Применение методов машинного обучения для обеспечения качества спецификаций требований

А.Д. Белоногова, П.А. Огнянович, К.И. Гайдамака

***Аннотация* — Данная статья посвящена проблеме обеспечения качества спецификаций требований для сложных технических систем. Целью данной статьи является оценка применимости нейронных сетей, методов классификации и кластеризации для проверки спецификаций требований на непротиворечивость и атомарность. Предполагается, что использование нейронных сетей позволит получить векторное представление текстовых формулировок требований для последующего определения противоречий в спецификациях и проверки атомарности отдельных требований. В настоящей статье демонстрируется применение следующих методов обработки естественного языка: fasttext, doc2vec и BERT. Для нахождения противоречий в спецификациях требований используется кластеризация методом k средних исходя из предположения, что попадание требований в один кластер сигнализирует о возможном противоречии. Проверка требований на атомарность осуществляется с использованием градиентного бустинга над решающими деревьями. Исследование показало, что использование предобученной нейронной сети BERT дает наилучшие векторные представления требований для решения задач кластеризации и классификации с использованием k-Means и градиентного бустинга соответственно. С другой стороны продемонстрировано, что обучение модели doc2vec на спецификации является нецелесообразным, принимая во внимание тот факт, что количество требований в спецификациях обычно ограничено и недостаточно для обучения, а fasttext не позволяет учесть семантику полной формулировки требования. В заключении приводятся сравнение результатов рассмотренных в статье методов обработки естественного языка.**

***Ключевые слова* — BERT, инженерия требований, машинное обучение, обработка естественного языка.**

I. Введение

Разработка качественных системных требований занимает определяющую, ключевую роль в процессе работы над проектом вне зависимости от его масштаба. Сроки реализации, стоимость разработки и прочие показатели эффективности проекта напрямую зависят от того, насколько точно и правильно системные требования описывают систему, которую необходимо создать. Принимая во внимание тот факт, насколько сложными, масштабными могут быть проекты, логично будет утверждать, что количество требований в них может исчисляться несколькими тысячами. И поскольку разработка требований целиком и полностью лежит на плечах инженеров по требованиям, логично будет предположить также и то, что она не может не зависеть от человеческого фактора: невнимательности, небрежности или спешки. Но в случаях, когда на кону стоят огромные денежные суммы, а из-за несовершенства человеческого ресурса исполнитель проекта может понести существенные потери, необходимо минимизировать человеческий фактор.

Требования позволяют определить, что хотят получить от создаваемой системы заинтересованные стороны и какими свойствами должна обладать система для удовлетворения их запросов [1].

О качестве требований можно судить по наличию у них следующих свойств [2]:

1. *атомарность* – каждая формулировка требования должна являться одним элементом иерархии требований, и последующее деление предложения на два и более требования не представляется возможным;
2. *уникальность* – каждое требование должно обладать уникальным идентификатором, что позволит отличать его от прочих;
3. *выполнимость* – должна присутствовать возможность в указанные сроки и при указанном бюджете реализовать требование;
4. *законность* – требование не должно противоречить действующим законам;
5. *ясность* – в требовании должна отсутствовать двусмысленность;
6. *точность* – требование должно быть лаконичным;
7. *проверяемость* – должна присутствовать стратегия по проверке того, что требование реализовано;
8. *абстрактность* – требование не должно определять реализацию.

Касательно набора требований имеются другие

Статья получена 12 июня 2021.

Белоногова Алена Дмитриевна, Национальный Исследовательский Ядерный Университет МИФИ, магистрант, alena.belonogova@yandex.ru

Огнянович Павел Александрович, Национальный Исследовательский Ядерный Университет МИФИ, магистрант, pasha2la71@gmail.com

Гайдамака Кирилл Игоревич, РТУ МИРЭА, аспирант, gaydamaka@ians.aero

критерии [2]:

1. *полнота* – требования охватывают весь функционал системы;
2. *непротиворечивость* – требования, противоречащие друг другу, отсутствуют;
3. *отсутствие избыточности* – повторы в требованиях отсутствуют;
4. *модульность* – требования, близкие по смыслу, находятся в одной категории;
5. *структурированность* - в спецификации требований прослеживается четкая структура;
6. *удовлетворенность* – все требования с отметкой «удовлетворяется посредством» ссылаются на существующие требования;
7. *тестируемость* – требования покрыты тестами на желаемом уровне.

