T.R.
SAKARYA UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ENHANCING MOBILE APPLICATION AGE RATINGS USING NATURAL LANGUAGE PROCESSING TECHNIQUES

MSc THESIS

Muhammed Emin IDELBI

Computer And Informatics Engineering Department

AUGUST 2023

T.R.
SAKARYA UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ENHANCING MOBILE APPLICATION AGE RATINGS USING NATURAL LANGUAGE PROCESSING TECHNIQUES

MSc THESIS

Muhammed Emin IDELBI

Computer And Informatics Engineering Department

Thesis Advisor: Assist. Prof. Dr. Muhammed Fatih ADAK

AUGUST 2023

The thesis work titled "Enhancing Mobile Application Age Ratings Using Natural Language Processing Techniques" prepared by Muhammed Emin IDELBI was accepted by the following jury on 10/08 /2023 by unanimously/majority of votes as a MSc THESIS in Sakarya University Graduate School of Natural and Applied Sciences, Computer and Informatics Engineering department.

Thesis Jury

Head of Jury :	**Assist. Prof. Dr. M. Fatih ADAK** (Advisor) Sakarya University	[illegible]
Jury Member :	**Assist. Prof. Dr. Kayhan AYAR** Sakarya University	[illegible]
Jury Member :	**Assist. Prof. Dr. Fatih VARÇIN** Sakarya University of Applied Sciences	[illegible]

STATEMENT OF COMPLIANCE WITH THE ETHICAL PRINCIPLES AND RULES

I declare that the thesis work titled "ENHANCING MOBILE APPLICATION AGE RATINGS USING NATURAL LANGUAGE PROCESSING TECHNIQUES", which I have prepared in accordance with Sakarya University Graduate School of Natural and Applied Sciences regulations and Higher Education Institutions Scientific Research and Publication Ethics Directive, belongs to me, is an original work, I have acted in accordance with the regulations and directives mentioned above at all stages of my study, I did not get the innovations and results contained in the thesis from anywhere else, I duly cited the references for the works I used in my thesis, I did not submit this thesis to another scientific committee for academic purposes and to obtain a title, in accordance with the articles 9/2 and 22/2 of the Sakarya University Graduate Education and Training Regulation published in the Official Gazette dated 20.04.2016, a report was received in accordance with the criteria determined by the graduate school using the plagiarism software program to which Sakarya University is a subscriber, I accept all kinds of legal responsibility that may arise in case of a situation contrary to this statement.

(03/07/2023)

Muhammed Emin IDELBI

To my parents, sister and my best friends...

ACKNOWLEDGMENTS

First and foremost I am extremely grateful to God and then to my supervisor, Assist. Prof. Dr. Muhammed Fatih ADAK for his invaluable advice, continuous support, and patience during my MSc study. Their immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. Finally, I would like to express my gratitude to my parents, my sister and my friends. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

Muhammed Emin IDELBI

TABLE OF CONTENTS

Sayfa

ABBREVIATIONS

BoW	: Bag of words
DT	: Decision trees
GloVe	: Global vectors
KNN	: Nearest neighbor
LDA	: Latent dirichlet allocation
LR	: Logistic regression
MLP	: Multi-layer perceptron
NB	: Naive bayes
NLP	: Natural language processing
NLTK	: Natural language toolkit
RF	: Random forests
RNN	: Recursive neural networks
SIF	: Smooth inverse frequency
SVM	: Support vector machines
TF-IDF	: Term frequency – inverse document frequency
VSM	: Vector space model

SYMBOLS

k	: Number of words
V	: Vocabulary size
V′	: Output vector
w	: Context of word
exp	: Exponential
T	: Transpose

LIST OF TABLES

Page

LIST OF FIGURES

ENHANCING MOBILE APPLICATION AGE RATINGS USING NATURAL LANGUAGE PROCESSING TECHNIQUES

SUMMARY

As the huge number of mobile applications in stores increases, it gets difficult to verify all the information about each application. Especially Age Rating. For instance, Play Store and App Store have an algorithm to determine the age rating of an application, the algorithm is applied after the developer is asked questions about the application such as if the application includes any violence scenes and if It is frequent or not or containing offensive language or crude humor and many other questions, also rules of each country are put into account with the developer's answers while giving the appropriate rating. The challenge here is that a developer may answer any of the questions wrongly and this would affect the process of the rating of the application negatively, either by making it accessible by users who must not use it because of age restriction or limiting users from reaching the application while the application is suitable for them.

In this thesis, a novel method is presented to classify mobile apps by analyzing their app store descriptions. The study utilizes a dataset obtained from the Apple App Store, consisting of over 365,000 app descriptions. The research aims to accurately classify and age-rate apps, enabling users to find apps suitable for their age group and preferences, and helping developers identify direct competitors and market trends.

