) is the most effective way of using BERT in most of the GEC corpora that I used in the experiments. I also show that the performance of GEC improves further by combining the BERT-fuse mask and BERT-fuse GED methods. The best-performing model achieves state-of-the-art results on the BEA-2019 and CoNLL-2014 benchmarks.

3.1 Related Work

Studies have reported that a language representation can improve the performance of GEC when it is employed either as a re-ranker (Chollampatt, Wang, & Ng, 2019; Kaneko, Hotate, Katsumata, & Komachi, 2019) or as a filtering tool (Asano, Mita, Mizumoto, & Suzuki, 2019; Kiyono et al., 2019). EncDec-based GEC models combined with language representations can also be used in combination with these pipeline methods. Kantor et al. (2019) and Awasthi, Sarawagi, Goyal, Ghosh, and Piratla (2019) proposed sequence labeling models based on correction methods. The proposed method can utilize the existing EncDec GEC knowledge, but these methods cannot be utilized due to the different architecture of the model. Besides, to the best of my knowledge, no research has yet been conducted that incorporates information of language representations for effectively training the EncDec GEC model.

Language representations are generally used in downstream tasks by fine-tuning (Y. Liu, 2019; H. Zhang et al., 2019), however, Zhu et al. (2020) demonstrated that it is more effective to provide the output of the final layer of a language representation to the EncDec model as contextual embeddings. Recently, Weng, Yu, Huang, Cheng, and Luo (2019) addressed the mismatch problem between contextual knowledge from pre-trained models and the target bilingual machine translation. Here, I also claim that addressing the gap between grammatically correct raw corpora and GEC corpora can lead to the improvement of GEC systems.

3.2 Methods for Using Pre-trained language representation in GEC Model

In this section, I describe these approaches for incorporating a pre-trained language representation into a GEC model. Specifically, I chose the following approaches: (1) initializing a GEC model using BERT; (2) using BERT output as additional features for a GEC model, and (3) using the output of BERT fine-tuned with the GEC corpora as additional features for a GEC model.

3.2.1 BERT-init

I create a GEC EncDec model initialized with BERT weights. This approach is based on Lample and Conneau (2019). Most recent state-of-the-art methods use pseudo-data, which is generated by injecting pseudo-errors to grammatically correct sentences. However, note that this method cannot initialize a GEC model with pre-trained parameters learned from pseudo-data.

3.2.2 BERT-fuse

I use the model proposed by Zhu et al. (2020) as a feature-based approach (BERT-fuse). This model is based on Transformer EncDec architecture. It takes an input sentence $\mathbf{X} = (x_1, ..., x_n)$, where n is its length. x_i is i-th token in $\mathbf{X}$. First, BERT encodes it and outputs a representation $\mathbf{B} = (\boldsymbol{b}_1, ..., \boldsymbol{b}_n)$. Next, the GEC model encodes $\mathbf{X}$ and $\mathbf{B}$ as inputs. $\boldsymbol{h}_i^l \in \mathbf{H}$ is the i-th hidden representation of the l-th layer of the encoder in the GEC model. $\boldsymbol{h}^0$ stands for word embedding of an input sentence $\mathbf{X}$. Then I calculate $\tilde{\boldsymbol{h}}_i^l$ as follows:

$$\tilde{\boldsymbol{h}}_i^l = \frac{1}{2}(A_h(\boldsymbol{h}_i^{l-1}, \mathbf{H}^{l-1}) + A_b(\boldsymbol{h}_i^{l-1}, \mathbf{B}^{l-1})) \tag{3.1}$$

where A_h and A_b are attention models for the hidden layers of the GEC encoder $\mathbf{H}$ and the BERT output $\mathbf{B}$, respectively. Then each $\tilde{\boldsymbol{h}}_i^l$ is further processed by the feedforward network F which outputs the l-th layer $\mathbf{H}^l = (F(\tilde{\boldsymbol{h}}_1^l), ..., F(\tilde{\boldsymbol{h}}_n^l))$. The decoder's hidden state $\boldsymbol{s}_t^l \in \mathbf{S}$ is calculated as follows:

$$\hat{\boldsymbol{s}}_t^l = A_s(\boldsymbol{s}_t^{l-1}, \mathbf{S}_{<t+1}^{l-1}) \tag{3.2}$$

$$\tilde{\boldsymbol{s}}_i^l = \frac{1}{2}(A_h(\hat{\boldsymbol{s}}_i^{l-1}, \mathbf{H}^{l-1}) + A_b(\hat{\boldsymbol{s}}_i^{l-1}, \mathbf{B}^{l-1})) \tag{3.3}$$

$$\boldsymbol{s}_t^l = F(\tilde{\boldsymbol{s}}_t^l) \tag{3.4}$$

Here, A_s represents the self-attention model. Finally, $\boldsymbol{s}_t^L$ is processed via a linear transformation and softmax function to predict the t-th word $\hat{\boldsymbol{y}}_t$. I also use the drop-net trick proposed by Zhu et al. (2020) to the output of BERT and the encoder of the GEC model.

Zhu et al. (2020) proposes a drop-net trick to utilize both the output of BERT and the encoder of the GEC model. The drop-net will affect Eq 3.1 and Eq 3.3 as described as follows; the BERT dropout ratio is set to $p_{\text{drop}} \in [0, 0.5]$. At each training iteration, for any layer l, I randomly sample $p_h^l \in [0, 1]$ and compare it to p_{drop}. In encoder, if p_h^l is smaller than p_{drop}, I calculate $A_h(\boldsymbol{h}_i^{l-1}, \mathbf{H}^{l-1})$, if it is larger than $1 - p_{\text{drop}}$, then I calculate $A_b(\boldsymbol{h}_i^{l-1}, \mathbf{B}^{l-1}))$. If $p_{\text{drop}} \leq p_h^l \leq 1 - p_{\text{drop}}$, then Eq 3.1 is calculated as $\boldsymbol{h}_i^l$. In the decoder, if p_s^l is smaller than p_{drop}, I calculate $A_h(\hat{\boldsymbol{s}}_i^{l-1}, \mathbf{H}^{l-1})$, if it is larger than $1 - p_{\text{drop}}$, then I calculate $A_b(\hat{\boldsymbol{s}}_i^{l-1}, \mathbf{B}^{l-1}))$. If $p_{\text{drop}} \leq p_s^l \leq 1 - p_{\text{drop}}$, then Eq 3.3 is calculated as $\tilde{\boldsymbol{s}}_i^l$.

3.2.3 BERT-fuse Mask and GED

The advantage of the BERT-fuse is that it can preserve pre-trained information from raw corpora, however, it may not be adapted to either the GEC task or the task-specific distribution of inputs. The reason is that in the GEC model, unlike the data used for training BERT, the input can be an erroneous sentence. To fill the gap between corpora used to train GEC and BERT, I additionally train BERT on GEC corpora (BERT-fuse mask) or fine-tune BERT as a GED model (BERT-fuse GED) and use it for BERT-fuse. GED is a sequence labeling task that detects grammatically incorrect words in input sentences (Rei & Yannakoudakis, 2016; Kaneko, Sakaizawa, & Komachi, 2017). Since BERT is also effective in GED (Bell, Yannakoudakis, & Rei, 2019; Kaneko & Komachi, 2019), it is considered to be suitable for fine-tuning to take into account grammatical errors.

GEC model	
Model Architecture	Transformer (big)
Number of epochs	30
Max tokens	4096
Optimizer	Adam
	$(\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8})$
Learning rate	3×10^{-5}
Min learning rate	1×10^{-6}
Loss function	label smoothed cross-entropy
	$(\epsilon_{ls} = 0.1)$
	(Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016)
Dropout	0.3
Gradient Clipping	0.1
Beam search	5
GED model	
Model Architecture	BERT-Base (cased)
Number of epochs	3
Batch size	32
Max sentence length	128
Optimizer	Adam
	$(\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-8})$
Learning rate	$4e - 5$
Dropout	0.1

Table 3.1: Hyperparameters values of GEC model and Fine-tuned BERT.

3.3 Experimental Setup

3.3.1 Train and Development Sets

I use the BEA-2019 workshop[1] (Bryant, Felice, Andersen, & Briscoe, 2019) official shared task data as training and development sets. Specifically, to train a GEC model, I use W&I-train (Granger, 1998; Yannakoudakis, Andersen, Ardeshir, Ted, & Diane, 2018), NUCLE (Dahlmeier, Ng, & Wu, 2013), FCE-train (Yannakoudakis

[1] https://www.cl.cam.ac.uk/research/nl/bea2019st/

et al., 2011) and Lang-8 (Mizumoto, Komachi, Nagata, & Matsumoto, 2011) datasets. I use W&I-dev as a development set. Note that I excluded sentence pairs that were not corrected from the training data. To train BERT for BERT-fuse mask and GED, I use W&I-train, NUCLE, and FCE-train as training, and W&I-dev was used as development data.

3.3.2 Evaluating GEC Performance

In GEC, it is important to evaluate the model with multiple datasets (Mita, Mizumoto, Kaneko, Nagata, & Inui, 2019). Therefore, I used GEC evaluation data such as W&I-test, CoNLL-2014 (Ng et al., 2014), FCE-test and JFLEG (Napoles, Sakaguchi, & Tetreault, 2017). I used ERRANT evaluation metrics (Felice, Bryant, & Briscoe, 2016; Bryant, Felice, & Briscoe, 2017) for W&I-test, M^2 score (Dahlmeier & Ng, 2012) for CoNLL-2014 and FCE-test sets, and GLEU (Napoles, Sakaguchi, Post, & Tetreault, 2015) for JFLEG. All the results (except ensemble) are the average of four distinct trials using four different random seeds.

3.3.3 Models

Hyperparameter values for the GEC model is listed in Table 3.1. For the BERT initialized GEC model, I provided experiments based on the open-source code[2]. For the BERT-fuse GEC model, I use the code provided by Zhu et al. (2020)[3]. While the training the GEC model, the model was evaluated on the development set and saved every epoch. If loss did not drop at the end of an epoch, the learning rate was multiplied by 0.7. The training was stopped if the learning rate was less than the minimum learning rate or if the learning epoch reached the maximum epoch number of 30.

[2]https://github.com/facebookresearch/XLM
[3]https://github.com/bert-nmt/bert-nmt

Training BERT for BERT-fuse mask and GED was based on the code from Wolf et al. (2019)[4]. The additional training for the BERT-fuse mask was done in the Devlin et al. (2019)'s setting. Hyperparameter values for the GED model is listed in Table 3.1. I used the BERT-Base cased model, for consistency across experiments[5]. The model was evaluated on the development set.

3.3.4 Pseudo-data

I also performed experiments utilizing BERT-fuse, BERT-fuse mask, and BERT-fuse GED outputs as additional features to the pre-trained on the pseudo-data GEC model. The pre-trained model using pseudo-data was initialized with the Pret-Large+SSE model used in the Kiyono et al. (2019)[6] experiments. This pseudo-data is generated by probabilistically injecting character errors into the output (Lichtarge et al., 2019) of a backtranslation (Z. Xie, Genthial, Xie, Ng, & Jurafsky, 2018) model that generates grammatically incorrect sentences from grammatically correct sentences (Kiyono et al., 2019).

3.3.5 Right-to-left (R2L) Re-ranking for Ensemble

I describe the R2L re-ranking technique incorporated in these experiments proposed by Sennrich, Haddow, and Birch (2016), which proved to be efficient for the GEC task (Grundkiewicz et al., 2019; Kiyono et al., 2019). Standard left-to-right (L2R) models generate the n-best hypotheses using scores with the normal ensemble and R2L models re-score them. Then, I re-rank the n-best candidates based on the sum of the L2R and R2L scores. I use the generation probability as a re-ranking score and ensemble four L2R models and four R2L models.

