УДК 004.89 DOI 10.18522/2311-3103-2024-3-187-196

**А.М. Мансур**

# АЛГОРИТМ НА ОСНОВЕ ТРАНСФОРМЕРОВ ДЛЯ КЛАССИФИКАЦИИ ДЛИННЫХ ТЕКСТОВ

*Статья посвящена актуальной проблеме представления и классификации длинных текстовых документов с использованием трансформеров. Методы представления текста, основанные на трансформерах, не могут эффективно обрабатывать длинные последовательности из-за их процесса самовнимания, который масштабируется квадратично с длиной последовательности. Это ограничение приводит к высокой вычислительной сложности и невозможности применения таких моделей для обработки длинных документов. Для устранения этого недостатка, в статье разработан алгоритм на основе трансформера SBERT, который позволяет построить векторное представление длинных текстовых документов. Ключевая идея алгоритма заключается в применении двух различных процедур к созданию векторного представления: первая основана на сегментации текста и усреднении векторов сегментов, а вторая – на конкатенации векторов сегментов. Такая комбинация процедур позволяет сохранить важную информацию из длинных документов. Для проверки эффективности алгоритма был проведен вычислительный эксперимент на группе классификаторов, построенных на основе предложенного алгоритма, и группе известных методов векторизации текста, таких как TF-IDF, LSA и BoWC. Результаты вычислительного эксперимента показали, что классификаторы на основе трансформеров в целом достигают лучших результатов по точности классификации по сравнению с классическими методами. Однако, это преимущество достигается за счет более высокой вычислительной сложности и, соответственно, более длительного времени обучения и применения таких моделей. С другой стороны, классические методы векторизации текста, такие как TF-IDF, LSA и BoWC, продемонстрировали более высокую скорость работы, что делает их более предпочтительными в случаях, когда предварительное кодирование не допускается и требуется работа в режиме реального времени. Предложенный алгоритм обработки и представления длинных документов доказал свою высокую эффективность и привел к увеличению точности классификации набора данных BBC на 0,5% по критерию F1.*

*Классификация документов; BERT; трансформеры; механизм внимания; Sentence BERT; TF-IDF; интеллектуальный анализ текста.*

**A.M. Mansour**

# A TRANSFORMER-BASED ALGORITHM FOR CLASSIFYING LONG TEXTS

*The article is devoted to the urgent problem of representing and classifying long text documents using transformers. Transformers-based text representation methods cannot effectively process long sequences due to their self-attention process, which scales quadratically with the sequence length. This limitation leads to high computational complexity and the inability to apply such models for processing long documents. To eliminate this drawback, the article developed an algorithm based on the SBERT transformer, which allows building a vector representation of long text documents. The key idea of the algorithm is the application of two different procedures for creating a vector representation: the first is based on text segmentation and averaging of segment vectors, and the second is based on concatenation of segment vectors. This combination of procedures allows preserving important information from long documents. To verify the effectiveness of the algorithm, a computational experiment was conducted on a group of classifiers built on the basis of the proposed algorithm and a group of well-known text vectorization methods, such as TF-IDF, LSA, and BoWC. The results of the computational experiment showed that transformer-based classifiers generally achieve better classification accuracy results compared to classical methods. However, this advantage is achieved at the cost of higher computational complexity and, accordingly, longer training and application times for such models. On the other hand, classical text vectorization methods, such as TF-IDF, LSA, and BoWC, demonstrated higher speed, making them more preferable in cases where pre-encoding is not allowed and real-time operation is required. The proposed algorithm has proven its high efficiency and led and led to an increase in the classification accuracy of the BBC dataset by 0.5% according to the F1 criterion.*

*Document classification; BERT; transformers; antennation mechanism Sentence BERT; TF-IDF; text mining.*

**Введение**. Классификация документов является задачей обработки естественного языка (NLP, Natural Language Processing), которая заключается в присвоении текстовым документам заранее определенных категорий или меток. Это фундаментальная задача в области извлечения информации, и она находит применение в различных областях, включая обнаружение спама, анализ настроений и тематическую категоризацию. Основная цель методов классификации документов заключается в автоматизации организации и индексации больших объемов текстовых данных, что упрощает поиск, извлечение и анализ информации [1, 2].