The approach consists of three major steps: data pre-processing, vectorization (using word embeddings and Bag-of-Words), and classification. Text pre-processing techniques such as lowercasing, tokenization, removal of non-ASCII characters, numbers, URLs, and stop-word removal are applied. Two vectorization methods are used: Bag-of-Words with a maximum of 1000 most-frequent words and word embeddings (GloVe, Word2Vec, fastText). The generated word embeddings represent the entire app description using simple unweighted averaging. Classification is performed using various algorithms such as Support Vector Machines, Multi-layer Perceptron, Random Forests, Nearest Neighbor, AdaBoost, Decision Trees and Logistic Regression. The performance of these classifiers is evaluated using accuracy, recall, precision, and F-measure.

The study compares word embeddings with topic modeling (LDA) and demonstrates that word embeddings outperform LDA in app classification tasks. The limitations of LDA with short text inputs are highlighted, emphasizing the advantage of word embeddings in capturing semantic meaning and achieving better classification performance. Furthermore, the study compares word embeddings with the bag-of-words approach (VSM) and shows that VSM outperform word embeddings in capturing the semantic relationships and similarities between words in app descriptions.

Among the word embedding models (GloVe, Word2Vec, fastText), GloVe performed the best, potentially due to its more comprehensive vocabulary. The inclusion of character n-grams in fastText did not significantly contribute to classification accuracy. Overall, the proposed approach has practical implications for users and developers, enabling better app discovery and market insights. Future directions include conducting more extensive analysis with expert-generated categorizations and developing a working prototype to bring the research closer to real-world application. In summary, the thesis presents an effective approach for app classification based on app store descriptions, utilizing word embeddings, and achieving accurate age rating predictions.

DOĞAL DİL İŞLEME TEKNİKLERİ KULLANILARAK MOBİL UYGULAMA YAŞ DERECELENDİRMELERİNİN İYİLEŞTİRİLMESİ

ÖZET

Mobil uygulama mağazalarındaki uygulama sayısının hızla artmasıyla, her bir uygulama hakkındaki tüm bilgileri doğrulamak zorlaşmaktadır. Özellikle Yaş Derecelendirmesi konusunda bu durum daha da zorlaşmaktadır. Örneğin, Play Store ve App Store'un bir uygulamanın yaş derecelendirmesini belirlemek için bir algoritması vardır. Geliştiricilere uygulama hakkında sorular sorulur, örneğin uygulama şiddet içeriyor mu, sıklığı nedir, hakaret içeren dil veya açık saçık espri içeriyor mu gibi sorular sorulur. Ayrıca her ülkenin kuralları da geliştiricinin verdiği cevaplarla birlikte uygun derecelendirmeyi belirlemede dikkate alınır. Buradaki zorluk, bir geliştiricinin soruların herhangi birini yanlış cevaplaması durumunda, bu uygulamanın derecelendirme sürecini olumsuz etkileyebilmesi veya uygulamayı kullanmaması gereken yaş grubundaki kullanıcıların erişimini sınırlamasıdır. Mobil uygulamalara yaş derecelendirmesi atamak için gösterilen çabalara rağmen, mevcut yaklaşımlar çeşitli sınırlamalardan muzdariptir. Bu sınırlamalar, önerilen NLP tabanlı yaklaşım gibi daha etkili ve güvenilir bir çözüme olan ihtiyacı vurgulamaktadır. Mevcut çözümlerin temel sınırlamalarından bazıları şunlardır: Öznellik ve Tutarsızlık: Mevcut yaş derecelendirmesi atama süreci, manuel değerlendirmeye ve geliştiricilerin öznel yargılarına dayanmaktadır. Farklı geliştiriciler kriterleri farklı yorumlayabileceğinden, bu öznellik yaş derecelendirmesi atamalarında tutarsızlıklara yol açabilir. Bu tutarsızlık, mobil uygulamalar için yaş derecelendirmelerinin güvenini ve güvenilirliğini baltalamaktadır. Zaman Alıcı ve Zahmetli: Yaş derecelendirmelerini atamak için kullanılan anket tabanlı yaklaşım, geliştiriciler için zaman alıcı ve külfetli olabilir. Uygulamanın içeriğiyle ilgili birden çok soruyu yanıtlama süreci, büyük çaba gerektirir ve geliştiricilerin doğru bilgi vermesini engelleyebilir. Bu, yanlış veya eksik yaş derecelendirmesi atamalarına neden olabilir. Sürekli İzleme Eksikliği: Bir uygulamaya bir kez yaş sınırı atandığında, içeriği genellikle sınırlı veya sürekli olarak izlenmez. Bu, ilk yaş derecelendirmesi atamasından sonra uygulamada yapılan herhangi bir değişikliğin veya güncellemenin, uygulamanın farklı yaş grupları için uygunluğu üzerindeki etkisinin yeterince değerlendirilmeyebileceği anlamına gelir. Sonuç olarak, bir zamanlar belirli bir yaş grubu için uygun görülen bir uygulama, içerik güncellemeleri nedeniyle artık uygun olmayabilir. Kültürel Farklılıkların Sınırlı Olarak Dikkate Alınması: Mevcut yaş derecelendirme sistemleri genellikle kültürel farklılıkların ve hassasiyetlerin kapsamlı bir şekilde değerlendirilmesinden yoksundur. Farklı ülkeler ve bölgeler, farklı yaş grupları için neyin uygun olduğu konusunda farklı normlara ve değerlere sahip olabilir. Mevcut sistemlerin herkese uyan tek yaklaşımı, bu kültürel nüansları doğru bir şekilde yansıtmayabilir, bu da tutarsızlıklara ve uygun olmayan içeriğe potansiyel erişime yol açabilir. Bağlamsal Bilgileri Yakalayamama:

