文の埋め込みに効果的な静的単語ベクトルの獲得

和田崇史　平川優伎　清水良太郎　川島貴大　斎藤侑輝
ZOZO Research
{takashi.wada,yuki.hirakawa,ryotaro.shimizu}@zozo.com
{takahiro.kawashima,yuki.saito}@zozo.com

概要

本研究は, 文の埋め込みに有効な静的単語ベクトル (Static Word Embedding) を提案する. 文ベクトルを出力するように事前学習された Sentence Transformer から単語ベクトルを抽出し, さらに主成分分析と知識蒸留を行うことで文表現に適した単語ベクトルを獲得する. 計 56 個の英語のデータセットから成る Massive Text Embedding Benchmark (MTEB) で実験を行い, 既存手法の精度を大幅に上回ることを示した.

1　はじめに

近年, 様々なタスク (検索・分類・クラスタリング等) に応用可能な文埋め込みのモデルの研究が盛んに行われている. 現在主流の手法は, BERT[1] や最近では LLaMA 3[2] のような大規模言語モデル (LLM) を様々なタスクのラベル付きデータで fine-tune し, 入力文の意味を捉えた固定長ベクトルを出力するモデルを学習する [3, 4]. しかし, こうした手法は多くの計算資源を必要とするため, 大量のテキストを低コストで高速に処理することや, スマートフォンのような計算資源が限られる機器でモデルを動かすことが困難となっている.

そこで本研究は, 文埋め込みに有効な静的単語ベクトル (Static Word Embedding) を提案する. 文埋め込みのタスクで事前学習された Transformer[5] ベースのモデル (Sentence Transformer)[3] から単語ベクトルを抽出し, 更にそれを主成分分析や知識蒸留で改善する. 推論時は, テキストを単語ベクトルの平均で表す所謂 bag-of-words モデルであるため, CPU 上でも高速に動かすことが出来る. 精度評価を計 56 個の英語のデータセットから成る Massive Text Embedding Benchmark (MTEB)[6] で行い, 既存の静的単語ベクトルの精度を大幅に上回ることを示した. また, SimCSE[7] のような基礎的な Sentence Transformer の精度も上回り, 実応用上における有効性も示した.

2　提案手法

2.1　静的単語ベクトルの抽出

まず, 文埋め込みに最適化された Transformer モデル (**S**entence **T**ransformer, 以下 **ST**) から静的単語ベクトル (**S**tatic **W**ord **E**mbedding, 以下 **SWE**) を抽出する. 任意の語彙集合 V に含まれる単語 $w \in V$ のベクトル $\mathrm{SWE}(w) \in \mathbb{R}^d$ を得るために, まず大規模な Web コーパス (CC-100[8, 9]) から w を含んだ文を $N(=100)$ 文サンプリングし, 各文 z_i $(i = 1, 2, ...N)$ を ST で埋め込む. そして, Transformer の最終層から出力される $|z_i|$ 個のベクトル ($|z_i|$ は z_i のトークン数) のうち, w が入力された位置のベクトルを抽出する. これを N 文でそれぞれ計算し, それらの平均を取ることで非文脈化された w の SWE を得る. なお, 本研究では語彙集合 V を (小文字化した)Web コーパスで最も使用頻度が高い 15 万単語と設定したため, 低頻度語は ST の tokenizer でサブワードに分割される. その場合, Transformer の最終層でサブワードの平均を取り, それを N 文で平均したものを SWE とみなす.

2.2　文ベクトルの主成分分析

前節で得られた SWE に主成分分析 (PCA) を適用し, 文表現により適したベクトルを獲得する. まず, 前節で用いた文集合から $M(=100\mathrm{k})$ 文をランダムに抽出し, 各文を SWE の平均で表す. そして得られた文ベクトル行列 $X \in \mathbb{R}^{M \times d}$ に PCA を適用し, 中心化された行列 $X - \bar{X}$ を新しい軸に写像する行列 $W \in \mathbb{R}^{d \times d}$ を得る. 元のベクトル空間を W で写像することで文の分散が捉えやすくなり, 文埋め込みのタスクで精度が改善した. なお, 文ベクトルを SWE の平均で表しているため, 事前に各単語 $w \in V$ の $\mathrm{SWE}(w)$ を ($\bar{X}$ を引いてから)W で写像することが可能となる ($\because W^\top(\frac{1}{a}\sum_x x - b) = \frac{1}{a}\sum_x W^\top(x - b)$).

また, PCA は次元削減の手法としても活用できる.