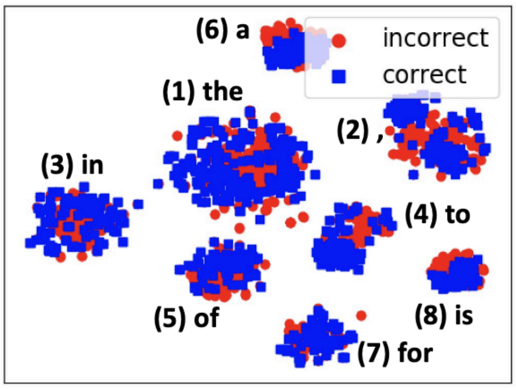
[4]https://github.com/huggingface/transformers
[5]https://github.com/google-research/bert
[6]https://github.com/butsugiri/gec-pseudodata

	BEA-test (ERRANT)			CoNLL-14 (M^2)			FCE-test (M^2)			JFLEG
	P	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**	**GLEU**
w/o BERT	51.5	43.2	49.6	59.2	31.2	50.2	61.7	46.4	57.9	52.7
BERT-init	55.1	43.7	52.4	61.3	31.5	51.4	62.4	46.9	58.5	53.0
BERT-fuse	57.5	**44.9**	54.4	62.3	31.3	52.0	64.0	47.6	59.8	54.1
BERT-fuse mask	57.1	44.7	54.1	62.9	32.2	52.8	64.3	48.1	60.2	54.2
BERT-fuse GED	**58.1**	44.8	**54.8**	**63.6**	**33.0**	**53.6**	**65.0**	**49.6**	**61.2**	**54.4**
w/o BERT	66.1	59.9	64.8	68.5	44.8	61.9	56.5	48.1	54.9	61.0
BERT-fuse	66.6	60.0	65.2	68.3	**45.7**	62.1	59.7	**48.5**	**57.0**	61.2
BERT-fuse mask	67.0	60.0	65.4	68.8	45.3	62.3	59.7	47.1	56.6	61.2
BERT-fuse GED	**67.1**	**60.1**	**65.6**	**69.2**	45.6	**62.6**	**59.8**	46.9	56.7	61.3
Lichtarge et al. (2019)	-	-	-	65.5	37.1	56.8	-	-	-	**61.6**
Awasthi et al. (2019)	-	-	-	66.1	43.0	59.7	-	-	-	60.3
Kiyono et al. (2019)	65.5	59.4	64.2	67.9	44.1	61.3	-	-	-	59.7
BERT-fuse GED + R2L	72.3	**61.4**	69.8	**72.6**	**46.4**	**65.2**	62.8	48.8	59.4	62.0
Lichtarge et al. (2019)	-	-	-	66.7	43.9	60.4	-	-	-	**63.3**
Grundkiewicz et al. (2019)	72.3	60.1	69.5	-	-	64.2	-	-	-	61.2
Kiyono et al. (2019)*	**74.7**	56.7	**70.2**	72.4	46.1	65.0	-	-	-	61.4

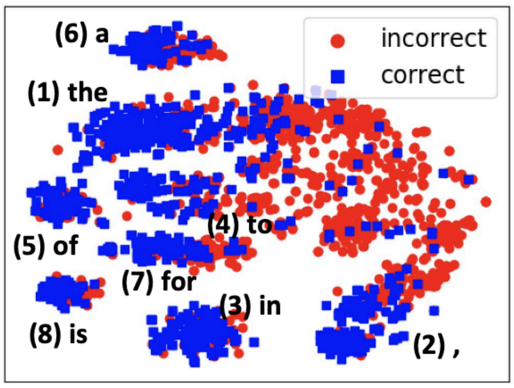
Table 3.2: Results of the GEC models. The top group shows the results of the single models without using pseudo-data and/or ensemble. The second group shows the results of the single models using pseudo-data. The third group shows ensemble models using pseudo-data. **Bold** indicates the highest score in each column. * reports the state-of-the-art scores for BEA test and CoNLL 2014 for two separate models: models with and without SED. I filled out a single line with the results from such two separate models.

3.4 Results

Table 3.2 shows the experimental results of the GEC models. A model trained on Transformer without using BERT is denoted as "w/o BERT." In the top group of results, it can be seen that using BERT consistently improves the accuracy of the GEC model. Also, BERT-fuse, BERT-fuse mask, and BERT-fuse GED outperformed the BERT-init model in almost all cases. Furthermore, I can see that using BERT considering GEC corpora as BERT-fuse leads to better correction results. And the BERT-fuse GED always gives better results than the BERT-fuse mask. This may be because the BERT-fuse GED is able to explicitly consider grammatical errors. In the second group, the correction results are improved by using BERT



(a) BERT



(b) Fine-tuned BERT

Figure 3.1: Hidden representation visualization for encoded grammatically correct and incorrect words.

as well. Also in this setting, BERT-fuse GED outperformed other models in all cases except for the FCE-test set, thus, achieving state-of-the-art results with a single model on the BEA2019 and CoNLL14 datasets. In the last group, the ensemble model yielded high scores on all corpora, improving state-of-the-art results by 0.2 points in CoNLL14.

3.5 Analysis

3.5.1 Visualizing Hidden Layers of Language Representation

I investigate the characteristics of the hidden representations of vanilla (i.e., without any fine-tuning) BERT and BERT fine-tuned with GED. I visualize the hidden representations of the same words from the last layer of BERT $\mathbf{H}^L$. They were chosen depending on correctness in a different context, using the above models. These target eight words (1. the 2. , 3. in 4. to 5. of 6. a 7. for 8. is) that have been mistaken more than 50 times, were chosen from W&I-dev. I sampled the same number of correctly used cases for the same word from the corrected side of

Error type	BERT-fuse GED	w/o BERT
PUNCT	**40.2**	36.8
OTHER	**20.4**	19.1
DET	**48.8**	45.4
PREP	**36.7**	34.8
VERB:TENSE	**36.0**	34.1

Table 3.3: The result of single Fine-tuned BERT-fuse and w/o BERT models without using pseudo-data on most error types including all the top-5 frequent types of error in W&I-dev

W&I-dev.

Figure 3.1 visualizes hidden representations of BERT and fine-tuned BERT. It can be seen that the vanilla BERT does not distinguish between correct and incorrect clusters. The plotted eight words are gathered together, and it can be seen that hidden representations of the same word gather in the same place regardless of correctness. On the other hand, fine-tuned BERT produces a vector space that demonstrates correct and incorrect words on different sides, showing that hidden representations take grammatical errors into account when fine-tuned on GEC corpora. Moreover, it can be seen that the correct cases are divided into 8 clusters, implying that BERT's information is also retained.

3.5.2 Performance for Each Error Type

I investigate the correction results for each error type. I use ERRANT (Felice et al., 2016; Bryant et al., 2017) to measure $F_{0.5}$ of the model for each error type. ERRANT can automatically assign error types from source and target sentences. I use single BERT-fuse GED and w/o BERT models without using pseudo-data for this investigation.

Table 3.3 shows the results of single BERT-fuse GED and w/o BERT models without using pseudo-data on most error types including all the top-5 frequent error types in W&I-dev. I see that BERT-fuse GED is better for all error types compared to w/o BERT. I can say that the use of BERT fine-tuned by GED for the EncDec model improves the performance independently of the error type.

4 | Multi-Head Multi-Layer Attention to Language Representations

In this chapter, I propose a method to effectively reduce the effect of grammatical bias in downstream tasks, instead of debiasing pre-trained language representations in advance. It has been demonstrated that utilizing language representation models pre-trained with large-scale data is effective for various downstream tasks. For example, recent studies have shown a significant improvement using large-scale data to train large deeper models for natural language understanding tasks (M. E. Peters et al., 2018; Alec, Karthik, Tim, & Ilya, 2018; Devlin et al., 2019).

Moreover, neural networks learn different representations for each layer. For example, Belinkov, Durrani, Dalvi, Sajjad, and Glass (2017) demonstrated that in a machine translation task, the lower layers of the network learn to represent the word structure, while higher layers are more focused on word meaning. M. E. Peters et al. (2018) showed that in learning deep contextualized word representations, constructing representations of layers corresponding to each task by a weighted sum improved the accuracy of six NLP tasks. M. Peters, Neumann, Zettlemoyer, and Yih (2018) empirically showed that lower layers are best-suited for local syntactic relationships, that higher layers better model longer-range relationships, and that the top-most layers specialize at the language modeling.

For tasks that emphasize the grammatical nature, such as grammatical error detection, the usual way of using language representations, using only the final layer of information, is considered to be more susceptible to grammatical bias, as it does not allow for effective learning of grammatical information. Therefore, I propose a model that uses multi-head multi-layer attention in order to construct hidden representations from different layers suitable to reduce grammatical bias for grammatical error detection.

4.1 Related Works

4.1.1 Grammatical Error Detection with Language Representations

Often, in sequence labeling tasks, recent supervised neural grammatical error detection models are built upon Bi-LSTM (Rei & Yannakoudakis, 2016; Kaneko et al., 2017; Rei et al., 2017; Rei, 2017; Kasewa et al., 2018; Rei & Søgaard, 2019). Rei and Søgaard (2019) used token-level predictions by Bi-LSTM for self-attention to predict sentence-level labels for grammatical error detection. However, I adopt a transformer model for token-level grammatical error detection due to the expressiveness and better performance of transformer model over LSTM model.

Several studies have exploited large quantities of raw data to create additional artificial data. Rei et al. (2017) artificially generated writing errors in order to create additional resources to learn a neural sequence labeling model following Rei (2017). Kasewa et al. (2018) employed a neural machine translation system to create error-filled artificial data for grammatical error detection. By contrast, I directly adopt a pre-trained language representation model trained with large-scale raw data. This way, there is no need to train an additional generation model and generate pseudo-data. It also eliminates the need for long training runs on large pseudo-data.

4.1.2 Using the Layer Representations

Contextualized Word Representations (ELMo) (M. E. Peters et al., 2018) used large-scale data for a language representation model. Their model learns task-specific weighting from all fixed hidden layers of the pre-trained Bi-LSTM to construct contextualized word embeddings optimized to a given task. In other words, ELMo learns task-specific representations exclusively in the first layer, whereas other parameters of a pre-trained model remain unchanged. On the contrary, I construct representations suited for given tasks by fine-tuning all parameters of the proposed pre-trained model, using multi-head multi-layer attention. All parameters and constructed representations of proposed model are trained to be best-suited for the given task.

Takase, Suzuki, and Nagata (2018) employed intermediate layer representations, including input embeddings, to calculate the probability distributions in order to solve a ranking problem in language generation tasks. Similarly, I considered the information of each layer, but my motivation is to seize the optimal information from each layer suitable for a given task using a multi-head multi-layer attention. Moreover, their model estimated probability distributions from each layer, whereas mine constructs hidden representations from each layer for the output layer.

Furthermore, there is a study that predicts information from the middle layer of the language model and learns the errors occurring owing to the model (Al-Rfou, Choe, Constant, Guo, & Jones, 2019). The use of the information of the middle layer of `transformer_block` is the same as in my research, however the information of each layer is not taken into account at the time of evaluation and is used only for learning. Furthermore, the information on the surface layer is less useful and learning is undertaken so that the influence of the surface layer decreases as learning progresses. In contrast, as the proposed method uses attention, it also learns which layer is utilized in the model itself.

4.2 Language Representations for Grammatical Error Detection

I propose a model that applies multi-head attention to each layer (multi-head multi-layer attention, MHMLA) to fine-tune pre-trained BERT (Devlin et al., 2019). Architectures of BERT and MHMLA for the grammatical error detection task are illustrated in Figure 4.1. In this section, I first introduce BERT and then explain the proposed model, MHMLA.

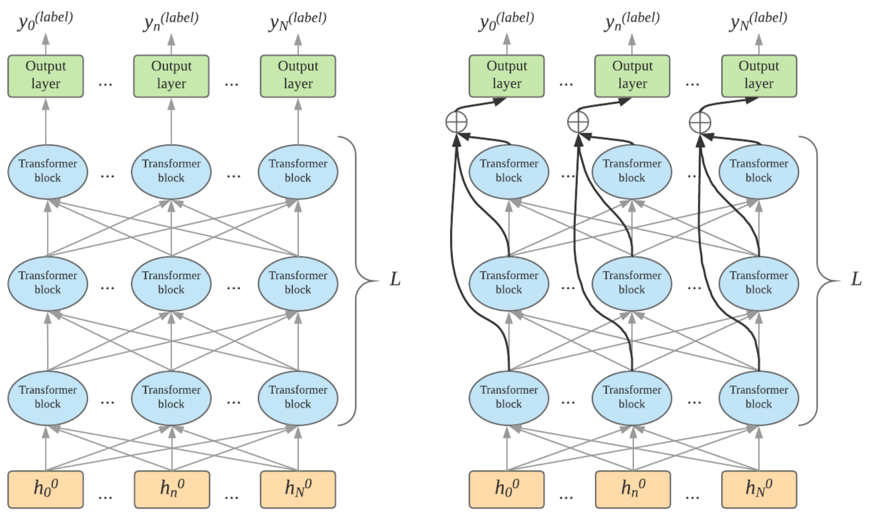
4.2.1 BERT

BERT is designed to learn deep bidirectional representations by jointly conditioning both the left and right contexts in all layers (Figure 4.1(a)). It is based on a multi-layer bidirectional transformer encoder (Vaswani et al., 2017). Insofar it is a language representation model pre-trained on large-scale data, it can be used for fine-tuning. It achieved state-of-the-art results for a wide range of tasks such as natural language understanding, name entity recognition, question answering, and grounded commonsense inference (Devlin et al., 2019).

BERT has a multi-layer bidirectional transformer encoder and can be used for different architectures, such as in classification and sequence-to-sequence learning tasks. Here, I explain the BERT's architecture for sequence labeling tasks. Given a sequence $\mathcal{S} = \boldsymbol{w}_0, \cdots, \boldsymbol{w}_n, \cdots, \boldsymbol{w}_N$ as input, BERT is formulated as follows:

$$\boldsymbol{h}_n^0 = \mathbf{W}_{\mathrm{e}} \boldsymbol{w}_n + \mathbf{W}_{\mathrm{p}} \tag{4.1}$$

$$\boldsymbol{h}_n^l = \text{transformer\_block}(\boldsymbol{h}_n^{l-1}) \tag{4.2}$$



(a) BERT. (b) MHMLA.