Mevcut yaş derecelendirmesi atama yöntemleri, öncelikle geliştiriciler tarafından belirli sorulara verilen yanıtlar şeklinde sağlanan açık bilgilere dayanmaktadır. Ancak, genellikle uygulamanın içeriğinin bağlamsal bilgilerini ve inceliklerini yakalayamazlar. Bu, uygulama içeriğinin bazı yönleri gözden kaçabileceği veya yeterince dikkate alınmayabileceği için, yetersiz veya yanlış yaş derecelendirme atamalarına neden olabilir. Sınırlı Ölçeklenebilirlik: Mobil uygulamaların sayısı hızla artmaya devam ettikçe, mevcut manuel değerlendirme yaklaşımının ölçeklendirilmesi giderek daha zor hale geliyor. Çok sayıda başvuru, kapsamlı bir şekilde incelemeyi ve her birine doğru yaş derecelendirmesi atamayı zorlaştırıyor. Bu ölçeklenebilirlik sorunu, yaş derecelendirme atamalarının gecikmesine veya uygulamaların farklı yaş grupları için uygunluğunun değerlendirilmesinde gözden kaçmasına neden olabilir.

Bu tez, mobil uygulamaları app store açıklamalarına dayalı olarak sınıflandırmak için yenilikçi bir yaklaşım önermektedir. Çalışma, Apple App Store'dan elde edilen 365.000'den fazla uygulama açıklamasını içeren bir veri kümesinden yararlanmaktadır. Araştırma, uygulamaları doğru bir şekilde sınıflandırmayı ve yaş derecelendirmesi yapmayı amaçlamaktadır, böylece kullanıcılar yaş grubu ve tercihlerine uygun uygulamaları bulabilir ve geliştiriciler doğrudan rakipleri ve pazar trendlerini belirleyebilir.

Yaklaşım, veri ön işleme, vektörleştirme (kelime gömme “Word Embeddings” ve Bag-of-Words kullanarak) ve sınıflandırma olmak üzere üç ana adımdan oluşmaktadır. Küçük harfe dönüştürme, belirteçlere ayırma, ASCII olmayan karakterlerin, sayıların, URL'lerin ve etkisiz kelimelerinin kaldırılması gibi metin ön işleme teknikleri uygulanır. İki vektörleştirme yöntemi kullanılır: en çok kullanılan 1000 kelimeye dayalı Bag-of-Words ve kelime gömme modelleri (GloVe, Word2Vec, fastText). Oluşturulan kelime gömmeleri, basit ağırlıksız ortalama kullanılarak tüm uygulama açıklamasını temsil eder. Sınıflandırma, Support Vector Machines, Naive Bayes, Multi-layer Perceptron, Random Forests, Nearest Neighbor, AdaBoost, Decision Trees ve Logistic Regression gibi çeşitli algoritmalar kullanılarak gerçekleştirilir. Sınıflandırıcıların performansı, hassasiyet, hatırlama, F-ölçütü ve doğruluk kullanılarak değerlendirilir.

Çalışma, kelime gömmelerini konu modellemesi (LDA) ile karşılaştırır ve kelime gömmelerinin uygulama sınıflandırma görevlerinde LDA'ya göre daha iyi performans gösterdiğini gösterir. LDA'nın kısa metin girdileriyle sınırlamaları vurgulanır ve kelime gömmelerinin anlamsal anlamı yakalama ve daha iyi sınıflandırma performansı elde etmedeki avantajı vurgulanır. Ayrıca, kelime gömmelerini Bag-of-Words yaklaşımı (VSM) ile karşılaştırır ve VSM uygulama açıklamalarındaki kelime ilişkilerini ve benzerlikleri yakalamada kelime gömmelerininden daha iyi performans gösterdiğini gösterir.

