МРНТИ 20.19.27 **https://doi.org/10.58805/kazutb.v.4.21-220**

АНАЛИЗ НАУЧНЫХ ТЕКСТОВ НА ОСНОВЕ ЯЗЫКОВЫХ МОДЕЛЕЙ АЛГОРИТМАМИ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ

Г.Ж. Шуйтенов[1], У.К. Турусбекова[1*], М.М. Муратбеков[2]

[1]Esil University, Астана, Казахстан,

[2]Евразийский национальный университет им. Л. Н. Гумилева, Астана, Казахстан,

e-mail: umut.t@mail.ru

Статья анализирует проблемы, связанные с обработкой больших объемов текстовых данных, таких как научные статьи, и обсуждает возможности применения распределенных систем обработки данных для повышения эффективности анализа. В частности, авторы статьи изучают использование языковых моделей, таких как N-граммы и рекуррентные нейронные сети, для извлечения смысла и классификации научных текстов. Статья представляет алгоритмические подходы и методы, основанные на распределенной обработке, и описывает возможности использование языковых моделей и алгоритмов распределенной обработки. В целом, предлагается новый подход к анализу научных текстов, основанный на использовании языковых моделей и фреймворков распределенной обработки данных.

Ключевые слова: язык программирования Scala, научный текст, большие данные, неструктурированные данные, обработка данных, Apache Spark, распределенные вычисления, математический аппарат

ANALYSIS OF SCIENTIFIC TEXTS BASED ON LANGUAGE MODELS BY DISTRIBUTED PROCESSING ALGORITHMS

G.Zh. Shuitenov[1], U.K. Turusbekova[1*], M.M. Muratbekov[2]

[1]Esil University, Astana, Republic of Kazakhstan,

[2]L.N. Gumilyov Eurasian National University, Astana, Republic of Kazakhstan,

e-mail: umut.t@mail.ru

The paper analyzes the problems associated with processing large amounts of text data, such as scientific articles, and discusses the possibilities of using distributed data processing systems to improve the efficiency of analysis. In particular, the authors of the paper study the use of language models such as N-grams and recurrent neural networks to extract meaning and classify scientific texts. The paper presents algorithmic approaches and methods based on distributed processing and describes the possibilities of using language models and distributed processing algorithms. In general, a new approach to the analysis of scientific texts is proposed, based on the use of language models and distributed data processing frameworks.

Keywords: Scala programming language, scientific text, big data, unstructured data, data processing, Apache Spark, distributed computing, mathematical apparatus

ТАРАТЫЛҒАН ӨҢДЕУ АЛГОРИТМДЕРІ НЕГІЗІНДЕ ҒЫЛЫМИ МӘТІНДЕРДІ ТАЛДАУ

Г.Ж. Шуйтенов[1], У.К. Турусбекова[1*], М.М. Муратбеков[2]

[1]Esil University, Астана, Қазақстан,

[2]Л.Н. Гумилев атындағы Еуразия ұлттық университеті, Астана, Қазақстан,

e-mail: umut.t@mail.ru

Мақалада ғылыми жұмыстар сияқты мәтіндік деректердің үлкен көлемін өңдеуге байланысты проблемаларды талдайды және талдаудың тиімділігін арттыру үшін деректерді өңдеудің таратылған жүйелерін

қолдану мүмкіндіктерін талқылайды. Атап айтқанда, мақала авторлары N-грамм және рекуррентті нейрондық желілер сияқты тілдік үлгілерді ғылыми мәтіндердің мағынасын алу және жіктеу үшін пайдалануды зерттейді. Мақала таратылған өңдеуге негізделген алгоритмдік тәсілдер мен әдістерді ұсынады және тілдік үлгілер мен бөліп өңдеудің алгоритмдерін пайдалану мүмкіндіктерін сипаттайды. Жалпы, ғылыми мәтіндерді талдауға тілдік модельдер мен деректерді бөліп өңдеу фреймворктарын пайдалануға негізделген жаңа тәсіл ұсынылады.