その場合, W の第 1 主成分から第 d' 主成分までの値を用いて $d'(< d)$ 次元に削減するのが一般的であるが, 本研究では第 k 主成分から第 $k+d'$ 主成分の値を用いて次元削減を行う. これは, 最初の第 k 主成分を取り除くことにより SWE の精度が良くなることが文献 [10] で示されているためで, 実際に本研究でも同様の結果となった. 文献 [10] に倣い, $k = \lfloor \frac{d}{100} \rfloor$ とした. ところで, SWE の次元削減は SWE の語彙行列 $Y \in \mathbb{R}^{|V| \times d}$ に PCA を適用するのが一般的であるが, 上記の手法よりも文埋め込みの精度が悪化した. これは, 文埋め込みでは単語より文の意味の分散を捉える方が重要であることに起因すると考えられる.

2.3 知識蒸留

最後に, 次元削減した SWE を知識蒸留 (Knowledge Distillation)[11] でさらに改善する. 知識蒸留の教師モデルは節 2.1 で SWE の抽出に用いた ST, 生徒モデルは SWE の平均で文を埋め込むモデルとする. 任意の 2 つの文が与えられた時, それらのコサイン類似度をまず教師モデルで計算し, その値を生徒モデルで再現できるように SWE を fine-tune する. 具体的には, $K(= 128)$ 文から成るミニバッチに対して以下の loss を下げるように学習する:

$$\text{loss} = -\frac{1}{K}\sum_{i=1}^{K}\sum_{i\neq j,j=1}^{K} \text{q}(t_{i,j})\log(\text{p}(s_{i,j})),$$

$$\text{p}(s_{i,j}) = \frac{e^{\frac{s_{i,j}}{\tau}}}{\sum_{k\neq i,k=1}^{K} e^{\frac{s_{i,k}}{\tau}}},\ \text{q}(t_{i,j}) = \frac{e^{\frac{t_{i,j}}{\tau}}}{\sum_{k\neq i,k=1}^{K} e^{\frac{t_{i,k}}{\tau}}}.$$

ここで, $t_{i,j}$ と $s_{i,j}$ はそれぞれ, 教師モデル及び生徒モデルで計算したミニバッチ内の i 番目と j 番目の文ベクトルのコサイン類似度である. なお, ミニバッチ内の文は節 2.1 で用いた文からランダムに抽出する. τ は類似度行列の尖度をコントロールするハイパーパラメータで, 本研究では $\tau = 0.05$ とした. 教師モデルのパラメーターを固定して生徒モデルのみを fine-tune することで, SWE をさらに文埋め込みに最適化する. なお, 文献 [12] はこれと同様の手法を用いて ST から BERT のような他の Transformer モデルに知識蒸留を行い, その有効性を示している.

なお, 節 2.1 の SWE の抽出と本節の知識蒸留を行う際, **ST にトークン数の多いテキストを入力すると, 精度が悪化した**. これは, 入力トークン数が多くなるにつれて (1) ST の最終層で入力単語 w 以外の情報を (self-attention により) 多く含むようになり, (2) 教師と生徒モデルの考慮できる情報 (=語順) の差が広がって知識蒸留が困難になる, ということに起因すると考えられる. 文の抽出に用いた CC-100 は 1 行に長い文章を含むことが多々あるため, それらをまず文に分割し,[1] $w \in V$ を含む文を最大 2,000 文ずつ抽出し, そのうちトークン数が最も少ない N 文を SWE の抽出, 主成分分析, 知識蒸留に用いた.

2.4 単語ベクトルのアンサンブル

機械学習の様々なタスクにおいて, 複数モデルの出力をアンサンブルすることの有効性が示されている. そこで, 本研究は L 個の SWE $(\text{SWE}_1, \ldots, \text{SWE}_L)$ をアンサンブルし, テキスト z を以下のように埋め込むことを提案する:

$$\text{E}(z) = \frac{1}{\sqrt{\sum_{i=1}^{L} a_i^2}}[a_1\text{f}(\text{E}_1(z)), \ldots, a_L\text{f}(\text{E}_L(z))], \quad (1)$$

$$\text{E}_i(z) = \sum_{w'\in z} \text{SWE}_i(w'). \quad (2)$$

ここで, f はベクトルのノルムを 1 に正規化する関数, $a_i > 0$ は各モデルの重みに対応するスカラー値, そして $[x_1, \ldots, x_L]$ は $x_1, \ldots, x_L$ を結合したベクトルを表す. $\text{E}(z)$ で 2 つの文の内積を取ると, $\text{SWE}_1, \ldots, \text{SWE}_L$ でそれぞれ計算したコサイン類似度を a_i^2 で重み付けした加重平均となり, アンサンブルとなる. 本研究では簡単のため $a_1 = \cdots = a_L = 1$ で実験を行う. なお, ST のアンサンブルとは異なり, **本手法は文埋め込みの計算コストをあまり増やさずに適用できる**という利点がある. なぜなら, 各単語 $w \in V$ について $\text{SWE}(w) = [\text{SWE}_1(w), \ldots, \text{SWE}_L(w)]$ と事前に結合しておけば, $\text{E}(z)$ は $\text{SWE}(w)$ の和を取った後に各モデルの次元でそれぞれ正規化することで, 簡単に求められるからである.[2]

