DOI 10.37882/2223-2966.2024.4-2.15

ПРИМЕНЕНИЕ НЕЙРОСЕТЕВЫХ АЛГОРИТМОВ И СТАТИСТИЧЕСКИХ МОДЕЛЕЙ ПРОГНОЗИРОВАНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕССОВ УПРАВЛЕНИЯ ПРОЕКТАМИ

APPLICATION OF NEURAL NETWORK ALGORITHMS AND STATISTICAL FORECASTING MODELS FOR AUTOMATION OF PROJECT MANAGEMENT PROCESSES

S. Efimtsev
M. Zhurankov
D. Melnikov

Summary. In the modern world, the process of managing technological projects is actively developing: new methodologies are being used, existing management models are being updated, new tools are being introduced to organize and monitor the work of employees of various companies. One of these tools is a neural network, a software based on mathematical models simulating the functioning of biological networks of nerve cells. Thanks to them and a voluminous knowledge base, it is possible to bring machine text answers as close as possible to human-like ones. Thus, with sufficient initial data, it is possible to train an algorithmic model to solve simple control tasks. This article examines and analyzes the subject area, designs and prototypes a neuro-network machine learning model in the Python 3 programming language to automate the processes of distribution and decomposition of work tasks between employees of the information project development team.

Keywords: project management, tasks allocation, machine learning, Python, neural network, text classification, logistic regression.

Ефимцев Станислав Максимович
МИРЭА — Российский технологический университет (г. Москва)
evencofr@gmail.com
Журанков Максим Александрович
МИРЭА — Российский технологический университет (г. Москва)
astronmax@yandex.ru
Мельников Денис Александрович
аспирант МИРЭА — Российский технологический университет (г. Москва),
dmbox2019@gmail.com

Аннотация. В современном мире процесс управления технологическими проектами активно развивается: используются новые методологии, дорабатываются уже существующие модели управления, внедряются новые инструменты для организации и регулирования действий сотрудников различных компаний. Одним из таких инструментов является нейронная сеть — программное обеспечение, построенное на математических моделях, симулирующих функционирование биологических сетей нервных клеток. Благодаря им и объёмной базе знаний возможно максимально приблизить машинные текстовые ответы к человекоподобным. Таким образом, при достаточных исходных данных, можно обучить алгоритмическую модель решать несложные задачи управления. В данной статье рассматривается и анализируется предметная область, проектируется и прототипируется нейро-сетевая модель машинного обучения на языке программирования Python 3 для автоматизации процессов распределения и декомпозиции рабочих задач между сотрудниками команды разработки информационных проектов.

Ключевые слова: управление проектами, распределение задач, машинное обучение, Python, нейронная сеть, классификация текстов, логистическая регрессия.

Введение

В наши дни разработка качественных программных продуктов невозможна без четко структурированной команды разработчиков, возглавляемую проектным менеджером. Он отвечает за все аспекты жизненного цикла проекта, начиная с исследования рынка и заканчивая разработкой стратегии развития проекта [1]. Развитие проекта подразумевает также его управление.

Управление проектами — это методология проектной и предпринимательской деятельности, содержащая в себе постановку целей и их достижение путём решения определённых задач. Без него не обходится ни одна реализация проекта [2].

Специалист, занятый в этой области, называется проектным менеджером.

В данной статье будет рассмотрена обязанность проектного менеджера по распределению задач внутри команды, а именно автоматизация данного процесса.

Традиционно распределение задач и обязанностей происходит с помощью составления таблиц ответственности, например, по методике RACI [1].

В данной методике сначала составляется таблица, где по вертикали указываются должности, имена, а по горизонтали — наименования задач. На пересечении столбцов и строк указывается тип роли участника.

Однако, при использовании такой методики и аналогичных ей, все действия по распределению задач производятся вручную. Они занимают достаточно много времени и сил. Расход физических и интеллектуальных ресурсов на простые и неприоритетные задачи является особенно нерациональным.