Kelime gömme modelleri (GloVe, Word2Vec, fastText) arasında GloVe'nin en iyi performansı gösterdiği belirlenmiştir, bu durum muhtemelen daha kapsamlı bir kelime hazinesine sahip olmasından kaynaklanmaktadır. fastText'te karakter n-gramlarının dahil edilmesi, sınıflandırma doğruluğuna önemli ölçüde katkı sağlamamıştır.Genel olarak, önerilen yaklaşım, kullanıcılara ve geliştiricilere daha iyi uygulama keşfi ve pazar içgörüleri sağlayarak pratik sonuçlar sunmaktadır. Gelecek yönelimler arasında uzmanların oluşturduğu kategorizasyonlarla daha kapsamlı analizler yapma vearaştırmayı gerçek dünya uygulamasına yaklaştırmak için bir çalışma prototipi geliştirme yer almaktadır. Özet olarak, bu tez, uygulama mağazası açıklamalarına dayalı olarak uygulama sınıflandırması için etkili bir yaklaşım sunar, kelime gömmelerini kullanır ve doğru yaş derecelendirme tahminleri elde eder.

1. INTRODUCTION

The advent of mobile devices and widespread internet connectivity has revolutionized the way children engage with technology. Nowadays, it is increasingly common for children to own or have access to mobile devices equipped with internet connections, opening up a world of possibilities and information at their fingertips. Furthermore, the continuous growth of mobile application marketplaces, housing an ever-expanding array of apps, presents both opportunities and challenges for young users.

The statistics surrounding children's internet usage underscore the importance of addressing the issue of age-appropriate content in mobile app marketplaces. In just a few years, internet usage among children aged 6-15 has witnessed a remarkable surge, rising from 50.8% in 2013 to a staggering 82.7% in 2021 [36] . This exponential growth in online activity among young users demands careful attention to safeguard them from potential dangers and ensure their digital experiences are both educational and secure.

Among the various activities carried out by children using the internet, two have emerged as prominent trends. Online classes have rapidly gained popularity, with 86.2% of children regularly participating in virtual learning sessions. This shift towards remote education, particularly during the global pandemic, highlights the pivotal role that technology plays in facilitating educational opportunities for young learners. Additionally, a significant number of children, accounting for 83.6%, turn to the internet for homework assistance and independent learning. The accessibility of online resources allows them to explore various subjects and expand their knowledge beyond the confines of traditional classroom settings.

Another prevalent activity among children is gaming, with 66.1% of young users engaging in playing or downloading games. Gaming has become a major form of entertainment and engagement for children, providing them with immersive experiences and opportunities for skill development. However, it also necessitates ensuring that the games they access align with their developmental stages and do not expose them to inappropriate content.

To address these challenges and protect children from accessing unsuitable apps, age ratings have been introduced. Mobile app marketplaces typically assign age ratings based on a questionnaire completed by developers during the app upload process. This questionnaire probes the app's content, including aspects such as violence, mature themes, and frequency of specific content. However, the current approach to assigning age ratings can be time-consuming and subjective, leading to potential inconsistencies and inaccuracies.

In light of these considerations, this paper aims to propose a novel method for generating age ratings for applications by employing Natural Language Processing (NLP) algorithms. The core idea is to leverage the textual information provided in the description section of each application's page to automate and enhance the accuracy of age rating assignments. By utilizing NLP techniques, the proposed algorithm seeks to streamline the process for developers while ensuring that apps are accurately categorized according to their suitability for different age groups.

The subsequent sections of this paper will delve deeper into the existing literature, elucidate the classification algorithm developed in this research, present the findings and effectiveness of the proposed method, and conclude with key insights and recommendations for future work. Ultimately, the goal is to create a more efficient and reliable system for assigning age ratings to mobile applications, fostering a safer and more inclusive digital environment for children.

2. RELATED WORK AND MOTIVATION

2.1. Related Work

Al-Subaihin et al. [1] performed an empirical assessment of text-based app classification strategies, including topic modeling Latent Dirichlet Allocation (LDA) [3] and keyword feature extraction methods [10] . The study made use of a dataset of 12,664 mobile app descriptions taken from Google Play Store. The results showed that LDA-based approaches performed most efficiently in terms of numerical grouping quality.

Gorla et al. [13] introduced CHABADA, an approach for detecting anomalies between the stated and implemented behavior of Android apps. The authors used LDA to identify topics from mobile app descriptions. The retrieved subjects were passed into a K-means algorithm, which formed different app categories. Sensitive APIs controlled by user permissions were found inside each group. SVMs were used to identify outliers in API consumption. These anomalies were regarded as potentially harmful activity. CHABADA was tested on a database of over 22,500 Android apps. The prototype detected various irregularities and identified 56% of unique malware.

Ebrahimi et al [9] proposed an automated method for organizing mobile applications into more detailed categories of functionally connected application domains. To build quantitative semantic representations of app descriptions, the authors used word embeddings. These representations are then grouped to provide more coherent app categories. The study employs a dataset of 600 apps gathered from the Apple App Store. The results reveal that when app descriptions are vectorized using GloVe, a count-based model of word embeddings [24] , the classification algorithms perform best. The results are then validated by 12 individual participants using a dataset of Sharing Economy apps. The results show that GloVe combined with Support Vector Machines can produce app classifications that closely resemble human-generated classifications.

