REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

QUALITY TEXT FILTERING AND STORYTELLING FROM WEB TEXTS

19011110 — AHMAD REİS
21011608 — AHMET SELİM SÖNMEZ

SENIOR PROJECT

Advisor
Res. Asst. Himmet Toprak KESGİN

January, 2024

ACKNOWLEDGEMENTS

I hope this message finds you well. I want to thank you for being supportive and guiding me during my semester project. Your feedback, suggestions, and motivation were critical to the success of this project.

Your commitment to nurturing an environment around me has greatly improved my skills and knowledge and also made the whole experience fascinating and enjoyable. I am very thankful for the time and effort you put in in trying to help me polish my ideas and improve the quality of my work.

Once again, I appreciate your mentoring. I will remember the lessons and apply them in my future work. We would also like to thank all the professors, assistant doctors, and all the working staff and faculty members of the Computer Engineering Department at Yildiz Technical University.

AHMAD REİS
AHMET SELİM SÖNMEZ

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Quality Text Filtering and Storytelling from Web Texts

AHMAD REİS
AHMET SELİM SÖNMEZ

Department of Computer Engineering
Senior Project

Advisor: Res. Asst. Himmet Toprak KESGİN

The exponential growth of textual data available on the internet presents significant challenges in efficiently extracting meaningful information. In this context, text filtering and storytelling become crucial in ensuring that only relevant and high-quality data is utilized for further processing. This project, titled Web Metinlerinden Kaliteli Metin Filtreleme ve Hikayeleştirme (Quality Text Filtering and Storytelling from Web Texts), aims to develop a robust pipeline for filtering and cleaning Turkish-language textual data, leveraging modern Natural Language Processing (NLP) techniques.

The project focuses on two main tasks: filtering out irrelevant or low-quality texts from a large corpus and refining the remaining data by extracting and enhancing the valuable content. To achieve this, both traditional feature extraction methods and state-of-the-art pre-trained models are employed. These models are used to convert text into meaningful feature vectors, which can then be utilized to classify and filter the texts effectively. Additionally, manual feature extraction techniques are implemented, including metrics like punctuation ratio, word length, and sentence structure, to enhance the filtering process.

The dataset utilized consists of web-based Turkish texts that are labeled as high-quality or low-quality. The project applies various classification models, trained using both extracted features and pre-trained embeddings, to distinguish between valuable and non-relevant content. The effectiveness of these models is evaluated through standard performance metrics such as accuracy and F1-score, which guide the iterative refinement of the pipeline.

Ultimately, the goal of this project is to develop a clean, high-quality dataset that can be used for further NLP tasks, such as sentiment analysis, topic modeling, or content summarization. The results of this work contribute to the broader field of text mining by providing a framework for filtering and enhancing web-based content, especially for languages with limited resources in NLP, such as Turkish.

Keywords: NLP, pre-trained models, classification, filtering process,

ÖZET

Web Metinlerinden Kaliteli Metin Filtreleme ve Hikayeleştirme

AHMAD REİS
AHMET SELİM SÖNMEZ

Bilgisayar Mühendisliği Bölümü
Bitirme Projesi

Danışman: Arş. Gör. Himmet Toprak KESGİN

İnternetteki metinsel verilerin hızla artışı, anlamlı bilgilerin verimli bir şekilde çıkarılması konusunda önemli zorluklar yaratmaktadır. Bu bağlamda, metin filtreleme ve hikayeleştirme, yalnızca ilgili ve yüksek kaliteli verilerin daha ileri işleme süreçlerine dahil edilmesini sağlamak için kritik öneme sahiptir. Web Metinlerinden Kaliteli Metin Filtreleme ve Hikayeleştirme başlıklı bu proje, Türkçe dilindeki metin verilerini filtreleme ve temizleme için sağlam bir iş akışı geliştirmeyi amaçlamaktadır. Proje, modern Doğal Dil İşleme (NLP) tekniklerini kullanarak metinleri analiz etmekte ve düzenlemektedir.

Proje iki ana görev üzerine yoğunlaşmaktadır: büyük bir metin kümesinden alakasız veya düşük kaliteli metinlerin filtrelenmesi ve geriye kalan verilerin değerli içeriğini çıkarmak ve geliştirmektir. Bu amaçla, hem geleneksel özellik çıkarım yöntemleri kullanılmaktadır. Bu modeller, metinleri anlamlı özellik vektörlerine dönüştürmek için kullanılır ve bu özellikler, metinlerin etkili bir şekilde sınıflandırılması ve filtrelenmesinde kullanılır. Ayrıca, noktalama işareti oranı, kelime uzunluğu ve cümle yapısı gibi metin içi özellikleri çıkaran manuel yöntemler de uygulanarak filtreleme süreci iyileştirilmektedir.

Kullanılan veri seti, Türkçe web metinlerinden oluşmakta olup, yüksek kaliteli veya düşük kaliteli olarak etiketlenmiştir. Proje, çıkarılan özellikler ve önceden eğitilmiş gömme (embedding) vektörleri ile eğitim almış çeşitli sınıflandırma modellerini kullanarak, değerli ve alakasız içerikleri ayırt etmektedir. Bu modellerin

etkinliği, doğruluk (accuracy) ve F1 skoru gibi standart performans metrikleriyle değerlendirilmekte ve bu değerlendirmeler, iş akışının sürekli iyileştirilmesini sağlamaktadır.

Sonuç olarak, bu proje, metin madenciliği alanına katkıda bulunarak, Türkçe gibi sınırlı kaynaklara sahip dillerde web tabanlı içeriğin filtrelenmesi ve geliştirilmesi için bir çerçeve sunmaktadır. Projenin nihai hedefi, daha ileri NLP görevleri için kullanılabilecek temiz ve kaliteli bir veri seti oluşturmaktır. Bu çalışma, duygu analizi, konu modelleme veya içerik özetleme gibi daha ileri metin işleme görevleri için sağlam bir temel oluşturmayı amaçlamaktadır.

Anahtar Kelimeler: Metin Filtreleme, Hikayeleştirme, Doğal Dil İşleme (NLP), Türkçe Metin Analizi,Derin Öğrenme,

1
Introduction

1.1 Project Description

The Internet is one of the most important technological innovations of the modern era, revolutionizing the way we communicate and conduct our daily activities. The Internet is defined as the global network that connects millions of devices and users around the world, giving them access to a huge and diverse content of information and services. With the rapid growth of the Internet and artificial intelligence, scientific research on these topics has flourished more and more, and there is no doubt that web texts are also important in this field, as text filtering and reformulation is very important. The digital world has noticed a tremendous increase in the amount of content that is created on the Internet, whether news articles, social media posts, blogs, or academic papers, every second dozens of unstructured or full data texts are created daily, despite the availability of a huge amount of valuable information. This issue is mainly exacerbated by the fact that despite the progress made in Natural Language Processing (NLP). Hence the importance of this project, which deals with our project.

In the field of NLP in our project, we are dealing with filtering texts previously collected from different and multiple sites, this matter has made it a little easier for us, as the main goal in this project is to convert raw, noisy and unorganized web content into high-quality, organized outputs that are the most suitable for a number of purposes The ways to achieve this were in following a plan of action to finish the project, starting with defining classification algorithms such as (-Nearest Neighbors LightGBM XGBoost Decision Tree) that are suitable, fast and at the same time with high performance, feature extraction, classification and fine-tuning of the latest and most advanced models, including Word2Vec, LightGBM and the Turkish language model-Llama-8b.