С этой проблемой может помочь справиться специально обученный алгоритм, реализованный на базе нейронной сети. Те задачи, что не требуют срочного выполнения, могут быть автоматически назначены какому-либо сотруднику, исходя из его компетенций, загруженности и других параметров. При ошибочном распределении команда не потерпит сильного ущерба, так как задачи не являются первоочередными к выполнению.

Исследование проблемы распределения задач

Грамотное планирование работы крайне необходимо для достижения положительных результатов. Однако, наблюдая за работой проектных менеджеров и их непосредственных подчинённых, можно заметить, что большое количество времени тратится именно на проведение совещаний, направленных на дальнейшую оптимизацию работы. Избежать этого невозможно, так как иначе на фактическое выполнение задач будет уходить ещё больше времени.

Сервис reclaim.ai проводил исследование тенденций управления задачами [3]. В нём отражена статистика траты времени проектных менеджеров и индивидуальных участников на распределение и выполнение задач. Для составления статистики были опрошены более 2000 специалистов. Необходимо было выяснить, как менеджеры и члены команд расставляют приоритеты и используют своё время для целенаправленной работы в течение рабочих недель, используя различные платформы управления проектами.

Согласно результатам исследования, только 12,4 % индивидуальных участников могут посвящать более 6 часов в день фактической работе над задачами, и только 53,3 % времени работы над задачами фактически тратятся на продуктивную работу. Среднестатистический менеджер также сообщил, что хочет каждую неделю выделять дополнительно 9,4 часа каждому члену команды на выполнение задач, освобождая их от отвлекающих совещаний и неприоритетных обязательств.

Анализ планирования статистики для независимых участников привёл к следующим выводам:

— Только 53,5 % запланированных задач выполняются во время еженедельного спринта (короткого временного интервала, в течение которого команда выполняет заданный объем работы);
— 16,8 % команд каждую неделю выполняют более 70 % запланированных задач;
— 31,3 % команд выполняют менее 40 % запланированных на неделю задач.

Предполагаемая причина таких показателей — большой расход времени на распределение задач и определение стратегии их выполнения.

Также были получены следующие результаты исследования распределения времени проектных менеджеров:

— Средний менеджер тратит 5 часов в неделю на распределение, расстановку и изменение приоритетов работы своей команды;
— 20,4 % менеджеров тратят более 8 часов в неделю на распределение, расстановку и изменение приоритетов задач для своей команды;
— 26,9 % менеджеров в среднем работают над своими задачами более 4 часов в день.

Распределение задач у четверти менеджеров занимает более 4 часов рабочего времени. При этом большинство команд выполняет только половину запланированных на неделю задач.

Таким образом, автоматизация процесса планирования и распределения задач позволила бы сэкономить значительную часть времени проектных менеджеров.

Предложение по решению проблемы

Проблему планирования и распределения задач можно свести к классификации текстов по типам. Под текстом подразумевается задача, а типами являются должности сотрудников. Решением подобных проблем занимается раздел дисциплины машинного обучения NLP (англ. Natural Language Processing, обработка естественных языков). Обучение нейронных сетей классификации небольших текстов является типичной задачей NLP [4]. Одним из множества алгоритмов решения данной задачи может стать метод логистической регрессии. Его суть заключается в предсказании вероятностей принадлежности объекта к тому или иному классу с помощью разделения этих данных гиперплоскостью [5]. Сочетая этот алгоритм с методами обработки естественных языков можно свести планирование и распределение задач к простой классификации.

Ввиду того, что подаваемые на вход текстовые сообщения будут иметь относительно небольшой размер, в среднем около 15 слов, то реализация свёрточной нейронной сети позволит получить более точный результат классификации. Сверточные сети — это тип нейронных сетей прямого распространения, когда сигнал идет последовательно по нейронам (от первого слоя к последнему) [6].

