サブワードに基づく単語分散表現の縮約モデリング

佐々木翔大 [1]　鈴木潤 [1,2]　乾健太郎 [1,2]
[1] 東北大学　[2] 理化学研究所 AIP センター
{sasaki.shota, jun.suzuki, inui}@ecei.tohoku.ac.jp

1 はじめに

Common Crawl (CC) コーパス[*1]のような大規模なテキストデータ上で学習された単語分散表現は，有用で基礎的な言語資源である．大規模で高品質な単語分散表現の典型的な例として，6 兆トークンで構成される CC コーパス上で fastText [4] を用いて学習された `fastText.600B`[*2]や，8.4 兆トークンで構成される CC コーパス上で GloVe [16] を用いて学習された `GloVe.840B`[*3]が挙げられる．実際に，固有表現抽出，構文解析，談話構造解析などの多くの自然言語処理タスクで，これらの単語分散表現を活用することで高いパフォーマンスを達成したことが報告されている [6–8, 21, 23]．また，昨今注目を集めている ELMo [17] などの深層ニューラル言語モデルも `GloVe.840B` を利用することで性能向上を達成しており，事前学習済みの単語分散表現の重要性は依然として高い．

しかしながら，事前学習済みの大規模な単語分散表現には利用者の観点からはいくつか解決したい改善課題が存在する．本研究では，i) 現状広く用いられている学習済み単語分散表現のモデルサイズが大きいため，実行時の必要記憶領域量（以下必要メモリ量と表記する）が比較的大きくなる点，ii) 語彙に含まれない単語（**未知語**）へ対応能力に欠ける点を，改善課題として取り上げる．これらの課題は特に実世界のオープンシステムへの応用を考えた時に，重大な要件となる．具体的には，語彙サイズ 200 万の `fastText.600B` を利用することを考えた時，全ての単語分散表現をシステムが保持するためには約 2 GB のメモリ量が必要となる．これは記憶領域が限られた計算環境においては許容し難いほど大きい．必要メモリ量を低減させる策として，頻度情報などを元に一部の単語を語彙から除外するという単純な方法が考えられる．しかし，このような方法は，語彙に含まれない単語に対応できなくなる **OOV 問題**の悪影響を増大させうる．

現在，単語を**サブワード**の組み合わせによって表すことで，OOV 問題を大幅に緩和する手法 [3, 4, 18] が注目を集めている．Bojanowski ら [4] は単語分散表現を学習する際に，文字 N-gram の情報を活用する手法 `fastText` を提案した．また，Zhao ら [3] は，事前学習済みの単語分散表現をサブワード（文字 N-gram）分散表現を用いて再構築する手法 `BoS` を提案した．Pinter ら [18] は，サブワードとして文字ユニグラム分散表現のみを用いて単語分散表現の再構築を行う手法 `MIMICK` を提案した．ただし，`MIMICK` はサブワード分散表現の混合関数として LSTM [10] を用いている．

本研究では既存のサブワードに基づくアプローチを拡張することで，モデルサイズ（保持するベクトルの総数）の削減と OOV 問題への対処を同時に行う．主なアイディアは（1）メモリ共有手法と（2）自己注意機構 [22] を応用した演算（以降，KVQ 演算と呼ぶ）の組み合わせである．実験では，メモリ共有と KVQ 演算を組み合わせた手法が元の単語分散表現の性能低減を 2 ∼ 8% に抑えながらモデルサイズを 1/4 に削減し，未知語分散表現予測の評価において従来法を上回る性能を達成したことを報告する．