Түйін сөздер: Scala бағдарламалау тілі, ғылыми мәтін, үлкен деректер, құрылымдалмаған деректер, деректерді өңдеу, Apache Spark, таратылған есептеулер, математикалық аппарат**.**

Введение. Научные тексты являются важным источником информации для исследователей и специалистов в различных областях знаний. Однако, с увеличением объема научных публикаций становится сложнее и затратнее проводить их полноценный анализ вручную. В связи с этим, все более актуальной становится разработка автоматизированных методов и алгоритмов для анализа научных текстов. Одним из подходов к анализу текстовых данных является использование языковых моделей. Языковые модели позволяют представить текст в виде математической модели, которая может быть использована для извлечения смысла, классификации и других задач анализа текста. Однако, при работе с большими объемами данных, возникают проблемы с производительностью и эффективностью работы с языковыми моделями. В свете этих проблем возможно стоит рассмотреть применение распределенной обработки для анализа научных текстов на основе языковых моделей. Распределенная обработка позволяет эффективно работать с большими объемами данных путем распределения и параллельной обработки данных на нескольких вычислительных узлах или устройствах. Целью статьи является исследование возможностей применения алгоритмов распределенной обработки для анализа текстов на основе языковых моделей. Конкретные задачи, рассматриваемые в статье, включают извлечение смысла, классификацию и другие аспекты анализа научных текстов. Для достижения поставленных целей необходимо разработать алгоритмические подходы и методы, основанные на распределенной обработке [1].

Материалы и методы. Языковые модели в виде математической модели обычно описываются с использованием вероятностной теории и статистики. Основная идея заключается в том, что каждое слово в предложении зависит от предыдущих слов и вероятности их сочетания. Одной из наиболее распространенных моделей является модель n-грамм, где n - это количество предыдущих слов, от которых зависит текущее слово. Например, в модели биграммы каждое слово зависит только от предыдущего слова. Модель языковой модели n-граммы определяет вероятности последовательности слов S, где $S = w_1, w_2, \ldots, w_n$, как произведение вероятностей отдельных слов в последовательности:

$$P(S) = P(w_1, w_2, \ldots, w_n) = P(w_n|w_1, w_2, \ldots, w_{n-1}) * P(w_{n-1}|w_1, w_2, \ldots, w_{n-2}) * \ldots * P(w_2|w_1) * P(w_1)$$

Для получения этих вероятностей можно использовать статистику, вычисленную на основе корпуса текстовых данных. Например, считая количество вхождений каждого слова и пары слов в корпусе, можно оценить вероятности $P(w_n|w_1, w_2, \ldots, w_{n-1})$ и $P(w_{n-1}|w_1, w_2, \ldots, w_{n-2})$.

Существует также другой подход к языковым моделям - рекуррентные нейронные сети (RNN). В этом случае модель языковой модели представляется в виде нейронной сети с использованием рекуррентных слоев, которые позволяют моделировать зависимости между словами в предложении. Этот подход может обрабатывать более сложные зависимости между словами, чем модели n-грамм [2-3].

Если детальнее рассматривать некоторые аспекты языковых моделей в виде математической модели, то выглядит это следующим образом.

1. Модель n-грамм:

В модели биграммы каждое слово w_i в предложении зависит только от предыдущего слова w_{i-1}. Для предсказания вероятности текущего слова, можно использовать условную вероятность $P(w_i|w_{i-1})$. Например, если хотим предсказать вероятность фразы «Я люблю КазУТБ», то используем следующую формулу:

$$P(,\ ,\ ) = P(|<\ >) * P(|) * P(|)$$

Аналогично, в модели триграммы каждое слово w_i зависит от двух предыдущих слов w_{i-2} и w_{i-1}, и для предсказания вероятности используется условная вероятность $P(w_i|w_{i-2},\ w_{i-1})$.

Оценка вероятности n-граммы может осуществляться с помощью метода максимального правдоподобия или сглаживания вероятностей для уменьшения эффекта сильной разреженности данных.