Её создание является более трудоёмким процессом, чем реализация математической модели, выполняющей заданный алгоритм, поэтому такая нейросеть может быть полезна при более детальной и масштабной разработке технического решения по распределению задач.

Для обучения нейронной сети необходимо собрать размеченный набор данных, содержащий формулировки задач и сотрудников, ответственных за их выполнение.

Такой размеченный набор данных может представлять собой внутреннюю базу знаний — платформу, где структурировано и логично хранятся все рабочие инструкции и материалы об услугах, процессах, отделах, всего, что связано с компанией, доступ к которой имеет каждый сотрудник [7].

Из базы знаний можно выделить необходимую информацию о задачах, которые решают сотрудники. Каждая задача может иметь такие поля, как название, описание, категория, исполнитель, статус, срок исполнения, степень готовности и другие.

Полученную информацию следует собрать в отдельной структуре данных, на основе которой будет обучаться реализуемая система распределения задач.

После структурирования необходимо разметить задачи по классам для обучения системы распределения — составить тренировочную выборку [5]. Эту задачу можно поручить как самим сотрудникам компании, так и передать на основе договора другой компании, компетентной в данном вопросе.

Классификация может быть следующей: всего n классов, где каждый класс обозначает определенный тип задач, например, класс 0 — творческая задача, класс 1 — backend-задача, класс 2 — frontend-задача и так далее.

После составления тренировочной выборки к системе нужно применить методы обучения нейронной сети.

Как только система обучена, можно приступать к классификации. Она реализуется с помощью описанного выше алгоритма логистической регрессии. Также можно использовать другие алгоритмы классификации [8].

Также важно рассмотреть процесс декомпозиции задач, на который затрачивается достаточно большое количество рабочего времени. Задача декомпозиции сложнее, чем категоризация текстов. Для её решения, как правило, используются мультимодальные большие языковые модели [9], например GPT (англ. Generative pre-trained transformer, тип нейронных языковых моделей, которые обучаются на больших наборах текстовых данных, чтобы генерировать осмысленный текст) [10].

Итого, первоначальная структура работ при разработке системы распределения задач внутри команды разработчиков будет следующей:

1. Составить базу знаний;
2. Извлечь из базы знаний необходимую информацию;
3. Структурировать информацию в формате, удобном для обучения нейронной сети;
4. Написать программный код системы распределения задач;
5. Обучить систему на основе полученной информации.

Аналог предлагаемого решения

Аналогом проектируемой нейро-сетевой модели является программа-виртуальный ассистент PMOtto.ai, использующий технологии нейросетей для помощи в управлении проектами. Данная программа основана на языковой модели GPT-4, помимо которой задействуется надстройка в виде корпоративной базы знаний. «Участники проекта могут … сообщать статус выполнения задач, информировать о рисках, запрашивать необходимую информацию. PMOtto распознает речь и текст, и преобразует это в команды для информационных систем. Кроме того, PMOtto может давать рекомендации по реализации проекта, основываясь на результатах машинного обучения и реализованных в нем алгоритмах» [10].

Итого, по крайней мере одно решение, занимающееся декомпозицией и распределением задач, существуют. Оно основано на технологии GPT.

Реализация прототипа предлагаемого решения

Необходимо обучить нейронную сеть решать задачу классификации текстов. В качестве классов будут выступать категории работников, которые способны выполнить предлагаемую задачу. Нейронная сеть должна выполнять классификацию задачи по её текстовой формулировке, составленной человеком [11].

Для демонстрации примера решения поставленной задачи воспользуемся набором данных, содержащим информацию о заказах, опубликованных на онлайн-платформе, где заказчики размещают необходимые к выполнению задачи, а исполнители откликаются на такие объявления [12]. Нейронная сеть должна предсказывать, к какой категории относится задача, используя только полученное описание. На рисунке 1 приведены первые 10 строк из набора данных.

