



Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Кузнецов Михаил Константинович

Векторные представления слов

ЭССЭ

Москва, 2021

Содержание

1 Введение	3
2 Классические способы представления слов	3
2.1 LSA	3
2.2 Вероятность появления слова	4
3 Вложение слов в непрерывное пространство	5
4 Безконтекстные представления слов	6
4.1 Word2vec	6
4.1.1 CBOW	6
4.1.2 Skip-gram	6
4.1.3 Улучшения	6
4.1.4 Свойства эмбеддингов	8
4.2 Fasttext	8
4.3 GloVe	9
4.4 JoSE	10
5 Контекстные представления слов	11
5.1 Tag lm	11
5.2 CoVe	11
5.3 ELMo	12
5.4 FLAIR	13
5.5 Трансформеры	13
5.5.1 BERT	13
5.5.2 ELECTRA	14
6 Заключение	15
7 Список литературы	16

Аннотация

В работе рассматриваются способы представления слов: классические, контекстно-свободные и контекстно-зависимые. Классические чаще всего опираются на вероятностное представление документа, как смесь тем. Контекстно-свободные обучаются без учителя на большом корпусе текстов и результатом работы является эмбеддинг, не зависящий от слов, стоящих рядом в новом предложении. В языке одни и те же слова могут иметь разное значение в зависимости от контекста. Контекстно-зависимые представления учитывают этот момент и часто используют контекстно-свободные представления, как начальное состояние для последующего обучения. Целью работы является обзор и сравнение различных способов представления слов.

§1. Введение

Векторные представления слов часто используются в современных моделях для решения задач обработки естественного языка (NLP). Их улучшение способствует не только более высоким показателям качества на задачах типа: определение части речи [1], вопрос—ответ [2], но и более интерпретируемым представлениям [3]. Размер эмбеддингов сильно влияет на портируемость той или иной модели на мобильные устройства. Например, в BERT [4] эмбеддинги составляют $\approx 21\%$ от размера всей модели.

§2. Классические способы представления слов

Самым простым способом кодирования слов является *one-hot-encoding* (ОНЕ) метод: слову сопоставляется вектор длины, равной размеру словаря \mathcal{V} , и содержащий единицу в компоненте, отвечающей за данное слово. Однако большая размерность и отсутствие хорошей близости делают этот метод не столь привлекательным. Если добавить к вектору слова его соседей в нескольких предложениях можно получить более нетривиальную оценку близостей с помощью косинусного расстояния между словами. Такое представление называется *counts*.

2.1. LSA

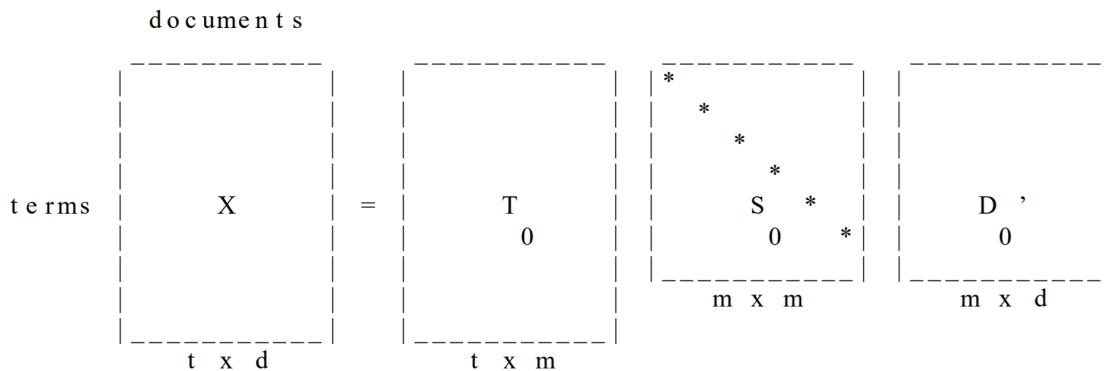


Рис. 1: Способ представления векторов слов в модели LSA. Каждому слову t сопоставляется соответствующая ему строчка из матрицы T_0 . [5]

Возникает естественное желание представить слова в виде вектора с малым количеством компонент, но с недискретным набором значений. В [5] это достигается с помощью разложения матрицы-слово документ X в произведение матриц $T_0 S_0 D_0^T$, где T_0 , D_0 — ортогональные матрицы с единичной нормой столбцов, S_0 — диагональная матрица с сингулярными значениями на диагонали (см. рис. 1).

2.2. Вероятность появления слова

Слово «поздно» в предложении «Лучше —, чем никогда» более вероятно, чем слово «рано». Данную информацию можно закодировать с помощью вероятностей встречаемости n-последовательных слов (n-грамм). Чаще всего в оценке насколько хорошо работает предложенная модель используется *правдоподобие выборки*:

$$\mathbb{P}(d) = \prod_{i=1}^N \mathbb{P}(w_i),$$

где w_i — слово в позиции i , содержащееся в документе d .

В [6] авторы предложили разделить все слова в \mathcal{V} на c классов, то есть каждому слову сопоставляется класс c_i . В случае 2-грамм доказано, что максимизация $\mathbb{P}(w_i|w_{i-1})$ соответствует максимизации взаимной информации $I(c_i, c_{i-1})$.