2. Рекуррентные нейронные сети (RNN):

Рекуррентные нейронные сети - это нейронные сети, в которых информация из предыдущего шага (в данном случае предыдущего слова) передается в следующий шаг. С использованием рекуррентных слоев, RNN может учитывать контекст и зависимости между словами в предложении. Модель языковой модели на основе RNN может быть представлена как последовательная модель, где каждое слово вводится в сеть на каждом шаге времени. На каждом шаге, RNN обрабатывает текущее слово w_i и скрытое состояние h_{i-1} (которое представляет предыдущий контекст) для предсказания следующего слова. Это можно записать следующим образом:

$$h_i = RNN(w_i,\ h_{i-1})$$

$$P(w_i|w_1,\ w_2,\ ...,\ w_{i-1}) = softmax(W * h_i + b)$$

где W и b - это параметры модели, которые оптимизируются в процессе обучения [4].

Рекуррентные нейронные сети имеют преимущество в обработке последовательностей переменной длины и в учете дальних зависимостей между словами.

Независимо от выбранного метода, языковые модели могут использоваться для автоматической генерации текста, оценки правильности предложений, автозаполнения текста, перевода, суммирования текстов и других задач обработки естественного языка. Они значительно улучшают качество и эффективность работы с текстовыми данными. Математические модели языковых моделей могут быть использованы для различных приложений, таких как автозаполнение или исправление опечаток в тексте, машинный перевод, распознавание речи и многое другое.

Теперь рассмотрим вопрос применения современных программных подходов для решения поставленной задачи и в качестве инструментария такой обработки выберем Apache Spark, который позволяет эффективно обрабатывать большие наборы данных на кластерах компьютеров, используя параллельные вычисления. В качестве языка программирования выберем Scala, который является одним из языков программирования, специально поддерживаемых в Apache Spark. Реккурентные нейронные сети (RNN) широко применяются в области обработки естественного языка, машинного перевода, распознавания речи и других задач машинного обучения, где учитывается последовательная природа данных. RNN обрабатывают данные пошагово, запоминая информацию о предыдущих шагах и используя ее для принятия решений на следующих шагах. Apache Spark может быть использован для обучения и применения RNN на больших объемах данных. Мощность распределенной обработки данных позволяет эффективно обрабатывать параллельное выполнение вычислений, что может быть особенно полезно при обработке больших последовательностей данных. Язык программирования Scala предоставляет гибкий и выразительный синтаксис для работы с Apache Spark. Scala позволяет использовать функциональную парадигму программирования, что облегчает разработку сложных алгоритмов и манипуляцию с данными.

Для реализации реккурентных нейронных сетей с использованием Apache Spark и Scala можно воспользоваться библиотекой Spark MLlib, которая предоставляет инструменты для обработки данных, обучения моделей и применения их на больших кластерах. Spark MLlib поддерживает различные типы моделей машинного обучения, в том числе и реккурентные нейронные сети. Пример реализации реккурентной нейронной сети с использованием Apache Spark и Scala может выглядеть следующим образом:

1. Подготовка данных: подготовьте данные, представленные последовательностями, которые будут использоваться для обучения и тестирования.

2. Создание модели: используйте библиотеку Spark MLlib для создания объекта модели реккурентной нейронной сети. Настройте архитектуру сети и параметры, такие как число скрытых слоев, размер входных и скрытых слоев и др.

3. Обучение модели: используйте метод обучения модели, предоставленный в Spark MLlib, чтобы обучить реккурентную нейронную сеть на подготовленных данных.

4. Применение модели: после обучения модель можно использовать для прогнозирования или классификации новых данных. Используйте метод применения модели, предоставленный в Spark MLlib, для получения прогнозов или классификации на новых данных.