Figure 4.1: Architectures of BERT and MHMLA for grammatical error detection.

$$\boldsymbol{y}_n^{(\mathrm{BERT})} = \mathrm{softmax}(\mathbf{W}_{\mathrm{o}}\boldsymbol{h}_n^L + \boldsymbol{b}_{\mathrm{o}}) \tag{4.3}$$

Where $\boldsymbol{w}_n$ is a current token, and N denotes the sequence length. Equation 4.1 thus creates an input embedding. Here, `transformer_block` includes self-attention and fully connected layers (Vaswani et al., 2017), and outputs $\boldsymbol{h}_n^l$. l is the number of the current layer, $l \geq 1$. L is the total number of layers of BERT. Equation 4.3 denotes the output layer. $\mathbf{W}_{\mathrm{o}}$ is an output weight matrix, $\boldsymbol{b}_{\mathrm{o}}$ is a bias for the output layer, and $\boldsymbol{y}_n^{(\mathrm{BERT})}$ is a prediction.

The parameters $\mathbf{W}_{\mathrm{e}}$, $\mathbf{W}_{\mathrm{p}}$ and `transformer_block` are pre-trained on a large document-level corpus using a masked language model (Taylor, 1953) and predicting a next sentence. Then, BERT uses a different task-specific matrix $\mathbf{W}_{\mathrm{o}}$ of the output layer (Equation 4.3) for a given sequence labeling task. To adapt BERT for specific tasks, all parameters of BERT are fine-tuned jointly by predicting a task-specific label with the task-specific output layer to maximize the log-probability of the correct label.

4.2.2 Multi-Head Multi-Layer Attention to Acquire Task-Specific Representations

Multi-head attention (Vaswani et al., 2017) is more beneficial than a single attention function. MHMLA on a sequence labeling model applies attention to each layer l of the output of `transformer_block` $\boldsymbol{h}_n^l$ of Equation 4.2 (Figure 4.1(b)). First, I calculate attention value $\boldsymbol{v}_n^l$:

$$\boldsymbol{v}_{n,j}^l = \mathbf{W}_{\mathrm{v}j}^l \boldsymbol{h}_n^l + \boldsymbol{b}_{\mathrm{v}j}^l \tag{4.4}$$

Here, $\mathbf{W}_{\mathrm{v}}$ is a weight matrix, $\boldsymbol{b}_{\mathrm{v}}$ is a bias, and j is a head number. I apply a non-linear layer to $\boldsymbol{h}_n^l$ to acquire $\boldsymbol{k}_n^l$. Attention score $\boldsymbol{a}_n^l$ is as follows:

$$\boldsymbol{k}_{n,j}^l = \mathrm{relu}(\mathbf{W}_{\mathrm{k}j}^l \boldsymbol{h}_n^l + \boldsymbol{b}_{\mathrm{k}j}^l) \tag{4.5}$$

$$\boldsymbol{a}_{n,j}^l = \mathbf{W}_{\mathrm{a}j}^l \boldsymbol{k}_n^l + \boldsymbol{b}_{\mathrm{a}j}^l \tag{4.6}$$

where $\mathbf{W}_{\mathrm{k}}$ and $\mathbf{W}_{\mathrm{a}}$ are weight matrices, and $\boldsymbol{b}_{\mathrm{k}}$ and $\boldsymbol{b}_{\mathrm{a}}$ are biases. Multi-heads are then calculated as follows:

$$\tilde{\boldsymbol{a}}_{n,j}^l = \frac{\exp(\boldsymbol{a}_{n,j}^l)}{\sum_{t=1}^{L} \exp(\boldsymbol{a}_{n,j}^t)} \tag{4.7}$$

$$\mathrm{head}_{n,j} = \sum_{t=1}^{L} \tilde{\boldsymbol{a}}_{n,j}^t \boldsymbol{v}_{n,j}^t \tag{4.8}$$

where $\tilde{a}^l$ is the attention weight, normalized to sum up to 1 over all values in the layers. These weights are then used to combine the context-conditioned hidden representations from Equation (5) into a single-token representation c_n:

$$\boldsymbol{c}_n = \mathrm{concat}(\mathrm{head}_{n,1}, \cdots, \mathrm{head}_{n,J}) \tag{4.9}$$

corpus	train	dev	test
FCE	28,731	2,222	2,720
CoNLL14	-	-	1,312
JFLEG	-	-	747

Table 4.1: Sentence statistics of used corpora.

where J is the total number of heads. Finally, I return task-specific predictions based on this representation:

$$\boldsymbol{y}_n^{(\text{label})} = \text{softmax}(\mathbf{W}_\text{o}\boldsymbol{c}_n + \boldsymbol{b}_\text{o}) \tag{4.10}$$

$\mathbf{W}_\text{o}$ is an output weight matrix and $\boldsymbol{b}_\text{o}$ is a bias of output layer. The proposed model is optimized by minimizing cross-entropy loss on the token-level annotation.

4.3 Experiments

4.3.1 Datasets

I focus on a supervised sequence labeling task: viz., grammatical error detection. Grammatical error detection is the task of identifying incorrect tokens that need to be edited in order to produce a grammatically correct sentence. I evaluated these approach on the three different grammatical error detection datasets. Table 4.1 shows statistics for each corpus.

FCE. I fine-tuned and searched the parameters of the model and evaluated the system on the First Certificate in English (FCE) dataset (Yannakoudakis et al., 2011), which contains error-annotated short essays written by language learners. The FCE dataset is a popular English learner corpus for grammatical error detection. I followed the official split of the data.

CoNLL14. I additionally used dataset from the CoNLL 2014 shared task (CoNLL14) dataset (Ng et al., 2014) in evaluation. This dataset was written by higher-proficiency learners on different technical topics. It was manually corrected by two separate annotators, and I report results on each of these annotations (CoNLL14-{1,2}).

JFLEG. I also evaluated this approach with the JHU FLuency-Extended GUG (JFLEG) corpus (Napoles et al., 2017). It contains a broad range of language-proficiency levels and focuses more on fluency edits and making the text more native-sounding, in addition to grammatical corrections. JFLEG is not labeled for grammatical error detection. Therefore, I used dynamic programming to label tokens in sentences as correct or incorrect. Because JFLEG is a recently developed corpus, there is only one prior study with experimental results (Rei & Søgaard, 2019). JFLEG is tagged by multiple annotators, like CoNLL14, so I followed Rei and Søgaard (2019) to build a version that combines the references: if a token is labeled as an error by any annotator, it is marked as an error[1].

4.3.2 Experimental Details

I used a publicly available pre-trained language representation model, namely the $\text{BERT}_{\text{BASE}}$ uncased model[2]. This model has 12 layers, 768 hidden size, and 16 heads of self-attention. Layer attention has 12 heads (J = 12). I fine-tuned the model over 5 epochs with a batch size of 32. The maximum training sentence length was 128 tokens. I used the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 5e-05. I applied dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) to $\boldsymbol{h}_n^l$, $\boldsymbol{k}_{n,j}^l$, and $\tilde{\boldsymbol{a}}_{n,j}^l$ with a dropout rate of

[1]Although JFLEG's experimental settings are not described in the paper, I confirmed them with the authors of the paper over e-mail.

[2]`https://github.com/google-research/bert`

0.3. $\tilde{\boldsymbol{a}}_{n,j}^{l}$ is referred to as attention dropout. I also used WordPiece embeddings (Wu et al., 2016). To make this compatible with sub-token tokenization, I inputted each tokenized word into the WordPiece tokenizer and used the hidden state corresponding to the first sub-token as input to the output layer, as with the original BERT.

I used $\mathrm{F}_{0.5}$ as the main evaluation measure. This measure was also adopted in the CoNLL14 shared task for the grammatical error correction task (Ng et al., 2014). It combines both precision and recall, while assigning twice as much weight to precision, because accurate feedback is often more important than coverage in error detection applications (Nagata & Nakatani, 2010).

4.3.3 Baselines and Comparisons

I compare with models of Rei (2017), Rei and Søgaard (2019), Rei et al. (2017), and Kasewa et al. (2018) which are based on the Bi-LSTM architecture. The first group, Rei (2017) and Rei and Søgaard (2019), was trained exclusively on the FCE dataset. The second group, Rei et al. (2017) and Kasewa et al. (2018) used additional artificial data along with the FCE dataset for training.

The baseline and the proposed models were trained on the transformer architecture. The first three are the descriptions of the baselines, and the fourth is a description of the proposed model:

BERT$_{\text{BASE}}$ w/o pre-train. This model is trained using only the FCE dataset and with random initialization. This baseline did not use any other corpus for training.

BERT$_{\text{BASE}}$. This is the original pre-trained model described in Section 4.3.2 fine-tuned on the FCE dataset. This baseline uses original BERT model (Devlin et al., 2019) and can be seen as surrogated version of the proposed method

	FCE			CoNLL14-1			CoNLL14-2			JFLEG		
	P	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**
Rei (2017)	58.88	28.92	48.48	17.68	19.07	17.86	25.22	19.25	23.62	-	-	-
Rei and Søgaard (2019)	65.53	28.61	52.07	25.14	15.22	22.14	37.72	16.19	29.65	72.53	25.04	52.52
Rei et al. (2017)	60.67	28.08	49.11	23.28	18.01	21.87	35.28	19.42	30.13	-	-	-
Kasewa et al. (2018)	-	-	55.6	-	-	28.3	-	-	35.5	-	-	-
$\text{BERT}_{\text{BASE}}$ w/o pre-train	48.85	11.30	29.34	11.45	7.80	10.47	18.24	9.31	15.30	58.85	13.22	34.81
$\text{BERT}_{\text{BASE}}$	**69.80**	37.37	59.47	34.08	**33.56**	33.97	46.01	33.89	42.93	**78.06**	36.28	63.45
AvgL	68.09	41.14	60.20	34.97	32.02	34.33	45.33	35.27	42.88	77.35	37.05	63.52
MHMLA	$68.87^{\dagger}$	**43.45***†	**61.65***†	**35.74***	$33.50^{\dagger}$	**35.26***†	**46.45**†	**35.47***	**43.74**†	77.74	**38.85***†	**64.77***†

Table 4.2: Results of grammatical error detection. These results are averaged over five runs. $*$ and $\dagger$ indicate that there is a significant difference against $\text{BERT}_{\text{BASE}}$ and AvgL, respectively.

J	FCE	CoNLL14-1	CoNLL14-2	JFLEG
1	61.16	33.75	42.89	63.98
2	61.62	33.44	42.42	63.72
3	**61.90**	34.50	43.17	64.45
4	61.55	33.74	42.80	64.37
6	61.22	34.26	43.29	64.48
8	61.27	34.72	43.02	64.10
12	61.65	**35.26**	**43.74**	**64.77**

Table 4.3: $F_{0.5}$ scores of MHMLA using different number of heads J. These results are averaged over five runs.

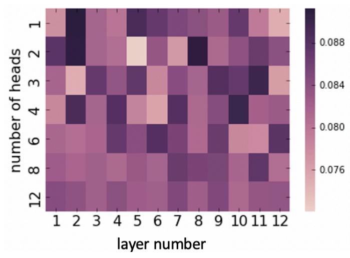
without multi-layer attention.

AvgL. This model is called averaged layers, which averages representations after linear transformation of $\boldsymbol{h}_n^l$ (Equation 4.2) for the output layer of $\text{BERT}_{\text{BASE}}$ model instead of using an attention.

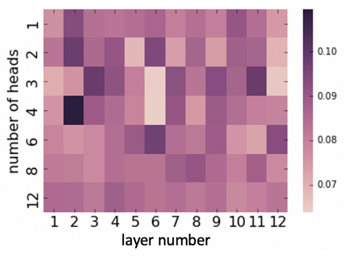
MHMLA. This is the proposed model that is an extension of $\text{BERT}_{\text{BASE}}$, with MHMLA to the pre-trained model while fine-tuning on the FCE dataset.

4.4 Results

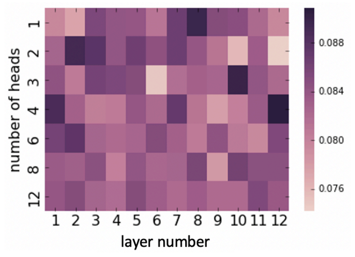
Table 4.2 shows the grammatical error detection results for the FCE, CoNLL14-{1,2}, and JFLEG datasets. Scores for Rei (2017), Rei and Søgaard (2019), Rei et al. (2017), and Kasewa et al. (2018) were taken from their respective papers.



(a) FCE.



(b) CoNLL14.



(c) JFLEG.

Figure 4.2: Attention visualization of MHMLA on each dataset using a different number of heads. MHMLA with 8 and 12 heads tends to attend to all layers more or less equally for all datasets.