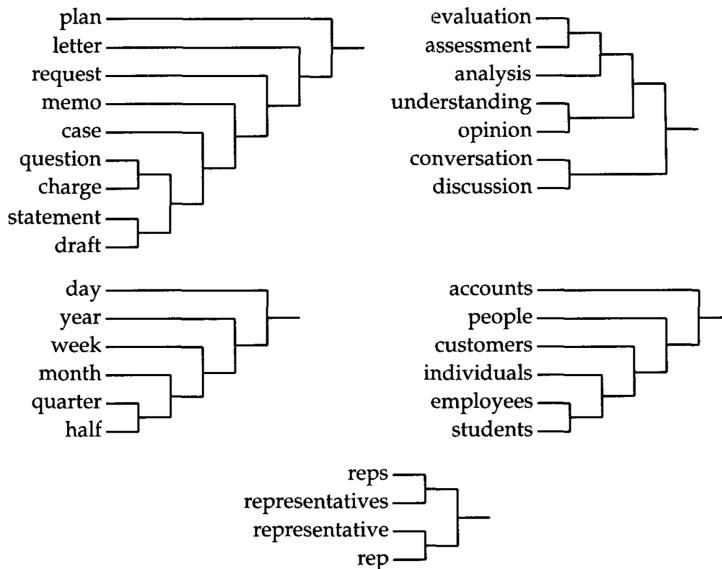


Рис. 2: Кластеризация похожих слов на основе взаимной информации между ними. [6]

Для кластеризации авторы использовали следующий метод:

1. все слова объявляются отдельным кластером,
2. на каждой итерации объединялись два класса, после объединения которых уменьшение взаимной информации будет наименьшим.

На Рис. 4 даны примеры нескольких кластеров, получившихся из 1000 начальных слов. Видно, что в большинстве случаев группы разделены семантически правильно.

Авторы [7] предложили более интересный способ (LDA) задать вероятностную модель на корпусе данных D . Изначальная генерация данных представляется в виде последовательности следующих действий. Для каждого документа d в корпусе D :

1. Сэмплируем $N \sim \text{Poisson}(\xi)$.
2. Сэмплируем $\theta \sim \text{Dir}(\alpha)$.
3. Для каждого слова $w_n, n = \overline{1, N}$:
 - (a) Сэмплируем тему $z_n \sim \text{Multinomial}(\theta)$.
 - (b) Сэмплируем слово w_n из распределения с плотностью $p(w_n | z_n, \beta)$

Вероятность $p(w | \theta, \beta)$ можно представить в виде $p(w | \theta, \beta) = \sum_z p(w | z, \beta)p(z | \theta)$. Из этого представления вытекает еще один способ получения корпуса документов.

Для каждого документа d :

1. Сэмплируем $\theta \sim \text{Dir}(\alpha)$.
2. Для каждого слова $w_n, n = \overline{1, N}$:
 - (a) Сэмплируем слово w_n из распределения с плотностью $p(w_n | \theta, \beta)$.

Если в таком представлении составить матрицу, в которой на (i, j) позиции будут стоять $p(w_i | z_j, \beta)$, то строки будут формировать векторное представление слова w_i . Обучение — максимизация правдоподобия + EM алгоритм.

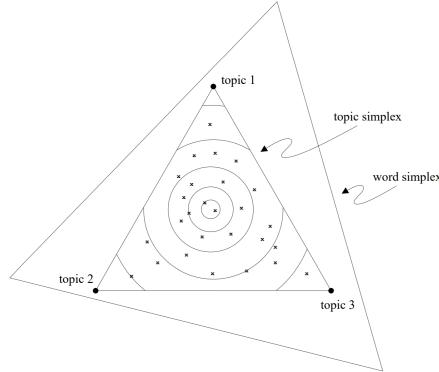


Рис. 3: Здесь нарисован симплекс — треугольник. В углах находится 3 слова. LDA пытается найти сглаженное распределение $p(\mathbf{w} | \mathbf{z})$, обозначенное эллипсами. Заметим, что само распределение $p(\mathbf{z} | \mathbf{d})$ фиксируется на предыдущем шаге. Жирным шрифтом обозначено векторное представление. [7]

§3. Вложение слов в непрерывное пространство

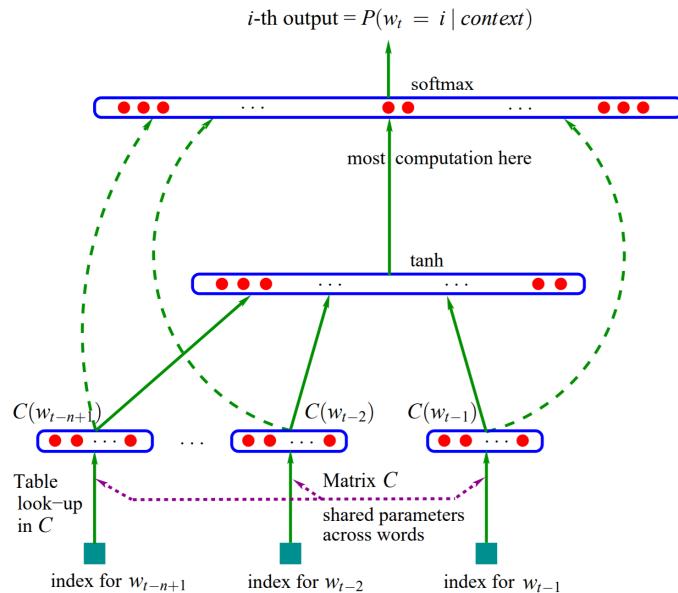


Рис. 4: Устройство работы алгоритма для получения вещественных векторов для слов, так называемых эмбеддингов (embeddings). C — матрица размером $|\mathcal{V}| \times$ размерность_эмбединга. [8]

В работе [8] векторы для слова получается в результате обучения нейросети предсказанию последнего слова из некоторого отрывка текста. Сперва создается матрица C , хранящая

эмбеддинги слов. После получения вещественного представления, эмбеддинги конкатенируются и подаются на вход нейросети, которая выдает вектор вероятностей. Каждая компонента отвечает за то, что слово ей соответствующее будет стоять в конце отрывка текста. В качестве лосса выбирается логарифм правдоподобия с регуляризацией на веса модели (MLP, без bias) и матрицы C . Данный подход показал лучшие результаты по сравнению с SOTA решениями 2003 года: Kneser-Ney back-off, class-based back-off.