Для эффективной классификации текста с использованием алгоритмов машинного обучения необходимо преобразовать неструктурированные текстовые данные в формат, который может быть обработан алгоритмами. Этот процесс называется векторизацией текста или извлечением признаков [3, 5]. Выбранный метод представления текста может значительно повлиять на производительность модели классификации.

Традиционные методы классификации документов полагаются на ручную разработку признаков и часто используют алгоритмы контролируемого обучения. Одним из хорошо известных примеров является метод классификации текста на основе правил и шаблонов, который использует заранее определенные критерии, такие как ключевые слова или фразы, для автоматического распределения документов по категориям [1, 7]. Этот подход широко применяется, например, в фильтрах спама для электронной почты. Он определяет категорию документа, основываясь на частоте появления ключевых слов или терминов, часто используя модель «мешок слов», которая представляет текст как набор уникальных слов без учета их порядка или грамматических связей [9]. Несмотря на свою простоту, этот метод классификации может быть эффективным для решения конкретных задач, где заранее известны характерные признаки документов различных категорий.

Современные методы классификации документов используют передовые техники машинного обучения, включая глубокое обучение и модели трансформеров [4]. Эти методы часто используют не контролируемое или полу-контролируемое обучение и могут захватывать более тонкие модели в текстовых данных. Трансформеры, такие как BERT [6] (Bidirectional Encoder Representations from Transformers), революционизировали обработку естественного языка, но они ограничены обработкой текстовых последовательностей до 512 токенов. Это создает проблему для классификации длинных документов, которые часто превышают этот лимит. Исследователи изучали различные методики адаптации BERT для классификации текстов большого объема, включая усечение, разбиение на фрагменты и специализированные архитектуры [8, 10, 11]. В этой статье мы рассмотрим проблему применения BERT к длинным документам и обзор некоторых ключевых подходов, предложенных для решения этой задачи.

Механизм внимания BERT, который является ключевым компонентом его архитектуры, имеет вычислительную сложность, которая растет квадратично с длиной входной последовательности. Это делает вычислительно дорогостоящим применение BERT напрямую к длинным документам. Наивные стратегии усечения, такие как использование только первых 512 токенов, могут привести к потере важной информации. Для эффективного использования полного контекста длинных текстов требуются более сложные методы.

Было исследовано несколько методов для расширения возможностей BERT для классификации текстов большого объема. Один из подходов заключается в разделении документа на более мелкие фрагменты, которые укладываются в лимит в 512 токенов, обработке каждого фрагмента независимо с помощью BERT и последующем агрегировании полученных представлений. Другим вариантом является использование специализированных архитектур трансформеров, таких как Longformer [12] или Big Bird [13], которые были разработаны для более эффективной обработки более длинных последовательностей.

Кроме того, были исследованы методы, такие как текстовое резюмирование, в качестве способа сжатия длинных документов до более управляемого размера перед подачей их в BERT. Извлекая наиболее важную информацию из исходного текста, эти методы

потенциально могут сохранить ключевое содержание, одновременно снижая вычислительную нагрузку. Однако результат классификации будет зависеть от эффективности процесса реферирования текста, и нельзя игнорировать дополнительные вычислительные затраты, необходимые для резюмирования текста.

В подтверждение этого теоретического анализа в этой статье представляется алгоритм классификации, который использует кодирование BERT для представления текста. Алгоритм характеризуется применением двух различных процедур для представления документа. Первая процедура включает в себя разделение документа на текстовые сегменты длиной 512 (максимально допустимый размер для входных данных BERT). Затем документ представляется как среднее значение векторов сегментов. Вторая процедура основана на предположении, что важная информация в документе часто сосредоточена в начале и заключении. Эта процедура учитывает кодирование первых и последних 512 токенов документа и представляет документ как вектор, полученный путем конкатенации двух векторов сегментов. Таким образом, основные вклады в эту работу:

1. Алгоритм представления текстовых документов в виде векторов с использованием двух разных процедуры: первая следует методу сегментации текса и усреднения векторов, а вторая – конкатенация векторов.

2. Программное приложение для классификации документов с использованием предложенного представления документов и проведения вычислительного эксперимента[1] по набору канонических методов.