In FCE, CoNLL14, and JFLEG, the $\text{BERT}_{\text{BASE}}$ model significantly outperformed existing methods and the baseline (without pre-training) in terms of precision, recall, and $F_{0.5}$. This demonstrates that using a pre-trained language representation model is highly effective for grammatical error detection. Furthermore, MHMLA achieved the highest $F_{0.5}$ on all datasets, outperforming $\text{BERT}_{\text{BASE}}$ by 2.18 points, 1.29 points, 0.81 points, and 1.32 points on FCE, CoNLL14-{1,2}, and JFLEG, respectively. The scores for the AvgL model were lower than that for the proposed MHMLA model, meaning that naively using information from layers is not as effective as using MHMLA. These results show that using MHMLA and learning task-specific representations improves the accuracy. These results show that the proposed method is able to make more effective use of language representations in which grammatical biases are learned.

To verify the effect of MHMLA, I examined the $F_{0.5}$ value for each head number. I investigated 1, 2, 3, 4, 6, 8, and 12 heads (i.e. the number of heads up to 12 by which the hidden layer size of 768 can be divided). Table 4.3 shows the $F_{0.5}$ values for each number of heads on FCE, CoNLL14-{1,2}, and JFLEG datasets. Regarding FCE, the highest $F_{0.5}$ score was achieved with 3 heads. For CoNLL14-{1,2} and JFLEG, the $F_{0.5}$ values were highest with 12 heads, demonstrating that adopting multi-head leads to improved accuracy.

4.5 Analysis of the Effect of MHMLA

The purpose of MHMLA is to construct representations not only from the final layer but also from various layers to reduce the effect of grammatical bias. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. Therefore, it is considered that increasing the number of heads leads to utilization of information from various layers. Hence, I investigate the effect of the number of heads on each layer by visualizing the averaged score of MHMLA that was calculated by considering the heads j of Equation 4.7 for all layers on test sets of the three datasets: FCE, CoNLL14, and JFLEG.

Figure 4.2 visualizes the average attention score to each layer of MHMLA for each head. The average attention score is calculated by averaging head_n in Equation (4.8). For all datasets, when there were a fewer numbers of heads, the multi-head attentions learned to attend to different layers but tended to focus on particular layers. For example, as shown in Figure 4.2(b), multi-head attention with heads of 2, 3, and 4 heads focused more on layers 2 and 3 while hardly attending to layers 5 and 6. Figure 4.2(b) shows that the same amount of attention is attended to each layer when the number of heads are 8 and 12. In Figure 4.2(c), attention is sharp, especially with the number of heads being 1, 2, 3, and 4. In contrast, with there are more heads, viz. 8 and 12, attention tended to attend to all layers more or less equally for all datasets. From this visualization, I conclude that the goal of utilizing the information from various layers has been achieved. Therefore, the proposed method effectively extracts information from language representations in which grammatical biases have been learned by exploiting information from various layers.

5 | Gender-Preserving Debiasing for Word Embeddings

In this chapter, we propose a method to remove semantic bias, especially gender bias, from word embeddings. Despite the impressive success stories behind word representation learning (Pennington et al., 2014; Devlin et al., 2019; M. E. Peters et al., 2018; Mikolov, Yih, & Zweig, 2013; Mikolov, Chen, & Dean, 2013), further investigations into the learnt representations have revealed several worrying issues. The semantic representations learnt, in particular from social media, have shown to encode significant levels of racist, offensive and discriminative language usage (Bolukbasi, Chang, Zou, Saligrama, & Kalai, 2016; J. Zhao, Zhou, Li, Wang, & Chang, 2018; Elazar & Goldberg, 2018; Rudinger, Naradowsky, Leonard, & Van Durme, 2018; J. Zhao, Wang, Yatskar, Ordonez, & Chang, 2018). For example, Bolukbasi et al. (2016) showed that word representations learnt from a large (300GB) news corpus to amplify unfair gender biases. Microsoft's AI chat bot *Tay* learnt abusive language from Twitter within the first 24 hours of its release, which forced Microsoft to shutdown the bot (The Telegraph, 2016). Caliskan, Bryson, and Narayanan (2017) conducted an implicit association test (IAT) (Greenwald, McGhee, & Schwatz, 1998) using the cosine similarity measured from word representations, and showed that word representations computed from a large Web crawl contain human-like biases with respect to gender, profession and ethnicity.

Given the broad applications of pre-trained word embeddings in various downstream NLP tasks such as machine translation (Zou, Socher, Cer, & Manning, 2013), sentiment analysis (Shi, Fu, Bing, & Lam, 2018), dialogue generation (S. Zhang et al., 2018) etc., it is important to debias word embeddings *before* they are applied in NLP systems that interact with and/or make decisions that affect humans. I believe that no human should be discriminated on the basis of demographic attributes by an NLP system, and there exist clear legal (European Union, 1997), business and ethical obligations to make NLP systems unbiased (Holstein, Vaughan, III, Dudík, & Wallach, 2018).

Despite the growing need for unbiased word embeddings, debiasing pre-trained word embeddings is a challenging task that requires a fine balance between removing information related to discriminative biases, while retaining information that is necessary for the target NLP task. For example, profession-related nouns such as *professor*, *doctor*, *programmer* have shown to be stereotypically male-biased, whereas *nurse*, *homemaker* to be stereotypically female-biased, and a debiasing method must remove such biases. On the other hand, one would expect[1], *beard* to be associated with male nouns and *bikini* to be associated with female nouns, and preserving such gender biases would be useful, for example, for a recommendation system (Garimella, Banea, & Mihalcea, 2017). As detailed later in section 5.1, existing debiasing methods can be seen as embedding word embeddings into a subspace that is approximately orthogonal to a gender subspace spanned by gender-specific word embeddings. Although unsupervised, weakly-supervised and adversarially trained models have been used for learning such embeddings, they primarily focus on the male-female gender direction and ignore the effect of words that have a gender orientation but not necessarily unfairly biased.

To perform an extensive treatment of the gender debiasing problem, I split a given vocabulary $\mathcal{V}$ into four mutually exclusive sets of word categories: (a) words $w_f \in \mathcal{V}_f$ that are female-biased but non-discriminative, (b) words $w_m \in \mathcal{V}_m$ that are male-biased but non-discriminative, (c) words $w_n \in \mathcal{V}_n$ that are gender-neutral,

[1]This indeed is the case for pre-trained GloVe embeddings

and (d) words $w_s \in \mathcal{V}_s$ that are stereotypical (i.e., unfairly[2] gender-biased). Given a large set of pre-trained word embeddings and small seed example sets for each of those four categories, I learn an embedding that (i) preserves the feminine information for the words in $\mathcal{V}_f$, (ii) preserves the masculine information for the words in $\mathcal{V}_m$, (iii) protects the neutrality of the gender-neutral words in $\mathcal{V}_n$, while (iv) removing the gender-related biases from stereotypical words in $\mathcal{V}_s$. The embedding is learnt using an encoder in a denoising autoencoder, while the decoder is trained to reconstruct the original word embeddings from the debiased embeddings that do not contain unfair gender biases. The overall model is trained end-to-end to dynamically balance the competing criteria (i)-(iv).

I evaluate the bias and accuracy of the word embeddings debiased by the proposed method on multiple benchmark datasets. On the SemBias (J. Zhao, Zhou, et al., 2018) gender relational analogy dataset, the proposed method outperforms previously proposed *hard-debiasing* (Bolukbasi et al., 2016) and *gender-neural Global Vectors* (GN-GloVe) (J. Zhao, Zhou, et al., 2018) by correctly debiasing stereotypical analogies. Following prior work, I evaluate the loss of information due to debiasing on benchmark datasets for semantic similarity and word analogy. Experimental results show that the proposed method can preserve the semantics of the original word embeddings, while removing gender biases. This shows that the debiased word embeddings can be used as drop-in replacements for word embeddings used in NLP applications. Moreover, experimental results show that the proposed method can also debias word embeddings that are already debiased using previously proposed debiasing methods such as GN-GloVe to filter out any remaining gender biases, while preserving semantic information useful for downstream NLP applications. This enables to use the proposed method in conjunction with existing debiasing methods.

[2] I use the term *unfair* as used in *fairness-aware machine learning*.

5.1 Related Work

Bias in Static Word Embeddings: Bolukbasi et al. (2016) proposed a post-processing approach that projects gender-neutral words into a subspace, which is orthogonal to the gender direction defined by a list of gender-definitional words. They refer to words associated with gender (e.g., *she*, *actor*) as gender-definitional, and the remainder as gender-neutral. They proposed a *hard-debiasing* method where the gender direction is computed as the vector difference between the embeddings of the corresponding gender-definitional words, and a *soft-debiasing* method, which balances the objective of preserving the inner-products between the original word embeddings, while projecting the word embeddings into a subspace orthogonal to the gender definitional words. They use a list of gender-definitional words to train a support vector machine classifier, and use it to expand the initial set of gender-definitional words. However, their method ignores gender-definitional words during the subsequent debiasing process, and focus only on words that are *not* predicted as gender-definitional by a classifier. Therefore, if the classifier erroneously predicts a stereotypical word as gender-definitional, it would not get debiased. J. Zhao, Zhou, et al. (2018) modified the original GloVe (Pennington et al., 2014) objective to learn gender-neutral word embeddings (GN-GloVe) from a given corpus. They maximise the squared ℓ_2 distance between gender-related sub-vectors, while simultaneously minimising the GloVe objective. GN-GloVe learns gender-debiased word embeddings from scratch from a given corpus, and cannot be used to debias pre-trained word embeddings. Moreover, similar to hard and soft debiasing methods described above, GN-GloVe uses pre-defined lists of feminine, masculine and gender-neutral words and *does not* debias words in these lists.

Adversarial learning (Q. Xie, Dai, Du, Hovy, & Neubig, 2017; Elazar & Goldberg, 2018; Li, Baldwin, & Cohn, 2018) for debiasing first encode the inputs and then two classifiers are jointly trained – one predicting the target task (for which we must ensure high prediction accuracy) and the other for protected attributes (that must not be easily predictable). Elazar and Goldberg (2018) showed that although it is possible to obtain chance-level development-set accuracy for the protected attributes during training, a post-hoc classifier trained on the encoded inputs can still manage to reach substantially high accuracies for the protected attributes. They conclude that adversarial learning alone does not guarantee invariant representations for the protected attributes. Ravfogel, Elazar, Gonen, Twiton, and Goldberg (2020) found that iteratively projecting word embeddings to the null space of the gender direction to further improve the debiasing performance.

Benchmarks for biases in Static Embeddings: Word Embedding Association Test (WEAT; Caliskan et al., 2017) quantifies various biases (e.g. gender, race and age) using semantic similarities between word embeddings. Word Association Test (WAT) measures gender bias over a large set of words (Du, Wu, & Lan, 2019) by calculating the gender information vector for each word in a word association graph created in the Small World of Words project (SWOWEN; Deyne, Navarro, Perfors, Brysbaert, & Storms, 2019) by propagating masculine and feminine words via a random walk (D. Zhou, Bousquet, Lal, Weston, & Schölkopf, 2003). SemBias dataset (J. Zhao, Zhou, et al., 2018) contains three types of word-pairs: (a) **Definition**, a gender-definition word pair (e.g. hero – heroine), (b) **Stereotype**, a gender-stereotype word pair (e.g., manager – secretary) and (c) **None**, two other word-pairs with similar meanings unrelated to gender (e.g., jazz – blues, pencil – pen). It uses the cosine similarity between the gender directional vector, $(\overrightarrow{he} - \overrightarrow{she})$, and the offset vector $(\boldsymbol{a} - \boldsymbol{b})$ for each word pair, (a, b), in each set to measure gender bias. WinoBias (J. Zhao, Wang, et al., 2018) uses the ability to predict gender pronouns with equal probabilities for gender neutral nouns such as occupations as a test for the gender bias in embeddings. WinoBias dataset contains two types of sentences that require linking gendered pronouns to either male or female stereotypical occupations. In **Type 1**, co-reference decisions must be made using world knowledge about some given circumstances. However, in **Type 2**, these tests can be resolved using syntactic information and understanding of the pronoun. It involves two conditions: the pro-stereotyped (**pro**) condition links pronouns to occupations dominated by the gender of the pronoun, and the anti-stereotyped (**anti**) condition links pronouns to occupations not dominated by the gender of the pronoun. For a correctly debiased set of word embeddings, the difference between **pro** and **anti** is expected to be small.

Bias in Contextualised Word Embeddings: May, Wang, Bordia, Bowman, and Rudinger (2019) extended WEAT using templates to create a sentence-level benchmark for evaluating bias called SEAT. In addition to the attributes proposed in WEAT, they proposed two additional bias types: *angry black woman* and *double binds* (when a woman is doing a role that is typically done by a man that woman is seen as arrogant). They show that compared to static embeddings, contextualised embeddings such as BERT, GPT and ELMo are less biased. However, similar to WEAT, SEAT also only has positive predictive ability and cannot detect the absence of a bias. Bommasani, Davis, and Cardie (2020) evaluated the bias in contextualised embeddings by first distilling static embeddings from contextualised embeddings and then using WEAT tests for different types of biases such as gender (male, female), racial (White, Hispanic, Asian) and religion (Christianity, Islam). They found that aggregating the contextualised embedding of a particular word in different contexts via averaging to be the best method for creating a static embedding from a contextualised embedding.