2 サブワードに基づく単語分散表現の再構築

2.1 定式化

本節では，本研究で対象とするサブワードに基づく単語分散表現の再構築を最適化問題として定式化する．$\mathcal{W}$ を単語の語彙，$\zeta(\cdot)$ を単語からその ID へのマップ関数とする．また $\boldsymbol{e}_w$ を単語 $w \in \mathcal{W}$ の D 次元の分散表現ベクトル，$\boldsymbol{E}$ を単語分散表現行列とする．同様に $\mathcal{S}$ を $\mathcal{W}$ に含まれる単語から得られるサブワードの語彙，$\eta_v(\cdot)$ をサブワードからその ID へのマップ関数，$\boldsymbol{v}_s$ をサブワード $s \in \mathcal{S}$ の D 次元の分散表現ベクトル，$\boldsymbol{V}$ をサブワード分散表現行列とすると，以下の関係が成り立つ．

$$\boldsymbol{e}_w = \boldsymbol{E}[z_e] \quad \text{ただし} \quad z_e = \zeta(w). \tag{1}$$

$$\boldsymbol{v}_s = \boldsymbol{V}[z_v] \quad \text{ただし} \quad z_v = \eta_v(s). \tag{2}$$

混合関数$\tau(\cdot)$: ある単語 w より得られるサブワードから w の分散表現の代替を計算する混合関数 $\tau(\cdot)$ としてサブワード分散表現の和が広く用いられている．

$$\tau_{\texttt{sum}}(\boldsymbol{V}, w) = \sum_{s \in \phi(w)} \boldsymbol{v}_s. \tag{3}$$

ここで $\phi(\cdot)$ はある単語から得られるサブワードを返す関数である．

損失関数$\Psi(\cdot)$: 損失関数の選択肢には様々な関数が考えられるが，本研究では次の二乗誤差関数を用いる．

$$\Psi(\boldsymbol{E}, \boldsymbol{V}, \tau) = \sum_{w \in \mathcal{W}} C_w \left\| \boldsymbol{e}_w - \hat{\boldsymbol{v}}_w \right\|_2^2. \tag{4}$$

[*1] http://commoncrawl.org
[*2] https://fasttext.cc/docs/en/english-vectors.html
[*3] https://nlp.stanford.edu/projects/glove/

表1: 各設定における保持する分散表現ベクトル数と必要メモリ量の統計情報: M は百万を表す．必要メモリ量は各実数を保持するのに必要なメモリ量を 4 バイトとして計算した．

ID	設定	ベクトル数	必要メモリ量 (GB)
(a)	`fastText.600B`	2.0 M	2.2 GB
(b)	文字 N-gram $N=1,2,3$	0.2 M	0.3 GB
(c)	$N=3,4,5,6$	6.2 M	7.1 GB
(d)	$N=1$ to 6	6.3 M	7.2 GB
(e)	$N=1$ to ∞	21.8 M	24.9 GB

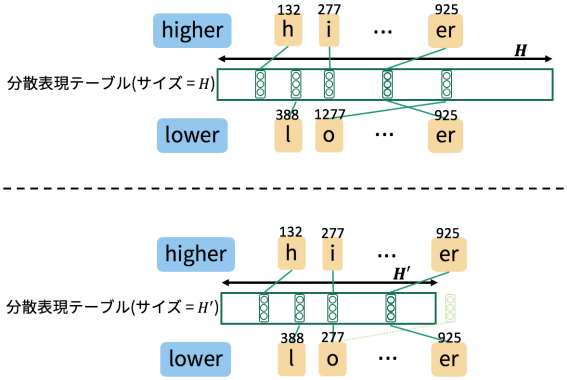

図1: ハッシュを用いたメモリ共有．$H' < H$ とする．

ここで C_w は重み係数である[*4]．ただし $\hat{\boldsymbol{v}}_w = \tau(\boldsymbol{V}, w)$ と置く．

このとき，$\boldsymbol{V}$ と $\tau(\cdot)$ を用いて $\boldsymbol{E}$ を再構築する問題を以下の最小化問題で表す．

$$\hat{\boldsymbol{V}} = \arg\min_{\boldsymbol{V}} \{\Psi(\boldsymbol{E}, \boldsymbol{V}, \tau)\}. \quad (5)$$