**1. Аналитический обзор методов векторизации текстов на основе Трансформеров.** *Трансформеры (англ. Transformers)*: Архитектура трансформеров, представленная [4], революционизировала обработку естественного языка. Модели, такие как BERT [6] и GPT [14] (Generative Pre-trained Transformer), достигли состояния искусства в различных задачах NLP, включая классификацию документов. Эти модели могут захватывать контекстуальные представления текста, что позволяет более точно и тонко классифицировать документы. Архитектура трансформеров особенно эффективна для классификации документов благодаря своей способности обрабатывать текст последовательно и параллельно, захватывая как местные, так и глобальные зависимости в данных.

*BERT* – это модель языкового представления на основе архитектуры трансформера [6, 19]. *BERT* предназначен для предварительного обучения глубоких двунаправленных представлений, что позволяет ему учитывать контекст с обеих сторон токена при его кодировании. Это отличается от предыдущих моделей, таких как *ELMo* [15], которые только однонаправленны. *BERT* использует блоки трансформеров, каждый из которых содержит слой многоголовой внимания и слой feed-forward neural network. Вход для *BERT* – это последовательность токенов, для представления каждого из которых *BERT* комбинирует векторные представления токенов, сегментов и позиций.

*BERT* предварительно обучается на двух неопределенных задачах: маскированном языковом моделировании (MLM, Masked Language Modeling) [22] и предсказании следующего предложения (NSP, Next Sentence Prediction) [21]. При MLM 15% входных токенов случайно маскируются, и модель должна предсказать оригинальный идентификатор лексемы маскированного токена на основе контекста. В задаче NSP по двум предложениям модель предсказывает, следует ли второе предложение за первым в исходном тексте. Эти цели предварительного обучения позволяют BERT уметь контекстуальные представления слов и понимать отношения между предложениями.

После предварительного обучения на больших неопределенных данных BERT может быть точным обучением на конкретных задачах, добавляя простой выходной слой. Этот процесс точного обучения требует намного меньше маркированных данных и изменений в архитектуре, специфичных для задачи, по сравнению с обучением модели с нуля.

[1] Ссылка на реализацию программы на GitHub https://github.com/Ali-MH-Mansour/ Document-classification/blob/main/Document_classification_BERT.ipynb.

BERT продемонстрировал лучшие результаты на различных задачах HLP, включая вопросно-ответную систему, вывод на основе языка, классификацию текста, определение именованных сущностей и анализ настроений.

*Sentence-BERT (SBERT)* [16] – это модификация предварительно обученной сети BERT, которая использует архитектуры сиамских и триплетных сетей для создания встраиваний предложений, которые можно эффективно сравнивать с помощью косинусного сходства.

Архитектура SBERT модифицирует оригинальную модель BERT, удаляя финальный классификационный слой и включая сиамскую архитектуру. Во время обучения SBERT обрабатывает пары предложений, при этом каждая подсеть BERT генерирует обобщенные встраивания предложений. Эти встраивания затем сравниваются с использованием косинусного сходства для получения оценки сходства. Сиамские сети состоят из двух или более идентичных подсетей с общими параметрами. Они широко используются в задачах, требующих сравнения, таких как поиск похожих предложений. SBERT использует триплетные потери, когда сеть обучается на якоре, положительной паре и отрицательной паре, что значительно снижает вычислительную нагрузку.

По сравнению с BERT, SBERT имеет ряд преимуществ. BERT требует совместной обработки обоих предложений в паре, что приводит к высокой вычислительной нагрузке. Например, поиск самой похожей пары из 10 000 предложений потребовал бы 50 миллионов вычислений вывода (~65 часов) с BERT. SBERT сокращает это время до около 5 секунд для вычисления встраиваний и 0,01 секунды для расчета косинусного сходства [17].

SBERT превосходит BERT в задачах семантического поиска, требующих нахождения предложений со схожими значениями. Он полезен для выявления аргументов с похожими темами и рассуждениями на различные темы. SBERT можно использовать для вычисления встраиваний предложений или проведения поиска по семантическому сходству с помощью простой библиотеки на Python.

В целом, SBERT значительно улучшает понимание предложений по сравнению с BERT, включая архитектуры сиамских и триплетных сетей. Его эффективное вычисление встраиваний предложений делает его идеальным для задач семантического поиска и других задач обработки естественного языка на уровне предложений.