Sanz et al. [27] developed a machine learning technique for app classification in the Android app store. The goal was to manage the Android market and detect fraudulent applications. The proposed method made use of features retrieved from app source code, requested permissions, and meta-data like as ratings, size, and advertised permissions. The dataset of 820 apps collected from 7 different Google Play categories was then classified using multiple classification techniques. The results showed that Bayesian networks outscored other methods, with an AUC of 93%.

In their study, Lulu and Kuflik [18] classified apps based on their capabilities using unsupervised machine learning. App features were collected from their app store descriptions and then supplemented with material from professional forums. TF.IDF weighted vectors of words were then used to represent app features. WordNet was used to resolve synonymy relationships. The authors then created hierarchies of functionally-related apps using hierarchical clustering. The efficiency of the proposed method was proved using a dataset of 120 Google Play apps.

Phan et al. [25] advocated using hidden themes to improve the categorization representation of short and sparse text. To avoid noise, the hidden themes are learned from an external data set using seed selection.

Sahami et al. [26] presented an innovative similarity measuring method for small text samples that can also be demonstrated using a kernel function. This strategy, in particular, makes use of a Web search engine to augment original textual information, which can then be used for short and sparse text classification.

Furthermore, Yih et al. [34] modified the measuring method by incorporating an additional learning mechanism to increase measurement efficiency. Broder et al. [5] advocated extracting data from the top related search outcomes of a keyword from a Web search engine, and Shen et al. [29] investigated query classification using a Web directory. Cao et al. [6] proposed using Web information to improve query classification by augmenting both the contextual and local aspects of web inquiries.

Martin et al. [20] studied the effect of sampling bias on the results of app store mining with the focus on user reviews. In the context of mobile apps, k-means clustering was used in conjunction with topic modeling for detecting the abnormal behavior of apps [13] .

Maarek et al. [19] , who built reusable libraries by automatically extracting concepts from natural language code-related artifacts. Dumitru et al. [15] [8] crawled softpedia.com product raw descriptions more recently to identify feature descriptors, which they represented using a TF-IDF vector. They employed incremental diffusive clustering, which they later extended to construct feature models [7] .

Kawaguchi et al. [17] used LSA to explore open source software sources for common phrases in the source code, and Tian et al. [31] used LDA to cluster software from SourceForge, with the existing classification serving as the basis for truth.

Wang et al. [33] use collaborative tags and application descriptions to categorize software only based on the program profile page obtained in the repository. They use TF-IDF to extract essential classifying terms from the program profile page, which includes both the software description and tags. They then employ SVM to classify systems into a hierarchical category tree manually customized from the one provided by SourceForge.

Seneviratne et al. [28] use app meta data to detect malware and malicious apps early in their development. Vakulenko and Muller [32] also sought to categorize apps only based on their product descriptions in order to improve the app store's existing categorization using LDA. They employ LDA to extract features for machine learning-based categorization, then compare their results to the real classification as training and truth set.

Gorla et al. [13] developed a method for clustering applications based on their reported behavior, which outperformed existing app store classification (when used as a harmful app detector). This important work by Gorla et al. supplied us with motivation and preliminary evidence that an efficient app clustering technique based on the developers' textual app descriptions would be achievable.

Related to this is the work of Guzman and Maalej [14] , who addressed the issue of extracting characteristics from app evaluations, and the work of Martin et al. [21] who employed NLP on new release text to determine the type of release and its effect.

In order to cover the Arabic applications categorization in the Google Play app store, Fuad et al. [11] created a long-lasting classification system employing the Latent Dirichlet Allocation (LDA) approach of topic modeling. Their study demonstrates that textual app descriptions may effectively propose categories for categorizing Arabic mobile applications when utilizing automated classification techniques like topic modeling.

Kalaivani et al. [16] used Latent Dirichlet Allocation (LDA), a topic modeling approach to extract themes that correspond to a probability distribution across words. Two distinct sets of parameters were used for K-means document clustering, and the assessment outcomes of both clusters were studied. Additionally, used deep learning and machine learning classification methods. Their system was efficient in categorizing misclassified apps.

In their research, Singla et al. [30] provided an app classification system that generates a more precise categorization for a specific app in accordance with a given taxonomy using object detection and identification in photos connected with applications, as well as text-based metadata of the apps. They showed that the study can improve the classification accuracy of a text based app classifier.

Qiu et al. [23] aimed at a multiple classification problem. As a data set, they used an extensive list of app descriptions. then examined the data using a number of different machine learning algorithms, like CountVectorizer, TfIdfVectorizer. After some parameter optimizations, they got an accuracy of 60%.