We researched a bunch of good algorithms in the field and ran tests such as F1 score, cosine similrity, and time that determined the best model for our project for Turkish. This included removing unwanted sections such as HTML markup, special characters, and extra spaces. This additional step not only made the dataset ink-free, but it also made it organized, making it ready for the next step's analytics. At this point in the process, the text was far from being of any practical use than it was at the beginning, as it was now just a collection of meaningless data that did not have the clarity or organization to allow analysis of any meaningful nature. Next, we needed to extract features from the texts, so we tested several models such as Word2Vec, distiluse-base-multilingual-cased-v1, fasttext-tr, dbm dz/bert-base-turkish-cased that capture the semantic and syntactic similarities between words in the text Word2Vec is a tool that can be used to capture semantic and syntactic similarities between words in a text. This system used the word vector representations created by the Word2Vec model, which was trained using the Skip-gram architecture, to represent the words in the dataset. The vectors formed due to the use of these representations helped the system to fully recognize the relationship between words and each other. That is, through them, the system gained the ability to identify content as either high-quality or low-quality with less difficulty. Through these vector embeddings, the system was able to delve deeper into the meaning of the text which gave the basis for the next stage: Categorization. Also, as mentioned earlier, tests were conducted to select the classification model or algorithm The classification was performed by LightGBM, a fast and efficient machine learning method that was able to handle high-dimensional data as it finds the best route to travel. LightGBM was one of the machine learning algorithms we chose to use for the classification task. This powerful algorithm is so fast and multifunctional that it can handle high-dimensional data. On the contrary, generating sentences from LDA is a task that involves many dimensions, such as word count, typographic dimensions, and thematic syntactic connections. This is because we used Word2Vec's extracted numeric vectors as inputs to train the LightGBM classifier and were able to recognize these vectors in two broad categories: high-quality contents and low-quality contents. However, there were also situations where it was not clear, for example, the predictions were not quite confident at one point, especially when it came to the boundary cases they had to distinguish between. To deal with this issue, we included confidence scores in the system so that the models had room for flexibility. Models above the threshold were better programmed to be adaptable and less rigid in the decision-making process.

Finally, after tuning the models and generating outputs for each of the four prompts, the results were evaluated for quality and consistency. To check the consistency between the resulting text and the original data, we calculated cosine similarity, which helped us estimate the degree of similarity between the fine-tuned outputs, the Gemini outputs, and the main text. This evaluation helped to accurately determine the effectiveness of the methodology while revealing gaps in it.

In the course of this project, we encountered some obstacles especially with the ambiguous cases that the model was struggling to categorize. Dealing with low-confidence predictions was difficult because there were a lot of extra steps needed to manage boundary conditions. However, in light of these challenges, using Word2Vec, LightGBM, and Turkish-Llama-8b fine-grained models provided a viable solution for filtering, categorizing, and rewriting web texts. The results showed that using this combined approach not only led to the creation of high-quality datasets but also to the creation of specific types of texts whether academic, journalistic, or even creative writing.

Finally, the work done in this project shows that advanced machine learning techniques can be used in conjunction with deep learning models to process web texts which is a very complex challenge. In addition to building a robust pipeline for processing, categorizing, and rewriting web content, this project also makes a significant contribution to the field of natural language processing by showing practical applications of these techniques in real-world situations. The knowledge gained from this research will help create more efficient, flexible, and accurate systems for content quality assessment and text generation, ultimately leading to more accurate and impactful digital content.

1.2 Purpose and Objectives of the Project

The main objective of the project is to obtain quality data by filtering unnecessary and erroneous information from Turkish texts. In addition, the usability of the datasets will increase by transforming filtered texts into meaningful summaries and stories. To this end, the following objectives have been set:

- To automatically filter out unnecessary and erroneous content from large text datasets. - Narrate the text by extracting meaningful and summary information from the filtered texts. - Measure the effectiveness of the filtering and storytelling processes by setting performance criteria.

In this project, different natural language processing methods and deep learning based models will be used and meaningful features will be extracted from texts with various filtering techniques. In addition, performance metrics will be used to measure the accuracy and effectiveness of the model obtained at the end of the project.

1.3 Pre-Review

Data mining and natural language processing (NLP) techniques have become a widespread application to the understanding, categorization, and summarization of large amounts of textual data. There is a growing volume of content on the internet which is cluttered and often contains incorrect and repeated data or information. In fact, this 'clutter' is most likely to be found in the texts of social media, forums, news sites, and other similar platforms. Thus, in order to yield a smart result from the extracted text data, the aforementioned process is necessary. The cleaning and filtering steps take the data quality to a higher level and set the baseline for further better analysis and modeling. Apart from purging the content that is incorrect or irrelevant, an in-depth analysis of the linguistic properties and the text's structure is a great way of accurately filtering the texts too. That is to say, NLP techniques for Turkish text processing should not contradict the language-specific requirements for Turkish but be harmonized with their linguistic properties. The aim of our project is to collect and process texts that are meaningful and of great value from the data sets. With this in mind, methods and models have been checked for their capabilities of correctly classifying and filtering texts.Most approaches in the field of text filtering and storytelling are found in the literature. Among them, statistical feature extraction methods, deep learning based models, and pre-trained language models are the best. As a part of this review text this paper provides many known approaches to estimated text filtering along with their performances. We also delve into the value achieved with language-specific Turkish models and algorithms.The success of the majority of the models in English, however, the remaining ones would be likely to be customized for Turkish. The main objective of the project is to obtain quality data by filtering unnecessary and erroneous information from Turkish texts. In addition, the usability of the datasets will increase by transforming filtered texts into meaningful summaries and stories. To this end, the following objectives have been set:

- To automatically filter out unnecessary and erroneous content from large text datasets. - Narrate the text by extracting meaningful and summary information from the filtered texts. - Measure the effectiveness of the filtering and storytelling processes by setting performance criteria.

In this project, different natural language processing methods and deep learning based models will be used and meaningful features will be extracted from texts with various filtering techniques. In addition, performance metrics will be used to measure the accuracy and effectiveness of the model obtained at the end of the project.

2
Literature Review

The aim of this project is to convert the large scale datasets into high-quality meaningful texts and cleaning of the Turkish text data in it. Data quality and the volume of training datasets are usually considered as two main sources of smallest training dataset's performance in extensive language models. Some of the studies in the area mention that preprocessing, filtering, and deleting duplicates have a high potential to increase the model's performance significantly. Here are analyses of three major studies relevant to this project:

2.1 DataComp-LM

Dataset Design and Filtering Approaches for Language Models DataComp-LM has been designed to be an experimental hub for testing different dataset designs and filtering strategies targeted at improving the quality of the training datasets of big language model platforms. The investigation contains human curated data lists that consist of keyworks interconnected to each other and a model is asked to find the correct connection among the given data. Here one of the most necessary steps is created which is checked by the right and wrong answers so that AI is learning more efficient. This study reveals the clear importance of constructing high-quality datasets. In the experiments, DataComp-LM showed that natural language processing with language model-based filtering can indeed create high-quality datasets which are crucially important for the success of a deep learning algorithm. The study was based on certain methods, for example, in medicine the structure of the heart was determined with the help of a robot that made the necessary detailing of the MRI or CT to the doctor. Yet in another area applications on language diversity were discussed.[**DataComp-LM**].

2.2 FineWeb

Filtering and Deduplication Strategies for Large-Scale Text Data Yet another study far away from the first one "DataComp-LM" is The FineWeb study which came out and described the structuring of 15 trillion tokens that contained Common Crawl. The study FTP data: FineWeb transforms 15 trillion-tokened Common Crawl data into structured data accepting at most three labels on each record entity and relation and cleans only records with strong cohesive context. This study targets the development of methods to clean (deduplicate), filter, and retrieve good structured text though it has an additional component named contrastive learning described in the emphasis section. Some mechanisms like deduplication (removal of duplicate or overlapping documents) and quality filtering (removing less relevant and low-quality documents) will help get rid of unneeded information; therefore, language model learning will be adequately supported. FineWeb-based models have been successful in various benchmark tests including knowledge- and reasoning-based tasks. This work emphasizes the fact that deduplication and other techniques in data management are critical to making large data sets more parsimonious and at the same time more meaningful.[1].

2.3 Cosmopedia

Creating Training Datasets with Synthetic Data Generation Cosmopedia is about the process of creating training data for large language models using synthetic data. To be true about it, Cosmopedia attempts to enhance the model performance and to waste less cost for the usage of data acquisition. Thus, the project uses synthetic content to develop affordable yet very valid datasets, which center on distinction to ward off duplication of content. Cosmopedia is fully packed with various topics, audiences, and text styles. It also explains synthetic data's potential and the impact of its use on model training in a meaningful way on a scalable basis[2].