**2. Постановка задачи.** Задача классификации документов может быть сформулирована следующим образом:

Пусть имеется множество документов $\boldsymbol{D} = \{d_1, d_2, ..., d_{|D|}\}$ и множество категорий (классов) $\boldsymbol{C} = \{\mathbf{c_1}, \mathbf{c_2}, ..., \mathbf{c_{|C|}}\}$. Существует неизвестная целевая функция $\mathbf{\Phi: D \times C \to \{0, 1\}}$, которая сопоставляет каждому документу и категории значение 1, если документ относится к этой категории, и 0 в противном случае. Задача состоит в построении функции $\mathbf{\Phi'}$, максимально близкой к $\mathbf{\Phi}$, то есть функции, которая корректно классифицирует документы по категориям. Для решения этой задачи используются методы машинного обучения, которые опираются на наличие коллекции заранее классифицированных документов $\mathbf{\Omega} = \{d_1, d_2, ..., d_{|\Omega|}\}$.

Тогда задача состоит в нахождении такой функции $\mathbf{\Phi'}$, которая минимизирует суммарные потери на коллекции $\mathbf{\Omega}$:

$$\min_{\mathbf{\Phi'}} \sum_{\mathbf{d} \in \mathbf{\Omega}} \mathbf{L}(\mathbf{\Phi}(\mathbf{d}), \mathbf{\Phi'}(\mathbf{d})),$$

где $\mathbf{L}(\mathbf{\Phi}(\mathbf{d}), \mathbf{\Phi'}(\mathbf{d}))$ – функция потерь, оценивающая качество аппроксимации $\mathbf{\Phi'}$ относительно $\mathbf{\Phi}$ для документа $\mathbf{d}$.

**3. Реализация алгоритмов классификации документов.** Предлагаемый алгоритм характеризуется применением двух различных процедур для представления документа. Ниже представлена идея каждого из них с подробным объяснением шагов.

Первая процедура включает в себя разделение документа на текстовые сегменты фиксированной длиной. Поскольку модель BERT принимает текстовую последовательность ограниченной длины 512 токенов, алгоритм делит текст документа на сегменты

длиной 512. Затем создается векторное представление каждого сегмента. Документ представляется как среднее значение векторов сегментов. Процесс векторного усреднения создает вектор центроида документа, который представляет документ лучше, чем просто представление первых 512 токенов и усечение остальных.

Векторы центроидов используются в качестве входных данных для классификатора, который, в свою очередь, определяет класс документа по заданному алгоритму. Ниже приведен псевдокод алгоритма с применением первой процедуры.

**Алгоритм с применением процедуры усреднения векторов**

**Ввод**: список документов ***D***
**Вывод**: векторное представление документов
Инициализировать пустой словарь ***document_embeddings = {}***
*Инициализировать максимальную длину сегментов* ***L***=512
Загрузить предварительно обученную модель BERT.
***For*** each ***d*** in the documents list ***D do***:
    Разделить документ на сегменты длиной `L` (за исключением последнего сегмента, который может быть короче).
    Игнорировать сегменты, размер которых меньше порогового значения
    сохранять сегменты в списке ***S***
    segment_embeddings = [] // *Инициализировать пустой список `segment_embeddings` для хранения векторных представлений сегментов*
    ***For*** each ***segment*** in ***S do***:
        Vector = ***BERT***(segment) //*Получить векторное представление сегмента, используя модель BERT.*
        segment_embeddings [] ← vector// добавить вектор в список
    ***End for***
    document_vector = ***mean***(segment_embeddings) // *Вычислить среднее значение векторов сегментов, чтобы получить векторное представление всего документа.*
    document_embeddings[doc_id] = document_vector // *Добавить вектор документа в словарь document_embeddings, используя идентификатор документа в качестве ключа*
***End for***
***Return*** словарь ***document_embeddings***

Алгоритм начинается с инициализации пустого словаря *document_embeddings*, который будет использоваться для хранения векторных представлений документов. Затем устанавливается максимальная длина сегментов ***L=512***, которая будет использоваться при разделении документов на части. После этого загружается предварительно обученная модель *BERT*, которая будет применяться для кодирования сегментов документов в векторные представления.

