DOI: 10.15514/ISPRAS-2020-32(2)-1

Векторные модели на основе символьных н-грамм для морфологического анализа текстов

Ц.Г. Гукасян, ORCID: 0000-0003-2389-517X <tsggukasyan@ispras.ru>
Российско-Армянский университет,
ул. Овсепа Эмина 123, Ереван, 119991 РА

Аннотация. В работе представляются модификации модели векторов fastText, основанные исключительно на н-граммах, для морфологического анализа текстов. fastText - библиотека для классификации текстов и обучения векторных представлений. Представление каждого слова вычисляется как сумма его отдельного вектора и векторов его символьных н-грамм. fastText хранит и использует отдельный вектор для целого слова, но во внесловарных случаях такой вектор отсутствует, что приводит к ухудшению качества получаемого вектора слова. Кроме того, в результате хранения векторов для целых слов, модели fastText обычно требуют много памяти для хранения и обработки Это становится особенно проблематично для морфологически богатых языков, учитывая многочисленность словоформ. В отличие от исходной модели fastText, предлагаемые варианты используют только информацию об н-граммах слова, избавляя от зависимости от векторов на уровне слов и в то же время помогая значительно сократить количество параметров в модели. Предлагается два способа извлечения информации из слова: внутренние символьные н-граммы и суффиксы. Модели тестируются на корпусе СинТагРус в задаче морфологической разметки и лемматизации русского языка, и показывают результаты, сравнимые с исходной моделью fastText.

Ключевые слова: вектора слов; морфологический анализ; lemmatization

Для цитирования: Гукасян Ц.Г. Векторные модели на основе символьных н-грамм для морфологического анализа текстов**.** Труды ИСП РАН, том 32, вып. 2, 2020 г., стр. 7-14. DOI: 10.15514/ISPRAS–2020–32(2)–1

Благодарности. Автор благодарит Ешилбашян Е.М., Аветисяна К.И., Королева С.Н. и Майорова В.Д. за помощь в разработке и экспериментах, а также Турдакова Д.Ю., Андрианова И.А., Хачатряна Г.А., Трифонова В.Д. за ценные отзывы и обсуждения.

Character N-gram-Based Word Embeddings for Morphological Analysis of Texts

Ts. Ghukasyan, ORCID: 0000-0003-2389-517X <tsggukasyan@ispras.ru>
Ivannikov Laboratory for System Programming at Russian-Armenian University,
123 Hovsep Emin str., Yerevan, 0051 Armenia

Abstract. The paper presents modifications of fastText word embedding model based solely on n-grams, for morphological analysis of texts. fastText is a library for classifying texts and teaching vector representations. The representation of each word is calculated as the sum of its individual vector and the vectors of its symbolic n-grams. fastText stores and uses a separate vector for the whole word, but in extra-vocabular cases there is no such vector, which leads to a deterioration in the quality of the resulting word vector. In addition, as a result of storing vectors for whole words, fastText models usually require a lot of memory for storage and processing. This becomes especially problematic for morphologically rich languages, given the large number of word forms. Unlike the original fastText model, the proposed modifications only pretrain and use vectors for the character n-grams of a word, eliminating the reliance on word-level vectors and at the same time helping to

significantly reduce the number of parameters in the model. Two approaches are used to extract information from a word: internal character n-grams and suffixes. Proposed models are tested in the task of morphological analysis and lemmatization of the Russian language, using SynTagRus corpus, and demonstrate results comparable to the original fastText.

Keywords: word embeddings; morphological analysis; lemmatization

For citation: Ghukasyan Ts. Character N-gram-Based Word Embeddings for Morphological Analysis of Texts. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 2, 2020. pp. 7-14 (in Russian). DOI: 10.15514/ISPRAS–2020–32(2)–1

Acknowledgments. The author thanks E. Yeshilbashyan, K. Avetisyan, S. N. Korolev and Mayorov V.D. for assistance in the development and experiments, as well as Turdakov D.Yu., Andrianov I.A., Khachatryan G.A., Trifonov V.D. for valuable feedback and discussions.