3
SYSTEM ANALYSIS AND FEASIBILITY

3.1 System Analysis

This project's system analysis is the characterization process of Turkish text datasets by rejecting irrelevant or incorrect texts to keep the meaningfulness and validity of the dataset which is the main operation of the project. To be more precise in doing so, we should aim to achieve a pipeline for filtering and feature extraction which will analyze the texts depending on some criteria. Using model-based and feature-based methods to evaluate the quality will be the processing of data. During system analysis, projects' objectives to describe the tools and methods which will be used with data processing and feature extraction procedures, model training and verification methods are explicitly stated. Our objectives are: - Turkish texts inspection, which let us know where the cleaning process should be focused, - Building a comprehensive feature set with the help of statistical and linguistic feature extraction from the texts, - The selective models will be used for training with the highest level of precision, and they should be optimized appropriately, thus offering the most satisfactory results.

3.2 Requirement Analysis

Data Requirements: Datasets of raw text data are required for the project. Software Requirements: Python libraries including transformers, torch, pandas, sklearn. Hardware Specifications: Powerful processing computer or computer with GPU capability (a powerful CPU or GPU, 16 GB of RAM or more).

3.3 Feasibility

3.3.1 Technical Feasibility

The project's technical feasibility hinges on the chosen technologies and infrastructure to ensure its successful rollout and completion. This part breaks down into hardware and software elements spotlighting the best picks to meet the project's needs .

Software Compatibility: This project will operate in a Python setting and will use open source libraries (transformers, torch, pandas, sklearn). These work well with most current operating systems and hardware and are common in Python-based projects. Plus, the ongoing backing of up-to-date versions of the libraries used gives the project an edge in terms of technical compatibility throughout its lifespan.

3.3.1.1 Hardware Feasibility

Hardware is an important component in that one has to possess good equipment to be able to execute the project that requires Turkish text filtering and story generation in the right way without the need to train and use these models in an efficient manner. Hardware that has a high technical ability in conjunction with high-performance hardware is recommended for this purpose. Hardware Requirements: The project that needs large models for text preprocessing, feature extraction, and classification requires large RAM and good processor. We suggest that the minimum RAM should be at least 16 GB, and the processor must be a fast and modern multi-core processor (Intel i7 or AMD Ryzen 7). Moreover, if it is possible, having an NVIDIA GPU will help to accelerate the training time of deep learning models. The SSD drive is also a good choice to quickly access data and at the same time speed up the model training process. In the case of initial local development and small-scale testing, the machine described above is ok. This practically means that all the tasks including the model training, code generation, and dataset preparation will be done without a delay. Still, the software and the size of the data are big enough, the need for hardware increases. Cloud-based Solutions: In a situation when the local hardware is not available, the cloud-based solutions such as Google Colab, AWS, or Microsoft Azure can be used to offer computing resources. They provide high-speed GPUs and easily scalable infrastructure making it possible to run huge datasets and deep learning models without the use of local hardware. These solutions are very beneficial in case one has a large dataset or running deep learning models that need a high level of computing resources.

Scalability and Long-Term Sustainability: The project's dynamic constitution and the increasing complexity of language models make scalability the most vital element. The platforms and hardware that we decide to use are the most suitable as the project can be scaled up easily along with the complexity of the model and the dataset sizes. Such adaptability will help the project to be well-adjusted to the technologies of the future and, to some extent, the hardware upgrades can be avoided. In summary, the project will be assigned with the capability of Turkish text processing, and narrative generation due to investing in the most high-spect hardware and the use of cloud-based solutions that the project gets the necessary computational resources and efficient rates for these functions. This hardware architecture is a certainty of the project's successful use and scalability in the future; therefore, the technology development of the project over the long period should also last.

3.3.2 Legal and Ethical Feasibility

This project's legal viability is assured because it uses public open-source data and frameworks. The main data comes from websites that allow research and educational use all following copyright laws. Also, the risk of infringing on commercial tech is low as all the computational software and machine learning platforms, like torch , have open-source licenses letting people use them for education. The project follows general data protection and intellectual property rights because it doesn't use identifiable or sensitive personal information. By working with text data and using emojis to analyze and predict, the project steers clear of legal issues. The project design and methods respect user privacy and ethical concerns making the project possible and viable for research and academic centers.

3.3.3 Economic Feasibility

Economic feasibility examines the cost-benefit analysis of the project to determine if the financial benefits outweigh the costs.

- Since we used the free versions of Google Colab, Jupyter, and Kaggle Notebook as development environments, We did not incur any development environment fees.
- Additionally, the libraries that we utilized for LLMs and machine learning are open source, and thus there were no charges for the software tools.
- The data set used in the project is also open source and free.
- Furthermore, our personal computers' hardware (Lenovo Legion 5 RTX-2060) provides sufficient capability to execute the project.

Table 3.1 Cost breakdown and specifications for different machines and services

Machine	Piece	Price	Total cost	CPU	GPU
Lenovo Legion 5	1	50,000 TL	50,000 TL	AMD 7-4800	RTX 2060
Google Colab	2	0	0	Varies	Tesla 4
Kaggle	3	0	0	Varies	Tesla P100
Google Colab Pro	6	169 TL	1014 TL	Varies	A100 Tensor Core

3.3.4 Labor and Time Feasibility

Workforce and time feasibility are critical to the successful completion of projects. In the project of Quality Text Filtering and Storytelling from Web Text, there are many phases such as text processing, feature extraction, model training and evaluation. The workforce, skill sets and time requirements for each phase must be planned accurately. In this section, the workforce and timeline required by the project are evaluated.

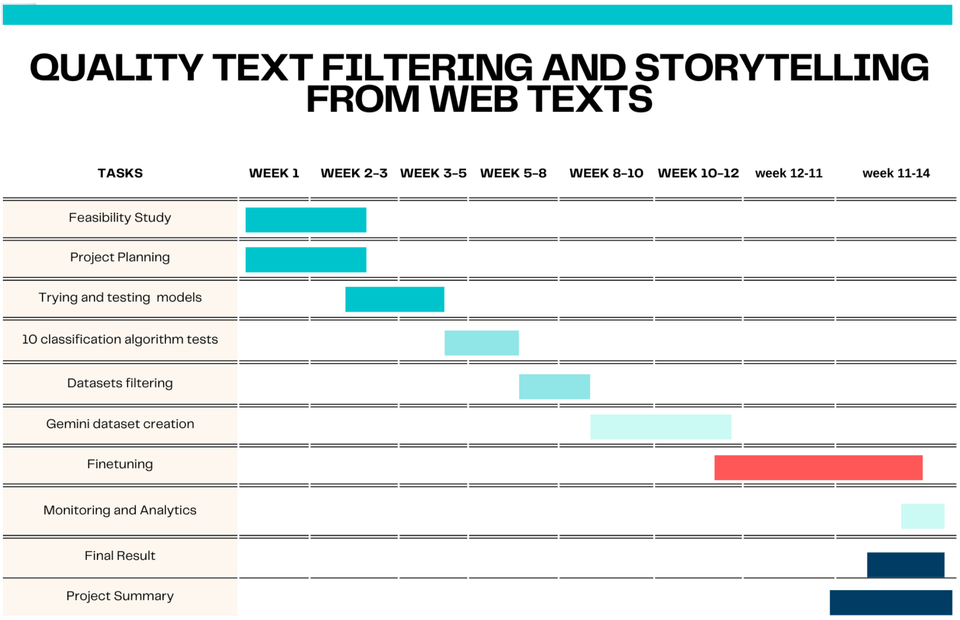


Figure 3.1 Gantt Chart for Time Feasibility

4
SYSTEM DESIGN

In our project, the goal is to obtain a quality text by filteri the texts and storytelling From Web texts. We have a main data set, 60 GB in size, collected from different websites with various types of content. Additionally, we have a test dataset, which we use to ensure that we are employing the fastest method to achieve our goal with the lowest error rate.