В данной работе осуществляется применение методов машинного обучения для автоматизации проверки спецификаций требований на непротиворечивость и отдельных требований на атомарность.

II. Описание набора исходных данных

Исходные данные представляют из себя таблицу из 287 записей со следующими колонками: «ID», «Requirement» и «Singular», где «ID» - уникальный идентификационный номер требования, «Requirement» - текстовая формулировка требования, а «Singular» - метка принадлежности требования к классу атомарных или не атомарных (1 и 0 соответственно). Столбец «Singular» необходим только для осуществления проверки требований на атомарность.

III. Формирование признаков

Поскольку методы машинного обучения работают с объектами, представленными в виде числовых векторов, первостепенной задачей является преобразование текстовых формулировок требований в вектора, что само по себе не тривиально. Рассмотрим три модели обработки естественного языка, базирующихся на нейронных сетях - doc2vec, fasttext, BERT.

Подход doc2vec базируется на модели Google word2vec [3], которая позволяет трансформировать слова в пространство числовых векторов с учетом семантики слов. Существуют два принципиально разных подхода к Word2Vec – CBOW и skip-gram. Метод CBOW (Continuous bag of words) позволяет предсказать слова на основе контекста. Метод skip-gram, в другом случае, позволяет предсказать контекст по одному слову. Результатом работы любого из алгоритмов Word2Vec являются N-мерные вектора, определяющие слова предложения. Подход Doc2Vec можно описать следующим образом - помимо того, чтобы использовать только вектора слов для предсказания другого слова, мы дополнительно вводим вектор предложения, являющийся уникальным для каждого предложения. Таким образом, результатом обучения нейронной сети является вектор предложения, искомый нами. Как и в случае Word2Vec, существуют два подхода к определению вектора предложения – DM (distributed memory) и DBOW (distributed bag of words). DM позволяет определять слово из контекста, DBOW – контекст из вектора предложения.

FastText – разработанная в Facebook библиотека, содержащая предобученные готовые векторные представления слов [4]. Алгоритм использует одновременно модели CBOW и skipgram. Для нас основной интерес представляет то, что FastText не требует обучения. Существуют готовые модели для 157 языков, в том числе и для русского. По умолчанию FastText использует вектора слов размерностью 300, но при желании ее можно поменять. Однако, FastText преобразует в вектор слово, а у нас в качестве исходных данных текст требования. Основная идея использования FastText для преобразования текстовой формулировки требования в векторное представление заключается в том, что требование очищается от всех специальных символов и знаков, токенизируется и лемматизируется [5], после чего список токенов подается на вход модели FastText. Далее осуществляется усреднение значений полученного набора векторных представлений отдельных слов формулировки требования.

Модель BERT была анонсирована и выложена в общий доступ в 2008 году [6]. Изначально модель BERT обучается на огромном количестве текста, взятого из книг, Википедии и т.д., причем на нескольких языках. В основе данной технологии лежат такие алгоритмы, разработанные NLP-сообществом, как обучение с частичным привлечением учителя, модели ELMo (Embeddings from Language Models - вложения из языковых моделей), ULMFiT (Universal Language Model Fine-Tuning – настройка универсальной языковой модели), OpenAI Transformer и другие.

В основе BERT лежит Трансформер, по сути, BERT является обученным стеком из энкодеров Трансформера. Энкодеры – это одинаковые по своей структуре объекты, однако имеющие разные веса. Каждый энкодер состоит из двух частей – слой внутреннего понимания и слой распространения. Энкодер получает на вход числовые вектора, полученные из слов, обрабатывает вектора на уровне слоя внутреннего понимания и передает в нейронную сеть прямого распространения. Результат выхода одного уровня энкодера является входными данными для следующего энкодера. Энкодеры образуют стек из 12 (24 для глубоко обученных моделей) элементов, стек энкодеров и является основой BERT, как уже было сказано. Каждое отправленное слово в BERT проходит через цепочку энкодеров и в результате для каждого слова формируется числовой вектор размерности 768 для базовой реализации BERT. Полученные векторы и являются искомыми в рамках этой задачи. Далее возможно производить различные операции с ними: кластеризовать, использовать как входные данные в алгоритмах машинного обучения с учителем, например, для задачи логистической регрессии.