J. Zhao et al. (2019) showed that contextualised ELMo embeddings also learn gender biases present in the training corpus. Moreover, these biases propagate to a downstream coreference resolution task. They showed that data augmentation by swapping gender helps more than neutralisation by a projection. However, data augmentation requires re-training of the embeddings, which is often costly compared to fine-tuning. Kurita, Vyas, Pareek, Black, and Tsvetkov (2019) created masked templates such as "__ is a nurse" and used BERT to predict the masked gender pronouns. They used the log-odds between male and female pronoun predictions as an evaluation measure and showed that BERT to be biased according to it. Karve, Ungar, and Sedoc (2019) learnt conceptor matrices using class definitions in the WEAT and used the negated conceptors to debias ELMo and BERT. Although their method was effective for ELMo, the results on BERT were mixed.

Dev, Li, Phillips, and Srikumar (2020) used natural language inference (NLI) as a bias evaluation task, where the goal is to ascertain if one sentence (i.e. premise) entails or contradictions another (i.e. hypothesis), or if neither conclusions hold (i.e. neutral). The premise-hypothesis pairs are constructed to elicit various types of discriminative biases. They showed that orthogonal projection to gender direction (Dev & Phillips, 2019) can be used to debias contextualised embeddings as well. However, their method can be applied only to the noncontextualised layers (ELMo's Layer 1 and BERT's subtoken layer).

5.2 Gender-Preserving Debiasing

5.2.1 Formulation

Given a pre-trained set of d-dimensional word embeddings $\{\boldsymbol{w}_i\}_{i=1}^{|\mathcal{V}|}$, over a vocabulary $\mathcal{V}$, I consider the problem of learning a map $E : \mathbb{R}^d \rightarrow \mathbb{R}^l$ that projects the original pre-trained word embeddings to a debiased l-dimensional space. I do not assume any knowledge about the word embedding learning algorithm that was used to produce the pre-trained word embeddings is given. Moreover, I do not assume the availability or access to the language resources such as corpora or lexicons that might have been used by the word embedding learning algorithm. Decoupling the debiasing method from the word embedding learning algorithm and resources increases the applicability of the proposed method, enabling to debias pre-trained word embeddings produced using different word embedding learning algorithms and using different types of resources.

I propose a debiasing method that models the interaction between the values of the protected attribute (in the case of *gender* I consider *male*, *female* and *neutral* as possible attribute values), and whether there is a stereotypical bias or not. Given four sets of words: *masculine* ($\mathcal{V}_m$), *feminine* ($\mathcal{V}_f$), *neutral* ($\mathcal{V}_n$) and *stereotypical* ($\mathcal{V}_s$), our proposed method learns a projection that satisfies the following four criteria:

(i) for $w_f \in \mathcal{V}_f$, I protect its feminine properties,

(ii) for $w_m \in \mathcal{V}_m$, I protect its masculine properties,

(iii) for $w_n \in \mathcal{V}_n$, I protect its gender neutrality, and

(iv) for $w_s \in \mathcal{V}_s$, I remove its gender biases.

By definition the four word categories are mutually exclusive and the total vocabulary is expressed by their disjunction $\mathcal{V} = \mathcal{V}_m \cup \mathcal{V}_f \cup \mathcal{V}_n \cup \mathcal{V}_s$. A key feature of the proposed method that distinguishes it from prior work on debiasing word embeddings is its ability to differentiate between undesirable (stereotypical) biases from the desirable (expected) gender information in words. The procedure I follow to compile the four word-sets is described later in subsection 5.3.1, and the words that belong to each of the four categories are shown in the supplementary material.

To explain the proposed gender debiasing method, let's first consider a *feminine* regressor $C_f : \mathbb{R}^l \rightarrow [0, 1]$, parameterised by $\boldsymbol{\theta_f}$, that predicts the degree of femininess of the word w. Here, highly feminine words are assigned values close to 1. Likewise, let's consider a *masculine* regressor $C_m : \mathbb{R}^l \rightarrow [0, 1]$, parametrised by $\boldsymbol{\theta_m}$, that predicts the degree of masculinity of w. I then learn the debiasing function as the encoder $E : \mathbb{R}^d \rightarrow \mathbb{R}^l$ of an autoencoder (parametrised by $\boldsymbol{\theta}_e$), where the corresponding decoder (parametrised by $\boldsymbol{\theta}_d$) is given by $D : \mathbb{R}^l \rightarrow \mathbb{R}^d$.

For feminine and masculine words, I require the encoded space to retain the gender-related information. The squared losses, L_f and L_m, given respectively by (5.1) and (5.2), express the extent to which this constraint is satisfied.

$$L_f = \sum_{w \in \mathcal{V}_f} ||C_f(E(\boldsymbol{w})) - 1||_2^2 + \sum_{w \in \mathcal{V} \setminus \mathcal{V}_f} ||C_f(E(\boldsymbol{w}))||_2^2 \tag{5.1}$$

$$L_m = \sum_{w \in \mathcal{V}_m} ||C_m(E(\boldsymbol{w})) - 1||_2^2 + \sum_{w \in \mathcal{V} \setminus \mathcal{V}_m} ||C_m(E(\boldsymbol{w}))||_2^2 \tag{5.2}$$

Here, for notational simplicity, I drop the dependence on parameters.

For the stereotypical and gender-neutral words, I require that they are embedded into a subspace that is orthogonal to a gender directional vector, $\boldsymbol{v}_g$, computed using a set, Ω, of feminine and masculine word-pairs $(w_f, w_m)(\in \Omega)$ as given by (5.3).

$$\boldsymbol{v}_g = \frac{1}{|\Omega|} \sum_{(w_f, w_m) \in \Omega} (E(\boldsymbol{w}_m) - E(\boldsymbol{w}_f)) \tag{5.3}$$

Prior work on gender debiasing (Bolukbasi et al., 2016; J. Zhao, Zhou, et al., 2018) showed that the vector difference between the embeddings for male-female word-pairs such as *he* and *she* accurately represents the gender direction in Word2Vec and GloVe embeddings. When training, I keep $\boldsymbol{v}_g$ fixed during an epoch, and re-estimate $\boldsymbol{v}_{\mathrm{g}}$ between every epoch. I consider the squared inner-product between $\boldsymbol{v}_g$ and the debiased stereotypical or gender-neutral words as the loss, L_g, as given by (5.4).

$$L_g = \sum_{w \in \mathcal{V}_n \cup \mathcal{V}_s} (\boldsymbol{v}_g w)^2 \tag{5.4}$$

It is important that I preserve the semantic information encoded in the word embeddings as much as possible when I perform debiasing. If too much information is removed from the word embeddings, not limited to gender-biases, then the debiased word embeddings might not be sufficiently accurate to be used in downstream NLP applications. For this purpose, I minimise the reconstruction loss, L_r, for the autoencoder given by (5.5).

$$L_r = \sum_{w \in \mathcal{V}} ||D(E(\boldsymbol{w})) - \boldsymbol{w}||_2^2 \tag{5.5}$$

Finally, I define the total objective as the linearly-weighted sum of the above-defined losses as given by (5.6).

$$L = \lambda_f L_f + \lambda_m L_m + \lambda_g L_g + \lambda_r L_r \tag{5.6}$$

Here, the coefficients $\lambda_f, \lambda_m, \lambda_g, \lambda_r$ are nonnegative hyper-parameters that add to 1. They determine the relative importance of the different constraints I consider and can be learnt using training data or determined via cross-validation over a dedicated validation dataset. In the experiments, I use the latter approach.

5.2.2 Implementation and Training

C_f and C_m are both implemented as feed forward neural networks with one hidden layer and the sigmoid function is used as the nonlinear activation. Increasing the number of hidden layers beyond one for C_f and C_m did not result in a significant increase in accuracy. Both the encoder E and the decoder D of the autoencoder are implemented as feed forward neural networks with two hidden layers. Hyperbolic tangent is used as the activation function throughout the autoencoder.

The objective (5.6) is minimised w.r.t. the parameters $\boldsymbol{\theta}_f$, $\boldsymbol{\theta}_m$, $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_d$ for a given pre-trained set of word embeddings. During optimisation, I used dropout with probability 0.01 and use stochastic gradient descent with initial learning rate set to 0.1. The hyper-parameters $\lambda_f, \lambda_m, \lambda_g, \lambda_r$ are estimated using a separate validation dataset as described later in subsection 5.3.1.

Note that it is possible to pre-train C_f and C_m separately using $\mathcal{V}_f$ and $\mathcal{V}_m$ prior to training the full objective (5.6). In the preliminary experiments, I found that initialising $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_m$ to the pre-trained versions of C_f and C_m to be helpful for the optimisation process, resulting in early convergence to better solutions compared to starting from random initialisations for $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_m$. For pre-training C_f and C_m I used Adam optimiser (Kingma & Ba, 2015) with initial learning rate set to 0.0002 and a mini-batch size of 512. Autoencoder is also pre-trained using a randomly selected 5000 word embeddings and dropout regularisation is applied with probability 0.05.

I note that $\mathcal{V}_f$ and $\mathcal{V}_m$ are separate word sets, not necessarily having corresponding feminine-masculine pairs as in Ω used in (5.4). It is of course possible to re-use the words in Ω in $\mathcal{V}_f$ and $\mathcal{V}_m$, and I follow this approach in the experiments, which helps to decrease the number of seed words required to train the proposed method. Moreover, the number of training examples across the four categories $\mathcal{V}_f, \mathcal{V}_m, \mathcal{V}_n, \mathcal{V}_s$ were significantly different, which resulted in an imbalanced learning setting. I conduct one-sided undersampling (Kubat & Matwin, 1997) to successfully overcome this data imbalance issue. The code and the debiased embeddings are publicly available[3].

5.3 Experiments

5.3.1 Training and Development Data

I use the feminine and masculine word lists (223 words each) created by J. Zhao, Zhou, et al. (2018) as $\mathcal{V}_f$ and $\mathcal{V}_m$, respectively. To create a gender-neutral word list, $\mathcal{V}_n$, I select gender-neutral words from a list of 3000 most frequent words in English[4]. Two annotators independently selected words and subsequently verified for gender neutrality. The final set of $\mathcal{V}$ contains 1031 gender-neutral words. I use the stereotypical word list compiled by Bolukbasi et al. (2016) as $\mathcal{V}_s$, which contains 166 professions that are stereotypically associated with one type of a gender. The four sets of words used in the experiments are shown in the supplementary material.

I train GloVe (Pennington et al., 2014) on 2017 January dump of English Wikipedia to obtain pre-trained 300-dimensional word embeddings for 322636 unique words. In the experiments, I set both d and l to 300 to create 300-dimensional de-biased word embeddings. I randomly selected 20 words from each of the 4 sets $\mathcal{V}_f$, $\mathcal{V}_m$, $\mathcal{V}_n$ and $\mathcal{V}_s$, and used them as a development set for pre-training C_f and C_m and to estimate the hyperparameters in (5.6). The optimal hyperparameter values estimated on this development dataset are: $\lambda_f = \lambda_m = \lambda_g = 0.0001$, and $\lambda_r = 1.0$.

[3]https://github.com/kanekomasahiro/gp_debias

[4]https://bit.ly/2SvBINY

In the preliminary experiments I observed that increasing λ_f, λ_m and λ_g relative to λ_r results in higher reconstruction losses in the autoencoder. This shows that the ability to accurately reconstruct the original word embeddings is an important requirement during debiasing.

5.3.2 Baselines and Comparisons

I compare the proposed method against several baselines.

GloVe: is the pre-trained GloVe embeddings described in subsection 5.3.1. This baseline denotes a non-debiased version of the word embeddings.

Hard-GloVe: I use the implementation[5] of hard-debiasing (Bolukbasi et al., 2016) method by the original authors and produce a debiased version of the pre-trained GloVe embeddings.[6]

GN-GloVe : I use debiased GN-GloVe embeddings released by the original authors[7], without retraining myself as a baseline.

AE (GloVe): I train an autoencoder by minimising the reconstruction loss defined in (5.5) and encode the pre-trained GloVe embeddings to a vector space with the same dimensionality. This baseline can be seen as surrogated version of the proposed method with $\lambda_f = \lambda_m = \lambda_g = 0$. **AE (GloVe)** does *not* perform debiasing and shows the amount of semantic information that can be preserved by autoencoding the input embeddings.