2.2 課題

本研究で議論の対象として取り上げている分散表現のベクトル数と必要メモリ量について議論する．表1にサブワードの構成方法の違いによる，生成されるサブワードの数とそこから算出される必要メモリ量を示す．まず，表1 (a) 行で示すように，`fastText.600B` の分散表現ベクトルは 300 次元，200 万単語からなり，必要メモリ量は 2.2 ギガバイト（GB）となる．もし全ての文字 N-gram をサブワードとし，各サブワードが分散表現ベクトルを持つとすると，(e) 行で示すように，必要メモリ量は 25GB と非常に大きくなる．

現実的には，(b)$N=1\sim3$ や (c)$N=1\sim6$ のように，より小さな範囲の N-gram を利用することが考えられる．しかしながら，(b) のような設定では元の単語分散表現の性能を大幅に劣化させてしまう可能性が高い．よって，必要メモリ量（保持する分散表現ベクトルの数）と性能のバランスが良い設定を探索する必要がある．

3 提案手法

モデルサイズの削減と高い性能を同時に達成することを目的に，2.1節で述べた学習の改良手法を提案する．改良手法は（1）マップ関数 $\eta_v(\cdot)$ の変更，（2）混合関数 $\tau(\cdot)$ の変更の大きく分けて 2 通りである．

[*4]本研究では従来法と同様に，$C_w = log(n_w)$ とする．ただし n_w は単語 w のコーパスにおける頻度とする．

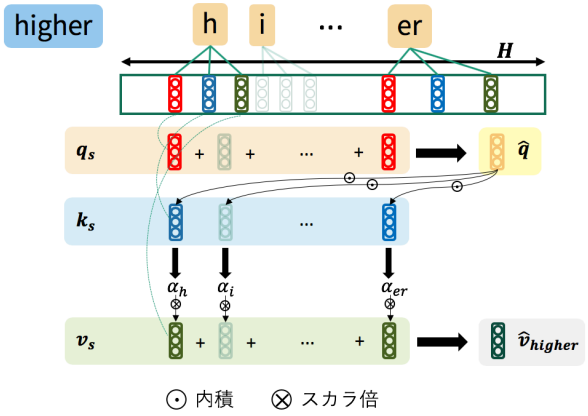

図2: KVQ 演算の概略図．

3.1 $\eta_v(\cdot)$ の変更

3.1.1 高頻度サブワード

$\mathcal{S}$ 中の全てのサブワードを利用する代わりに，語彙 $\mathcal{W}$ 内での頻度上位 F 件のサブワードのみを利用する手法が考えられる．頻度上位 F 件のサブワードの集合を $\mathcal{S}_F \subseteq \mathcal{S}$ とすると，新たなマップ関数 $\eta_{v,F}(\cdot)$ を以下のように定義する．

$$\eta_{v,F}(\cdot) : \mathcal{S}_F \to \mathcal{I}_F \text{ ただし } \mathcal{I}_F = \{1, \ldots, |\mathcal{S}_F|\}. \quad (6)$$

3.1.2 メモリ共有

2.1節で説明した $\eta_v(\cdot)$ は全単射であるが，ここではその代替として全射である新たなマップ関数 $\eta_{v,H}(\cdot)$ を以下のように定義する．

$$\eta_{v,H}(\cdot) : \mathcal{S} \to \mathcal{I}_H \text{ ただし } \mathcal{I}_H = \{1, \ldots, H\}. \quad (7)$$

ここで H はハイパーパラメータで，$H < |\mathcal{S}|$ とする．このマップ関数 $\eta_{v,H}(\cdot)$ は各サブワードから対応する ID にマップするが，各 ID は 1 つのサブワードに固有ではなく，複数のサブワードによって共有されうる．ゆえに複数のサブワードによって 1 つの分散表現ベクトルを共有することで，保持するベクトル数を H に削減することができる．実際には $\eta_{v,H}(\cdot)$ に Fowler-Noll-Vo ハッシュ関数[*5]を用いた．これはサブワード分散表現が不規則に選ばれた複数のサブワードによって共有されることを意味する．図1にメモリ共有手法の概要を示す．