Gallego et al. [12] Presented a new NLP-based feature extraction pipeline “TransFeatEx”, which combines the use of a RoBERTa-based model with the application of consolidated syntactic and semantic techniques. The pipeline is designed as a customizable, standalone service to be used either as a playground, experimentation tool or as a software component.

2.2. Motivation

The rapid growth of mobile application marketplaces and the increasing prevalence of children using mobile devices raise significant concerns regarding age-appropriate content. Ensuring that children are protected from accessing unsuitable apps becomes crucial to safeguard their well-being and provide a secure digital environment. Age ratings have been introduced as a means to address this issue, but the current manual assessment process is time-consuming, subjective, and prone to inconsistencies.

The motivation behind this research stems from the need to enhance the accuracy and efficiency of age rating assignments for mobile applications. By leveraging advancements in Natural Language Processing (NLP) algorithms, we aim to automate the age rating process and provide a more reliable and standardized approach. The use of NLP techniques allows us to analyze the textual information available in the description section of app pages and extract valuable insights to determine the appropriate age rating.

By developing a robust NLP-based classification algorithm, we can streamline the age rating assignment process for developers, saving them time and effort. Moreover, the automated nature of the proposed method ensures a more objective evaluation of the app's content, reducing the potential for human errors and inconsistencies.

This study also contributes to the larger objective of making the internet a safer and more inclusive place for children. By accurately categorizing mobile applications according to their suitability for different age groups, we can protect children from accessing content that may be inappropriate or harmful to their development. Additionally, providing a reliable age rating system allows parents and guardians to make informed decisions about the apps their children can access, fostering responsible digital engagement.

The potential impact of this research extends beyond the realm of age rating for mobile applications. The development and application of NLP algorithms in this context can serve as a foundation for future advancements in automated content evaluation and moderation. This research can pave the way for more efficient and effective systems for filtering and categorizing digital content, ensuring that users, especially children, are protected from harmful or unsuitable materials.

In conclusion, the motivation behind this research lies in the urgent need to improve the age rating assignment process for mobile applications. By harnessing the power of

NLP algorithms, we aim to automate and enhance the accuracy of age rating predictions, contributing to a safer and more inclusive digital environment for children.

2.3. Limitations of Existing Solutions

Despite the efforts made to assign age ratings to mobile applications, the current approaches suffer from several limitations. These limitations highlight the need for a more effective and reliable solution, such as the proposed NLP-based approach. Some of the key limitations of existing solutions are:

Subjectivity and Inconsistency: The current process of assigning age ratings relies on manual assessment and subjective judgments by developers. This subjectivity can lead to inconsistencies in age rating assignments, as different developers may interpret the criteria differently. This inconsistency undermines the trust and reliability of age ratings for mobile applications.

Time-consuming and Burdensome: The questionnaire-based approach used to assign age ratings can be time-consuming and burdensome for developers. The process of answering multiple questions about the content of the application requires considerable effort and may deter developers from providing accurate information. This can result in incorrect or incomplete age rating assignments.

Lack of Continuous Monitoring: Once an application is assigned an age rating, there is often limited or no continuous monitoring of its content. This means that any changes or updates made to the application after the initial age rating assignment may not be adequately assessed for their impact on the appropriateness of the app for different age groups. As a result, an application that was once deemed suitable for a particular age group may no longer be appropriate due to content updates.

Limited Consideration of Cultural Differences: The current age rating systems often lack comprehensive consideration of cultural differences and sensitivities. Different countries and regions may have varying norms and values regarding what is considered appropriate for different age groups. The one-size-fits-all approach of the current systems may not accurately reflect these cultural nuances, leading to inconsistencies and potential access to unsuitable content.

Inability to Capture Contextual Information: The current methods of assigning age ratings primarily rely on explicit information provided by developers in the form of responses to specific questions. However, they often fail to capture the contextual information and subtleties of the application's content. This can result in inadequate or inaccurate age rating assignments, as certain aspects of the application's content may be overlooked or not adequately considered.

Limited Scalability: As the number of mobile applications continues to grow rapidly, the current manual assessment approach becomes increasingly challenging to scale. The sheer volume of applications makes it difficult to thoroughly review and assign accurate age ratings to each one. This scalability issue can result in delayed age rating assignments or oversights in assessing the suitability of applications for different age groups.

Addressing these limitations requires a more automated, objective, and scalable approach, which the proposed NLP-based solution aims to provide. By leveraging NLP algorithms to analyze the textual information in the description section of app pages, we can overcome these limitations and enhance the accuracy, efficiency, and reliability of age rating assignments for mobile applications.

3. METHODS

3.1. Bag Of Words

The bag-of-words model is a way of representing text data when modelling text with machine learning algorithms.