Далее алгоритм переходит к обработке каждого документа ***d*** из входного списка ***D***. Для каждого документа он разделяется на сегменты длиной ***L***, за исключением последнего сегмента, который может быть короче. Все сегменты сохраняются в списке ***S***. Затем для каждого сегмента в списке ***S*** используется модель BERT для получения векторного представления. Все векторы сегментов сохраняются в списке *segment_embeddings*. После обработки всех сегментов вычисляется среднее значение векторов сегментов, чтобы получить векторное представление всего документа. Это векторное представление документа добавляется в словарь *document_embeddings*, используя идентификатор документа в качестве ключа.

**Применение процедуры конкатенации векторов**. Вводные и заключительные разделы документа часто включают в себя основные термины и ключевые слова, относящиеся к теме документа. В большинстве случаев эти ключевые слова предоставляют достаточно информации для точной категоризации документа. Второй алгоритм учитывает это предположение, рассматривая только векторные представления начального и

конечного сегментов документа (512 токенов каждого). Тогда документ представляется как вектор, полученный объединением (конкатенации) двух векторов сегментов. Именно это отличает предложенный алгоритм от других, так как первого и последнего разделов достаточно, чтобы размерности не становились слишком большими, что отрицательно влияет на эффективность классификатора. Ниже приведен псевдокод алгоритма с применением второй процедуры.

**Алгоритм с применением процедуры конкатенации векторов**

**Ввод**: список документов ***D***
**Вывод**: векторное представление документов
Инициализировать пустой словарь ***document_embeddings = {}***
*Инициализировать максимальную длину сегментов* **L** // *по умолчанию L*=512
Load the BERT model
***For*** each ***d*** in the documents list ***D do***:
    ***First = d[:L]*** *//Получить первые 512 токенов*
    ***Last = d[-L:]*** *//Получить последние 512 токенов*
    vector1, vector2 = ***BERT*** ([***First, Last***]) //*закодировать первый и последний сегмент с использованием модели BERT*
    document_vector = ***concatenate*** (vector1, vector2) // *объединить векторы первого и последнего сегмента, чтобы получить вектор документа*
    document_embeddings[doc_id] = document_vector // *Добавить вектор документа в словарь document_embeddings, используя идентификатор документа в качестве ключа*
***End for***
***Return*** словарь ***document_embeddings***

**4. Вычислительный эксперимент и анализ полученных результатов.** Эффективность предложенных алгоритмов оценивалась и сравнивалась с несколькими основными методами, включая TF-IDF, *BoWC*, *BERT* и *LSA*, которые кратко описаны ниже.

*Метод TF-IDF* (Term Frequency-Inverse Document Frequency) – это статистический подход к оценке важности слова в контексте документа или корпуса документов. Он вычисляется как произведение частоты слова в документе (TF) и обратной частоты этого слова во всем корпусе (IDF), что позволяет выделить наиболее значимые термины для представления содержания документа.

*Метод BoWC* (*bag of weighted concepts*) – это расширение традиционной модели мешка слов, которое учитывает семантическую информацию о концептах, представленных в тексте. Вместо простого подсчета частоты слов, *BoWC* использует веса, основанные на семантической близости концептов, извлеченных из анализируемых текстов. Это позволяет лучше отразить смысловое содержание документа и повысить эффективность задач классификации текстов.

*Метод LSA* (*Латентно-семантический анализ*) – это метод обработки текстовой информации, основанный на гипотезе распределения и использовании сингулярного разложения для выявления скрытых взаимосвязей между терминами и концепциями в наборе документов. LSA позволяет представить текст в виде латентных признаков меньшей размерности, чем пространство слов, что помогает в задачах информационного поиска и автоматической категоризации документов.

Для проведения эксперимента использовался набор данных *BBC* [18] который включает 2225 документов, полученных с веб-сайта новостей *BBC*, охватывающих новостные истории по пяти различным тематическим областям за период 2004-2005 гг.