3.1.3 高頻度サブワードとメモリ共有の組み合わせ

また $\eta_{v,F}(\cdot)$ と $\eta_{v,H}(\cdot)$ の組み合わせである $\eta_{v,F,H}(\cdot)$ も考えられる．

$$\eta_{v,F,H}(\cdot) : \mathcal{S}_F \to \mathcal{I}_H \text{ ただし } \mathcal{I}_H = \{1, \ldots, H\}. \quad (8)$$

はじめにサブワードの集合 $\mathcal{S}$ を頻度上位 F 件の $\mathcal{S}_F$ に絞り込んだ上で，メモリ共有手法を適用する手法である．

3.2 $\tau(\cdot)$ の変更

既存研究においては，混合関数 $\tau(\cdot)$ として和が用いられている（式3）．しかしながら，3.1.2節で述べたメモリ共有の設定に置いては表現力に欠ける可能性がある．ここではその対処として文脈依存の重み係数を導入した次

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

表2: 実験に用いた評価データセット．OOV の行は `fastText.600B` を用いた際の未知語を含む評価インスタンス数を表す．

	サイズ	OOV
単語類似度判定タスク		
MEN [5]	3,000	0
M&C [15]	30	0
MTurk [19]	287	0
RW [13]	2,034	37
R&G [20]	65	0
SCWS [11]	2,003	2

	サイズ	OOV
SLex [9]	998	0
WSR [1]	252	0
WSS [1]	203	0
単語アナロジータスク		
GL [2]	19,544	0
MSYN [14]	8,000	1000

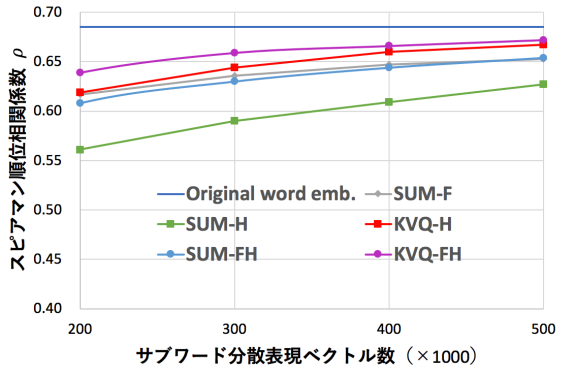

図3: 単語類似度判定タスクにおけるモデルサイズと性能の関係．x 軸と y 軸はそれぞれサブワード分散表現ベクトルの数，スピアマン順位相関係数 ρ を表す．

の混合関数 $\tau_{\mathrm{kvq}}(\cdot)$ を定義する．

$$\tau_{\mathrm{kvq}}(\boldsymbol{V}, w) = \sum_{s \in \phi(w)} a_{s,w} \boldsymbol{v}_s. \tag{9}$$

ここで $a_{s,w}$ はサブワード s の文脈依存重み係数である．この場合の‘文脈’は単語 w から得られる全てのサブワードを意味する．$a_{s,w}$ の計算のために，$\boldsymbol{v}_s$（式2）と同様に $\boldsymbol{k}_s$ と $\boldsymbol{q}_s$ を定義する．

$$\boldsymbol{k}_s = \boldsymbol{V}[z_k] \quad \text{ただし} \quad z_k = \eta_k(s). \tag{10}$$

$$\boldsymbol{q}_s = \boldsymbol{V}[z_q] \quad \text{ただし} \quad z_q = \eta_q(s). \tag{11}$$

ここで $\eta_k(\cdot)$，$\eta_q(\cdot)$ は $\eta_v(\cdot)$ と同様のマップ関数である．これらを用いて，Key-value-query（KVQ）演算を以下のように定義する．

$$a_{s,w} = \frac{\exp(Z\hat{\boldsymbol{q}} \cdot \boldsymbol{k}_s)}{\sum_{s' \in \phi(w)} \exp(Z\hat{\boldsymbol{q}} \cdot \boldsymbol{k}_{s'})}, \tag{12}$$