Our objective is to label and classify the text, filtering it into either 'good' or 'bad' text.

4.1 Materials and Dataset

As you know, in classical machine learning, we have features and labels in our data set. With these, we choose a model, train it, and get a model that can make predictions. But texts are strings and such properties are not readily available. Therefore, we need to create these features.

We have a main data set, 60 GB in size, collected from different websites with various types of content. First of all we have raw data that have to be filled out, then we can use it so for this raw data we need to perform feature extraction, we have tried many models to make sure which is giving us the shortest time to encode it as the result of our reserching we have found the Word2Vece has the fastest time you can check the tabele below

Model	Encoded_Lines	Time_Seconds
distiluse-base-multilingual-cased-v1	5000	76.35150981
TURKCELL/roberta-base-turkish-uncased	5000	237.7839816
Word2Vec	5000	0.269585133
dbmdz/bert-base-turkish-cased	5000	202.5804005
fasttext-tr	5000	232.0462003

Figure 4.1 Time

All this model in the table below we have been trying it and we chose Word2Vec and multilingual because of short time that they have Word2Vec: A word embedding model that transforms text into vectors. It captures semantic relationships between words As a model, it is specialized for Turkish then to make filltering we have to use a Classification Algorithms to classify the texts into categories in our project we have two categoriers good or bad (1,0) in this step we should select an algorithm fast and high accuracy we also tried many algorithms like Naive Bayes (Gaussian) Quadratic Discriminant Analysis Linear Discriminant Analysis K-Nearest Neighbors LightGBM XGBoost Decision Tree Multi-Layer Perceptron (MLP) AdaBoost Extra Trees CatBoost Random Forest SVM but we select these algorithms which are fast and good K-Nearest Neighbors LightGBM XGBoost Extra Trees

Naive Bayes (Gaussian)
Quadratic Discriminant Analysis
Linear Discriminant Analysis
K-Nearest Neighbors
LightGBM
XGBoost
Decision Tree
Multi-Layer Perceptron (MLP)
AdaBoost
Extra Trees
CatBoost
Random Forest
SVM

Figure 4.2 Time

In figure above you can check the algorithm that we test and we select algorithms written in black.

4.2 Methodology

The algorithms chosen were a result of thorough examination of the project. The Word2Vec model was preferred because it is a comprehesive model that, in a matter of seconds, turns text data into meaningful low dimensional vectors. For classification operations, the LightGBM algorithm was preferred. LightGBM is a high-performance classification method that is particulary designed for large scale datasets because of their improved speed and accuracy.

For the achievement of the project goals, we utilized a systematic and well-structured methodology, structured into classification and preprocessing stages. The process comprised the following stages:

4.2.1 Data Preparation:

Took the gathered text data from the web and it was divided into individual lines, after which we more detailedly treated it.

The data was clean and was ready for the analysis after unwanted elements were removed including HTML tags, special character, and empty spaces.

4.2.2 Feature Extraction:

We could make the text suitable for analysis by using the Word2Vec model. Through the model, not only the semantic but also the syntactic relationships of the words can be taken apart so that the representation could be more meaningful.

The data was used for training Word2Vec with which Skip-gram, an algorithm which works extremely well for processing words at the level of larger corpora, was employed

We applied Word2Vec on the dataset using the Skip-gram architecture, which is best suited for learning word relationships from large corpora.

4.2.2.1 Word2vec

Word2vec is a technique in natural language processing (NLP) for obtaining vector representations of words. These vectors capture information about the meaning of the word based on the surrounding words also another defention of word2vec is not a singular algorithm, rather, it is a family of model architectures and optimizations that can be used to learn word embeddings from large datasets. Embeddings learned through word2vec have proven to be successful on a variety of downstream natural language processing tasks.

4.2.3 Classification with LightGBM:

Through the process of Word2Vec, vectorized strings were obtained and then they were fed into the LightGBM classifier. This technique that was preferred to deploy is very fast, efficient, and can cope with lots of dimensions effectively. The LightGBM classifier was fitted with the numerical vectors produced by Word2Vec. LightGBM was the algorithm that was that was mainly chosen because it is swift, deal with a large amount of data efficiently, and produce low dimensionality. Identifying segment labels as either low-quality or high-quality content, the classifier allocated the values of 0 and 1 to each text piece.

Classifier made a decision about each piece of text as being either of the low-quality or high quality category by employing labels 1 or 0.

LightGBM:

Preferred for its fast speed and high accuracy. Effective in large datasets or feature space. Leaf-wise growth strategy provides deeper trees and therefore better performance.

4.2.4 Incorporating Confidence Scores:

The confidence scores given by LightGBM to its predictions were the indicators that assured how certain the model was about each classification. For instances that were classified as low-quality, we added a step in the process where we generate a random number from 0 to 1. If the confidence score of the prediction surpassed 0.8, the result was secured no matter what initial labeling the content had. In this way, we could deal with borderline cases a bit flexibly.

4.3 Manual and Hybrid Feature Extraction

4.3.1 Manual Features

The manually extracted features used in the project are listed below:

- `id`

- word_count
- character_count
- sentence_count
- longest_word_length
- shortest_word_length
- average_sentence_length
- average_word_length
- upper_case_ratio
- lower_case_ratio
- punctuation_ratio
- digit_ratio
- whitespace_ratio
- vowel_ratio
- consonant_ratio
- upper_lower_ratio
- unique_word_ratio
- all_caps_ratio
- words_ending_with_m_ratio
- words_ending_with_z_ratio
- words_second_last_char_d_ratio
- words_ending_with_sh_ratio
- four_digit_sequences_ratio
- words_ending_with_r_ratio
- de_da_ratio
- yok_ratio

- MÖ_ratio
- emoji_ratio
- en_ratio
- ve_ratio
- com_ratio
- english_not_turkish_ratio
- ben_ratio
- sen_ratio
- biz_ratio
- o_ratio
- siz_ratio
- onlar_ratio
- time_reference_ratio
- positive_stop_words_ratio
- negative_stop_words_ratio
- neutral_stop_words_ratio
- english_word_ratio
- me_ratio
- ma_ratio
- me_suffix_ratio
- ma_suffix_ratio
- label

4.3.2 Hybrid Feature Extraction

Hybrid features were created by combining manually extracted features with vector-based features, as shown below:

- `multilingual_x_manual`: Combination of multilingual embeddings with manual features.
- `multilingual_x_word2vec`: Combination of multilingual embeddings with Word2Vec embeddings.
- `word2vec_x_manual`: Combination of Word2Vec embeddings with manual features.

4.3.3 Results

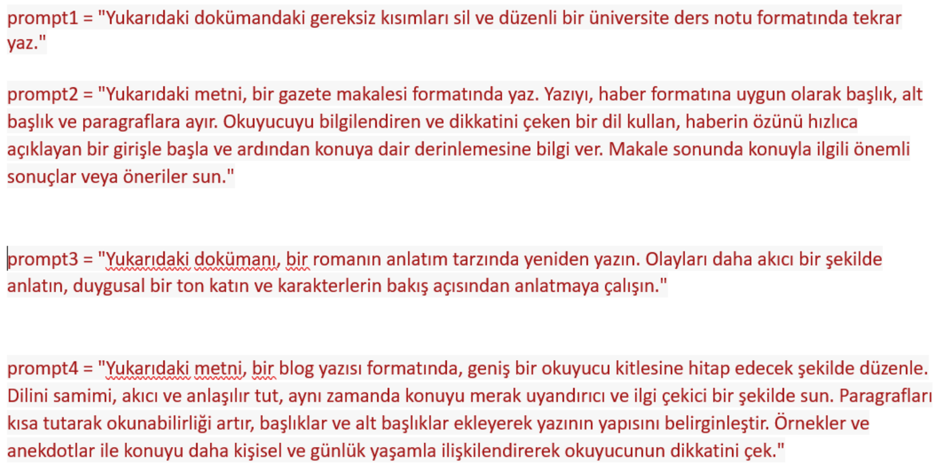
The hybrid feature extraction approach was evaluated by concatenating Word2Vec, multilingual embeddings, and manual features. Notable improvements were observed in the performance of the model:

- The `multilingual_x_manual` feature set demonstrated a 0.01 increase in performance compared to other configurations.