§4. Безконтекстные представления слов

4.1. Word2vec

Разумно предположить, что слово в предложении зависит от его контекста и наоборот. В [9] с помощью линейной сети (автоэнкодера) предлагается обучаться задаче предсказания слова по контексту (CBOW) и наоборот (skip-gram). В данном случае размечать данные не нужно, так как их можно получать из больших корпусов внешних текстов.

4.1.1 CBOW

Математически конечную функцию можно записать в виде:

$$p(w_t | \text{context}(w_t)) = \text{softmax} \left(V \left(W \sum_{w_i \in \text{context}(w_t)} \text{OHE}(w_i) \right) \right),$$

где $\text{context}(w_t)$ — слова, стоящие рядом с w_t в некотором окне определенного размера; $\text{OHE}(w_i)$ — вектор с одной единицей в компоненте, отвечающей w_i ; V, W — обучаемые матрицы размерности $|\mathcal{V}| \times k$, $k \times |\mathcal{V}|$ соответственно. Естественно взять i -ый столбец W или i -ую строчку V в качестве эмбеддинга слова w_i . Если усреднить для w_i все скрытые представления $W \sum_{w_j \in \text{context}(w_i)} \text{OHE}(w_j)$ для разнообразного контекста $\text{context}(w_i)$ тоже получится своеобразный эмбеддинг.

4.1.2 Skip-gram

Восстановление контекста по слову более нетривиальная задача, решаемая также с помощью линейной сети:

$$\mathbb{P}(w_i \in \text{context}(w_t) | w_t) = \text{softmax}(VW\text{OHE}(w_t)).$$

На каждой итерации обучения целью является предсказание только одного слова из окружения.

4.1.3 Улучшения

В [10] предложены улучшения касательно качества и скорости обучения сети. В английских текстах более всего распространены артикли и другие не очень влияющие на смысл предложения элементы. Для того, чтобы снизить огромную популярность таких слов и дать малоизвестным словам также влиять на процесс обучения, вероятности просэмплировать слово w_i заменяют на:

$$p(w_i) = \left(\sqrt{\frac{\text{freq}(w_i)}{0.001}} + 1 \right) \cdot \frac{0.001}{\text{freq}(w_i)},$$

где freq — частота появления слова в корпусе.

На этапе обучения skip-gram предсказывается только одно слово. Учитывая, что верен только один ответ из многих, в лосс можно добавлять не все пары (w_t, w_i) , $w_i \in \mathcal{V}$, а только какую-то подвыборку (w_t, w_i) , $w_i \in \mathcal{V}_{\text{sampled}}$, $|\mathcal{V}_{\text{sampled}}| << |\mathcal{V}|$. При этом для сэмплирования слова в $|\mathcal{V}_{\text{sampled}}|$, его частота возведена в степень 3/4. Данный прием называется *negative sampling*.

Большая сложность содержится в вычислении softmax. С помощью *иерархического софтмакса* (Hierarchical Softmax) можно получить те же вероятности за сложность $O(\log_2(|\mathcal{V}|))$. Это достигается построением бинарного дерева, в узле которого содержится вероятность пройти по левому или правому ребенку. В основе формулы, определяющей вероятности $p(w_i | w_j)$ лежат случайные блуждания по дереву. В отличии от классического софтмакса, в каждом узле дерева формируется свой эмбеддинг слова.

Порядок слов имеет значение в предложении и, как следствие, для предсказания маскированного слова тоже. Авторы SSG [11] предложили использовать не один эмбеддинг для слова, а столько какова ширина окна при проходе по тексту. Тогда вероятность

$$p(w_{t+i} | w_t) = \frac{\exp\left(\sum_{r=-c}^c V_{w_{t+i}}^{(r)\top} W_{w_t}\right)}{\sum_{w_{t+i} \in V} \exp\left(\sum_{r=-c}^c V_{w_{t+i}}^{(r)\top} W_{w_t}\right)}, \quad (1)$$

где V_{w_t} — строка, соответствующая слову w_t , W_{w_t} — столбец, соответствующий слову w_t , c — половина ширины окна.

При увеличении размера окна общая размрность эмбеддингов возрастает сильно, т.к. обычно словарь содержит десятки тысяч слов. На помощь приходит DSG [1]. Вместо обучения 2с векторов, обучается 2 вектора: W_{w_t}, Δ_{w_t} . К вероятности (1) добавляется вероятность:

$$g(w_{t+i}, w_t) = \frac{\exp\left(\Delta_{w_{t+i}}^\top W_{w_t}\right)}{\sum_{w_{t+i} \in \mathcal{V}} \exp\left(\Delta_{w_{t+i}}^\top W_{w_t}\right)},$$

которая отвечает за информацию о расположении w_{t+i} относительно w_t . Обновление векторов происходит согласно формулам:

$$\begin{aligned} W_{w_t}^{(new)} &= W_{w_t}^{(old)} - \gamma (\sigma(W_{w_t}^\top \Delta_{w_{t+i}}) - \mathcal{D}) \Delta_{w_{t+i}}, \\ \Delta_{w_{t+i}}^{(new)} &= \Delta_{w_{t+i}}^{(old)} - \gamma (\sigma(W_{w_t}^\top \Delta_{w_{t+i}}) - \mathcal{D}) W_{w_t}, \end{aligned}$$

$$\text{где } \mathcal{D} = \begin{cases} 1, & i < 0, \\ 0, & i > 0. \end{cases}, \quad \sigma \text{ — сигмоида, } \gamma \text{ — learning rate.}$$

В задачах предсказания похожести и части речи слов DSG имеет лучшие результаты (см. рис. 5, 6).