1. Введение

Вектора слов широко и успешно используются во многих задачах обработки естественного языка, но они имеют серьезные недостатки для обработки редких слов или слов из словарного запаса, для которых вложения либо недоступны, либо неудовлетворительны. Это особенно проблематично для морфологически богатых языков, где лексемы имеют много редких словоформ. Векторные представления слов на основе подслов были популярным направлением исследований в последние годы. В них слово рассматривается как мешок подслов, и используется исключительно эта внутренняя информация для составления вектора целого слова, позволяя получать вектора для внесловарных слов и тем самым помогая преодолеть проблему разреженности данных. Ранние попытки были сосредоточены на восстановлении предобученных векторов [1-4]. Однако, эти подходы по-прежнему требовали предварительного обучения векторов на уровне слов и были разработаны специально для обработки внесловарных случаев. Последующие подходы были направлены на обучение векторов морфемоподобных подслов непосредственно. В этих работах получение вектора слова производилось через агрегирование векторов подслов с помощью обычного усреднения, рекуррентных сетей или механизма self-attention [5-6].

fastText [7] – библиотека для классификации текстов и обучения векторных представлений. В последнем режиме fastText учит представления слов с использованием символьных н-грамм, обучая нейронную сеть вида SkipGram или CBOW на неразмеченных текстах. Представление каждого слова вычисляется как сумма его отдельного вектора и векторов его символьных н-грамм. Отсюда вытекает преимущество fastText по сравнению с другими моделями встраивания слов, заключающееся в том, что он может вычислять представление для слова вне словарного запаса (OOV), используя его символьные н-граммы. Как видно, fastText хранит и использует отдельный вектор для целого слова, но во внесловарных случаях такой вектор отсутствует, что приводит к ухудшению качества получаемого вектора слова. Кроме того, в результате хранения векторов для целых слов, модели fastText обычно требуют много памяти для хранения и обработки (модели векторов от Facebook, обученные на Common Crawl, весят 7,3 ГБ и 4,5 ГБ в форматах .bin и .vec соответственно [8]). Это становится особенно проблематично для морфологически богатых языков, учитывая многочисленность словоформ. В результате, для таких языков получаются модели, имеющие большое количество параметров и требующие много памяти. По сравнению со словоформами, словарь подслов имеет, как правило, меньший размер и позволяет получать модели со значительно более маленьким числом параметров.

В этой работе рассматриваются модификации fastText, которые удаляют вектора на уровне слов из модели и основываются только на символьных н-граммах для обучения и генерации представлений. За исключением использования только н-грамм при вычислении вектора слов, представленные модели в остальном не отличаются от fastText. Предложенные модели

тестируются в задаче морфологического анализа и лемматизации русских текстов и показываем, что они демонстрируют результаты, сопоставимые с исходной версией fastText.

2. Обзор моделей

В этом разделе описаны модель векторов fastText и две ее модификации – no-fastText и so-fastText, позволяющие получать вектора на основе символьных н-грамм.

2.1 fastText

В режиме без учителя fastText обучает представления слов. При обучении используется нейронная сеть с архитектурой CBOW или SkipGram [13]. SkipGram принимает в качестве входных данных векторное представление слова и применяет полносвязанный слой с softmax активацией для прогнозирования его контекстных слов. Сеть хранит отдельный вектор для каждого слова и ограниченное количество векторов для н-грамм символов. Представление каждого слова вычисляется как сумма его вектора и векторов н-грамм его символов:

$$\mathrm{V} = \mathrm{w}^{\mathrm{T}}\mathrm{E}_{\mathrm{w}} + \sum_{\mathrm{k=min}}^{\mathrm{max}} \sum_{\mathrm{i=1}}^{\mathrm{n-k+1}} \mathrm{s_{ki}}^{\mathrm{T}}\mathrm{E_s} \quad (1)$$

$$\boldsymbol{Output} = \boldsymbol{V} * \boldsymbol{E'} \quad (2)$$

где W – множество слов; S – множество символьных н-грамм; $w \in \mathbb{Z}_{+}^{|W|\mathrm{x}1}$ – унитарный код слова; $s_{ki\epsilon} \in \mathbb{Z}_{+}^{|S|\mathrm{x}1}$ – унитарный код символьной н-граммы; $E_w \in \mathbb{R}^{|\mathrm{W}|\mathrm{xdim}}$ и $E_S \in \mathbb{R}^{|\mathrm{S}|\mathrm{xdim}}$ – матрицы векторов слов и символьных н-грамм соответственно; k – длина н-граммы (min и max – гиперпараметры); $E' \in \mathbb{R}^{\mathrm{dimx}|\mathrm{W}|}$ – параметры выходного слоя. В реализации модели символьные н-граммы отображаются во множество меньшего размера с помощью хеширования. Когда встречается незнакомое слово, его представление вычисляется как сумма векторов н-грамм.

В fastText вместо softmax выходные значения обрабатываются отдельно, как в задаче бинарной классификации. Матрицы векторов и параметры выходного слоя обучаются путем обратного распространения ошибки с использованием негативного семплирования и стохастического градиентного спуска.