[5] https://github.com/tolga-b/debiaswe

[6] Bolukbasi et al. (2016) released debiased embeddings for word2vec only and for comparison purposes with GN-GloVe, I use GloVe as the pre-trained word embedding and apply hard-debiasing on GloVe

[7] https://github.com/uclanlp/gn_glove

Embeddings	SemBias			SemBias-subset		
	Definition ↑	Stereotype ↓	None ↓	Definition ↑	Stereotype ↓	None ↓
GloVe	80.2	10.9	8.9	57.5	20	22.5
Hard-Glove	84.1	9.5	6.4	25	47.5	27.5
GN-GloVe	97.7	1.4	0.9	75	15	10
AE (GloVe)	82.7	8.2	9.1	62.5†	17.5†	20
AE (GN-GloVe)	98.0$^{\dagger *}$	1.6$^{\dagger *}$	**0.5**$^{\dagger *}$	77.5	17.5†	**5**$^{\dagger *}$
GP (GloVe)	84.3*	8.0	7.7*	65†	15†	20
GP (GN-GloVe)	**98.4**$^{\dagger *}$	**1.1**$^{\dagger *}$	**0.5**$^{\dagger *}$	**82.5**$^{\dagger *}$	**12.5**$^{\dagger *}$	**5**$^{\dagger *}$

Table 5.1: Prediction accuracies for gender relational analogies. $*$ and $\dagger$ indicate statistically significant differences against respectively **GloVe** and **Hard-GloVe**.

AE (GN-GloVe): Similar to **AE (GloVe)**, this method autoencodes the debiased word embeddings produced by **GN-GloVe**.

GP (GloVe): I apply the proposed *gender-preserving* (**GP**) debiasing method on pre-trained GloVe embeddings to debias it.

GP (GN-GloVe): To test whether I can use the proposed method to further debias word embeddings that are already debiased using other methods, I apply it on GN-GloVe.

5.3.3 Evaluating Debiasing Performance

I use the SemBias dataset created by J. Zhao, Zhou, et al. (2018) to evaluate the level of gender bias in word embeddings. Each instance in SemBias consists of four word pairs: a gender-definition word pair (**Definition**; e.g. "waiter - waitress"), a gender-stereotype word pair (**Stereotype**; e.g., "doctor - nurse") and two other word-pairs that have similar meanings but not a gender relation (**None**; e.g., "dog - cat", "cup - lid"). SemBias contains 20 gender-stereotype word pairs and 22 gender-definitional word pairs and use their Cartesian product to generate

440 instances. Among the 22 gender-definitional word pairs, 2 word-pairs are not used as the seeds for training. Following, J. Zhao, Zhou, et al. (2018), to test the generalisability of a debiasing method, I use the subset (SemBias-subset) of 40 instances associated with these 2 pairs. I measure relational similarity between (he, she) word-pair and a word-pair (a, b) in SemBias using the cosine similarity between the $\overrightarrow{he} - \overrightarrow{she}$ gender directional vector and $\boldsymbol{a} - \boldsymbol{b}$ using the word embeddings under evaluation. For the four word-pairs in each instance in SemBias, I select the word-pair with the highest cosine similarity with $\overrightarrow{he} - \overrightarrow{she}$ as the predicted answer. In Table 5.1, I show the percentages where a word-pair is correctly classified as **Definition**, **Stereotype**, or **None**. If the word embeddings are correctly debiased, I would expect a high accuracy for **Definitions** and low accuracies for **Stereotypes** and **Nones**.

From Table 5.1, I see that the best performances (highest accuracy on **Definition** and lowest accuracy on **Stereotype**) are reported by **GP (GN-GloVe)**, which is the application of the proposed method to debias word embeddings learnt by **GN-GloVe**. In particular, in both SemBias and SemBias-subset, **GP (GN-GloVe)** statistically significantly outperforms **GloVe** and **Hard-Glove** according to Clopper-Pearson confidence intervals (Clopper & Pearson, 1934). Although **GN-GloVe** obtains high performance on SemBias, it does not generalise well to SemBias-subset. However, by applying the proposed method, I can further remove any residual gender biases from **GN-GloVe**, which shows that the proposed method can be applied in conjunction with **GN-GloVe**. I see that **GloVe** contains a high percentage of stereotypical gender biases, which justifies the need for debiasing methods. By applying the proposed method on **GloVe** (corresponds to **GP (GloVe)**) I can decrease the gender biases in **GloVe**, while preserving useful gender-related information for detecting definitional word-pairs. Comparing corresponding **AE** and **GP** versions of **GloVe** and **GN-GloVe**, I see that autoencoding

Embeddings	sem	syn	total	MSR	SE
GloVe	80.1	62.1	70.3	53.8	38.8
Hard-GloVe	80.3	**62.7**	**70.7**	**54.4**	39.1
GN-GloVe	77.8	60.9	68.6	51.5	39.1
AE (GloVe)	**81.0**	61.9	70.5	52.6	38.9
AE (GN-GloVe)	78.6	61.3	69.2	51.2	39.1
GP (GloVe)	80.5	61.0	69.9	51.3	38.5
GP (GN-GloVe)	78.3	61.3	69.0	51.0	**39.6**

Table 5.2: Accuracy for solving word analogies.

alone is insufficient to consistently preserve gender-related information.

5.3.4 Bias in Downstream Task

5.3.5 Preservation of Word Semantics

It is important that the debiasing process removes only gender biases and preserve other information unrelated to gender biases in the original word embeddings. If too much information is removed from word embeddings during the debiasing process, then the debiased embeddings might not carry adequate information for downstream NLP tasks that use those debiased word embeddings.

To evaluate the semantic accuracy of the debiased word embeddings, following prior work on debiasing (Bolukbasi et al., 2016; J. Zhao, Wang, et al., 2018), I use them in two popular tasks: semantic similarity measurement and analogy detection. I recall that I do *not* propose novel word embedding learning methods in this dissertation, and what is important here is whether the debiasing process preserves as much information as possible in the original word embeddings.

Datasets	#Orig	#Bal
WS	353	366
RG	65	77
MTurk	771	784
RW	2,034	2,042
MEN	3,000	3,122
SimLex	999	1,043

Table 5.3: Number of word-pairs in the original (**Orig**) and balanced (**Bal**) similarity benchmarks.

Analogy Detection

Given three words a, b, c in analogy detection, I must predict a word d that completes the analogy "a is b as c is to d". I use the CosAdd (Levy & Goldberg, 2014) that finds $\boldsymbol{d}$ that has the maximum cosine similarity with $(\boldsymbol{b} - \boldsymbol{a} + \boldsymbol{c})$. I use the semantic (**sem**) and syntactic (**syn**) analogies in the Google analogy dataset (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) (in **total** contains 19,556 questions), **MSR** dataset (7,999 syntactic questions) (Mikolov, Yih, & Zweig, 2013) and SemEval dataset (**SE**, 79 paradigms) (Jurgens, Mohammad, Turney, & Holyoak, 2012) as benchmark datasets. The percentage of correctly solved analogy questions is reported in Table 5.2. I see that there is no significant degradation of performance due to debiasing using the proposed method.

Semantic Similarity Measurement

The correlation between the human ratings and similarity scores computed using word embeddings for pairs of words has been used as a measure of the quality of the word embeddings (Mikolov, Yih, & Zweig, 2013). I compute cosine similarity between word embeddings and measure Spearman correlation against human ratings for the word-pairs in the following benchmark datasets: Word Similarity 353 dataset (**WS**) (Finkelstein et al., 2001), Rubenstein-Goodenough dataset

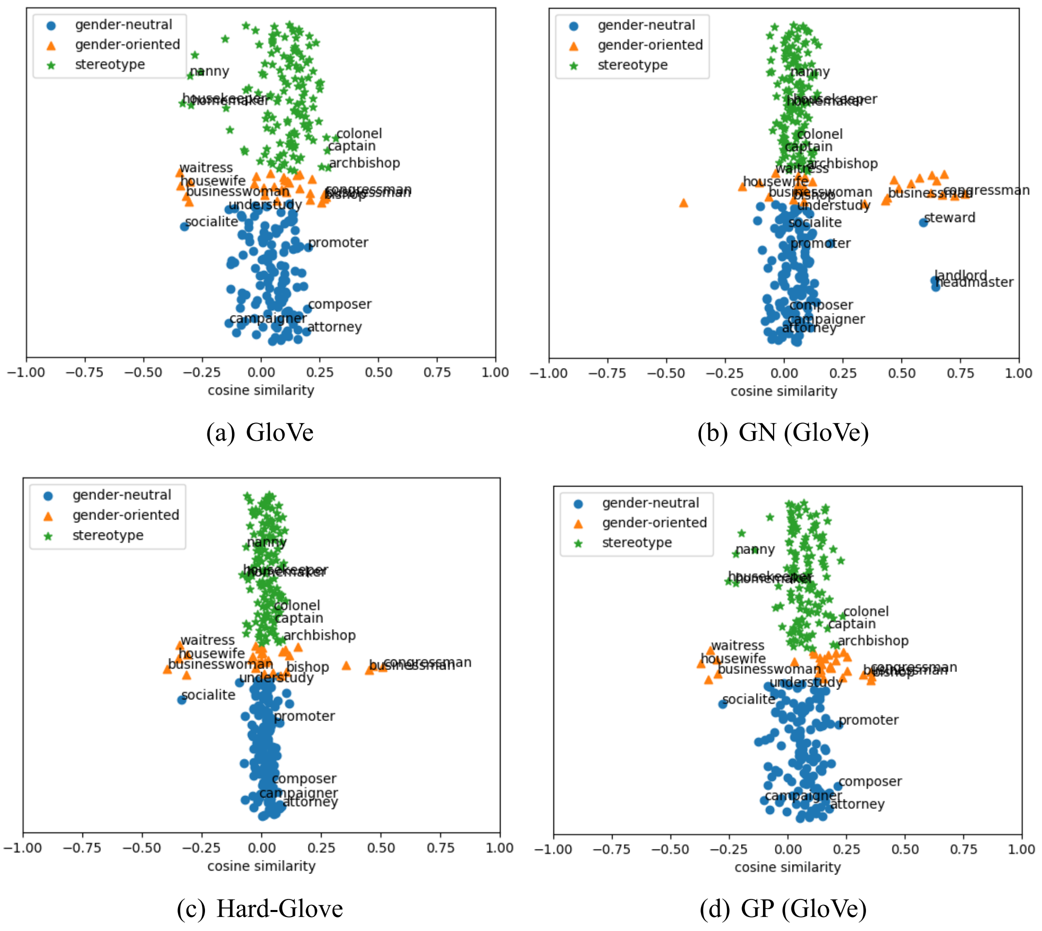
Embeddings	WS		RG		MTurk		RW		MEN		SimLex	
	Orig	Bal	Orig	Bal	Orig	Bal	Orig	Bal	Orig	Bal	Orig	Bal
GloVe	61.6	62.9	75.3	75.5	64.9	63.9	37.3	37.5	73.0	72.6	34.7	35.9
Hard-GloVe	61.7	63.1	76.4	76.7	65.1	64.1	37.4	37.4	72.8	72.5	35.0	36.1
GN-GloVe	62.5	63.7	74.1	73.7	66.2	65.5	40.0	40.1	74.9	74.5	37.0	38.1
AE (GloVe)	61.3	62.6	**77.1**	**76.8**	64.9	64.1	35.7	35.8	71.9	71.5	34.7	35.9
AE (GN-GloVe)	61.3	62.6	73.0	74.0	66.3	65.5	38.7	38.9	73.8	73.4	36.7	37.7
GP (GloVe)	59.7	61.0	75.4	75.5	63.9	63.1	34.7	34.8	70.8	70.4	33.9	35.0
GP (GN-GloVe)	**63.2**	**64.3**	72.2	72.2	**67.9**	**67.4**	**43.2**	**43.3**	**75.9**	**75.5**	**38.4**	**39.5**

Table 5.4: Spearman correlation between human ratings and cosine similarity scores computed using word embeddings for the word-pairs in the original and balanced versions of the benchmark datasets.

(**RG**) (Rubenstein & Goodenough, 1965), **MTurk** (Halawi, Dror, Gabrilovich, & Koren, 2012), rare words dataset (**RW**) (Luong, Socher, & Manning, 2013), **MEN** dataset (Bruni, Boleda, Baroni, & Tran, 2012) and **SimLex** dataset (Hill, Reichart, & Korhonen, 2015).

Unfortunately, existing benchmark datasets for semantic similarity were not created considering gender-biases and contain many stereotypical examples. For example, in **MEN**, the word *sexy* has high human similarity ratings with *lady* and *girl* compared to *man* and *guy*. Furthermore, masculine words and *soldier* are included in multiple datasets with high human similarity ratings, whereas it is not compared with feminine words in any of the datasets. Although prior work studying gender bias have used these datasets for evaluation purposes (Bolukbasi et al., 2016; J. Zhao, Wang, et al., 2018), I note that high correlation with human ratings can be achieved with biased word embeddings.

To address this issue, I *balance* the original datasets with respect to gender by including extra word pairs generated from the opposite gender with the same human ratings. For instance, if the word-pair (*baby*, *mother*) exists in the dataset, I add a new pair (*baby*, *father*) to the dataset. Ideally, I should re-annotate this



(a) GloVe (b) GN (GloVe)

(c) Hard-Glove (d) GP (GloVe)

Figure 5.1: Cosine similarity between gender, gender-neutral, stereotypical words and the gender direction.

balanced version of the dataset to obtain human similarity ratings. However, such a re-annotation exercise would be costly and inconsistent with the original ratings. Therefore, I resort to a proxy where I reassign the human rating for the original word-pair to its derived opposite gender version. Table 5.3 shows the number of word-pairs in the original (**Orig**) and balanced (**Bal**) similarity benchmarks.