Для написания программного кода был использован язык программирования Python 3 и платформа Google Colab, содержащая множество предустановленных библиотек машинного обучения [13, 14, 15, 16]. Исходный код приведён в листинге 1.

	Category Name	Description
0	Design	We are looking to improve the banner images on...
1	Video, Photo & Image	Hello \n\nI need a quick designer to make 4 pi...
2	Business	Hi - I need a bookkeeper to assist with bookke...
3	Business	Hi - I need an accountant to assist me with un...
4	Digital Marketing	Hi, I am currently running a project where I w...
5	Technology & Programming	Brief\nThe requirements of this brief is to fi...
6	Design	I need to build web site for my tutoring compa...
7	Technology & Programming	I am currently working on a new freelancer com...
8	Design	Looking for a quote for an introductory e-lear...
9	Music & Audio	1. need native speaker from US or UK or CA\n2...

Рис. 1. Первые 10 строк из набора данных

Листинг 1 — Подготовка данных для обучения нейронной сети

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import plotly.express as px
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.corpus import stopwords
df = pd.read_csv('dataset.csv')
category_df = df[['Category Name', 'Description']]
print(«Categories:», set([c for c in df['Category Name']]))

def preprocess(text):
 tokenizer = nltk.tokenize.TreebankWordTokenizer()
 tokens = tokenizer.tokenize(text)
 stemmer = nltk.stem.WordNetLemmatizer()
 return ' '.join(stemmer.lemmatize(token) for token in tokens)

sentences_train, sentences_test, Y_train, Y_test = train_test_split(category_df['Description'].map(preprocess), category_df['Category Name'], test_size=0.25, random_state=42)
```

На данном этапе выполнено разделение набора данных на тренировочные (содержащие правильные категории для каждого описания) и тестовые (которые будут использованы для проверки качества обучения) в соотношении 25 % и 75 % соответственно. Для обучения будет использоваться библиотека sklearn. Доступны следующие категории: Marketing, Branding & Sales, Video, Photo & Image, Music & Audio, Design, Writing & Translation, Technology & Programming', Digital Marketing, Social Media, Business. Для проверки качества обучения кроме процента правильных ответов будет использоваться матрица ошибок, показывающая количество правильных и неправильных ответов нейронной сети. Программный код для её генерации представлен в листинге 2 [13]. В качестве аргумента функция получает массив ответов, которые предоставила нейронная сеть.

Листинг 2 — Генерация матрицы ошибок

```
def show_matrix(y_predict):
    plt.rcParams['figure.figsize'] = (10, 10)
    fig = px.imshow(confusion_matrix(Y_test, y_predict), text_auto=True)
    fig.update_layout(xaxis_title=»Target», yaxis_title=»Prediction»)
    fig.show()
```

Приступая к обучению, необходимо свести текстовую информацию к числовому представлению. Для этого делается подсчёт количества вхождений каждое слово в описании. Затем строится вектор, и нейронная сеть учится сопоставлять подобные векторы с категориями. Для прояснения работы алгоритма приведём пример.

Допустим, есть две строки: «John likes ice cream» и «John hates chocolate». Сначала необходимо определить, сколько различных слов встречается в этих строках: {"John": 0, "chocolate": 1, "cream": 2, "hates": 3, "ice": 4, "likes": 5}. Затем необходимо построить вектор для каждой строки. Размерность вектора будет равна количеству слов (то есть вектор будет содержать 6 элементов). Каждый элемент — это число, показывающее сколько раз слово встретилось в этой строке. Такой вектор будет построен для каждой строки. В результате получится массив [[1, 0, 1, 0, 1, 1], [1, 1, 0, 1, 0, 0]]. Для первой строки: слово «John» встретилось 1 раз, слово «chocolate» встретилось 0 раз и так далее. Программный код, выполняющий это действие, представлен в листинге 3 [13, 15].

Листинг 3 — Построение вектора для текстового описания

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=1500,min_df=5, max_df=0.7,stop_words=stopwords.words('english'))
vectorizer.fit(sentences_train)
X_train = vectorizer.transform(sentences_train)
X_test = vectorizer.transform(sentences_test)
```

На данном этапе выполнено преобразование текстовой информации в вектор чисел [11]. Это означает, что теперь можно применить стандартные алгоритмы машинного обучения.