IV. Проверка на непротиворечивость

Наличие даже одного противоречия в наборе требований говорит о том, что спецификация – некачественная, и что при дальнейшем проектировании и реализации системы, скорее всего, возникнут проблемы. Поэтому единственным решением является своевременное исправление ошибки на самых ранних этапах развития проекта. Прежде всего, для того, чтобы внести изменения в неверную формулировку, необходимо обнаружить проблему. Однако, когда спецификация требований исчисляется, в случае крупномасштабных проектов десятком тысяч требований, проблема заключается именно в поиске ошибки. В таком случае привлекается экспертная команда, которая проводит повторную проверку формулировок требований, что может отнять значительное количество ресурсов, даже не принимая во внимание человеческий фактор (т. е. что ошибка не обнаружится и при повторной проверке). Поэтому, в данном случае имеет смысл создания интеллектуального метода, который поможет команде системных инженеров в процессах управления требованиями. Этот метод должен определять ближайшие по смыслу требования к заданному, и таким образом вопрос «противоречит ли данное требование спецификации?» потребует повторной проверки требований внутри группы предложений, близких друг другу по смыслу, что в разы сокращает время на выполнение описанной задачи. А процесс группирования объектов таким образом, что похожие объекты оказываются в одной группе, а разные – в разных, является задачей кластеризации.

При решении задачи кластеризации у нас имеется N объектов - x1, …, xN, каждый из которых должен принадлежать к какому-то классу, а также метки классов y1, …, yM, $M \leq N$. Задача кластеризации заключается в том, чтобы правильно расставить метки на объектах таким образом, чтобы похожие объекты друг на друга попали в один класс, а объекты, кардинально отличающиеся – в разные классы. Самый простой способ кластеризации – метод k-средних [7]. Объекты, которые мы кластеризуем, являются точками в N-мерном пространстве. Вначале необходимо задать число k – число кластеров. Потом из всего множества точек необходимо случайным образом выбрать k штук – это будут центры кластеров. Затем измеряются расстояния каждой точки для каждого центра, и точка определяется в тот кластер, к центру которого она находится ближе всех. Затем центры кластеров пересчитываются – для каждого полученного кластера определяется его новый центр, после чего операция повторяется до тех пор, пока изменение положения центра кластера не будет существенно большим (меньше некоторого порогового значения). Это метод K-means имени Болла Холла. В реализации МакКина центры кластеров пересчитываются при каждом переходе объекта. Чтобы не считать среднее арифметическое по всем объектам кластера, из него берется произвольная выборка, и центр считается по ней – это вариант реализации Mini Batch. Есть еще одна реализация K-means – k++. В ней изначальное приближение выбирается не случайным образом, а с применением равномерного распределения. И каждое следующее приближение центра кластера устанавливается как наиболее правдоподобное в данной ситуации.

Решено было применить методы doc2vec, BERT для определения противоречий в спецификации требований. В качестве тестовых данных использовалась спецификация требований из 282-х требований, причем несколько из них были добавлены искусственно для создания противоречий, которые необходимо будет отыскать нашим алгоритмам. Поэтому были созданы две группы противоречий по три требования в каждом. Первая группа противоречит в части цены деления таймера, а вторая – в части погрешностей:

R-0224: Цена деления таймера должна составлять 1 минуту

R-0278: Цена деления шкалы таймера 1 мин

R-0282: Цена деления шкалы таймера 2 мин

R-0238: Погрешность таймера должна составлять 0,5 мин.

R-0221: Погрешность таймера должна составлять 1 мин.