*Метрика оценки*. Для оценки производительности алгоритмов классификации текста используется метрика оценки F-мера [20]. Это среднее гармоническое между точностью и полнотой, которое определяется следующими формулами:

$$Precision_{C_i} = \frac{TP_{C_i}}{TP_{C_i}+FP_{C_i}}, \quad (1)$$

$$Recall_{C_i} = \frac{TP_{C_i}}{TP_{C_i}+FN_{C_i}}, \quad (3)$$

$$F-score = \frac{(1+\beta)\cdot Precision_{C_i}\cdot Recall_{C_i}}{\beta\cdot Precision_{C_i}+Recall_{C_i}}. \quad (4)$$

Здесь *C* – это метка класса, *TP* (True Positive) – это количество документов, правильно отмеченных классификатором как относящиеся к классу *C*, а *FP* (False Positive) – это количество документов, неправильно отмеченных классификатором как относящиеся к классу *C*. Между тем, *FN* (False Negative) ложноотрицательный – это количество документов, которые относятся к классу C и которые классификатор ошибочно определил как не относящиеся к классу *C*, а *TN* (True Negative) – это количество документов, не принадлежащих к классу C, правильно отмеченных классификатором как не относящиеся к классу C. Коэффициент β позволяет настраивать баланс между точностью и полнотой алгоритма кластеризации. Значение β>1 делает акцент на точности, а β<1 – на полноте. По умолчанию β=1.0.

**Реализация и анализ результатов.** Для реализации алгоритмов, основанных на BERT, была использована модель SBERT. Библиотека LSA из *Scikit-learn*[2] была применена, при этом размерность векторов была установлена равной 300. Для TF-IDF и BoWC реализация была выполнена в точности, как описано в предыдущей работе автора [3].

Векторы, сгенерированные вышеупомянутыми методами, использовались в качестве входных данных для нескольких классификаторов, включая: метод опорных векторов (SVM) с различными ядрами (полиномиальное Poly (P), радиальная базисная функция RBF (R), линейное Linear (L)), классификатор на основе искусственной нейронной сети (ANN) и классификатор ближайших соседей (K-Nearest Neighbors, KNN). Для реализации классификаторов также применялась библиотека Scikit-learn. Для нейросетевого классификатора использовался многослойный персептрон (Multi-Layer Perceptron, MLP) с настройками по умолчанию и 500 итерациями обучения. Количество ближайших соседей было установлено равным 5, что соответствует количеству классов документов в стандартном наборе данных.

Результаты вычислительного эксперимента представлены в табл. 1 и на рис. 1. В целом, можно наблюдать, что алгоритмы, основанные на BERT, достигают наилучших результатов со всеми классификаторами.

Таблица 1

**Точность классификации, измеренная по F1**

| Методы | SVM (P) | SVM (R) | SVM (L) | ANN | KNN |
|---|---|---|---|---|---|
| **SBERT$_{\text{базовый (768)}}$** | 97,60 | 98,05 | 97,30 | 97,30 | 96,40 |
| **SBERT$_{\text{усреднения векторов (768)}}$** | **98,2** | **98,50** | 98,05 | 98,05 | 97,00 |
| **SBERT$_{\text{конкатенации векторов (1536)}}$** | 97,30 | 98,05 | **98,95** | **98,35** | **97,30** |
| **TF-IDF$_{(4590)}$** | 87,40 | 97,70 | 98,20 | 97,90 | 94,10 |
| **LSA $_{(300)}$** | 97,30 | 97,75 | 97,60 | 98,20 | 94,60 |
| **BoWC $_{(200)}$** | **98,2** | 97,75 | 97,60 | 97,03 | 97,30 |

Эксперимент также подтверждает, что два предложенных алгоритма превосходят базовое кодирование BERT (С кодированием только первых 512 токенов). Использование конкатенации векторов первого и последнего текстовых сегментов для представления всего документа достигает наилучшего результата, поскольку добавляет дополнительную различительную информацию к представлению документа. Усреднение векто-

[2] Библиотеку можно посмотреть по URL scikit-learn: machine learning in Python — scikit-learn 1.5.1 documentation.

ров параграфов документа (каждый длиной 512) достигают более высокой точности, чем базовый BERT, но ниже, чем конкатенация. Это имеет смысл, потому что по мере увеличения длины вектора со более плотными признаками, точность классификации увеличивается, но вычислительные затраты также возрастают.