This indicates that combining multilingual embeddings with manually extracted features offers a slight advantage in model performance.

4.3.4 Prompt Creation and Text Rewriting:

After filtering the data, we designed four distinct prompts to interact with the Gemini model, enabling us to obtain outputs formatted for different contexts. These prompts are:


prompt1 = "Yukarıdaki dokümandaki gereksiz kısımları sil ve düzenli bir üniversite ders notu formatında tekrar yaz."

prompt2 = "Yukarıdaki metni, bir gazete makalesi formatında yaz. Yazıyı, haber formatına uygun olarak başlık, alt başlık ve paragraflara ayır. Okuyucuyu bilgilendiren ve dikkatini çeken bir dil kullan, haberin özünü hızlıca açıklayan bir girişle başla ve ardından konuya dair derinlemesine bilgi ver. Makale sonunda konuyla ilgili önemli sonuçlar veya öneriler sun."

prompt3 = "Yukarıdaki dokümanı, bir romanın anlatım tarzında yeniden yazın. Olayları daha akıcı bir şekilde anlatın, duygusal bir ton katın ve karakterlerin bakış açısından anlatmaya çalışın."

prompt4 = "Yukarıdaki metni, bir blog yazısı formatında, geniş bir okuyucu kitlesine hitap edecek şekilde düzenle. Dilini samimi, akıcı ve anlaşılır tut, aynı zamanda konuyu merak uyandırıcı ve ilgi çekici bir şekilde sun. Paragrafları kısa tutarak okunabilirliği artır, başlıklar ve alt başlıklar ekleyerek yazının yapısını belirginleştir. Örnekler ve anekdotlar ile konuyu daha kişisel ve günlük yaşamla ilişkilendirerek okuyucunun dikkatini çek."

Figure 4.3 prompts

Prompt 1: "Yukarıdaki dokümandaki gereksiz kısımları sil ve düzenli bir üniversite ders notu formatında tekrar yaz." - This prompt guided the model to rewrite the document in a concise, structured format suitable for university lecture notes.

Prompt 2: - "Yukarıdaki metni, bir gazete makalesi formatında yaz. Yazıyı, haber formatına uygun olarak başlık, alt başlık ve paragraflara ayır. Okuyucuyu bilgilendiren ve dikkatini çeken bir dil kullan, haberin özünü hızlıca açıklayan bir girişle başla ve ardından konuya dair derinlemesine bilgi ver. Makale sonunda konuyla ilgili önemli sonuçlar veya öneriler sun." - This prompt required the model to transform the input into a well-structured newspaper article with a headline, subheadings, and paragraphs, providing detailed and engaging information.

Prompt 3: - "Yukarıdaki dokümanı, bir romanın anlatım tarzında yeniden yazın. Olayları daha akıcı bir şekilde anlatın, duygusal bir ton katın ve karakterlerin bakış açısından anlatmaya çalışın." - With this prompt, the model was tasked with rewriting the document as a novel, emphasizing narrative flow, emotional tone, and character perspectives.

Prompt 4: - "Yukarıdaki metni, bir blog yazısı formatında, geniş bir okuyucu kitlesine hitap edecek şekilde düzenle. Dilini samimi, akıcı ve anlaşılır tut, aynı zamanda konuyu merak uyandırıcı ve ilgi çekici bir şekilde sun. Paragrafları kısa tutarak okunabilirliği artır, başlıklar ve alt başlıklar ekleyerek yazının yapısını belirginleştir. Örnekler ve anekdotlar ile konuyu daha kişisel ve günlük yaşamla ilişkilendirerek okuyucunun dikkatini çek." - This prompt instructed the model to create a blog post that is engaging, accessible, and appealing to a broad audience, with clear structure and relatable examples.

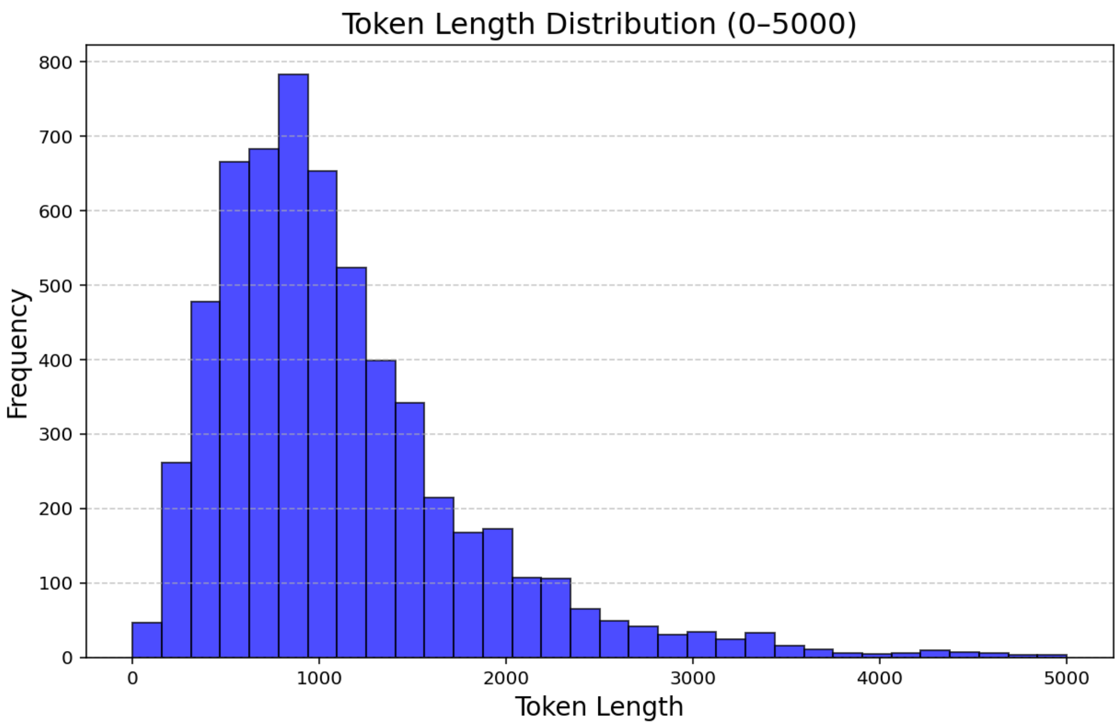


Figure 4.4 Token Length Distribution-1

4.3.5 Fine-tuning with Turkish-Llama-8b

In the (gemini_input, gemini_output) dataset, examples with more than 2048 tokens were excluded to create a reduced dataset. Using this improved dataset, we fine-tuned the "ytu-ce-cosmos/Turkish-Llama-8b-Instruct-v0.1" model on four different prompts, resulting in four separately fine-tuned models.

For fine-tuning, we utilized LoRA and the Adam optimizer. Various parameter settings were explored, including:

- **LoRA alpha:** 16 or 32
- **Optimizer:** AdamW (standard) or Adam8
- **Learning rate:** 1e-4 or 5e-5
- **Number of epochs:** 2 or 3
- **Weight decay:** 0.01 (constant)
- **Warmup ratio:** 0.1 (optional)
- **Maximum gradient norm:** 2.0 (optional)

Overall, the fine-tuning process reduced the validation loss by approximately 0.03 on average.

4.3.6 Output Generation Configuration

The fine-tuned models were configured with specific parameters during the text generation process to ensure the quality and diversity of the outputs. These parameters are detailed below:

We ensuere that only one output sequence was generated for each input prompt to maintain simplicity and focus on the primary result.

Also we prevent the model from repeating any sequence of four consecutive words, enhancing the fluency and coherence of the generated text.

also sampling was enabled,which allowing the model to explore different possibilities in text generation instead of always selecting the most probable word.

temperature was seted to 0.6 which means a lower temperature value makes the model's predictions more focused and deterministic while still retaining some level of variability.