R-0277: Таймер должен производить отсчет времени с погрешностью не более 0.5 мин

Применение doc2vec не демонстрирует приемлемых результатов. Вне зависимости от того, производить ли лемматизацию над предложениями или нет, выставлять ли метод распределенной памяти или сумки слов, результаты оставляют желать лучшего. Приведем здесь один вариант – DBOW. Был дважды произведен поиск противоречий – к требованию R-0224, R-0238. Результатом стали кластеры из 15-ти и 9-ти требований соответственно. Несмотря на то, что оба кластера содержат информацию о таймере, в первом случае у нас имеется ни одного выявленного противоречия и 14 требований, не противоречащих заданному, однако помеченных, как потенциально противоречащих. Во втором случае картина несколько лучше – два противоречия всё-таки найдены, но ошибки по-прежнему значительны (7 ложных противоречий). Это связано с тем, что мы используем необученную модель doc2vec. Прежде ее нужно обучить на наших требованиях, а 280 штук – ничтожно мало. Поэтому модели не достает информации для того, чтобы построить словарь. Но нельзя здесь оперировать лишь тем, что спецификация недостаточно объемная. Бывают и небольшие проекты, и в таких случаях количество требований исчисляется несколькими сотнями, а значит алгоритм не будет работать. Нам необходимо брать уже обученную модель для наших вычислений. Однако у данной ситуации есть обратная сторона - обучение doc2vec на большом количестве отраслевых текстов, возможно, позволит достичь лучших результатов, но этот вопрос требует дальнейшего исследования.

Поиск кластеров с применением метода BERT приносит

впечатляющие результаты (рис. 1): обе группы оказываются в одном кластере и в полном объеме. Этому алгоритму удалось определить, как простые противоречия, связанные с изменением одной цифры в тексте требования, так и переформулировки. Это первый алгоритм, определивший 6 противоречий из 6 лишь с одной ошибкой – BERT добавил в кластер требование R-0209, которое не противоречит остальным.

	ID	Requirement
206	R-0209	Длительность звукового сигнала должна составля...
218	R-0221	Погрешность таймера должна составлять 1 мин.
221	R-0224	Цена деления таймера должна составлять 1 минуту
235	R-0238	Погрешность таймера должна составлять 0.5 мин.
274	R-0277	Таймер должен производить отсчет времени с пог...
275	R-0278	Цена деления шкалы таймера 1 мин
279	R-0282	Цена деления шкалы таймера 2 мин

Рисунок 1: кластер противоречий BERT.

V. Проверка на атомарность

В общем случае задачу классификации можно описать следующим образом: имеется некоторая выборка объектов, для которых определена принадлежность к классу [8, 9]. Множество классов конечное и известное. Требуется реализовать алгоритм, который будет способен определять классовую принадлежность для всего множества объектов.

Задача определения атомарности формулировки требования может быть сформулирована как задача классификации: множество классов представлено двумя элементами: атомарно, не атомарно. Адаптируя формулировку задачи классификации к рассматриваемой проблеме, получаем следующую постановку: необходимо построить алгоритм, который будет осуществлять бинарную классификацию требования.

Поскольку задача классификации является частным случаем обучения с учителем, то возникает необходимость составления и разметки обучающей выборки. Для этого была сформирована и размечена спецификация требований из 287 элементов. Соотношение элементов разных классов в выборке – 43 на 57, что позволяет утверждать о ее сбалансированности. Спецификация требований была поделена в соотношении 3 к 7 – на тестовую и тренировочную выборки соответственно.

Рассмотрим подробнее градиентный бустинг. Он представляет собой композицию из N базовых алгоритмов, построенных направлено [10]. Они строятся последовательно один за другим, чтобы i-ый алгоритм исправлял ошибки композиции предыдущих i-1 штук. Из-за такого способа построения композиции становится возможным использование сравнительно простых базовых алгоритмов, например неглубоких деревьев. Иногда за первый базовый алгоритм можно принять константный классификатор, который относит все объекты к одному классу.

Применение FastText в качестве метода предобработки данных для градиентного бустинга привело к получению следующих результатов: точность 81% и площадь под ROC кривой 0.803. График ROC кривой представлен на рисунке 2.