ただし，$\hat{\boldsymbol{q}} = \sum_{s \in \phi(w)} \boldsymbol{q}_s$ とする．Z はハイパーパラメータである．この KVQ 演算は，機械翻訳で大幅な性能向上に成功した `Transformer` [22] で用いられている自己注意機構を参考にした．KVQ 演算の概要を図2に示す．

4 実験

本節ではモデル削減の観点，未知語分散表現予測の観点それぞれにおける提案手法の性能評価を行い，有効性を検証する．サブワードとしては文字 N-gram を用いる．以降，混合関数 $\tau(\cdot)$ として和（式3），KVQ 演算（式9）を用いた場合をそれぞれ `SUM`, `KVQ` で表し，$\eta_v(\cdot)$ として3.1節で導入した式6，7，8を用いた場合をそれぞれ `F`，`H`，`FH` で表す．また，各手法をこの組み合わせ（例えば `SUM-FH`）として表す．

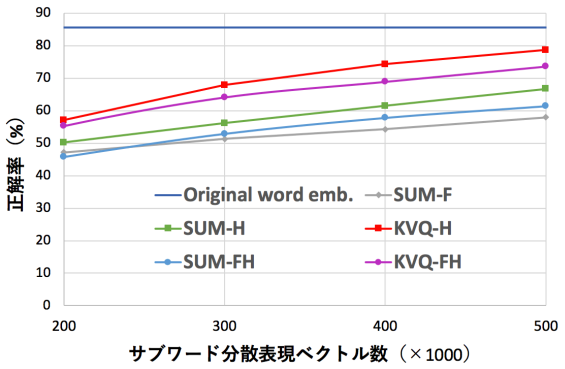

図4: 単語アナロジータスクにおけるモデルサイズと性能の関係．x 軸と y 軸はそれぞれサブワード分散表現ベクトルの数，正解率を表す．

4.1 実験：モデル削減

実験設定: 評価データとして，9 つの単語類似度判定タスクと 2 つの単語アナロジータスク（表2）を用い，それぞれスピアマン順位相関係数 ρ，正解率（%）で性能評価する．本節の評価においては，評価インスタンス中に 1 単語でも未知語が存在した場合，そのインスタンスを評価データから除外する．この設定は既存研究でも広く用いられている標準的な設定であることに注意されたい．再構築のターゲットとなる事前学習済みの単語分散表現 $\boldsymbol{E}$ として，$D = 300$, $|\mathcal{W}| = 2\mathrm{M}$ である `fastText.600B` を用いた．式9中の Z は，全ての実験において $Z = \sqrt{D}$ を用いた．式5の最適化には Adam [12]（学習率 $\alpha = 0.0001$）を用い，300 epoch 訓練した．

実験結果: 単語類似度判定タスクと単語アナロジータスクにおける性能とモデルサイズの関係を，それぞれ図3，4に示す．図中の点は各データセットにおける性能の平均値を表している．両タスクにおいて，メモリ共有と KVQ 演算を用いた手法が他の手法の性能を上回った．この結果の理由として，メモリ共有手法と KVQ 演算の相性の良さが挙げられる．メモリ共有手法は必要メモリ量を抑える代わりにハッシュ値の衝突が生じる問題があるが，KVQ 演算は各サブワードの重要度に基づいて重み付けすることができ，実質的に使用するサブワード分散表現の選択を行うことができるため，ハッシュ値の衝突による性能低下を軽減できていると考えられる．

また元の単語分散表現の性能と比べた時，KVQ 演算を用いた手法は $H = 0.5\mathrm{M}$ において，性能低減を 2 ～ 8% に抑えていることから，モデルサイズを 1/4 削減することに成功したといえる．