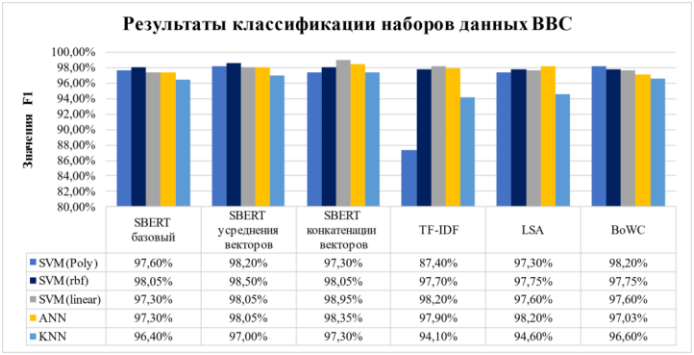


*Рис. 1. Результаты классификации набора данных BBC, измеренные мерой F1*

Также заметим, что классификатор с линейным ядром в алгоритме конкатенации векторов «*S-BERT*$_{конкатенации\ векторов}$» достиг наилучшего результата классификации набора данных BBC. Причина в том, что конкатенация векторов означает увеличение количества признаков (размерностей), что делает нелинейные ядра менее эффективными из-за «проклятия размерности».

Важно отметить, что базовое кодирование BERT не превосходит традиционные методы, такие как *TF-IDF* и *LSA*. Более того, метод *BoWC* достиг более высокой точности классификации, чем базовый BERT в сочетании с полиномиальным ядром *SVM*. Это указывает на то, что в некоторых случаях традиционные подходы могут быть более эффективными, чем современные методы на основе нейронных сетей, особенно когда последние требуют больших вычислительных ресурсов.

Относительно временных затрат, ссылаясь на табл. 2, заметим, что кодирование набора данных из 2225 документов заняло 5111.59 секунд, в то время как TF-IDF был гораздо быстрее (2,45 секунды). Время выполнения для каждого метода колеблется в зависимости от используемого классификатора и характеристик входных векторов. При одновременном учете скорости и точности метод TF-IDF, как правило, обеспечивает наилучшую общую производительность.

Таблица 2

**Сравнение времени представления документов разными методами**

| | Vectorization time | SVM | ANN | KNN |
|---|---|---|---|---|
| **S-BERT**$_{базовый\ (768)}$ | 1050.55 | 5.72 | 6.22 | 0.13 |
| **S-BERT**$_{усреднения\ векторов\ (768)}$ | 5111.59 | 6.01 | **6** | 0.11 |
| **S-BERT**$_{конкатенации\ векторов\ (1536)}$ | 2108 | 18.9 | 11 | 0.44 |
| **TF-IDF**$_{(4590)}$ | 2.45 | 20.4 | 18 | 21.1 |
| **BoWC** $_{(200)}$ | 1887 | 0.991 | 13.3 | **0.1** |
| **LSA** $_{(300)}$ | **2.11** | **2.33** | 7.36 | 0.8 |

В будущем планируется расширить эксперименты и усилить результаты на других наборах данных. Также планируется протестировать большие языковые модели в классификации документов, для которых существующие методы не смогли определить правильный класс.

**Заключение.** В данной работе представлен эффективный алгоритм классификации длинных текстовых документов на основе трансформера SBERT. Проведенное исследование продемонстрировало, что предложенный алгоритм превосходит по точности классификации классификаторы, основанные на классических методах векторизации текстов, таких как TF-IDF, *LSA* и *BoWC*. Несмотря на то, что классификаторы на базе трансформеров показали лучшие результаты в целом, они уступают классическим методам по скорости работы. Поэтому традиционные подходы остаются предпочтительными в приложениях, где необходима сверхбыстрая обработка текстов в режиме реального времени без предварительного кодирования. Предложенный алгоритм может быть применен в различных областях, таких как анализ тональности, извлечение сущностей и ключевых фраз, а также в других задачах обработки естественного языка, где необходимо работать с длинными текстовыми последовательностями. Дальнейшие исследования могут быть направлены на оптимизацию алгоритма для повышения скорости обработки и расширение его применения на другие задачи обработки естественного языка.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК