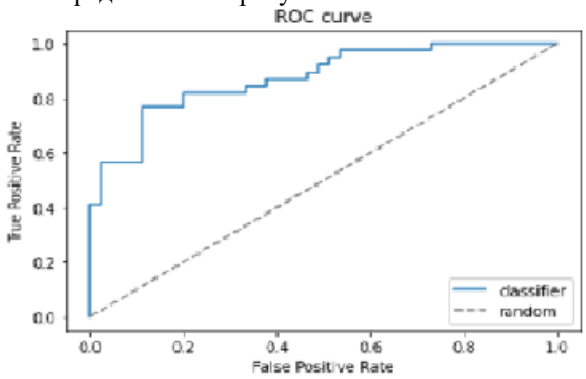

Рисунок 2: ROC кривая FastText.

Предобработка формулировок требований с использованием BERT позволила получить более высокие оценки на тестовой выборке. Результатом стали точность на уровне 88% в сочетании с площадью под ROC кривой в 0.874, что отражено на рисунке 3

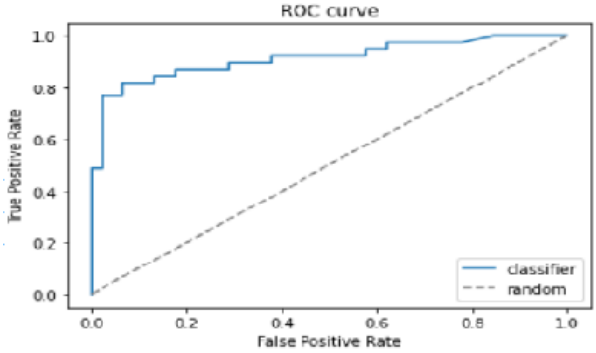

Рисунок 3: ROC кривая BERT.

Таким образом, полученные результаты позволяют сделать предположение о том, что особенности BERT, а именно – учитывание семантики при формировании векторного представления текста требования, положительно сказывается на точности итогового сочетания метода предобработки исходных данных и алгоритма классификации. Однако есть предположение о том, что вручную сформированное пространство признаков, которое не учитывает семантику и основано на синтаксических, пунктуационных особенностях предложения, позволит достичь лучших показателей точности при определении атомарности требований.

VI. Заключение

В работе рассматривалось применение методов обработки естественного языка doc2vec, FastText и BERT в контексте проверки спецификаций требований на непротиворечивость и отдельных требований на атомарность. Исследование показало, что спецификаций требований сравнительного небольшого объема

недостаточно для обучения модели doc2vec с целью получения удовлетворительных результатов при решении задачи нахождения противоречий в наборах требований. Использование предобученной модели BERT хорошо показало себя при в данной задаче и позволило определить переформулирования в тестовой выборке. Кроме того, преобразование с помощью FastText показало себя несколько хуже при проверке требований на атомарность, чем BERT: 81% точности против 88% соответственно. Это может быть связано с тем, что BERT учитывает семантику текста целиком, в отличие от FastText. Таким образом, можно сделать вывод о том, что модели, основанные на нейронных сетях, могут быть применены для улучшения качества спецификаций требований. В дальнейшем планируется проверка требований на атомарность с использованием пространства признаков сформированного с учетом синтаксических и пунктуационных особенности формулировок требований, и поиск противоречий в требованиях с doc2vec, обученного на большом количестве отраслевых текстов

Библиография

[1] Требования [электронный ресурс]: System Engineering Thinking Wiki. - Режим доступа: http://sewiki.ru/Категория:Требования (дата обращения: 14.03.2021г.)

[2] Hall E. Requirements Engineering. US, Kent, Gray Publishing, 2005

[3] A gentle introduction to Doc2Vec [электронный ресурс]: Medium. - Режим доступа: https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e (дата обращения: 12.04.2021г.)

[4] fastText: Library for efficient text classification and representation learning [электронный ресурс] fastText – Режим доступа: https://fasttext.cc/. (Дата обращения: 23.04. 2021г.)

[5] Data Preparation: полет нормальный – что такое нормализация данных и зачем она нужна [электронный ресурс]: Школа больших данных. - Режим доступа: https://www.bigdataschool.ru/blog/нормализация-feature-transformation-data-preparation.html (дата обращения: 10.05.2021г.)