Выполним категоризацию тестовых значений и посмотрим, какой процент задач удалось классифицировать правильно. Исходный код этапа обучения приведён в листинге 4 [14, 15].

Листинг 4 — Обучение нейронной сети

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
classifier = LogisticRegression(max_iter=300)
classifier.fit(X_train, Y_train)
predictions = classifier.predict(X_test)
score = classifier.score(X_test, Y_test)
print(«Accuracy:», score)
show_matrix(predictions)
```

По итогам обучения получается ~70 % правильных ответов (Accuracy: 0.6969895287958116). Матрица ошибок приведена на рисунке 2.

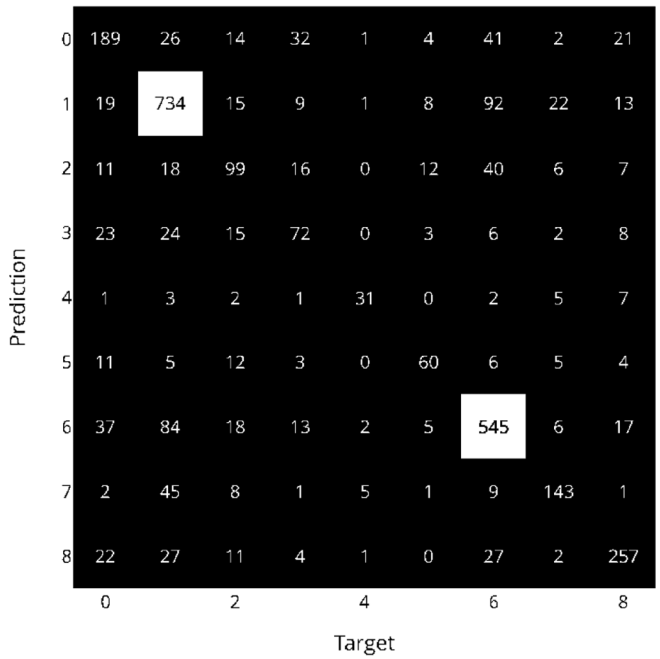

Рис. 2. Матрица ошибок

На главной диагонали матрицы содержатся количества правильных ответов. Строка i содержит количество предсказаний для описаний, которые относились к категории i. Столбец j содержит количества предсказаний для описаний, которые нейронная сеть отнесла к категории j. Таким образом, 26 задач, относящихся к категории 0, нейронная сеть ошибочно отнесла к категории 1. При этом 189 задач, также относящихся к категории 0, нейронная сеть смогла правильно классифицировать.

Продемонстрируем процесс классификации. Для этого придумаем 4 задачи и проверим, какая категория будет им назначена. Исходный код тестового запуска представлен в листинге 5.

Листинг 5 — Тестовый запуск