As shown in Table 5.4, **GP (GloVe)** and **GP (GN-GloVe)** obtain the best performance on the balanced versions of all benchmark datasets. Moreover, the performance of **GP (GloVe)** on both original and balanced datasets is comparable to that of **GloVe**, which indicates that the information encoded in GloVe embeddings are preserved in the debiased embeddings, while removing stereotypical gender biases. The autoencoded versions report similar performance to the original input embeddings.

Overall, the results on the analogy detection and semantic similarity measurement tasks show that the proposed method removes only gender-biases and preserve other useful gender-related information.

5.3.6 Visualising the Effect of Debiasing

To visualise the effect of debiasing on different word categories, I compute the cosine similarity between the gender directional vector $\overrightarrow{he} - \overrightarrow{she}$, and selected *gender-oriented* (female or male), *gender-neutral* and stereotypical words. In Figure 5.1, horizontal axises show the cosine similarity with the gender directional vector (positive scores for masculine words) and the words are alphabetically sorted within each category.

From Figure 5.1, I see that the original **GloVe** embeddings show a similar spread of cosine similarity scores for gender-oriented as well as stereotypical words. When debiased by hard-debias (**Hard-GloVe**) and **GN-GloVe**, I see that stereotypical and gender-neutral words get their gender similarity scores equally reduced. Interestingly, **Hard-GloVe** shifts even gender-oriented words towards the masculine direction. On the other hand, **GP (GloVe)** decreases gender bias in the stereotypical words, while almost preserving gender-neutral and gender-oriented words as in **GloVe**.

Considering that a significant number of words in English are gender-neutral, it is essential that debiasing methods do not adversely change their orientation. In particular, the proposed method's ability to debias stereotypical words that carry

Embeddings	OntoNotes	T1-p	T1-a	AVG	|Diff|	T2-p	T2-a	AVG	|Diff|
GloVe	60.4	**69.2**	52.6	60.9	16.6	**79.4**	68.2	**73.8**	11.2
Hard-GloVe	60.5	64.2	**58.7**	**61.5**	**5.5**	60.9	63.2	62.1	2.3
GN-GloVe	60.6	68.7	50.5	59.6	18.2	76.1	66.2	71.2	9.9
GP (GloVe)	60.8	68.6	54.3	**61.5**	14.3	73.3	69.0	71.2	4.3
GP (GN-GloVe)	**61.0**	69.0	53.4	61.2	15.6	72.4	**71.3**	71.9	**1.1**

Table 5.5: F_1 on OntoNotes and WinoBias test sets.

unfair gender-biases, while preserving the gender-orientation in feminine, masculine and neutral words is important when applying the debiased word embeddings in NLP applications that depend on word embeddings for representing the input texts.

5.3.7 Measuring Bias with Coreference Resolution

Finally, I verify the effectiveness of the proposed method in the downstream task. Specifically, I evaluate whether I can debias coreference resolution model, which finds all expressions that refer to the same entity in a text, using the **WinoBias** data set (J. Zhao, Wang, et al., 2018). It contains two types of sentences that require linking gendered pronouns to either male or stereotypical female occupations. A system is considered to be gender-biased if it links pronouns to occupations dominated by the gender of the pronoun (which is pro-stereotyped condition) more accurately than occupations not dominated by the gender of the pronoun (which is anti-stereotyped condition). I used the model implemented in AllenNLP[8]. I initialized word embeddings of the model by these five word embeddings: GloVe, Hard-GloVe, GN-GloVe, GP (GloVe), and GP (GN-GloVe). Additionally, I also use OntoNotes data (Weischedel et al., 2013) to show that the pre-trained information is not forgotten after debiasing.

[8]`http://docs.allennlp.org/v0.9.0/api/allennlp.models.coreference_resolution.html`

Table 5.5 shows the results on OntoNotes and Winobias test sets. In T1, coreference decisions must be made using the world knowledge about given circumstances. T2 can be resolved using syntactic information and understanding of the pronoun. Here, p and a stand for pro-stereotyped and anti-stereotyped conditions. A system passes if, for both T1 and T2 examples, pro-stereotyped and anti-stereotyped coreference decisions are made with the same accuracy. It can be seen that my proposed method can debias both T1 and T2 without loss of precision.

6 | Conclusions

6.1 Conclusions

In this thesis, I studied the problems of grammatical and semantic biases learned from raw datasets in representation learning. A grammatical bias is a bias learned from the raw corpus, which mostly consists of grammatically correct text but includes very little erroneous text, which has a negative effect on tasks that deal with grammatically incorrect data. Semantic bias is learned from the biased co-occurrence of words in the raw corpora. As a result, discriminatory information is learned in the meanings of some words. I resolved these biases in word embeddings and language representations.

First, I solved the problem of grammatical bias in word embeddings by using pseudo-errors for word embeddings in chapter 2. Therefore, by assigning pseudo-errors to the raw corpus and learning word embeddings from it, grammatically correct and grammatically incorrect information can be considered equally. Experimental results showed that these grammatical error-aware word embeddings improve the performance of GED for grammatical errors.

In chapter 3, I proposed a method of considering grammatical errors in language representations to remove grammatical bias for GEC. Experiments showed that fine-tuning pre-trained language representations with GED corpus was the best way to account for grammatical errors. Furthermore, experiments showed that the language representations fine-tuned by the GED can distinguish between grammatically correct and grammatically incorrect texts.

In chapter 4, I proposed a method which effectively use language representations with learned grammatical bias. In a neural network model, different information is learned at each layer, and my method allowed the model to dynamically select the information from the layer that is best suited to handle grammatically incorrect information. I showed that the proposed method that uses each layer's information is better than the previous method using only the information of the final layer. The analysis results showed that it is important to use information from all layers in the GED.

Finally, I addressed removing gender discrimination information in word embeddings due to semantic bias in chapter chapter 5. Furthermore, I showed that debiased word embeddings retain useful information such as analogy and semantic similarity.

In this thesis, I have shown that biases are learned from raw data unintentional to humans, and that we must explicitly remove or effectively exploit them. In addition to the biases discussed in this thesis, many other biases are learned that may cause poor performance of the model or cause socially problematic behavior in natural language processing systems. Therefore, in order for artificial intelligence to be accepted by humans, we need to be aware of known and unknown biases in our research and development of representation learning.

6.2 Future Work

Here, I address the issue of bias as future work, which has not been addressed in this dissertation or in existing research.

Grammatical bias in languages other than English: Cross-lingual GEC (Yamashita, Katsumata, Kaneko, Imankulova, & Komachi, 2020) models have been proposed using language representations, and they reported improvements in correction performance for low-resource languages. On the other hand, it is not obvious that the methods discussed in this dissertation for removing grammatical bias in word embeddings and language representations are effective for languages other than English.

Using grammatical bias for building robust NLP models: I resolved the grammatical bias in the language teaching and language learning tasks. Grammatical bias causes significant performance degradation in various NLP tasks, such as machine translation (S. Zhou, Zeng, Zhou, Anastasopoulos, & Neubig, 2019) and sentiment analysis (Pruthi, Dhingra, & Lipton, 2019), when grammatically incorrect text is given as input. Therefore, the proposed techniques for reducing grammatical bias may be used to build robust models for other tasks.

Semantic bias in language representations: It is known that language representations also contain semantic bias (Kurita et al., 2019; May et al., 2019). In order to adapt the proposed methods to language representations, we need to define a gender vector that takes the context into account. It is an interesting question whether the debiasing by orthogonality with the gender vector will work for more expressive language representations with more parameters.

Bibliography

Alec, R., Karthik, N., Tim, S., & Ilya, S. (2018). Improving Language Understanding with Unsupervised Learning. *Technical report, OpenAI*.

Alikaniotis, D., Yannakoudakis, H., & Rei, M. (2016). Automatic Text Scoring Using Neural Networks. In ACL (pp. 715–725).

Al-Rfou, R., Choe, D., Constant, N., Guo, M., & Jones, L. (2019). Character-Level Language Modeling with Deeper Self-Attention. In AAAI.

Asano, H., Mita, M., Mizumoto, T., & Suzuki, J. (2019). The AIP-Tohoku System at the BEA-2019 Shared Task. In BEA (pp. 176–182).

Awasthi, A., Sarawagi, S., Goyal, R., Ghosh, S., & Piratla, V. (2019). Parallel Iterative Edit Models for Local Sequence Transduction. In EMNLP-IJCNLP (pp. 4259–4269).

Belinkov, Y., Durrani, N., Dalvi, F., Sajjad, H., & Glass, J. (2017). What do Neural Machine Translation Models Learn about Morphology? In ACL (pp. 861–872).

Bell, S., Yannakoudakis, H., & Rei, M. (2019). Context is Key: Grammatical Error Detection with Contextual Word Representations. In BEA (pp. 103–115).

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*(Feb), 1137–1155.

Bolukbasi, T., Chang, K., Zou, J. Y., Saligrama, V., & Kalai, A. (2016). Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. In NeurIPS.

Bommasani, R., Davis, K., & Cardie, C. (2020). Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings. In ACL (pp. 4758–4781).

Bruni, E., Boleda, G., Baroni, M., & Tran, N. K. (2012). Distributional Semantics in Technicolor. In ACL (pp. 136–145).

Bryant, C., Felice, M., Andersen, Ø. E., & Briscoe, T. (2019). The BEA-2019 Shared Task on Grammatical Error Correction. In BEA (pp. 52–75).

Bryant, C., Felice, M., & Briscoe, T. (2017). Automatic Annotation and Evaluation of Error Types for Grammatical Error Correction. In ACL (pp. 793–805).

Caliskan, A., Bryson, J. J., & Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. *Science*, *356*, 183–186.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., & Robinson, T. (2013). One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In EMNLP (pp. 1724–1734).

Chollampatt, S., Hoang, D. T., & Ng, H. T. (2016). Adapting Grammatical Error Correction Based on the Native Language of Writers with Neural Network Joint Models. In EMNLP (p. 1901-1911).

Chollampatt, S., Taghipour, K., & Ng, H. T. (2016). Neural Network Translation Models for Grammatical Error Correction. *arXiv*.

Chollampatt, S., Wang, W., & Ng, H. T. (2019). Cross-Sentence Grammatical Error Correction. In ACL (pp. 435–445).

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In ICLR.

Clopper, C. J., & Pearson, E. S. (1934). The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial. *Biometrika*, *26*(4), 404 – 413.

Collobert, R., & Weston, J. (2008). A Unified Architecture for Natural Language

Processing: Deep Neural Networks with Multitask Learning. In ICML (pp. 160–167).

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research*, *12*, 2493–2537.

Dahlmeier, D., & Ng, H. T. (2012). Better Evaluation for Grammatical Error Correction. In NAACL (pp. 568–572).

Dahlmeier, D., Ng, H. T., & Wu, S. M. (2013). Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In BEA (pp. 22–31).

Dev, S., Li, T., Phillips, J., & Srikumar, V. (2020). On Measuring and Mitigating Biased Inferences of Word Embeddings. In AAAI.

Dev, S., & Phillips, J. M. (2019). Attenuating Bias in Word vectors. In AISTATS (Vol. 89, pp. 879–887).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In NAACL (pp. 4171–4186).

Deyne, S. D., Navarro, D. J., Perfors, A., Brysbaert, M., & Storms, G. (2019). The "Small World of Words" English word association norms for over 12,000 cue words. *Behavior Research Methods*, *51*, 987-1006.

Du, Y., Wu, Y., & Lan, M. (2019). Exploring Human Gender Stereotypes with Word Association Test. In EMNLP (pp. 6133–6143).

Elazar, Y., & Goldberg, Y. (2018). Adversarial Removal of Demographic Attributes from Text Data. In EMNLP (pp. 11–21).

European Union. (1997). Treaty of Amsterdam (Article 13).

Felice, M., Bryant, C., & Briscoe, T. (2016). Automatic Extraction of Learner Errors in ESL Sentences Using Linguistically Enhanced Alignments. In

COLING (pp. 825–835).

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., & Ruppin, E. (2001). Placing Search in Context: The Concept Revisited. In WWW (pp. 406–414).

Garimella, A., Banea, C., & Mihalcea, R. (2017). Demographic-aware word associations. In EMNLP (pp. 2275–2285).

Granger, S. (1998). Developing an Automated Writing Placement System for ESL Learners. In LEC (pp. 3–18).

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5), 602–610.

Greenwald, A. G., McGhee, D. E., & Schwatz, J. L. K. (1998). Measuring Individual Differences in Implicit Cognition: The Implicit Association Test. *Journal of Personality and Social Psychology*, *74*(6), 1464–1480.