Результаты и обсуждения. Вот пример кода на языке Scala, реализующий алгоритм линейной регрессии с использованием библиотеки Spark MLlib:

```
scala

import org.apache.spark.sql.SparkSession

import org.apache.spark.ml.feature.VectorAssembler

import org.apache.spark.ml.regression.LinearRegression

// Создание сессии Spark

val spark = SparkSession.builder()

.appName(”Linear Regression”)

.getOrCreate()

// Загрузка данных из файла в формате libsvm и создание DataFrame

val data = spark.read.format(”libsvm”).load(”path/to/data/file”)

// Создание вектора признаков

val assembler = new VectorAssembler()

.setInputCols(Array(”feature1”, ”feature2”, ...))

.setOutputCol(”features”)

val inputData = assembler.transform(data).select(”features”, ”label”)

// Разделение данных на обучающую и тестовую выборки

val Array(trainData, testData) = inputData.randomSplit(Array(0.7, 0.3))

// Создание и обучение модели линейной регрессии

val lr = new LinearRegression()

.setFeaturesCol(”features”)

.setLabelCol(”label”)

val model = lr.fit(trainData)

// Предсказание для тестовой выборки
```

```
val predictions = model.transform(testData)

// Вывод предсказаний и фактических значений

predictions.select(”prediction”, ”label”).show()

// Оценка модели с помощью среднеквадратичной ошибки

val evaluator = new org.apache.spark.ml.evaluation.RegressionEvaluator()

.setLabelCol(”label”)

.setPredictionCol(”prediction”)

.setMetricName(”mse”)

val mse = evaluator.evaluate(predictions)

println(s”Mean Squared Error = \$mse”)

// Закрытие сессии Spark

spark.stop()
```

Этот код реализует алгоритм линейной регрессии с использованием библиотеки Spark MLlib. Алгоритм используется для предсказания цены дома на основе его характеристик. Сначала создается сессия Spark. Затем данные загружаются из файла в формате libsvm и преобразуются в DataFrame. Далее векторы признаков объединяются в один вектор с помощью класса VectorAssembler. Данные разделены на обучающую и тестовую выборки с соотношением 70% на 30%. Затем создается объект LinearRegression, указываются входные и выходные колонки, и модель обучается на обучающей выборке. Предсказания для тестовой выборки делаются с помощью метода transform. Результаты предсказаний и фактических значений выводятся на экран. Модель оценивается среднеквадратичной ошибкой, которая вычисляется с помощью model.summary.meanSquaredError.

Алгоритм линейной регрессии является одним из самых простых и широко используемых алгоритмов машинного обучения. Его эффективность зависит от нескольких факторов:

1. Размер и качество данных: Чем больше данных у вас есть для обучения модели, тем более точные и надежные будут ее прогнозы. Кроме того, данные должны быть репрезентативными и хорошо представлять изучаемую проблему.

2. Зависимость между предикторами и целевой переменной: Линейная регрессия предполагает линейную зависимость между предикторами и целевой переменной. Если данные не подчиняются этому предположению, то линейная регрессия может быть неэффективной.

3. Предобработка данных: Важно провести предварительную обработку данных, чтобы устранить выбросы, заполнить пропущенные значения, нормализовать признаки и т. д. Чистые и хорошо подготовленные данные могут существенно улучшить эффективность алгоритма.

4. Выбор подходящей модели: Линейная регрессия является простой моделью и может не отражать сложности и нелинейные зависимости в данных. В некоторых случаях может потребоваться использование более сложных моделей, таких как градиентный бустинг или нейронные сети, для достижения лучшей эффективности.

5. Параметры модели: Некоторые гиперпараметры модели, такие как регуляризация и скорость обучения, могут повлиять на ее эффективность. Подбор оптимальных значений параметров может помочь улучшить производительность модели.

В целом, линейная регрессия может быть эффективной моделью, если данные подходят для линейной модели и, если все факторы, перечисленные вы-

ше, оптимизированы.

Выводы. В данной статье был предложен и исследован подход к анализу научных текстов на основе языковых моделей и алгоритмов распределенной обработки. Было проведено исследование эффективности и точности данного подхода. Исследованные математические подходы и обзор программного обеспечения показали, что использование языковых моделей и алгоритмов распределенной обработки позволяет улучшить точность анализа научных текстов. Этот подход позволяет обрабатывать большие объемы данных и получать более точные и надежные результаты. Практическая значимость заключается в том, что разработанный подход может быть использован в различных областях, связанных с анализом научных текстов, например, в машинном обучении, информационном поиске, анализе данных и других. Дальнейшие исследования могут быть направлены на совершенствование алгоритмов и методов анализа научных текстов, а также на расширение области применения данного подхода. Также можно провести сравнительное исследование различных моделей и алгоритмов для более полного понимания их эффективности и применимости.