4.2 実験：未知語分散表現の予測

4.2.1 人工未知語実験

実験設定: 評価データとして，4.1節でも用いた 9 つの単語類似度判定タスクを用い，スピアマン順位相関係数 ρ の平均値で性能評価する．表2に示したように，`fastText.600B` の語彙を前提にした時，未知語を含む評価インスタンス数はごく少量となっている．これは `fastText.600B` の語彙サイズが 200 万と非常に大きい

表3: 人工未知語実験の結果.

method	$\|\mathcal{W}\|$	$\|\mathcal{S}\|$	mem. (GB)	ρ
Random	2M	-	2.23GB	.053
SUM-F $F = 0.5$M	2M	0.5M	0.59GB	.603
SUM-H $H = 0.5$M	2M	21.8M	0.59GB	.568
KVQ-H $H = 0.5$M	2M	21.8M	0.59GB	.572
SUM-FH $H = 0.5$M	2M	1.0M	0.59GB	.600
KVQ-FH $H = 0.5$M	2M	1.0M	0.59GB	**.606**
SUM-F $F = 0.2$M	2M	0.2M	0.23GB	.582
SUM-H $H = 0.2$M	2M	21.8M	0.23GB	.515
KVQ-H $H = 0.2$M	2M	21.8M	0.23GB	.536
SUM-FH $H = 0.2$M	2M	1.0M	0.23GB	.571
KVQ-FH $H = 0.2$M	2M	1.0M	0.23GB	**.587**

表4: 従来法との比較実験の結果. * は [3] における報告値を表す.

method	$\|\mathcal{W}\|$	$\|\mathcal{S}\|$	mem. (GB)	ρ
Random	0.16M	-		.452
MIMICK	0.16M	<1K		.201
BoS	0.16M	0.53M	0.62GB	.46*
SUM-F $F = 0.04$M	0.16M	0.04M	0.05GB	.513
SUM-H $H = 0.04$M	0.16M	2.03M	0.05GB	.485
KVQ-H $H = 0.04$M	0.16M	2.03M	0.05GB	.509
SUM-FH $H = 0.04$M	0.16M	0.5M	0.05GB	.488
KVQ-FH $H = 0.04$M	0.16M	0.5M	0.05GB	**.522**
fastText	0.16M	0.53M	0.62GB	.48*

ことが理由である．それゆえ，未知語分散表現の予測性能を直接的に測ることは難しくなっている．

ここでは，訓練時に評価データ中の単語を語彙 $\mathcal{W}$ から除外し人工的に未知語を作ることで，その未知語分散表現の予測性能を測る．これによって全ての評価インスタンスに擬似的な未知語が含まれることになる．他の設定は4.1節の実験と同じ設定を用いる．

実験結果: 人工未知語実験の結果を表3に示す．Random は未知語の分散表現としてランダムベクトルを割り当てるベースライン手法である．Random の性能は $\rho = 0$ に近い値であり，これは Random の類似度スコアと人手類似度スコアに相関がないことを示している．それと比較し，提案手法は $\rho = 0.515 \sim 0.606$ を達成しており，未知語の分散表現予測に成功していることがわかった．また提案手法の間で比較すると，SUM-F，SUM-FH，KVQ-FH が同等で最も良い性能であった．

4.2.2 従来法との比較実験

1節で挙げたように，BoS や MIMICK などの従来法においても OOV 問題の解決に取り組まれてきた．しかしながら，これらの手法（もしくは入手可能な実装[*6][*7]）は fastText.600B のような大きな語彙を持つ分散表現に対してスケールしない．そのため，ここでは正確に Zhao ら [3] の設定に従い，同じ条件において従来法との性能比較を行う．

実験設定: 評価データとして表2中の RW を用い，スピアマン順位相関係数 ρ で性能評価する．再構築のターゲットとなる事前学習済みの単語分散表現 $\boldsymbol{E}$ として，Zyao ら [3] の用いた $D = 300$, $|\mathcal{W}| = 0.16$M の単語分散表現を用いる．