2.2. Ngrams-only fastText

В ngrams-only fastText (no-fastText), первой рассматриваемой модификации fastText, во время обучения и генерации представления слова игнорируется вектор целого слова и учитываются только н-граммы символов. Вектор вычисляется следующим образом:

$$\boldsymbol{V} = \sum_{\boldsymbol{k=min}}^{\boldsymbol{max}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n-k+1}} \boldsymbol{s_{ki}}^{\boldsymbol{T}} \boldsymbol{E_s} \quad (3)$$

В остальном модель идентична fastText. Данная модель основана на предположении, что символьные н-граммы слова несут достаточно информации, чтобы восстановить его значение. По сравнению со словами, словарный запас символов н-грамм ограничен, и с использованием хеширования они отображаются в фиксированное число векторов. Следовательно, модифицированная модель имеет значительно меньше параметров, чем исходная.

2.3 Suffixes-only fastText

Вторая предложенная модификация (suffixes-only fastText; so-fastText) аналогична первой, но сжимает модель еще больше. Она исключает внутренние н-граммы символов из модели,

оставляя только н-граммы, которые в конце слова, то есть суффиксы. Представление слова вычисляется как сумма векторов его суффиксов s_k:

$$\boldsymbol{V} = \sum_{\boldsymbol{k=min}}^{\boldsymbol{max}} \boldsymbol{s_k}^{\boldsymbol{T}} \boldsymbol{E_s} \quad (4)$$

Предположение, лежащее в основе этой модели, заключается в том, что суффиксы могут содержать достаточно информации о форме слова, необходимой для морфологических задач [14]. Несмотря на то, что изменения позволяют еще больше сократить словарный запас, с учетом реализации и использования хеширования количество параметров остается неизменным по сравнению с no-fastText. В то же время, в этой модификации теоретически менее выражена проблема коллизий из fastText и no-fastText, где разные н-граммы отображаются в один и тот же вектор.

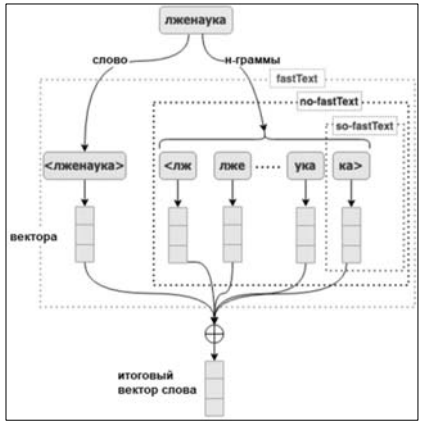

Рис. 1. Вычисление вектора слова в моделях fastText, no-fastText, so-fastText
Fig. 1. Word vector calculation in fastText, no-fastText, so-fastText models

3. Эксперименты

В этом разделе описаны параметры экспериментов по определению эффективности предложенных изменений в модели векторов fastText. Были проведены эксперименты для оценки производительности no-fastText и so-fastText в задаче морфологического анализа и лемматизации русского языка по сравнению с оригинальной версией fastText.

3.1 Векторные представления

Для всех векторов был использован размер 200 вместо значения по умолчанию 100, так как CoNLL Shared Task 2018 показала, что более высокие размеры лучше подходят для морфологического анализа [15]. В символьных н-граммов во всех моделях использовалась минимальная длина 3 и максимальная длина 6, и было задано ограничение 100000 для общего числа их векторов. Также, словарь был ограничен 400000 наиболее частыми словами. Для остальных параметров оставили значения по умолчанию.

3.2 Архитектура анализатора

Для морфологического анализатора и лемматизации была использована нейронная сеть, в целом придерживающаяся архитектуре COMBO [16] из CoNLL Shared Task 2018, с небольшими изменениями, как использование highway-слоев [17] для признаков на основе символов слова.

В качестве входных данных используются 2 компонента: вектор слова и результаты 3-слойной расширяющейся сверточной сети над символами.

Входные данные обрабатываются двумя двунаправленными рекуррентными слоями на уровне предложения, затем их результат передается в отдельные слои для конкретных задач. Для лемматизатора используется 3-слойная расширяющаяся сверточная сеть, которая выдает отдельный вектор признаков для каждого символа леммы. Лемма составляется конкатенацией отдельно предсказанных символов на основе этих векторов признаков. Прогнозирование морфологических признаков выполняется гранулярно, используя отдельный полносвязный слой для каждого из них. Важно отметить, что в этой сети параметры типичной предобученной модели векторов, как fastText или GloVe, будут составлять почти 90% от общего количества.