```
tasks = [«I want to build REST API for our new service», «I need to promote my Instagram account», «I want a new logo for my YouTube channel», «Make a translation from French to English»]
vectr = CountVectorizer(max_features=1500, stop_words=stopwords.words('english'))
vectr.fit(tasks)
data = vectorizer.transform(tasks)
preds = classifier.predict(data)
print(preds)
```

Был получен следующий результат работы программного кода: ['Technology & Programming' 'Social Media' 'Design''Writing & Translation'].

Все задачи в данном примере были классифицированы правильно. Это доказывает изначально сформулированный тезис: поставленная задача может быть решена с помощью технологий машинного обучения. Однако, нейронная сеть даёт лишь 70 % правильных ответов, но этот показатель может быть улучшен, если применить технологии глубокого обучения [5].

Заключение

Таким образом, была рассмотрена предметная область решаемой проблемы, составлено описание будущей автоматизированной системы, найден и проанализирован её аналог, реализован прототип рекомендательной системы, показавший распознавание и назначение задач в 70 % случаев.

Для получения лучшего результата необходимо составить объемную качественную базу знаний и применить алгоритмы глубокого обучения.

Рассматриваемая информационная система имеет большой потенциал и при успешной реализации и грамотной эксплуатации способна изменить в лучшую сторону работу менеджеров проектов.

ЛИТЕРАТУРА

1. Руководство к своду знаний по управлению проектом (Руководство PMBOK), 6-е изд., Коллектив переводчиков, Олимп-Бизнес, 2018.
2. Боронина, Л.Н., З.В. Сенук. Основы управления проектами: учебное пособие 2-е изд., доп., Екатеринбург, УрФУ, 2016, URL: https://e.lanbook.com/book/98746 (дата обращения: 29.10.2023).
3. Task Management Trends Report — исследование тенденций управления задачами, URL: https://reclaim.ai/blog/task-management-trends-report (дата обращения 29.10.2023).
4. Л. Хобсон. Обработка естественного языка в действии, СПБ.: Питер, 2020.
5. В. Вьюгин. Математические основы машинного обучения и прогнозирования. Электронное издание, Москва: МЦНМО, 2014, URL: http://iitp.ru/upload/publications/6256/vyugin1.pdf (дата обращения: 20.12.2023).
6. Дли, М.И., О.В. Булыгина Особенности применения нейро-сетевых моделей для классификации коротких текстовых сообщений. Программные продукты и системы, 2019, № 4, С. 650–654, URL: https://e.lanbook.com/journal/issue/312296 (дата обращения: 20.12.2023).
7. Тугускина Г.Н., Рожкова Л.В., Сальникова О.В. Управление знаниями в современных организациях, Известия высших учебных заведений. Поволжский регион. Общественные науки, 2019, № 2, С. 210–218, URL: https://e.lanbook.com/journal/issue/315267 (дата обращения: 29.10.2023).
8. П. Флах. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных, пер. с англ. А.А. Слинкина, Москва: ДМК Пресс, 2015.
9. Лезгян А.С. Автоматическое реферирование текстов: классификация, архитектуры, современные подходы и проблемы. Математическое моделирование, компьютерный и натурный эксперимент в естественных науках, 2023, №1, URL: https://cyberleninka.ru/article/n/avtomaticheskoe-referirovanie-tekstov-klassifikatsiya-arhitektury-sovremennye-podhody-i-problemy (дата обращения: 21.12.2023).
10. PMOtto — Помощник на базе искусственного интеллекта во всех вопросах управления проектами, URL: https://www.pmotto.ai/. (дата обращения 29.10.2023).
11. Гольдберг, Й. Нейросетевые методы в обработке естественного языка: руководство, пер. с англ. А.А. Слинкина, Москва, ДМК Пресс, 2019, URL: https://e.lanbook.com/book/131704 (дата обращения: 17.11.2023).
12. Freelance Platform Projects, набор данных, URL: https://www.kaggle.com/datasets/prtpljdj/freeelance-platform-projects. (дата обращения 15.12.2023).
13. Дж. Вандер Плас, пер. И. Пальти, Python для сложных задач. Наука о данных и машинное обучение, Питер, 2017.
14. Б. Шарден. Крупномасштабное машинное обучение вместе с Python, пер. с англ. А.В. Логунова, Москва: ДМК Пресс, 2018.
15. С. Рашка. Python и машинное обучение: машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow 2, 3-е изд., пер. с англ. СПб.: ООО «Диалектика», 2020.
16. Документация к Python-библиотеке NLTK, URL: https://www.nltk.org/index.html (дата обращения: 15.12.2023).