[6] BERT, ELMO и Ко в картинках (как в NLP пришло трансферное обучение) [электронный ресурс]: Habr. - Режим доступа: https://habr.com/ru/post/487358/ (дата обращения: 24.04.2021г.)

[7] Московский физико-технический институт Яндекс & E-Learning Development Fund Поиск структуры в данных [электронный ресурс]: Coursera. – Режим доступа: https://www.coursera.org/learn/unsupervised-learning/home/welcome (дата обращения: 25.04.2021г.)

[8] Задача классификации [электронный ресурс] Википедия – Режим доступа: https://ru.wikipedia.org/wiki3адача_классификации. (Дата обращения: 01.05.2021г.)

[9] Классификация [электронный ресурс] MachineLearning – Режим доступа: http://www.machinelearning.ru/wiki/index.php?title=Классификация. (Дата обращения: 01.05.2021г.)

[10] Бустинг [электронный ресурс] Википедия – Режим доступа: https://ru.wikipedia.org/wiki/Бустинг. (Дата обращения: 08.05.2021г.)

Applying machine learning algorithms to provide quality requirements specification

A.D. Belonogova, P.A. Ognyanovich, K.I. Gaydamaka

***Annotation* — This article is devoted to the problem of ensuring the quality of requirements specifications for complex technical systems. The purpose of this article is to use neural networks, classification and clustering algorithms to check requirements specifications for consistency and atomicity. It is believed that the use of neural networks will provide a vector representation of textual requirements formulations in order to identify inconsistencies in requirements specifications and to check the atomicity of individual requirements. This article demonstrates the use of such natural language processing techniques as fasttext, doc2vec, and BERT. K-means clustering is used to find inconsistencies in requirements specifications based on the assumption that the requirements of one cluster are potentially conflicting. Requirements are checked for atomicity by using gradient boosting over decision trees. The study showed that using the pretrained BERT neural network gives the best vector representations of requirements for solving clustering and classification problems using k-Means and gradient boosting, respectively. In addition, training the doc2vec model on requirements specifications is impractical, because the number of requirements in the specifications is usually limited and not enough for training, and FastText does not consider the semantics of the full requirement statement. In conclusion, a comparison of the results of the natural language processing methods considered in the article is given.**

***Keywords* - BERT, machine learning, natural language processing, requirements engineering.**

REFERENCES

[1] Requirements [electronic resource]: System Engineering Thinking Wiki. - Access mode: http://sewiki.ru/Категория:Требования (date of visit: 14.03.2021г.)
[2] Hall E. Requirements Engineering. US, Kent, Gray Publishing, 2005
[3] A gentle introduction to Doc2Vec [electronic resource]: Medium. - Access mode: https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e (date of visit: 12.04.2021г.)
[4] fastText: Library for efficient text classification and representation learning [electronic resource] fastText – Access mode: https://fasttext.cc/. (date of visit: 23.04. 2021г.)
[5] Data Preparation: normal flight - what is data normalization and why is it needed [electronic resource]: School of Big Data. - Access mode: https://www.bigdataschool.ru/blog/нормализация-feature-transformation-data-preparation.html (date of visit: 10.05.2021г.)
[6] BERT, ELMO and Co in pictures (how transfer learning came to NLP) [electronic resource]: Habr. - Access mode: https://habr.com/ru/post/487358/ (date of visit: 24.04.2021г.)
[7] MIPT Yandex & E-Learning Development Fund Finding a structure in the data [electronic resource]: Coursera. – Access mode: https://www.coursera.org/learn/unsupervised-learning/home/welcome (date of visit: 25.04.2021г.)
[8] The task of classification [electronic resource] Wikipedia – Access mode: https://ru.wikipedia.org/wiki3адача_классификации. (date of visit: 01.05.2021г.)
[9] Classification [electronic resource] MachineLearning – Access mode: http://www.machinelearning.ru/wiki/index.php?title=Классификация. (date of visit: 01.05.2021г.)
[10] Boosting [electronic resource] Wikipedia – Access mode: https://ru.wikipedia.org/wiki/Бустинг. (date of visit: 08.05.2021г.)