Finally Nucleus sampling was applied, where the model considers only the top 90% of the probability mass for generating the next word. This ensures diversity in the output while avoiding unlikely words.

These parameters were chosen after experimenting with different configurations to balance between diversity and relevance in the generated text.

4.3.7 Cosine Similarity Evaluation

Cosine similarity was deployed to calculate the absolute similarity between the data fine-tuned by the model, data produced from Gemini outputs, and the original one, thus revealing the extent of the content's relevance to the anticipated results.

id	original_vs_finetune	gemini_vs_finetuned	original_vs_gemini	Versus
42838	0.461641119	0.42055343	0.312329087	gemini_vs_finetuned
42839	0.611734507	0.583176048	0.40911987	gemini_vs_finetuned
42840	0.922850732	0.639487592	0.642872636	original_vs_gemini
42841	0.660958657	0.594359634	0.582471775	gemini_vs_finetuned
42842	0.492088121	0.480888965	0.195354194	gemini_vs_finetuned
42843	0.414253457	0.544689855	0.198658071	gemini_vs_finetuned
42844	0.226511866	0.506267863	0.135450138	gemini_vs_finetuned
42845	0.837324252	0.819289568	0.747750424	gemini_vs_finetuned
42846	0.655283755	0.39820424	0.364946671	gemini_vs_finetuned
42847	0.41946515	0.46101809	0.17465546	gemini_vs_finetuned
42848	0.63540331	0.611947866	0.484350323	gemini_vs_finetuned
42849	0.485767471	0.469332115	0.29032795	gemini_vs_finetuned
42850	0.906717554	0.818340538	0.77037462	gemini_vs_finetuned
42851	0.502052524	0.481087643	0.280826514	gemini_vs_finetuned
42853	0.583498424	0.56205187	0.40185615	gemini_vs_finetuned
42854	0.333657173	0.315966421	0.06583564	gemini_vs_finetuned
42855	0.879625382	0.732320901	0.759365944	original_vs_gemini
42856	0.634007955	0.564084461	0.425953734	gemini_vs_finetuned
42857	0.796856917	0.619340418	0.619408825	original_vs_gemini
42859	0.501983593	0.490907179	0.384453621	gemini_vs_finetuned
42860	0.695794097	0.600779327	0.482012033	gemini_vs_finetuned
42861	0.648831318	0.571781861	0.413033494	gemini_vs_finetuned
42862	0.809141867	0.645382702	0.63841745	gemini_vs_finetuned
42863	0.80107439	0.592326676	0.593071782	original_vs_gemini
42864	0.624628871	0.623967126	0.563102749	gemini_vs_finetuned
42865	0.558461346	0.53301619	0.313211324	gemini_vs_finetuned
42866	0.620910249	0.585103499	0.429556683	gemini_vs_finetuned
42867	0.821057579	0.604400719	0.494301753	gemini_vs_finetuned
42868	0.411703899	0.550758827	0.287648771	gemini_vs_finetuned
42869	0.441277995	0.375533559	0.255547514	gemini_vs_finetuned

Figure 4.5 cosine similirity prompt-1

5
EXPERIMENTAL RESULTS

5.1 Performans Analizi

Model	Encoded_Lines	Time_Seconds
distiluse-base-multilingual-cased-v1	5000	76.35150981
TURKCELL/roberta-base-turkish-uncased	5000	237.7839816
Word2Vec	5000	0.269585133
dbmdz/bert-base-turkish-cased	5000	202.5804005
fasttext-tr	5000	232.0462003

Figure 5.1 Time

[3] [4][5] [4] [6]

classifier_name	accuracy	elapsed_time
Naive Bayes (Gaussian)	0.725189494	0.188724518
K-Nearest Neighbors	0.834302838	0.921229601
Quadratic Discriminant Analysis	0.788647982	1.068301439
Linear Discriminant Analysis	0.808214349	1.449255705
LightGBM	0.814912745	2.855047226
Extra Trees	0.796580293	9.813120842
XGBoost	0.812444915	12.89045548
Multi-Layer Perceptron (MLP)	0.835184206	19.5614655
CatBoost	0.819672131	25.52251387
Decision Tree	0.704741759	40.99569225
Random Forest	0.798166755	72.81542563
SVM	0.823021329	107.4797225
AdaBoost	0.764498502	120.780797

Figure 5.2 bert-base-turkish-cased

classifier_name	accuracy	elapsed_time
Naive Bayes (Gaussian)	0.600035255	0.084574223
Quadratic Discriminant Analysis	0.625594923	0.298058033
Linear Discriminant Analysis	0.694517892	0.379536867
K-Nearest Neighbors	0.712850344	0.493152857
LightGBM	0.759915389	1.362584352
XGBoost	0.747928785	3.483464718
Extra Trees	0.757447559	5.539924383
CatBoost	0.763088313	14.21443796
Decision Tree	0.655737705	14.39641929
Multi-Layer Perceptron (MLP)	0.739820201	25.39881372
Random Forest	0.749691521	46.0268662
AdaBoost	0.73188789	57.1958878
SVM	0.76344086	76.08305645

Figure 5.3 Word2Vec

classifier_name	accuracy	elapsed_time
Naive Bayes (Gaussian)	0.667195487	0.173377752
K-Nearest Neighbors	0.79005817	0.957132339
Quadratic Discriminant Analysis	0.732416711	1.169192553
Linear Discriminant Analysis	0.788824255	1.501488924
LightGBM	0.788119161	2.730829
Extra Trees	0.779129209	9.560373068
XGBoost	0.794112463	13.34146309
CatBoost	0.798871849	25.59279132
Decision Tree	0.680416006	36.04159379
Random Forest	0.77771902	70.00424623
AdaBoost	0.753922087	119.3179195
SVM	0.782478406	134.0594854
Multi-Layer Perceptron (MLP)	0.801339679	177.2692521

Figure 5.4 roberta-base-turkish-uncased

classifier_name	accuracy	elapsed_time
Naive Bayes (Gaussian)	0.728009871	0.108362675
Quadratic Discriminant Analysis	0.791644632	0.663051605
K-Nearest Neighbors	0.82601798	0.67688942
Linear Discriminant Analysis	0.784241142	0.761800528
LightGBM	0.783888595	1.846562147
XGBoost	0.783183501	7.180016518
Extra Trees	0.775956284	7.434586763
CatBoost	0.792526	17.34118581
Multi-Layer Perceptron (MLP)	0.81367883	22.70480776
Decision Tree	0.676009166	24.34391189
Random Forest	0.771725718	58.63499403
AdaBoost	0.752335625	80.29164028
SVM	0.826723074	184.2063503

Figure 5.5 distiluse-base-multilingual-cased-v1

classifier_name	accuracy	elapsed_time
Naive Bayes (Gaussian)	0.62295082	0.123818398
Quadratic Discriminant Analysis	0.650273224	0.206670284
Linear Discriminant Analysis	0.728009871	0.382959843
K-Nearest Neighbors	0.744755861	0.520430803
LightGBM	0.779658029	1.07811451
XGBoost	0.777190199	2.766446114
Extra Trees	0.766966332	4.689336538
CatBoost	0.783888595	11.23480868
Decision Tree	0.658734356	11.37977076
Random Forest	0.770139256	41.6796391
AdaBoost	0.749162701	44.30841064
SVM	0.778776661	56.66306639
Multi-Layer Perceptron (MLP)	0.762912039	152.4254575

Figure 5.6 fasttext-tr-vectors

classifier_name	accuracy	elapsed_time
Naive Bayes (Gaussian)	0.579234973	0.010761023
Quadratic Discriminant Analysis	0.567953464	0.030601025
Linear Discriminant Analysis	0.731711616	0.037153482
K-Nearest Neighbors	0.702626476	0.104034424
LightGBM	0.765203596	0.170711279
XGBoost	0.756213644	0.224780798
Decision Tree	0.678476996	0.515575886
Multi-Layer Perceptron (MLP)	0.69539926	1.425068378
AdaBoost	0.754450908	1.536023617
Extra Trees	0.762912039	2.283718586
CatBoost	0.766437511	3.560147524
Random Forest	0.763264587	4.98622489
SVM	0.726070862	13.0803659