Научно-исследовательская работа выполняется в рамках ГФ Министерством науки и высшего образования Республики Казахстан AR19677733 по теме «Разработка интеллектуальной распределенной системы параллельного анализа научных текстов» на 2023-2025 гг.

Литература

1.Boranbayev, A., Shuitenov, G., Boranbayev, S. The Method of Analysis of Data from Social Networks Using Rapidminer.- Advances in Intelligent Systems and Computing. - 2020. - 1229 AISC. - pp. 667-673.

3.Сухов, К. Node.js. Путеводитель по технологии / К. Сухов. - М.: ДМК Пресс, 2015. - 416 с.

4.Одерски М., Спун Л., Веннерс Б. Scala. Профессиональное программирование - Programming in Scala: Updated for Scala 2.12. - СПб: Питер, 2018. - 688 с.

5.Прокопец А. Конкурентное программирование на SCALA. - М.: ДМК Пресс, 2017. - 342 с.

6.Хостманн, К. Функциональное программирование. SCALA для нетерпеливых/К. Хостманн. - М.: ДМК, 2015. - 408 с.

7.Dean Wampler, Alex Payne. Programming Scala: Scalability- Functional Programming + Objects -1st. - O'Reilly Media, 2009. - p. 448.

8.Применяем Apache POI, docx4j и springframework.jdbc -https://habr.com/ru/articles/214435/.- Дата обращения: 27.09.2023

References

1. Boranbayev, A., Shuitenov, G., Boranbayev, S. The Method of Analysis of Data from Social Networks Using Rapidminer.-Advances in Intelligent Systems and Computing.-2020. - 1229 AISC.- pp. 667-673.

2. Kantelon, M. Node.js v dejstvii / M. Kantelon. -SPb: Piter.- 2015.-810 s.

3. Sukhov, K. Node.js. Putevoditel' po tekhnologii / K. Sukhov. - M.: DMK Press, 2015. - 416 s.

4. Oderski M., Spun L., Venners B. Scala. Professional'noe programmirovanie - Programming in Scala: Updated for Scala 2.12. - SPb: Piter.-2018. - 688 s.

5. Prokopecz A. Konkurentnoe programmirovanie na SCALA.- M.: DMK Press, 2017. - 342 s.

6. Khostmann, K. Funkczional'noe programmirovanie. SCALA dlya neterpelivy'kh/K. Khostmann. - M.: DMK, 2015. - 408 s.

7. Dean Wampler, Alex Payne. Programming Scala: Scalability- Functional Programming + Objects -1st. - O'Reilly Media, 2009. - p. 448.

8.Primenyaem Apache POI, docx4j i springframework.jdbc - https://habr.com/ru/articles/214435

/. - Data obrashheniya: 27.09.2023

Сведения об авторах

ШуйтеновГ.Ж. -к.п.н.,Esil University, Республика Казахстан, г. Астана, e-mail: g.shuitenov@mail.ru

Турусбекова У.К.- PhD, и.о. доцента, Esil University, Астана, Казахстан, e-mail: umut.t@mail.ru

Муратбеков М.М.- к.ф.-м.н., PhD, Евразийский национальный университет им. Л. Н. Гумилева, Астана, Казахстан, e-mail: madimm@list.ru

Information about the authors

Shuitenov G.Zh.- Candidate of Pedagogical Sciences,Esil University, Astana, Kazakhstan, e-mail: g.shuitenov@mail.ru

Turusbekova U. K. - PhD, Acting Associate Professor, Esil University, Astana, Kazakhstan, e-mail: umut.t@mail.ru

Muratbekov M.M.- c.ph.-math.sc., PhD, Acting Associate Professor, L.N. Gumilyov Eurasian National University, Astana, Kazakhstan, e-mail: madimm@list.ru