	MEN-3k	SimLex-999	WS-353
CBOW	70.96	34.32	69.25
CWin	74.28	36.06	72.21
SG	71.90	34.35	70.11
SSG	71.26	31.80	69.46
SSSG	72.07	33.62	70.90
DSG	73.76	36.10	72.60

	MEN-3k	SimLex-999	WS-353
CBOW	58.23	26.67	64.40
CWin	59.68	25.19	62.82
SG	60.19	27.14	65.23
SSG	55.42	24.00	61.95
SSSG	62.70	26.55	66.10
DSG	63.18	27.51	66.71

Рис. 5: Результаты работы методов (корреляция Спирмена x100) на задаче похожести слов при обучении на большом (слева) и малом (справа) корпусах. [1]

	WSJ		ARK	
	Dev	Test	Dev	Test
CBOW	96.86	97.01	89.36	88.36
CWin	96.98	97.25	90.03	89.94
SG	96.95	97.12	89.26	88.77
SSG	97.08	97.31	90.05	90.15
SSSG	97.01	97.19	89.83	89.78
DSG	97.16	97.37	90.12	90.43

Рис. 6: Точность на задаче предсказания части речи слов. [1]

4.1.4 Свойства эмбеддингов

Оказывается над полученными эмбеддингами можно применять арифметические операции и получать ожидаемые результаты (см. рис. 7). Кроме того, ближайшие соседи к некоторому слову, например, «peace» (Peaceful, Friendship, Nonviolence, Democracy) имеют схожее значение [12]. Однако если взять слова «black» и «white», то близость будет равна 0.8, что говорит о не очень «хорошем» представлении цветов в пространстве.

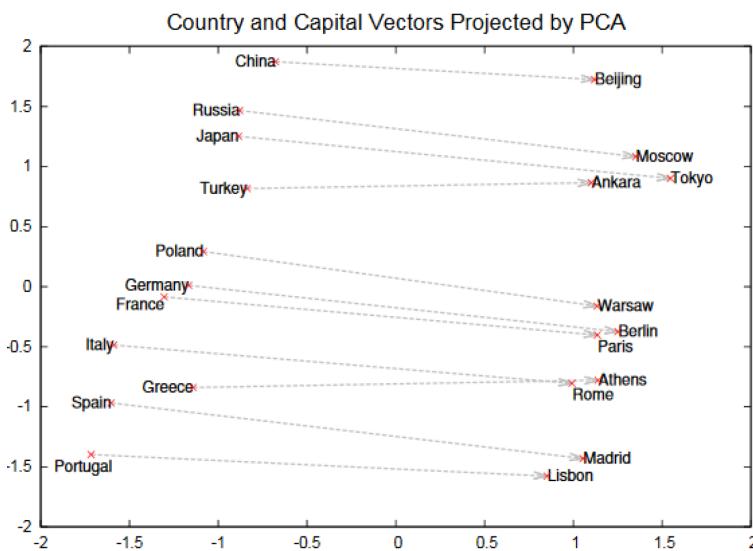


Рис. 7: Представление стран в двух главных компонентах эмбеддингов. Если вычесть из названия столиц стран название стран, то получатся приблизительно параллельные и одинаковые по длине вектора, что говорит о возможности перехода между страной и ее столицей с помощью, например, усредненного вектора. [12]

4.2. Fasttext

В случае появления новых слов в модели word2vec (skip-gram) получить эмбеддинги напрямую не выйдет. Если разбить слово на n-граммы и закодировать каждую из них, как отдельное слово, то удастся частично решить данную проблему. Путем хэширования n-грамм сокращается количество используемой памяти. Данный подход называется fasttext [13].

4.3. GloVe

Word2vec проходится окном по корпусу текстов и не учитывает встречаемость слов со статистической точки зрения. В GloVe [3] данная информация содержится в матрице A размером $|\mathcal{V}| \times |\mathcal{V}|$. Целью является получение таких матриц V, W размерности $|\mathcal{V}| \times k$, $k \times |\mathcal{V}|$ соответственно, что $A \approx VW$. Формально это можно записать следующим образом:

$$\sum_{i,j} f(\#ij) \left(V_{w_j} W_{w_i} + b_{w_i} + \tilde{b}_{w_j} - \log(\#ij) \right)^2 \rightarrow \min,$$

$$f(x) = \begin{cases} \left(\frac{x}{x_{\max}} \right)^\alpha & x < x_{\max}, \\ 1, & x \geq x_{\max}, \end{cases}$$

где $\#ij$ — сколько раз слово j встречается в контексте слова i (на расстоянии $\leq k$ слов). Введение весовой функции $f()$ позволяет избежать слишком сильное влияние часто встречающихся пар слов на обучение. В данном представлении евклидово или косинусное расстояние между двумя словами является хорошим инструментом для поиска не только лингвистически, но и семантически похожих слов.

Среди GloVe, SVD, SG CBOW показал лучшие результаты в задаче предсказания похожих слов (см. рис. 8).

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	<u>72.7</u>	75.1	56.5	37.0
CBOW [†]	6B	57.2	65.6	68.2	57.0	32.5
SG [†]	6B	62.8	65.2	69.7	<u>58.1</u>	37.2
GloVe	6B	<u>65.8</u>	<u>72.7</u>	<u>77.8</u>	53.9	<u>38.1</u>
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	75.9	83.6	82.9	59.6	47.8
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

Рис. 8: Корреляция Спирмена $\times 100$ для задачи предсказания похожих слов. [3]

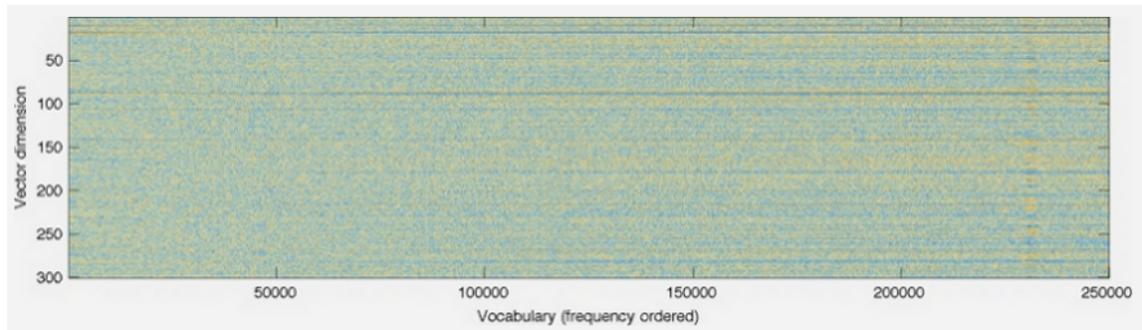


Рис. 9: Здесь изображена обученная с помощью GloVe матрица (частота слова, номер компоненты эмбеддинга слова). Из-за использования в модели скалярного произведения появляются горизонтальные полосы. Вертикальные полосы в районе 230k-233k частоты скорее всего соответствует числам, которые имеют похожие частоты. [14]