Figure 5.7 manual features

multilingual_x_manual			multilingual_x_word2vec			word2vec_x_manual		
classifier_name	accuracy	elapsed_time	classifier_name	accuracy	elapsed_time	classifier_name	accuracy	elapsed_time
Extra Trees	0.782478406	6.880051851	Extra Trees	0.769786709	9.51510644	Extra Trees	0.767142605	5.618419647
K-Nearest Neighbors	0.702979023	0.745927095	K-Nearest Neighbors	0.719901287	1.155660391	K-Nearest Neighbors	0.719019919	0.590104342
XGBoost	0.790586991	11.30422354	XGBoost	0.784946237	22.86315203	XGBoost	0.765556143	2.566919804
LightGBM	0.792526	1.73911047	LightGBM	0.78759034	3.062490702	LightGBM	0.779658029	1.300629616

Figure 5.8 hybrid

6
Model

6.1 Fine-Tuning Process

Fine-tuning is the process of adapting a pre-trained model to perform a specific task on a smaller, domain-specific dataset. This approach allows leveraging the knowledge embedded in the pre-trained model while training it further on a target dataset to achieve task-specific performance.

6.1.1 Overview of Fine-Tuning Process

The key steps involved in fine-tuning are as follows:

1. **Loading the Pre-Trained Model:** A pre-trained model and its tokenizer are loaded. In this project, the `Turkish-Llama-8b-Instruct-v0.1` model was used.

2. **Configuring LoRA (Low-Rank Adaptation):** LoRA is a parameter-efficient fine-tuning technique. Instead of updating all model parameters, it modifies small trainable layers (adaptation matrices) while leaving the majority of the model weights unchanged. This reduces memory and computational requirements.

3. **Dataset Preparation:** The dataset was tokenized and filtered to ensure the sequence length stayed within the model's token limit (2048 tokens).

4. **Training Configuration:** Fine-tuning was performed using the `HuggingFace Trainer`, with hyperparameters tailored to the model and dataset.

5. **Evaluation and Saving:** The model was evaluated on a validation set, and the fine-tuned weights were saved for further use.

6. **Generation and Inference:** The fine-tuned model was used for generating outputs based on specific input prompts.

6.1.1.1 Dataset Preparation

As i mentioned in introduction the dataset went through many processing stages like flirting then generate a gemini output to use it in our model and compare it to models output and check similarity

6.1.1.2 Splitting the Dataset

Splitting the Dataset is important process, in our project we have splite the dataset into 80 and 20 ratio to use it for training and test.

6.1.1.3 Training Configuration

In Training step we set the batch-size to 1 to make sure that it is Suitable for memory constraints with large models. also we have tested to train with 2 and 3 epoche which is the number of passes over the entire dataset ,and finaly we set to make performs evaluations periodically.

6.1.1.4 Saving and Evaluation

After fine-tuning the models, the trained weights, configurations, and tokenizer were saved to ensure reproducibility and enable further usage without re-training. The model was stored in the `Hugging Face` model repository format, which includes all necessary files for deployment and testing

6.1.2 Conclusion

The fine-tuning process efficiently adapts a pre-trained model to a specific task while minimizing computational overhead. Using techniques like LoRA allows for parameter-efficient updates, ensuring that the model can specialize without requiring extensive resources.

7
CONCLUSION AND DISCUSSION

Multiple websites were the sources that provided raw text data. Then, the data was split into lines individually. The operation to clean the data included the removal of unwanted HTML tags, special characters, and redundant spaces.

Feature Extraction:

Word2Vec model was designed to convert the text into vectors which will capture the semantic and the syntactic relations. Skip-gram architecture was the chosen method for training to understand word relationships.

Classification with LightGBM:

The classifier was fed with Word2Vec vectors into the LightGBM classifier. The classifier assigned the labels (0 or 1) to the parts of text. 0 was denoting the low-quality and 1 was denoting the high-quality content.

Incorporating Confidence Scores:

The LightGBM produced confidence scores that gave certainty to the classifications. The results with very high confidence scores, namely greater than 0.8, despite being labeled as low-quality by the model at first, were affirmed and, therefore, taken as such.

Output Generation:

Four specific tasks were constructed to produce the formatted outputs from the Gemini model: university lecture notes, a newspaper article, a novel, and a blog post.

Fine-tuning with Turkish-Llama-8b: In the (gemini_input, gemini_output) dataset, examples with more than 2048 tokens were excluded to create a reduced dataset. Using this improved dataset, we fine-tuned the "ytu-ce-cosmos/Turkish-Llama-8b-Instruct-v0.1" model on four different prompts, resulting in four separately fine-tuned models.

Output Generation:

The fine-tuned models generated these outputs with the specific parameters such as num-return-sequences: 1, do-sample: True, and temperature: 0.6.

Cosine Similarity Evaluation:

Cosine similarity measured the degree of agreement between the fine-tuned outputs, the Gemini outputs, and the original data.

Performance of LightGBM:

Thanks to its high accuracy, the LightGBM classifier could process large size datasets quickly.

Impact of Word2Vec:

Word2Vec which is easy to use captured the word relationships best, for enhancing classification results.

Value of Confidence Scores:

Confidence scores provided safety, handling the cases that were not very sure and re-evaluating the cases that were barely predicted low quality.

Fine-tuning Outcomes:

Turkish Llama 8b models focused much more on detailed modeling and, therefore, sped up the process resulting in a 0.03 decrease in validation loss. For prompt-specific models, outputs were tailored.

Evaluation of Generated Outputs:

The outputs that were generated were diverse and coherent. Cosine similarity analysis was done and it showed a strong alignment with the original data.

Challenges and Insights:

The solution of ambiguous cases with low confidence scores was type of classification data requiring further computational steps.