1. *Kadhim A.I.* Survey on supervised machine learning techniques for automatic text classification, *Artificial Intelligence Review,* 2019, Vol. 52, No. 1, pp. 273-292.
2. *Minaee S., Kalchbrenner N., Cambria E., Nikzad N., Chenaghlu M., Gao J.* Deep learning-based text classification: A comprehensive review, *arXiv preprint arXiv:.03705,* 2020.
3. *Mansour A., J.M. and Y.K.* Text Vectorization Method Based on Concept Mining Using Clustering Techniquesm *2022 VI International Conference on Information Technologies in Engineering Education (Inforino)*. IEEE, 2022, pp. 1-10.
4. *Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł., Polosukhin I.* Attention is all you need, *Advances in neural information processing systems,* 2017, Vol. 30.
5. *Mansour A.M., Mohammad J.H., Kravchenko Y.A.* Text vectorization using data mining methods, *Izvestia SFedU. Technical science,* 2021, No. 2.
6. *Devlin J., Chang M.-W., Lee K., Toutanova K.* Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:.04805,* 2018.
7. *Ni P., Li Y., Chang V.* Research on text classification based on automatically extracted keywords, *International Journal of Enterprise Information Systems (IJEIS),* 2020, Vol. 16, No. 4, pp. 1-16.
8. *Grootendorst M., Vanschoren J.* Beyond Bag-of-Concepts: Vectors of Locally Aggregated Concepts, *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2019, pp. 681-696.
9. *Qader W.A., Ameen M.M., Ahmed B.I.* An overview of bag of words; importance, implementation, applications, and challenges, *2019 International Engineering Conference (IEC)*. IEEE, 2019, pp. 200-204.
10. *Kim H.K., Kim H., Cho*. Bag-of-concepts: Comprehending document representation through clustering words in distributed representation, *Neurocomputing,* 2017, Vol. 266, pp. 336-352.
11. *Rücklé A., Eger S., Peyrard M., Gurevych I.* Concatenated power mean word embeddings as universal cross-lingual sentence representations, arXiv preprint arXiv:.01400, 2018.
12. *Beltagy I., Peters M.E., Cohan A.* Longformer: The long-document transformer, *arXiv preprint arXiv:.05150,* 2020.
13. *Zaheer M., Guruganesh G., Dubey K.A., Ainslie J., Alberti C., Ontanon S., Pham P., Ravula A., Wang Q., Yang L.* Big bird: Transformers for longer sequences, *Advances in neural information processing systems,* 2020, Vol. 33. Big bird, pp. 17283-17297.
14. *Floridi L., Chiriatti M.* GPT-3: Its Nature, Scope, Limits, and Consequences, *Minds and Machines,* 2020, Vol. 30. GPT-3. No. 4, pp. 681-694.
15. *Ethayarajh K.* How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings, *How Contextual are Contextualized Word Representations?*, 2019.
16. *Reimers N., Gurevych I.* Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, *arXiv preprint arXiv,* 2019, Vol. abs/1908.10084.
17. Decoding Sentence-BERT | Continuum Labs. Available at: https://training.continuumlabs.ai/knowledge/vector-databases/decoding-sentence-bert (accessed 06 July 2024).
18. *Greene,D., Cunningham P.* Practical solutions to the problem of diagonal dominance in kernel document clustering, *Proceedings of the 23rd international conference on Machine learning,* 2006, pp. 377-384.

19. *Adhikari A., Ram A., Tang R., Lin J.* Docbert: Bert for document classification, *arXiv preprint arXiv:.08398,* 2019.
20. *Sabbah T., Selamat A., Selamat M.H., Al-Anzi F.S., Viedma E.H., Krejcar O., Fujita H.* Modified frequency-based term weighting schemes for text classification, *Applied Soft Computing,* 2017, Vol. 58, pp. 193-206.
21. *Sun Y., Zheng Y., Hao C., Qiu H.* NSP-BERT: A Prompt-based Few-Shot Learner Through an Original Pre-training Task Next Sentence Prediction, 2022. NSP-BERT.
22. *Wettig A., Gao T., Zhong Z., Chen D.* Should You Mask 15% in Masked Language Modeling?, *arXiv preprint arXiv:2202.08005,* 2022.

Статью рекомендовал к опубликованию к.т.н. С.Г. Буланов.

**Мансур Али Махмуд** – Южный федеральный университет; e-mail: mansur@sfedu.com; г. Таганрог, Россия; тел.: 89880158697; кафедра систем автоматизированного проектирования; аспирант.

**Mansour Ali Mahmoud** – Southern Federal University; e-mail: mansur@sfedu.com; Taganrog, Russia; phone: +79880158697; the Department of Computer Aided Design; graduate student.