4.4. JoSE

Обычно для получения оценки схожести слов или текстов применяют косинусное расстояние к их эмбеддингам. В случае обучения в евклидовом пространстве, найденные представления могут быть не оптимальны с точки зрения конечного косинусного расстояния. В JoSE (Joint Spherical Embedding) [15] авторы обучают эмбеддинги сразу в сферическом пространстве (на единичной сфере), тем самым обходя данную проблему.

Каждое слово w_t имеет два представления: основное и контекстное. Основное представление используется для генерации так называемого *центрального слова* с вероятностью:

$$p(w^{(cent)} | d) \propto \exp(\cos(\mathbf{w}^{(cent)}, \mathbf{d})),$$

где жирным шрифтом выделено векторное представление слова/документа. А *контекстные слова* на основе центрального сэмплируются из распределения:

$$p(w^{(cont)} | w^{(cent)}) \propto \exp(\cos(\mathbf{w}^{(cont)}, \mathbf{w}^{(cent)})).$$

В качестве лосса взята:

$$\begin{aligned} \mathcal{L}(w^{(cont)}, w^{(cent)}, d) &= \max \left(0, m - \log p(w^{(cont)}, w^{(cent)} | d) + \log p(w^{(cent)'}, w^{(cont)} | d) \right) \\ &= \max \left(0, m - \cos(\mathbf{w}^{(cont)}, \mathbf{w}^{(cent)}) - \cos(\mathbf{w}^{(cent)}, \mathbf{d}) \right. \\ &\quad \left. + \cos(\mathbf{w}^{(cont)}, \mathbf{w}^{(cent)'}) + \cos(\mathbf{w}^{(cent)'}, \mathbf{d}) \right), \end{aligned}$$

где m — отступ (> 0), $w^{(cent)'}$ — случайно выбранное из словаря слово. Второе равенство доказано в статье. Также как и в word2vec, используется negative sampling. Процесс обучения похож на word2vec: по корпусу текстов проходят окном ширины 10 и оптимизируют лосс для центрального слова. Так как вся оптимизация происходит на сфере, обычный SGD не подходит и используется Риманово многообразие и градиент.

Процесс получение новой точки \mathbf{x}_{t+1} из \mathbf{x}_t состоит из:

1. получения градиента $(\nabla f(\mathbf{x}_t))$,
2. проекции его на касательную гиперплоскость в данной точке $(I - \mathbf{x}_t \mathbf{x}_t^\top)$,
3. умножения на косинусное расстояние между \mathbf{x}_t и $-\nabla f(\mathbf{x}_t)$ $(1 + \frac{\mathbf{x}_t^\top \nabla f(\mathbf{x}_t)}{\|\nabla f(\mathbf{x}_t)\|})$,
4. умножение на learning rate $(-\eta_t)$,
5. проекции на единичную сферу $(R_{\mathbf{x}_t})$

Сокращенно:

$$\mathbf{x}_{t+1} = R_{\mathbf{x}_t} \left(-\eta_t \left(1 + \frac{\mathbf{x}_t^\top \nabla f(\mathbf{x}_t)}{\|\nabla f(\mathbf{x}_t)\|} \right) (I - \mathbf{x}_t \mathbf{x}_t^\top) \nabla f(\mathbf{x}_t) \right).$$

Доказывается, что процесс при бесконечном количестве шагов и выполнении стандартных условий на learning rate ($\sum_t \eta_t^2 < \infty, \sum_t \eta_t = \infty$) сходится.

JoSE показал значительно лучшее качество, чем BERT (усредненные контекстные эмбеддинги), Doc2Vec в задачи классификации документов (см. рис. 10, слева). В задаче поиска похожих слов/документов JoSE дал небольшой прирост по сравнению с Doc2Vec (см. рис. 10, справа).

JoSE обучается почти в два раза быстрее GloVe, при этом получая лучше с точки зрения задачи поиска похожих слов/документов представления (см. рис. 11).

Table 3: Document classification evaluation using k -NN ($k = 3$).

Embedding	20 Newsgroup		Movie Review	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1
Avg. W2V	0.630	0.631	0.712	0.713
SIF	0.552	0.549	0.650	0.656
BERT	0.380	0.371	0.664	0.665
Doc2Vec	0.648	0.645	0.674	0.678
JoSE	0.703	0.707	0.764	0.765

Table 1: Spearman rank correlation on word similarity evaluation.

Embedding Space	Model	WordSim353	MEN	SimLex999
Euclidean	Word2Vec	0.711	0.726	0.311
	GloVe	0.598	0.690	0.321
	fastText	0.697	0.722	0.303
	BERT	0.477	0.594	0.287
Poincaré	Poincaré GloVe	0.623	0.652	0.321
Spherical	JoSE	0.739	0.748	0.339

Рис. 10: Сравнение методов в задачи классификации документов (слева) и поиска схожих слов/документов (справа). [15]

Table 4: Training time (per iteration) on the latest Wikipedia dump.

Word2Vec	GloVe	fastText	BERT	Poincaré GloVe	JoSE
0.81 hrs	0.85 hrs	2.11 hrs	> 5 days	1.25 hrs	0.73 hrs

Рис. 11: Сравнение времени работы методов (одна итерация — один проход по корпусу текстов). Для обучения BERT использовались 8x GeForce GTX 1080 GPU. Для остальных методов только 20 cores of Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80 GHz. [15]

§5. Контекстные представления слов

5.1. Tag lm

Рассмотрим пары предложений:

- Я люблю *печь* пироги. — *Печь* стоит у правой стены.
- *Лечу* людей по призванию. — Завтра *лечу* на другой континент.