The "Bag of Words" (BoW) model is a simple and commonly used technique in natural language processing (NLP) that represents text documents as a collection of words, disregarding grammar and word order. In this model, the frequency or presence of words in a document is used to create a numerical representation. The BoW model operates on the principle that the order of words in a document may not be as important as the occurrence of words themselves. It treats a document as a "bag" or a set of words, hence the name "Bag of Words." This approach provides a straightforward and efficient representation of text data. The process of creating a BoW representation involves a few steps. First, a vocabulary is created by collecting unique words from the entire corpus or set of documents. Each word in the vocabulary becomes a feature or dimension in the vector space representation. Then, for each document, a vector is created, where each dimension corresponds to a word in the vocabulary, and the value represents the frequency or presence of that word in the document. Typically, the BoW representation uses a count-based approach, where the vector values indicate the frequency of words. However, binary or presence-based representations can also be used, where the value is either 0 or 1, indicating whether a word is present or absent in the document.

The BoW model is a simple yet effective representation for various text-based tasks, such as text classification, sentiment analysis, and information retrieval. It allows for efficient computation and can handle large amounts of text data. However, it has limitations, such as the loss of word order and context, disregarding important semantic relationships between words.

3.1.1. Vector Space Model (VSM)

The Vector Space Model (VSM) is a widely used mathematical framework for representing and analyzing text documents in information retrieval and natural language processing. It treats documents as vectors in a high-dimensional space, where each dimension corresponds to a unique term or word. The core idea behind the VSM is to capture the semantic relationships between documents and queries by measuring the similarity or distance between their corresponding vectors. The assumption is that documents with similar content will have vectors that are close to each other in this vector space. The VSM representation begins with creating a vocabulary by collecting unique terms from the entire corpus or set of documents. Each term becomes a dimension in the vector space. Then, for each document, a vector is constructed, where each dimension represents a term in the vocabulary, and the value represents the importance or weight of that term in the document. There are several common techniques to determine the weights of terms in the vectors. One of the simplest approaches is the term frequency-inverse document frequency (TF-IDF) weighting, which assigns higher weights to terms that appear frequently in a document but infrequently across the corpus. This helps to emphasize important and discriminative terms. Once the vectors are constructed, various similarity measures can be used to compare documents or queries in the vector space. The most commonly used measure is cosine similarity, which calculates the cosine of the angle between two vectors. A higher cosine similarity indicates greater similarity between the documents or queries. The VSM has been successfully applied in various text-based tasks, including document retrieval, text classification, clustering, and recommendation systems. It provides a flexible and efficient framework for organizing and comparing text documents based on their content. However, the VSM also has its limitations. It treats terms as independent and disregards the context and word order within documents. Additionally, the VSM can be sensitive to noise, such as stop words or irrelevant terms that may affect the similarity calculations.

3.2. Word Embedings

Word embeddings are a powerful technique in natural language processing (NLP) that represent words as dense vector representations in a continuous space. Unlike traditional sparse representations, such as one-hot encoding, word embeddings capture semantic and syntactic relationships between words, enabling machines to better understand and process language. The key idea behind word embeddings is to learn word representations based on the contextual information from a large corpus of text data. The embeddings are typically generated using neural network models, such as Word2Vec, GloVe, or FastText. These models aim to map words to vector representations in a way that preserves semantic relationships and captures linguistic regularities. Word embeddings possess several desirable properties. One of them is the ability to capture semantic similarity. Words with similar meanings or in related contexts tend to have similar vector representations in the embedding space. For example, the vectors of "king" and "queen" are expected to be close together, indicating their semantic similarity. Word embeddings also exhibit interesting linear relationships. By performing vector arithmetic operations in the embedding space, we can explore analogical relationships. For instance, by subtracting the vector of "man" from "woman" and adding it to the vector of "king," we obtain a vector close to the vector representation of "queen." This ability to capture analogies showcases the semantic understanding encoded within word embeddings. These dense vector representations have found applications in various NLP tasks. They enhance the performance of tasks such as sentiment analysis, named entity recognition, machine translation, text classification, and document clustering. Embeddings provide valuable contextual information and improve the generalization ability of models. Pretrained word embeddings are widely available and offer a convenient way to leverage the semantic knowledge captured from extensive text corpora. They can be utilized out of the box or fine-tuned on specific downstream tasks, depending on the availability of training data and the nature of the problem at hand.

In recent years, contextualized word embeddings have gained popularity. Models like BERT, GPT, and ELMO generate embeddings that consider the surrounding context of a word, resulting in richer and more nuanced representations. These embeddings excel in capturing word sense disambiguation and context-dependent semantics.