3 実験

3.1 モデル

本提案手法の学習に用いる ST として GTE-base[3] [13] を選んだ. このモデルは様々なラベル付きデータを用いて BERT を fine-tune したもので, ベクトルの次元数が 768 と比較的軽量ながら Massive Text Embedding Benchmark (MTEB) [6] と呼ばれる文ベクトル評価データにおいて高い精度を達成している. まず GTE から SWE を節 2.1 の手法で抽出し, 節

1) https://github.com/microsoft/BlingFire を用いた
2) ただし, 類似度計算のコストやメモリー消費量は増加する.
3) https://huggingface.co/Alibaba-NLP/gte-base-en-v1.5

2.2 で PCA を用いて次元数を $d' = \{256, 512\}$ まで削減した. 節 2.3 では学習率 0.001 の Adam[14] を用いて最大 3 万ステップ学習し, 検証データの loss が下がらなくなれば学習を止めた. 学習と検証データは節 2.1 で用いた文からランダムに抽出した.

提案手法と直接比較可能な SWE のベースラインとして GloVe [15] と Sent2Vec [16] を選んだ. 両モデルともラベルなしのコーパスを用いて単語の共起情報を元に学習されたモデルであるが, Sent2Vec は名前の通り文埋め込みに有効なモデルとなっている. また, 提案手法と同様に ST を用いて学習した SWE が最近 2 つの異なる GitHub プロジェクトで公開されたため (WordLlama[4] [17], Model2Vec[5] [18]), これらもベースラインに含め評価した. ただし, **これらのモデルには論文や詳細なレポートが存在しない**ため, 実験設定や学習の細部を把握するのが困難となっている. WordLlama は LLaMa 3[2] の入力層で使われる SWE を抽出し, それを様々なラベル付きデータを用いて改善するモデル, Model2Vec は, BGE[6] [19] に $w \in V$ を 1 単語ずつ (文脈なしで) 入力して出力されるベクトルを SWE(w) とみなし, 更に, 任意の文を BGE に入力して出力される文ベクトルと SWE の平均で得られる文ベクトルの二乗誤差を最小化する学習を行い SWE(w) を改善する. Model2Vec はその後に SWE の語彙行列に対して PCA を適用して次元削減を行い, 最後に高 (低) 頻度語のノルムを小さく (大きく) するという古典的な heuristic を用いている.

3.2 データと評価

文埋め込みの評価には, Massive Text Embedding Benchmark (MTEB)[6] を用いた. このデータは 7 個のタスク (classification, pair classification, clustering, reranking, retrieval, semantic textual similarity (STS), summarisation evaluation) を網羅する 56 個の英語のデータセットで構成されている. 評価するモデルに各データセットのテキストを入力し, 得られたベクトルをテキスト間の類似度計算や分類器の特徴量として用いる. 分類器の学習等は MTEB のライブラリー内部で自動で行われる. そして, それぞれのタスク・指標で精度を算出し, 全データセットでの平均精度 (**Avg**) でモデルの評価を行う. 更に, 本研究では全 56 データのうち Sentence-to-Sentence (S2S) とタグ付けされた, 入力テキストが比較的短い 33 個のデータセットでの平均精度 (**Avg-s2s**) も比較する. これは, 1,000 単語を超えるような長い文章を SWE で表すのは非常に困難で, あまり現実的でないためである.

3.3 結果

実験結果を表 1 に示す. 最初の 3 モデルは ST で, それ以降は全て SWE である. SimCSE[7] [7] は BERT ベースの代表的な ST, GTE-base は提案手法の教師モデル, そして LLM2Vec[4] は LLaMa 3 (8B) を fine-tune した ST である. 表が示すように, SWE の中では提案手法 (OURS) が Avg 及び Avg-s2s において最も精度が高くなった. 加えて, OURS が SimCSE の精度を上回ったのは驚きの結果である. ただし, 他の強力な ST に対しては Avg で大きく下回っており, 特に Retrieval のタスクで大きく差をつけられている. これは, Retrieval では単語数の多い文書が入力となることが多く, 語順を考慮できない SWE にとって不利な条件だからである. 事実, Avg-s2s では ST との差が小さくなっており, 入力テキストが短い場合においては SWE が有効であることが確かめられた.