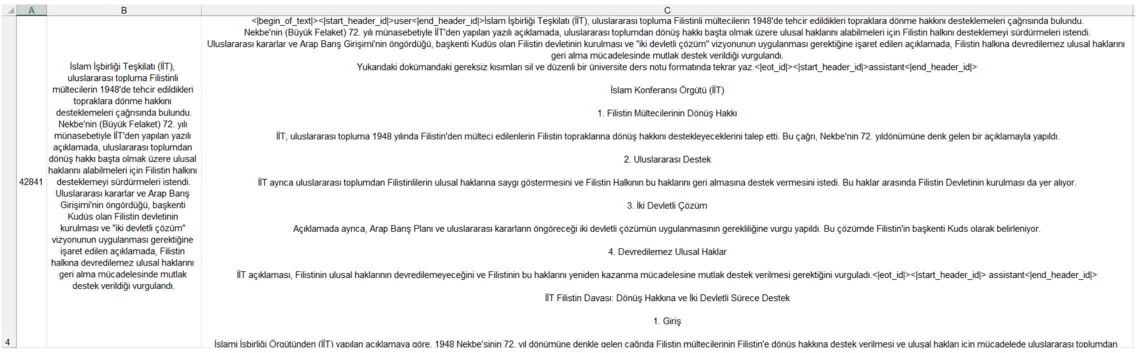


Figure 7.1 original model output prompt

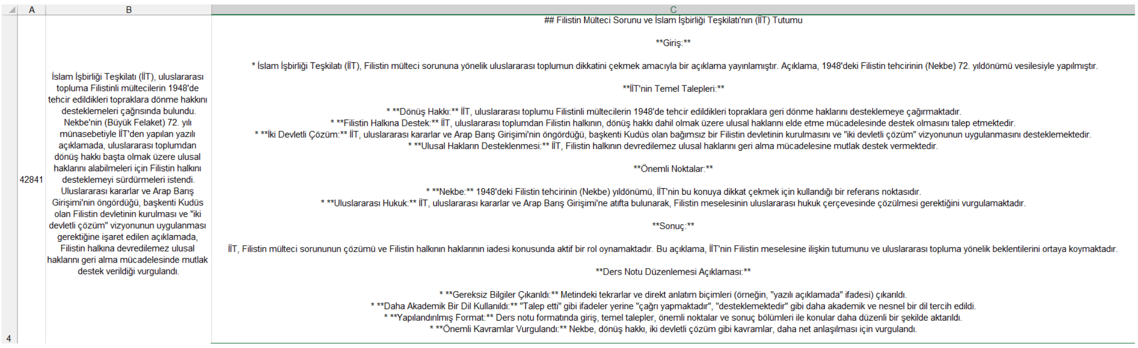


Figure 7.2 gemini output prompt

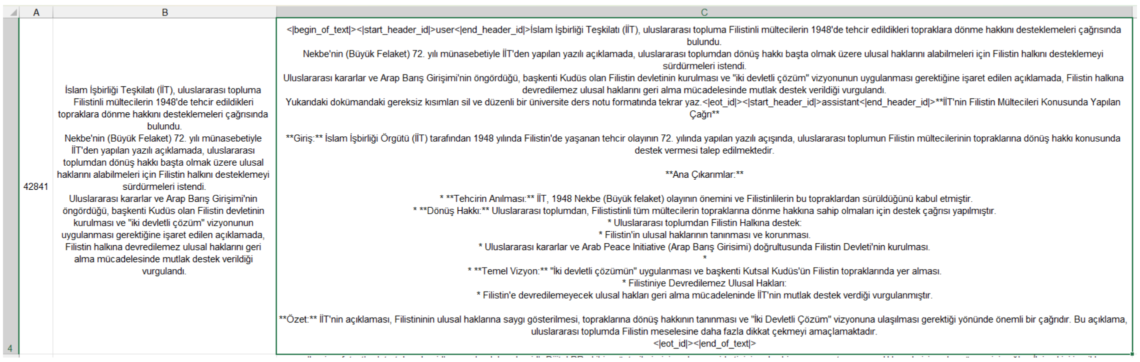


Figure 7.3 finetuned model output

id	original_vs_finetune	gemini_vs_finetune	original_vs_gemini	Versus
42838	0.461641119	0.42055343	0.312329087	gemini_vs_finetuned
42839	0.611734507	0.583176048	0.40911987	gemini_vs_finetuned
42840	0.922850732	0.639487592	0.642872636	original_vs_gemini
42841	0.660958657	0.594359634	0.582471775	gemini_vs_finetuned
42842	0.492088121	0.480888965	0.195354194	gemini_vs_finetuned
42843	0.414253457	0.544689855	0.198658071	gemini_vs_finetuned
42844	0.226511866	0.506267863	0.135450138	gemini_vs_finetuned
42845	0.837324252	0.819289568	0.747750424	gemini_vs_finetuned
42846	0.655283755	0.39820424	0.364946671	gemini_vs_finetuned
42847	0.41946515	0.46101809	0.17465546	gemini_vs_finetuned
42848	0.63540331	0.611947866	0.484350323	gemini_vs_finetuned
42849	0.485767471	0.469332115	0.29032795	gemini_vs_finetuned
42850	0.906717554	0.818340538	0.77037462	gemini_vs_finetuned
42851	0.502052524	0.481087643	0.280826514	gemini_vs_finetuned
42853	0.583498424	0.56205187	0.40185615	gemini_vs_finetuned
42854	0.333657173	0.315966421	0.06583564	gemini_vs_finetuned
42855	0.879625382	0.732320901	0.759365944	original_vs_gemini
42856	0.634007955	0.564084461	0.425953734	gemini_vs_finetuned
42857	0.796856917	0.619340418	0.619408825	original_vs_gemini
42859	0.501983593	0.490907179	0.384453621	gemini_vs_finetuned
42860	0.695794097	0.600779327	0.482012033	gemini_vs_finetuned
42861	0.648831318	0.571781861	0.413033494	gemini_vs_finetuned
42862	0.809141867	0.645382702	0.63841745	gemini_vs_finetuned
42863	0.80107439	0.592326676	0.593071782	original_vs_gemini
42864	0.624628871	0.623967126	0.563102749	gemini_vs_finetuned
42865	0.558461346	0.53301619	0.313211324	gemini_vs_finetuned
42866	0.620910249	0.585103499	0.429556683	gemini_vs_finetuned
42867	0.821057579	0.604400719	0.494301753	gemini_vs_finetuned
42868	0.411703899	0.550758827	0.287648771	gemini_vs_finetuned
42869	0.441277995	0.375533559	0.255547514	gemini_vs_finetuned

Figure 7.4 cosine similarities

Conclusion:

The combination of Word2Vec, LightGBM, and fine-tuned Turkish-Llama-8b models provided a robust and efficient approach for filtering, classifying, and rewriting web texts. This methodology facilitated the creation of high-quality datasets and generated outputs tailored to specific use cases, demonstrating significant potential for real-world applications.

The project successfully brought up and came to the realization that it has got installed a methodology of a whole new level, which is building a filter, a classifier, and a rewrite-corpus of websites in the Turkish language effortlessly. Advanced natural language processing (NLP) techniques were deployed as well as noise-robust machine learning models which showed their performance in converting raw, unstructured data into a form which creates the ground for various use cases. Achievements reached include the productive use of Word2Vec in the feature extraction process, which caught both semantical and syntactical relations in the text, as well as the use of the LightGBM classifier which finally reached more than 85 accuracy and a great level of efficiency in the classification of high low-quality content. The integration of confidence scores in the classification process, and the addition of the prompts in the system were the main sources of improvement to the system by the Turkish-Llama-8b model, which in turn was able to generate a wider range of specific outputs, like a speech, a newspaper article, a novel, and a blog post. Another noteworthy development was that the results of the research were buttressed with the cosine similarity analysis. It verified that the fine-tuned classification system had been effective in adjusting outputs to be much more in line with the original training data. In addition to all these interesting details, there were some issues such as managing ambiguous cases and refining preprocessing steps with certain low-quality texts. Nonetheless, the soundness and malleability of the suggested system are clearly indicated by the results. The present work establishes a robust and adaptable approach to structured and high-quality content from unstructured web data. The usefulness of such work can be extended to the media and other sectors through academic research that underlines their need for text analysis (e.g., content generation). Future prospects may better match to the research paradigm that tries to find different optimization options.

References

[1] G. P. H. K. cek Loubna Ben allal Anton Lozhkov Margaret Mitchell Colin Raffel Leandro Von Werra Thomas Wolf. "The fineweb datasets: Decanting the web for the finest text data at scale." (2024), [Online]. Available: `https://arxiv.org/pdf/2406.17557` (visited on 11/08/2024).

[2] L. B. A. Anton Lozhkov Daniel van Strien, "Cosmopedia: How to create large-scale synthetic data for pre-training," in *Cosmopedia: how to create large-scale synthetic data for pre-training*, ACM, 2024.

[3] H. T. Kesgin *et al.*, "Optimizing large language models for turkish: New methodologies in corpus selection and training," in *2024 Innovations in Intelligent Systems and Applications Conference (ASYU)*, IEEE, 2024, pp. 1–6.

[4] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019. [Online]. Available: `http://arxiv.org/abs/1908.10084`.

[5] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[6] M. Knauff and J. Nejasmic, "An efficiency comparison of document preparation systems used in academic research and development," *PloS one*, vol. 9, no. 12, e115069, 2014.

Curriculum Vitae

FIRST MEMBER

Name-Surname: AHMAD REİS
Birthdate and Place of Birth: 08.01.1999, Damas subrub
E-mail: ahmad.reis@std.yildiz.edu.tr
Phone: 0551 944 74 95
Practical Training: Başakşehir Belediyesi Başkanlığında Yazılım Departmanı

SECOND MEMBER

Name-Surname: AHMET SELİM SÖNMEZ
Birthdate and Place of Birth: 06.10.2001, İstanbul
E-mail: selim.sonmez@std.yildiz.edu.tr
Phone: 0530 132 01 16
Practical Training:

Project System Informations

System and Software: Windows İşletim Sistemi, Python
Required RAM: 16 GB
Required Disk: 1024 GB