Grundkiewicz, R., Junczys-Dowmunt, M., & Heafield, K. (2019). Neural Grammatical Error Correction Systems with Unsupervised Pre-training on Synthetic Data. In BEA (pp. 252–263).

Halawi, G., Dror, G., Gabrilovich, E., & Koren, Y. (2012). Large-scale Learning of Word Relatedness with Constraints. In ACM KDD (pp. 1406–1414). ACM.

Han, N.-R., Chodorow, M., & Leacock, C. (2006). Detecting errors in English article usage by non-native speakers. *Natural Language Engineering.*, 115–129.

Harris, Z. S. (1954). Distributional structure. *Word*, *10*(2-3), 146–162.

Hill, F., Reichart, R., & Korhonen, A. (2015). SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation. *Computational Linguistics*, *41*(4), 665–695.

Hinton, G. E. (1984). Distributed representations. *arXiv*.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780.

Holstein, K., Vaughan, J. W., III, H. D., Dudík, M., & Wallach, H. (2018). Improving fairness in machine learning systems: What do industry practitioners need? *arXiv*.

Jurgens, D., Mohammad, S., Turney, P., & Holyoak, K. (2012). SemEval-2012 Task 2: Measuring Degrees of Relational Similarity. In *SEM (pp. 356–364).

Kaneko, M., Hotate, K., Katsumata, S., & Komachi, M. (2019). TMU Transformer System Using BERT for Re-ranking at BEA 2019 Grammatical Error Correction on Restricted Track. In BEA (pp. 207–212).

Kaneko, M., & Komachi, M. (2019). Multi-Head Multi-Layer Attention to Deep Language Representations for Grammatical Error Detection. *Computación y Sistemas*, *23*.

Kaneko, M., Sakaizawa, Y., & Komachi, M. (2017). Grammatical Error Detection Using Error- and Grammaticality-Specific Word Embeddings. In IJCNLP (pp. 40–48).

Kantor, Y., Katz, Y., Choshen, L., Cohen-Karlik, E., Liberman, N., Toledo, A., … Slonim, N. (2019). Learning to combine Grammatical Error Corrections. In BEA (pp. 139–148).

Karve, S., Ungar, L., & Sedoc, J. (2019). Conceptor Debiasing of Word Representations Evaluated on WEAT. In Gender Bias in NLP (pp. 40–48).

Kasewa, S., Stenetorp, P., & Riedel, S. (2018). Wronging a Right: Generating Better Errors to Improve Grammatical Error Detection. In EMNLP (pp. 4977–4983).

Kenter, T., & De Rijke, M. (2015). Short text similarity with word embeddings. In CIKM (pp. 1411–1420).

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. In ICLR.

Kiyono, S., Suzuki, J., Mita, M., Mizumoto, T., & Inui, K. (2019). An Empirical Study of Incorporating Pseudo Data into Grammatical Error Correction. In EMNLP-IJCNLP (pp. 1236–1242).

Kochmar, E., & Briscoe, T. (2014). Detecting Learner Errors in the Choice of Content Words Using Compositional Distributional Semantics. In COLING (pp. 1740–1751).

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In ICML (pp. 179 – 186).

Kurita, K., Vyas, N., Pareek, A., Black, A. W., & Tsvetkov, Y. (2019). Measuring Bias in Contextualized Word Representations. In Gender Bias in NLP (pp. 166–172).

Lample, G., & Conneau, A. (2019). Cross-lingual Language Model Pretraining. *arXiv*.

Levy, O., & Goldberg, Y. (2014). Linguistic regularities in sparse and explicit word representations. In CoNLL (pp. 171–180).

Li, Y., Baldwin, T., & Cohn, T. (2018). Towards Robust and Privacy-preserving Text Representations. In ACL (pp. 25–30).

Lichtarge, J., Alberti, C., Kumar, S., Shazeer, N., Parmar, N., & Tong, S. (2019). Corpora Generation for Grammatical Error Correction. In NAACL (pp. 3291–3301).

Liu, X., Han, B., Li, K., Stiller, S. H., & Zhou, M. (2010). SRL-based verb selection for ESL. In EMNLP (pp. 1068–1076).

Liu, Y. (2019). Fine-tune BERT for Extractive Summarization. *arXiv*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., … Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv*.

Luong, T., Socher, R., & Manning, C. (2013). Better Word Representations with Recursive Neural Networks for Morphology. In CoNLL (pp. 104–113).

May, C., Wang, A., Bordia, S., Bowman, S. R., & Rudinger, R. (2019). On Measuring Social Biases in Sentence Encoders. In NAACL-HLT (pp. 622–628).

Mikolov, T., Chen, K., & Dean, J. (2013). Efficient estimation of word representation in vector space. In ICLR.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In Advances in Neural Information Processing Systems 26 (pp. 3111–3119).

Mikolov, T., Yih, W., & Zweig, G. (2013). Linguistic Regularities in Continuous Space Word Representations. In NAACL-HLT (pp. 746–751).

Mita, M., Mizumoto, T., Kaneko, M., Nagata, R., & Inui, K. (2019). Cross-Corpora Evaluation and Analysis of Grammatical Error Correction Models — Is Single-Corpus Evaluation Enough? In NAACL (pp. 1309–1314).

Mizumoto, T., Komachi, M., Nagata, M., & Matsumoto, Y. (2011). Mining Revision Log of Language Learning SNS for Automated Japanese Error Correction of Second Language Learners. In IJCNLP (pp. 147–155).

Nagata, R., & Nakatani, K. (2010). Evaluating performance of grammatical error detection to maximize learning effect. In COLING (pp. 894–900).

Napoles, C., Sakaguchi, K., Post, M., & Tetreault, J. (2015). Ground Truth for Grammatical Error Correction Metrics. In NAACL (pp. 588–593).

Napoles, C., Sakaguchi, K., & Tetreault, J. (2017). JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction. In EACL (pp. 229–234).

Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., & Bryant, C. (2014). The CoNLL-2014 Shared Task on Grammatical Error Correction. In CoNLL (pp. 1–14).

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: global vectors for word representation. In EMNLP (pp. 1532–1543).

Peters, M., Neumann, M., Zettlemoyer, L., & Yih, W.-t. (2018). Dissecting Contextual Word Embeddings: Architecture and Representation. In EMNLP (pp. 1499–1509).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In NAACL-HLT (pp. 2227–2237).

Pruthi, D., Dhingra, B., & Lipton, Z. C. (2019). Combating Adversarial Misspellings with Robust Word Recognition. In ACL (pp. 5582–5591).

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., & Huang, X. (2020). Pre-trained Models for Natural Language Processing: A Survey. *arXiv*.

Ravfogel, S., Elazar, Y., Gonen, H., Twiton, M., & Goldberg, Y. (2020). Null It Out: Guarding Protected Attributes by Iterative Nullspace Projection. In ACL (pp. 7237–7256).

Rei, M. (2017). Semi-supervised Multitask Learning for Sequence Labeling. In ACL (pp. 2121–2130).

Rei, M., Felice, M., Yuan, Z., & Briscoe, T. (2017). Artificial Error Generation with Machine Translation and Syntactic Patterns. In BEA (pp. 287–292).

Rei, M., & Søgaard, A. (2018). Zero-Shot Sequence Labeling: Transferring Knowledge from Sentences to Tokens. In NAACL-HLT (pp. 293–302).

Rei, M., & Søgaard, A. (2019). Jointly Learning to Label Sentences and Tokens. In AAAI (pp. 6916–6923).

Rei, M., & Yannakoudakis, H. (2016, August). Compositional Sequence Labeling Models for Error Detection in Learner Writing. In ACL (pp. 1181–1191).

Rothe, S., Narayan, S., & Severyn, A. (2019). Leveraging pre-trained checkpoints for sequence generation tasks. *arXiv*.

Rubenstein, H., & Goodenough, J. B. (1965). Contextual Correlates of Synonymy. *ACM*, *8*(10), 627–633.

Rudinger, R., Naradowsky, J., Leonard, B., & Van Durme, B. (2018). Gender Bias in Coreference Resolution. In NAACL-HLT (pp. 8–14).

Sawai, Y., Komachi, M., & Matsumoto, Y. (2013). A Learner Corpus-based Approach to Verb Suggestion for ESL. In ACL (pp. 708–713).

Sennrich, R., Haddow, B., & Birch, A. (2016). Edinburgh Neural Machine Translation Systems for WMT 16. In WMT (pp. 371–376).

Shi, B., Fu, Z., Bing, L., & Lam, W. (2018). Learning Domain-Sensitive and Sentiment-Aware Word Embeddings. In ACL (pp. 2494–2504).

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., & Manning, C. D. (2011). Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In EMNLP (pp. 151–161).

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In (Vol. 15, pp. 1929–1958).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In IEEE.

Takase, S., Suzuki, J., & Nagata, M. (2018). Direct Output Connection for a High-Rank Language Model. In EMNLP (pp. 4599–4609).

Taylor, W. L. (1953). Cloze Procedure: A New Tool for Measuring Readability. In (Vol. 30, pp. 415–433).

Tetreault, J. R., & Chodorow, M. (2008). The Ups and Downs of Preposition Error Detection in ESL Writing. In COLING (pp. 865–872).

The Telegraph. (2016). Microsoft deletes 'teen girl' ai after it became a hitlter-loving sex robot within 24 hours. https://goo.gl/mE8p3J.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., …

Polosukhin, I. (2017). Attention is All you Need. In NeurIPS (pp. 5998–6008).

Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., … others (2013). Ontonotes release 5.0. *Linguistic Data Consortium*, *23*.

Weng, R., Yu, H., Huang, S., Cheng, S., & Luo, W. (2019). Acquiring Knowledge from Pre-trained Model to Neural Machine Translation. *arXiv*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., … Brew, J. (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., … others (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv*.

Xie, Q., Dai, Z., Du, Y., Hovy, E., & Neubig, G. (2017). Controllable Invariance through Adversarial Feature Learning. In NeurIPS (pp. 585–596).

Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., & Ng, A. Y. (2016). Neural Language Correction with Character-Based Attention. *arXiv*.

Xie, Z., Genthial, G., Xie, S., Ng, A., & Jurafsky, D. (2018). Noising and Denoising Natural Language: Diverse Backtranslation for Grammar Correction. In NAACL (pp. 619–628).

Yamashita, I., Katsumata, S., Kaneko, M., Imankulova, A., & Komachi, M. (2020). Cross-lingual Transfer Learning for Grammatical Error Correction. In COLING (pp. 4704–4715).

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. In Advances in Neural Information Processing Systems 32 (pp. 5753–5763).

Yannakoudakis, H., Andersen, Ø. E., Ardeshir, G., Ted, B., & Diane, N. (2018).

Developing an Automated Writing Placement System for ESL Learners. In Applied Measurement in Education (pp. 251–267).

Yannakoudakis, H., Briscoe, T., & Medlock, B. (2011). A New Dataset and Method for Automatically Grading ESOL Texts. In NAACL (pp. 180–189).

Yin, F., Long, Q., Meng, T., & Chang, K. (2020). On the Robustness of Language Encoders against Grammatical Errors. In ACL (pp. 3386–3403).

Zhang, H., Gong, Y., Yan, Y., Duan, N., Xu, J., Wang, J., … Zhou, M. (2019). Pretraining-Based Natural Language Generation for Text Summarization. In CoNLL (pp. 789–797).

Zhang, S., Dinan, E., Urbanek, J., Szlam, A., Kiela, D., & Weston, J. (2018). Personalizing Dialogue Agents: I have a dog, do you have pets too? In ACL (pp. 2204–2213).

Zhao, J., Wang, T., Yatskar, M., Cotterell, R., Ordonez, V., & Chang, K.-W. (2019). Gender Bias in Contextualized Word Embeddings. In NAACL-HLT (pp. 629–634).

Zhao, J., Wang, T., Yatskar, M., Ordonez, V., & Chang, K.-W. (2018). Gender Bias in Coreference Resolution: Evaluation and Debiasing Methods. In NAACL-HLT (pp. 15–20).

Zhao, J., Zhou, Y., Li, Z., Wang, W., & Chang, K.-W. (2018). Learning Gender-Neutral Word Embeddings. In EMNLP (pp. 4847–4853).

Zhao, W., Wang, L., Shen, K., Jia, R., & Liu, J. (2019). Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data. In NAACL (pp. 156–165).

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., & Schölkopf, B. (2003). Learning with Local and Global Consistency. In NeurIPS (pp. 321–328).

Zhou, S., Zeng, X., Zhou, Y., Anastasopoulos, A., & Neubig, G. (2019). Improving Robustness of Neural Machine Translation with Multi-task Learning. In

WMT (pp. 565–571).

Zhu, J., Xia, Y., Wu, L., He, D., Qin, T., Zhou, W., … Liu, T. (2020). Incorporating BERT into Neural Machine Translation. In ICLR.

Zou, W. Y., Socher, R., Cer, D., & Manning, C. D. (2013). Bilingual Word Embeddings for Phrase-Based Machine Translation. In EMNLP (pp. 1393 – 1398).