最後の 2 行は, OURS (256) とベースラインを節 2.4 の手法でアンサンブルした結果である. OURS 単体と比較して, 特に Reranking と Retrieval で精度が改善した. これらのタスクでは WordLlama が最も精度が高い SWE なため, アンサンブルにより OURS の弱点を補うことができたと考えられる. 一方, WordLlama が苦手な Classification や STS では精度がやや悪化したため, 式 1 の重み a_i を各モデルの強みに応じて調整することで, 更なる精度改善が期待できる.

次に, STS データセットの 1 つである STS15 での精度と実行速度を表 2 で比較した. 最初の 2 つの ST と比べて提案手法は精度でやや劣るものの, CPU 上でも非常に高速に文を埋め込むことができる.

4 分析

節 2.1, 2.2, 2.3 で得られる SWE の精度 (Avg) はそれぞれ *44.1, 49.7, 51.9* であり ($d' = 256$), PCA と知識蒸留で精度が改善する結果となった. また, 節 2.2 で述べた通り, 第 1 から第 d' 主成分までを用いて次元削減すると節 2.2 での精度が *49.7* から *48.7* に悪化し, PCA を語彙行列に適用すると *48.1* に悪化した.

各段階で SWE がどう変化しているかを示すために, 各単語のノルムの変化に注目して分析を行った. 以下, 節 2.1, 2.2, 2.3 における単語 w のノルム

4) https://github.com/dleemiller/WordLlama
5) https://github.com/MinishLab/model2vec
6) https://huggingface.co/BAAI/bge-base-en-v1.5
7) princeton-nlp/sup-simcse-bert-base-uncased

Models (d)	# Data Sets	Class. 12	Clust. 11	PairClass. 3	Rerank. 4	Retr. 15	STS 10	Summ. 1	Avg-s2s 33	Avg 56
Transformers										
SimCSE-supervised (768)		67.32	33.43	74.90	47.54	21.82	79.12	**31.17**	62.86	48.93
GTE-base (768)		**77.17**	**46.82**	85.33	57.66	54.09	81.97	**31.17**	71.51	64.11
LLM2Vec-LLaMa3 (4,096)		75.92	46.45	**87.80**	**59.68**	**56.63**	**83.58**	30.94	**72.94**	**65.01**
Static Word Embeddings										
GloVe (300)		57.29	27.73	70.92	43.29	21.62	61.85	28.87	52.63	41.97
Sent2vec (700)		61.16	31.75	71.88	47.31	27.79	65.80	29.51	56.33	46.29
WordLlama (256) [WL]		60.06	35.87	73.59	**52.68**	32.41	72.25	30.12	59.99	49.74
Model2Vec (256) [MV]		64.67	32.94	76.62	49.72	30.43	73.24	29.20	60.67	49.74
OURS (256) [**OURS**$_s$]		67.21	36.76	78.72	50.86	31.03	**75.88**	30.04	63.76	51.87
OURS (512)		**68.07**	36.17	**79.14**	50.90	31.50	75.65	29.82	63.82	52.04
OURS$_s$ + WL (512)		66.23	**37.25**	**79.14**	52.61	33.36	75.61	30.22	63.82	52.48
OURS$_s$ + WL + MV (768)		67.12	37.17	79.03	52.09	**33.97**	75.28	**30.27**	**63.86**	**52.72**

表 1 MTEB での実験結果. 括弧内の数字は各モデルのベクトルの次元数 (d) を表す.

Models (d)	ρ	実行速度 (秒)	
		GPU	CPU
LLM2Vec (4,096)	0.88	30.3	>10,000
GTE-base (768)	0.87	3.4	52.7
OURS (256)	0.83	–	0.5

表 2 各モデルの STS15 での精度 (スピアマンの相関係数 ρ) 及び実行速度. 速度は NVIDIA A100-SXM4-40GB で GPU あり/なしで 3 回独立に実行し, 平均を取った.