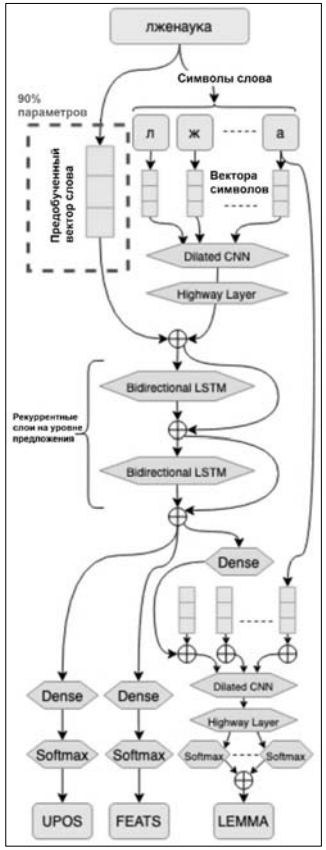

Рис. 2. Архитектура анализатора
Fig. 2. Analyzer architecture

Для реализации, обучения и оценки сети использовалась платформа BabylonDigger[1]. Во время экспериментов в сеть передавались готовые вектора, предварительно обученные и сгенерированные для каждого слова из обучающего, валидационного и тестового наборов. Эти вектора не обновлялись во время обучения, которое длилось 100 эпох, с использованием

[1] https://github.com/ispras-texterra/babylondigger

размера пакета 25 и оптимизатора Adam [18] с начальной скоростью обучения 0.01 и линейным затуханием. Каждый эксперимент был повторен с 10 различными случайными инициализациями параметров сети, и результаты были усреднены.

3.3 Данные

Для обучения векторных представлений был использован набор текстовых данных из Википедии и Common Crawl, взятый с сайта CoNLL Shared Task 2017 [15]. Для обучения анализатора использовался синтаксически размеченный корпус русского языка SynTagRus [19] версии Universal Dependencies v2.4.

4. Результаты

Табл. 1. Точность морфологического анализа (UPOS, FEATS) и лемматизации (LEMMA) с использованием разных моделей векторного представления слов
Table 1. The accuracy of morphological analysis (UPOS, FEATS) and lemmatization (LEMMA) using different models of vector representation of words

	Accuracy		
Модель	**UPOS**	**FEATS**	**LEMMA**
fastText	98.00	93.70	**96.49**
no-fastText	**98.02**	**93.72**	96.31
so-fastText	97.75	92.88	92.88

Представленные модели демонстрируют сопоставимый уровень точности по сравнению с исходной версией fastText, при этом no-fastText даже немного опережает в категориях UPOS и FEATS (Таблица 1). Несмотря на строгое ограничение на использование только суффиксов, вектора so-fastText продемонстрировали относительно хорошие результаты для русского языка [20]. В то же время нужно отметить, что эта модель исключает значительный объем информации, которая могла бы понадобиться в семантических задачах, где применяются векторные представления слов [21].

Табл. 2. Размер и количество параметров в моделях векторного представления слов
Table 2. The size and number of parameters in the models of the vector embedding of words

Модель	**.bin (Мб)**	**.txt* (Мб)**	**Количество параметров x10^6**
fastText	730	694	100
(no/so)-fastText	406	167.1	20

*текстовый файл, в каждой строке которого хранятся значения вектора слов/подслов

Благодаря небольшому размеру словаря, модели no-fastText и so-fastText заметно легче, чем обычные вложения на уровне слов, с точки зрения количества параметров и памяти (табл. 2). В них общее количество параметров меньше в 5 раз. Также, хранение параметров, необходимых для вычисления вектора слова, в текстовом формате требует более чем 4 раза меньше памяти в модифицированных моделях, чем в исходной версии.

5. Заключение

В работе были представлены модификации модели векторного представления слов fastText, которые используют только н-граммы символов для генерации представления слова. Помимо внутренних н-грамм, также была рассмотрена модель, где используются только суффиксы слова. Эти модификации устраняют зависимость fastText от векторов целых слов, и приводят к более легким моделям с точки зрения числа параметров. Модели были протестированы в

задаче морфологического анализа и лемматизации русского языка, где продемонстрировали уровень точности, сравнимый с исходной моделью.

Список литературы

[1]. Pinter Y., Guthrie R., Eisenstein J. Mimicking Word Embeddings using Subword RNNs. In Proc. of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 102-112.