実験結果: 実験結果を表4に示す．Random は未知語の分散表現としてランダムベクトルを割り当てるベースライン手法である．提案手法が，これまで最も性能の良い手法であった BoS の性能を上回った．また，KVQ-FH が最も良い性能を達成した．

5 おわりに

本研究では単語分散表現のサブワードに基づく再構築を通してモデルサイズの削減を行う手法を提案した．実験では，KVQ 演算を用いた手法が元の単語分散表現からの性能低減を 2 ～ 8% に抑えながら，モデルサイズを 1/4 に削減できることを示した．また，未知語分散表現の予測性能に関して，提案手法が従来法の性能を上回ることを示した．

謝辞 本研究は一部は JST CREST（課題番号: JPMJCR1513）の支援を受けて行った．

参考文献

[1] Eneko Agirre et al. “A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches”. In: *Proceedings of NAACL*. 2009.
[2] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *CoRR* (2013).
[3] Zhao et al. “Generalizing Word Embeddings using Bag of Subwords”. In: *Proceedings of EMNLP*. 2018.
[4] Piotr Bojanowski et al. “Enriching Word Vectors with Subword Information”. In: *TACL* (2017).
[5] Elia Bruni et al. “Multimodal Distributional Semantics”. In: *J. Artif. Int. Res.* (2014).
[6] Li Dong et al. “Coarse-to-Fine Decoding for Neural Semantic Parsing”. In: *Proceedings of ACL*. 2018.
[7] Carlos Gómez-Rodríguezz et al. “Constituent Parsing as Sequence Labeling”. In: *Proceedings of EMNLP*. 2018.
[8] Jonas Groschwitz et al. “AMR dependency parsing with a typed semantic algebra”. In: *Proceedings of ACL*. 2018.
[9] Felix Hill et al. “SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation”. In: *arXiv* (2014).
[10] Sepp Hochreiter et al. “Long Short-Term Memory”. In: *Neural Computation* (1997).
[11] Eric H. Huang et al. “Improving Word Representations via Global Context and Multiple Word Prototypes”. In: *Proceedings of ACL*. 2012.
[12] Diederik Kingma et al. “Adam: A method for stochastic optimization”. In: *arXiv* (2014).
[13] Thang Luong et al. “Better Word Representations with Recursive Neural Networks for Morphology”. In: *Proceedings of CoNLL*. 2013.
[14] Tomas Mikolov et al. “Linguistic Regularities in Continuous Space Word Representations”. In: *Proceedings of NAACL*. 2013.
[15] George A. Miller et al. “Contextual Correlates of Semantic Similarity”. In: *Language & Cognitive Processes* (1991).
[16] Jeffrey Pennington et al. “Glove: Global Vectors for Word Represqentation”. In: *Proceedings of EMNLP*. 2014.
[17] Matthew Peters et al. “Deep Contextualized Word Representations”. In: *Proceedings of NAACL*. 2018.
[18] Yuval Pinter et al. “Mimicking Word Embeddings using Subword RNNs”. In: *Proceedings of EMNLP*. 2017.
[19] Kira Radinsky et al. “A Word at a Time: Computing Word Relatedness Using Temporal Semantic Analysis”. In: *Proceedings of WWW*. 2011.
[20] Herbert Rubenstein et al. “Contextual Correlates of Synonymy”. In: *Commun. ACM* (1965).
[21] Jun Suzuki et al. “An Empirical Study of Building a Strong Baseline for Constituency Parsing”. In: *Proceedings of ACL*. 2018.
[22] Ashish Vaswani et al. “Attention is All you Need”. In: *NIPS*. 2017.
[23] Nan Yu et al. “Transition-based Neural RST Parsing with Implicit Syntax Features”. In: *Proceedings of COLING*. 2018.

[*6] https://github.com/jmzhao
[*7] https://github.com/yuvalpinter/Mimick