を $n_1(w), n_2(w), n_3(w)$ と記す. また, w を品詞情報がタグ付けされた Brown Corpus[20] で使用頻度が最も高い 1 万語に限定する. まず, $n_2(w)/n_1(w)$ の下位 100 単語に多く現れる品詞は動詞 (23), 前置詞 (19), 人称代名詞 (8), be 動詞 (7) と続き,[8] 下位 3 単語は *of, to, an* であった. 一方, 上位 100 単語の品詞は名詞 (71), 固有名詞 (19), 動詞 (5) の順で, 上位 3 単語は *railroads, orioles, churchyard* だった. つまり, PCA によって**前置詞や冠詞などの文の意味にあまり影響しない単語のノルムが小さくなり, 反対に名詞のノルムが大きくなる**ことを表している. なお, 文献 [21] は ST にも名詞の情報を多く埋め込むバイアスがあることを報告している. また, $n_3(w)/n_2(w)$ の下位 100 単語の品詞は副詞 (16), 人称代名詞 (14), 名詞 (13) の順で, 下位 3 単語は *they, we, it* であった. 知識蒸留では, たとえ名詞であっても情報量をあまり持たない単語のノルムが小さくなるように調整されていると考えられる. 一方, 上位 100 単語の品詞は順に動詞 (25), 名詞 (24), 形容詞 (18) と様々で, 上位 3 単語は *which, unable, provide* であった.*which* や *unable* は高頻度語であるものの, 入力文に疑問や否定が含まれるかどうかの手がかりになる場合があるので, ノルムが大きく調整された可能性がある. 最後に、単語の出現頻度と $n_1(w), n_2(w), n_3(w)$ のスピアマンの順位相関を見ると順に –0.34, –0.36, –0.37 で, 高頻度語のノルムが徐々に小さくなることが分かった.

8) 品詞が複数ある単語は一番頻度が高い品詞を参照した.

5 関連研究

文献 [22, 23, 24] は, 節 2.1 と同様の手法を用いて BERT 等の言語モデルから SWE を抽出し, それを単語アナロジー (word analogy) のような単語レベルのタスクで評価し有効性を示した. 文埋め込みモデルの研究は盛んに行われているものの, 近年は計算コストを度外視した LLaMa3[2] や Mistral[25] 等の LLM ベースの手法が注目を集めている. 一方, 計算コストの低い SWE ベースの手法は少なく, 古典的なモデルでは Sent2Vec[16], 最近では WordLlama[17] や Model2Vec[18] が GitHub 上で公開され話題となった.

6 結論

本研究は, 文埋め込みに有効な静的単語ベクトル (SWE) を提案した. Transformer の文埋め込みモデルから SWE を抽出し, 主成分分析と知識蒸留により精度を高めた. 文埋め込みの代表的な評価データ MTEB で, 既存の SWE より精度が高いことを示した.

参考文献

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] AI@Meta. Llama 3 model card. 2024.

[3] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)**, pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics.

[4] Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. LLM2vec: Large language models are secretly powerful text encoders. In **First Conference on Language Modeling**, 2024.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In **Advances in Neural Information Processing Systems 30**, pp. 5998–6008. Curran Associates, Inc., 2017.

[6] Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In **Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics**, pp. 2014–2037, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[7] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In **Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing**, pp. 6894–6910, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[8] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In **Proceedings of the Twelfth Language Resources and Evaluation Conference**, pp. 4003–4012, Marseille, France, May 2020. European Language Resources Association.

[9] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 8440–8451, Online, July 2020. Association for Computational Linguistics.

[10] Jiaqi Mu and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. In **International Conference on Learning Representations**, 2018.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In **NIPS Deep Learning and Representation Learning Workshop**, 2015.

[12] Jiahao Xu, Wei Shao, Lihui Chen, and Lemao Liu. DistillCSE: Distilled contrastive learning for sentence embeddings. In **Findings of the Association for Computational Linguistics: EMNLP 2023**, pp. 8153–8165, Singapore, December 2023. Association for Computational Linguistics.

[13] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning, 2023.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In **Proceedings of the 3rd International Conference on Learning Representations**, May 2015.

[15] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**, pp. 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[16] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)**, pp. 528–540, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[17] D. Lee Miller. Wordllama: Recycled token embeddings from large language models, 2024.

[18] Model2vec: The fastest state-of-the-art static embeddings in the world, 2024.

[19] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

[20] H. Kucera and W. N. Francis. **Computational analysis of present-day American English**. Brown University Press, 1967.

[21] Dmitry Nikolaev and Sebastian Padó. Representation biases in sentence transformers. In **Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics**, pp. 3701–3716, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[22] Kawin Ethayarajh. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)**, pp. 55–65, Hong Kong, China, November 2019. Association for Computational Linguistics.

[23] Rishi Bommasani, Kelly Davis, and Claire Cardie. Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings. In **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 4758–4781, Online, July 2020. Association for Computational Linguistics.

[24] Takashi Wada, Timothy Baldwin, Yuji Matsumoto, and Jey Han Lau. Unsupervised lexical substitution with decontextualised embeddings. In **Proceedings of the 29th International Conference on Computational Linguistics**, pp. 4172–4185, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics.

[25] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.