[2]. Schick T., Schütze H. Attentive Mimicking: Better Word Embeddings by Attending to Informative Contexts. In Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1 (Long and Short Papers), 2019, pp. 489-494.

[3]. Zhao J., Mudgal S., Liang Y. Generalizing Word Embeddings using Bag of Subwords. In Proc. of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 601-606.

[4]. Sasaki S., Suzuki J., Inui K. Subword-based Compact Reconstruction of Word Embeddings. In Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1 (Long and Short Papers), 2019, pp. 3498-3508.

[5]. Heinzerling B., Strube M. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In Proc. of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), 2018, pp. 2989-2993.

[6]. Zhu Y., Vulić I., Korhonen A. A Systematic Study of Leveraging Subword Information for Learning Word Representations. In Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1 (Long and Short Papers), 2019, pp. 912-932.

[7]. Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, vol. 5, 2017, pp. 135–146.

[8]. Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, Tomas Mikolov. Learning Word Vectors for 157 Languages. In Proc. of the Eleventh International Conference on Language Resources and Evaluation, 2018, pp. 3483-3487.

[9]. Shibata Y. et al. Byte Pair encoding: A text compression scheme that accelerates pattern matching. Technical Report DOI-TR-161, Department of Informatics, Kyushu University, 1999.

[10]. Pennington J., Socher R., Manning C. D. Glove: Global vectors for word representation. In Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1532-1543.

[11]. Üstün A., Kurfalı M., Can B. Characters or Morphemes: How to Represent Words? In Proc. of the Third Workshop on Representation Learning for NLP, 2018, pp. 144-153.

[12]. Devlin J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding In Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1 (Long and Short Papers), 2019, pp. 4171-4186.

[13]. Mikolov T. et al. Advances in Pre-Training Distributed Word Representations. In Proc. of the Eleventh International Conference on Language Resources and Evaluation, 2018, pp. 52-55.

[14]. Zhu Y. et al. On the Importance of Subword Information for Morphological Tasks in Truly Low-Resource Languages. In Proc. of the 23rd Conference on Computational Natural Language Learning (CoNLL), 2019, pp. 216-226.

[15]. Zeman D. et al. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In Proc. of the CoNLL 2018 Shared Task: Multilingual parsing from raw text to universal dependencies, 2018, pp. 1-19.

[16]. Rybak P., Wróblewska A. Semi-supervised neural system for tagging, parsing and lemmatization In Proc. of the CoNLL 2018 Shared Task: Multilingual parsing from raw text to universal dependencies, 2018, pp. 45-54.

[17]. Srivastava R. K., Greff K., Schmidhuber J. Highway networks. arXiv preprint arXiv:1505.00387, 2015.

[18]. Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Proc. of the 3rd International Conference on Learning Representations, 2015, pp. 1-15.

[19]. Boguslavsky I. SynTagRus – a Deeply Annotated Corpus of Russian. In Les émotions dans le discours – Emotions in Discourse, Peter Lang GmbH, Internationaler Verlag der Wissenschaften, 2014, pp. 367-380.

[20]. Турдаков Д., Астраханцев Н., Недумов Я., Сысоев А., Андрианов И., Майоров В., Федоренко Д., Коршунов А., Кузнецов С. Texterra: инфраструктура для анализа текстов. Труды ИСП РАН, том 26, вып. 1, 2014 г., стр. 421-438 / Turdakov D., Astrakhantsev N., Nedumov Y., Sysoev A., Andrianov I., Mayorov V., Fedorenko D., Korshunov A., Kuznetsov S. Texterra: A Framework for Text Analysis. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 1, 2014, pp. 421-438 (in Russian). DOI: 10.15514/ISPRAS-2014-26(1)-18.

[21]. Андрианов И.А., Майоров В.Д., Турдаков Д.Ю. Современные методы аспектно-ориентированного анализа эмоциональной окраски. Труды ИСП РАН, том 27, вып. 5, 2015 г., стр. 5-22 / Andrianov I.A., Mayorov V.D., Turdakov, D.Y. Modern approaches to aspect-based sentiment analysis. Trudy ISP RAN/ Proc. ISP RAS, vol. 27, issue 5, 2015, pp. 5-22 (in Russian). DOI: 10.15514/ISPRAS-2015-27(5)-1

Информация об авторе / Information about the author

Цолак Гукасович ГУКАСЯН является аспирантом кафедры системного программирования Российско-Армянского университета. Его научные интересы включают обработку естественного языка, машинное обучение.

Tsolak Gukasovitch GHUKASYAN is a postgraduate student of the Department of System Programming of Russian-Armenian University. His research interests include natural language processing, machine learning.