Слова, выделенные наклонным шрифтом, имеют разные значения в зависимости от контекста. В моделях, основанных на word2vec, данные слова будут представлены одним эмбеддингом. Чтобы преодолеть такую статическую векторизацию, авторы [16] сделали представления, в котором конкатенировалось 2 представления:

- предобученные представления слов, например, skip-gramm,
- эмбеддинги слов нейронной сети (lm), предобученной на задачу предсказания следующего слова в предложении.

5.2. CoVe

Разовьем идею зависимости центрального слова в окне от других рядом стоящих. В [17] используется блок внимания (attention) (см. рис. 12) для параметризации общей зависимости слова от контекста.

В задаче типа вопрос-ответ Char+CoVe побила SOTA в 2017 год (см. рис. 13).

Данный подход хоть и перспективный, сталкивается с задачей машинного перевода. Таким образом решение задачи содержит решение не менее сложной подзадачи.

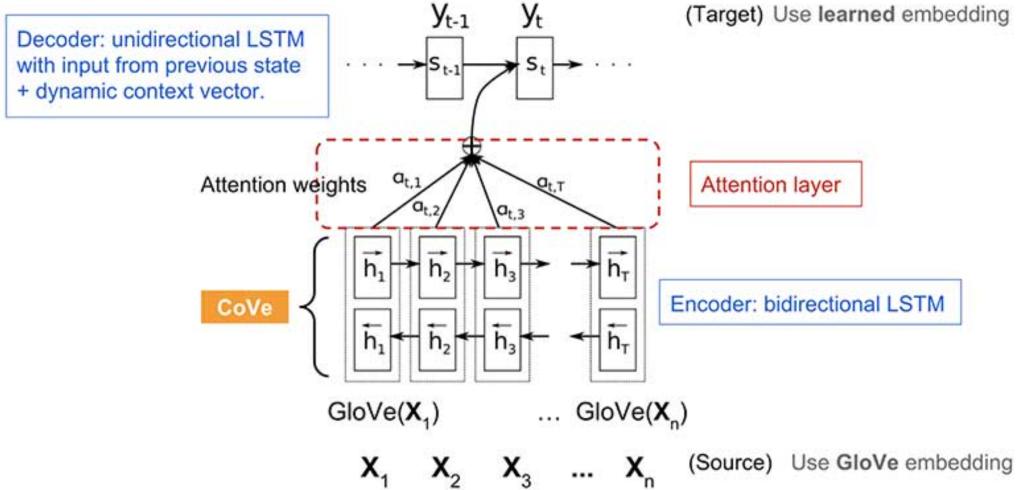


Рис. 12: Модель, используемая в CoVe для получения эмбеддингов. Вначале выбираются Glove представления для слов, обучается нейронная сеть задаче машинного перевода (NMT). Контекстные эмбеддинги получаются из скрытых слоев h_i . Итоговый эмбеддинги — конкатенация GloVe и полученных контекстных представлений. [18]

Model	EM	F1
LR [Rajpurkar et al., 2016]	40.0	51.0
DCR [Yu et al., 2017]	62.5	72.1
hM-LSTM+AP [Wang and Jiang, 2017]	64.1	73.9
DCN+Char [Xiong et al., 2017]	65.4	75.6
BIDAF [Seo et al., 2017]	68.0	77.3
R-NET [Wang et al., 2017]	71.1	79.5
DCN+Char+CoVe [Ours]	71.3	79.9

Рис. 13: Результаты SQuAD (EM характеризует точное совпадение, F1 — f1 оценка качества). Char — character n-gram. [17]

5.3. ELMo

В данном методе представления получаются в результате предтренировки без учителя. Обучается двунаправленная языковая модель (biLM) на большом корпусе текстов. Новое предложение в нашей задаче пропускается через biLM. Представление слова = линейная комбинация скрытых состояний слова (см. рис. 14).

Есть несколько хороших свойств, получающихся в результате использования таких представления, они:

- зависят от всего предложения,
- захватывают разный уровень абстракции,
- есть возможность дообучиться под конкретную задачу (обучить линейные коэффициенты)

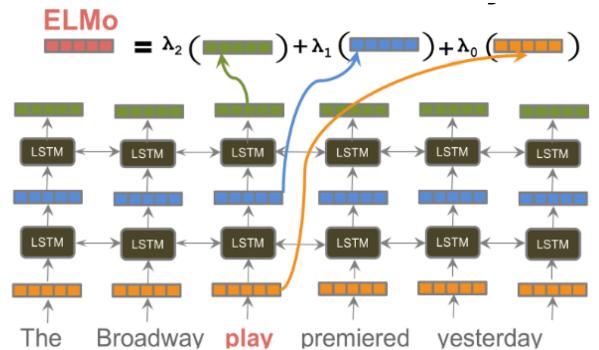


Рис. 14: Устройство работы ELMMo. [18]

5.4. FLAIR

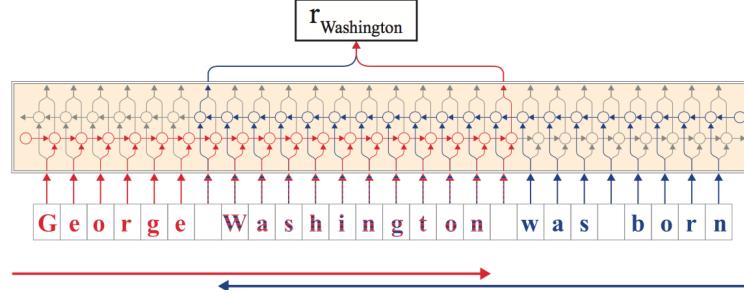


Рис. 15: Устройство работы FLAIR. Для получения эмбеддингов берется пара скрытых состояний, соответствующих концам слова. [19]

Спустимся на уровень символов, обучим языковую модель. Возьмем скрытые представления концов слова, как показано на рис. 16. Сконкатенируем их и GloVe, получим FLAIR эмбеддинги. [19]

Task	PROPOSED	Previous best
NER English	93.09 \pm 0.12	92.22 \pm 0.1 (Peters et al., 2018)
NER German	88.32 \pm 0.2	78.76 (Lample et al., 2016)
Chunking	96.72 \pm 0.05	96.37 \pm 0.05 (Peters et al., 2017)
PoS tagging	97.85 \pm 0.01	97.64 (Choi, 2016)

Рис. 16: Оценки качества для FLAIR и предыдущих SOTA моделей. Задачи включают CONLL03 NER на английском и немецком, где FLAIR сильно выигрывает. [19]

5.5. Трансформеры

В данном разделе рассмотрены примеры моделей, в которых используется энкодер/декодер трансформера [20].

5.5.1 BERT

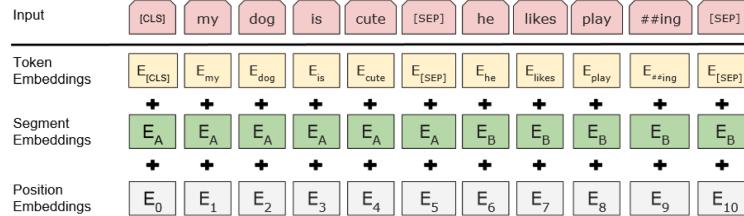


Рис. 17: Представление эмбеддингов в BERT. [4]

В BERT эмбеддинги входной последовательности из двух предложений состоят из (см. рис. 17):

- эмбеддингов токена,
- эмбеддингов номера предложения,
- эмбеддингов позиции токена.

Существует две постановки задачи, для которых получаются «хорошие» эмбеддинги:

- Замаскируем 15% случайно выбранных токенов из корпуса текстов и обучим нейронную сеть (BERT) на задачу их предсказания (MLM).
- Возьмем два предложения A, B. Задача — ответить на вопрос: стоит ли B после A или нет.

BERT представляет из себя последовательность энкодеров трансформера [20]. После каждого слоя получается своеобразное представление токена. Естественный вопрос заключается в выборе наилучшей комбинации этих представлений (см. рис. 18).

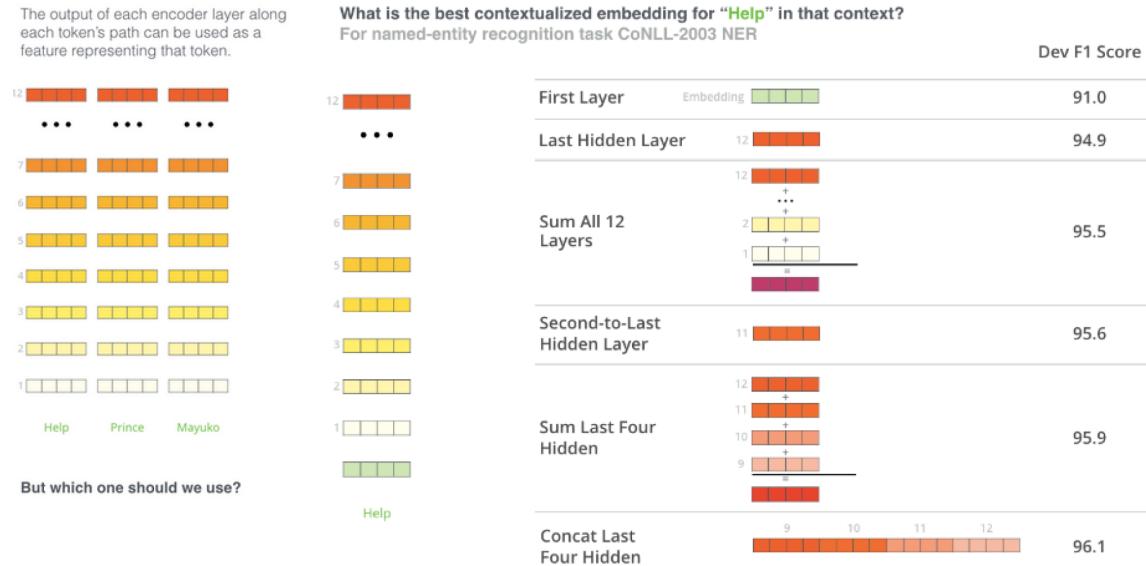


Рис. 18: Сравнение способов получения эмбеддингов токенов в BERT. [4]

Видно, что лучшим методом оказалась конкатенация выходов четырех последних слов.

5.5.2 ELECTRA

Model	Train FLOPs	Params	SQuAD 1.1 dev		SQuAD 2.0 dev		SQuAD 2.0 test	
			EM	F1	EM	F1	EM	F1
BERT-Base	6.4e19 (0.09x)	110M	80.8	88.5	—	—	—	—
BERT	1.9e20 (0.27x)	335M	84.1	90.9	79.0	81.8	80.0	83.0
SpanBERT	7.1e20 (1x)	335M	88.8	94.6	85.7	88.7	85.7	88.7
XLNet-Base	6.6e19 (0.09x)	117M	81.3	—	78.5	—	—	—
XLNet	3.9e21 (5.4x)	360M	89.7	95.1	87.9	90.6	87.9	90.7
RoBERTa-100K	6.4e20 (0.90x)	356M	—	94.0	—	87.7	—	—
RoBERTa-500K	3.2e21 (4.5x)	356M	88.9	94.6	86.5	89.4	86.8	89.8
ALBERT	3.1e22 (44x)	235M	89.3	94.8	87.4	90.2	88.1	90.9
BERT (ours)	7.1e20 (1x)	335M	88.0	93.7	84.7	87.5	—	—
ELECTRA-Base	6.4e19 (0.09x)	110M	84.5	90.8	80.5	83.3	—	—
ELECTRA-400K	7.1e20 (1x)	335M	88.7	94.2	86.9	89.6	—	—
ELECTRA-1.75M	3.1e21 (4.4x)	335M	89.7	94.9	88.0	90.6	88.7	91.4

Table 4: Results on the SQuAD for non-ensemble models.

Рис. 19: Сравнение моделей на задаче SQuAD. [2]

Одной из модификаций, показывающей более высокие оценки качества, является ELECTRA [2] (см. рис. 19). Обучение состоит из двух этапов (см. рис. 20):

1. Обучение генератора задачи предсказания замаскированных слов. Лосс — NLL.

- Фиксируем генератор, инициализируем весы дискриминатора весами генератора и обучаем его отличать маскированные и немаскированные слова, полученные из генератора. Причем если генератор сэмплировал корректное слово (которое стоит на этом месте в данных), то метка этого слова меняется на немаскированную. Лосс — поэлементная кросс-энтропия.

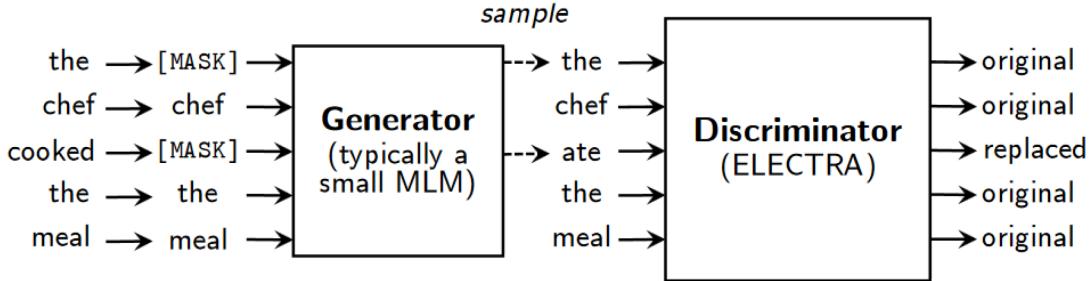


Рис. 20: Иллюстрации работы ELECTRA. [2]

В ELECTRA эмбеддинги имеют размер скрытого состояния дискриминатора. Генератор использует эти же представления слов после прохода через линейные слои размерности скрытого состояния генератора.

§6. Заключение

Контекстно-зависимые векторные представления на данный момент самые популярные, но их применение осложнено большим количеством параметров. Постановка задачи играет большую роль в эффективности обучения и качестве конечных представлений. ELECTRA быстрее и лучше учится, чем BERT, хотя использует те же «базовые единицы».

Контекстно-независимые модели являются инструментом для построения контекстно-зависимых. Последние работы связаны с переходом пространства обучения в пространство, более подходящее для получения конечных оценок качества.

Отдельно можно выделить появление эмбеддингов, специфичных для определенных областей, например, [21].

§7. Список литературы

- [1] Song Y. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. — 2018. — Vol. 2. — P. 175–180.
- [2] Clark K. Electra: Pre-training text encoders as discriminators rather than generators // arXiv preprint arXiv:2003.10555. — 2020.
- [3] Pennington J., Socher R., Manning C. D. Glove: Global vectors for word representation // Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). — 2014. — P. 1532–1543.
- [4] Devlin J. Bert: Pre-training of deep bidirectional transformers for language understanding // arXiv preprint arXiv:1810.04805. — 2018.
- [5] Deerwester S. Indexing by latent semantic analysis // Journal of the American society for information science. — 1990. — Vol. 41, no. 6. — P. 391–407.
- [6] Brown P. F. Class-based n-gram models of natural language // Computational linguistics. — 1992. — Vol. 18, no. 4. — P. 467–480.
- [7] Blei D. M., Ng A. Y., Jordan M. I. Latent dirichlet allocation // The Journal of machine Learning research. — 2003. — Vol. 3. — P. 993–1022.
- [8] Bengio Y. A neural probabilistic language model // The journal of machine learning research. — 2003. — Vol. 3. — P. 1137–1155.
- [9] Mikolov T. Efficient estimation of word representations in vector space // arXiv preprint arXiv:1301.3781. — 2013.
- [10] Mikolov T. Distributed representations of words and phrases and their compositionality // arXiv preprint arXiv:1310.4546. — 2013.
- [11] Ling W. Two/too simple adaptations of word2vec for syntax problems // Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. — 2015. — P. 1299–1304.
- [12] Group Turku NLP. — Access mode: http://bionlp-www.utu.fi/wv_demo/.
- [13] Bojanowski P. Enriching word vectors with subword information // Transactions of the Association for Computational Linguistics. — 2017. — Vol. 5. — P. 135–146.
- [14] Group Turku NLP. GloVe: Global Vectors for Word Representatio. — Access mode: <https://nlp.stanford.edu/projects/glove/>.
- [15] Meng Y. Spherical text embedding // arXiv preprint arXiv:1911.01196. — 2019.
- [16] Peters M. E. Semi-supervised sequence tagging with bidirectional language models // arXiv preprint arXiv:1705.00108. — 2017.
- [17] McCann B. Learned in translation: Contextualized word vectors // arXiv preprint arXiv:1708.00107. — 2017.

- [18] Weng L. Generalized Language Models: CoVe, ELMo Cross-View Training. — 2019. — Access mode: <https://www.topbots.com/generalized-language-models-cove-elmo/>.
- [19] Akbik A., Blythe D., Vollgraf R. Contextual string embeddings for sequence labeling // Proceedings of the 27th international conference on computational linguistics. — 2018. — P. 1638–1649.
- [20] Vaswani A. Attention is all you need // arXiv preprint arXiv:1706.03762. — 2017.
- [21] Tshitoyan V. Unsupervised word embeddings capture latent knowledge from materials science literature // Nature. — 2019. — Vol. 571, no. 7763. — P. 95–98.