3.2.1. Word2Vec

Word2Vec is a popular natural language processing approach that seeks to represent words as dense vector embeddings. Tomas Mikolov et al. [22] first proposed it in 2013. Word2Vec's core concept is to capture semantic and syntactic links between words by mapping them to continuous vector representations in a high-dimensional space. Word2Vec uses a neural network architecture to learn word embeddings from massive text datasets. It works on the Continuous Bag-of-Words (CBOW) or Skip-gram models' principles. The CBOW model predicts a target word based on the context words around it, whereas the Skip-gram model predicts context words given a target word. These models are trained by adjusting the neural network weights to reduce the difference between predicted and actual word occurrences. Word2Vec's learnt word vectors capture significant correlations between words. They have intriguing qualities such as the capacity to conduct vector arithmetic operations such as analogies. For example, by subtracting the vector representation of "king" from the vector representation of "queen" and adding the resulting vector to the vector representation of "woman," the resulting vector is near to the vector representation of "man." Word2Vec has been used in a variety of natural language processing tasks, including word similarity, document categorization, sentiment analysis, and recommendation systems. It allows the model to recognize the contextual meanings of words and record semantic linkages in a distributed representation, improving the model's performance on a variety of language-related tasks.

3.2.2. Global Vectors (GloVe)

Pennington et al. [24] introduced Global Vectors (GloVe) in 2014 as a technique for producing word embeddings. GloVe, like Word2Vec, tries to represent words as continuous vector embeddings, capturing semantic and syntactic links. GloVe combines the benefits of both global matrix factorization and local context window approaches. It learns word embeddings by leveraging the co-occurrence statistics of words in a huge corpus. GloVe's core finding is that word vectors can be learned by evaluating word co-occurrence probability ratios rather than just the probabilities itself. GloVe embeddings have several advantageous qualities. They exhibit linear substructures and capture semantic links between words, such as word analogies. For example, the vector formed by subtracting the vector representation of "man" from the vector representation of "woman" and adding it to the vector representation of "king" is like the vector representation of "queen". Overall, GloVe is a sophisticated word embedding technique that uses co-occurrence statistics to capture meaningful associations between words. Because of its capacity to acquire high-quality word representations, it is a popular choice in natural language processing.

3.2.3. fastText

Facebook's AI Research (FAIR) team created fastText, an open-source framework for text classification and word representation. Joulin et al. [4] first proposed it in 2016. fastText extends the success of classic word embeddings such as Word2Vec with subword information, allowing it to handle out-of-vocabulary (OOV) words and morphologically rich languages more well. The main idea underlying fastText is to represent words as bags of character n-grams, which are subword units made up of continuous character sequences. FastText can capture morphological and semantic similarities between words even for unseen or rare words by using character-level information. FastText's training technique entails learning continuous vector representations for words using a skip-gram model version. FastText, on the other hand, predicts the center target word based on its character n-grams rather than nearby words, as Word2Vec does. FastText can now handle OOV words by using the subword information in the training data. fastText also allows for effective text classification. It trains classifiers that can assign labels to text documents using a hierarchical softmax or negative sampling strategy. As a result, it is well-suited to tasks like sentiment analysis, topic classification, and spam identification. FastText's processing efficiency is a key advantage. When subword information is used, the dimensionality of the word representations is reduced, allowing for faster training and inference when compared to models that only employ whole words. FastText can also handle big vocabularies efficiently due to its small memory footprint. Because of its effectiveness and efficiency in many natural language processing applications, fastText has acquired favor in the scientific community and industry. It has been used successfully in a variety of real-world applications including as text classification, information retrieval, and language modeling.

4. DATASET

The empirical investigation we conducted utilized a dataset obtained from the Apple App Store. The dataset was collected by a group of students who scraped information from over 10,000,000 apps and it was obtained through Kaggle [35] . However, in order to ensure data quality, rows with incomplete information were filtered out, resulting in a remaining dataset of 365,000 entries. Upon analyzing the dataset, we observed that it comprised of eighteen columns. For our investigation, our primary focus was on the Content Rating and description columns. Consequently, we removed other columns to improve performance and streamline our analysis. The Content Rating column contained four ratings: 4+, 9+, 12+, and 17+. Additionally, we filtered out apps with non-English descriptions. As a result of these filtering processes, our final collection consisted of 283,352 app descriptions belonging to four Content Ratings.

5. ANALYSIS

Our recommended approach (**Figure 5.1**) includes the following processes: data pre-processing, vectorization, and classification. Every one of these phases is described in greater detail underneath.

5.1. Pre-Processing

Text classification problems typically use combinations of text pre-processing algorithms to decrease distortion and improve the classifier's prediction skills. For our investigation, app descriptions were initially converted to lower case. Non-ASCII characters, URLs and digits were removed. The NLTK stop-word list was also used to remove English stop-words such as the, in, and would.

5.2. Vectorization

In this phase, we used two methods as giver before, Bag of words and Word embeddings. Both methods are used to turn the list of pre-processed tokens in each app's description into a vector. Creating a Bag-of-Words model with a maximum of 1000 most-frequent words (as including all the words will make the dataset sparse and will only add noise). As for the word embeddings, pre-trained models are used. We used the word embeddings produced for representing the entire description. In the computation of word collection embeddings, various techniques can be employed, such as unweighted averaging which we used in our investigation to build an embedding for each app description.

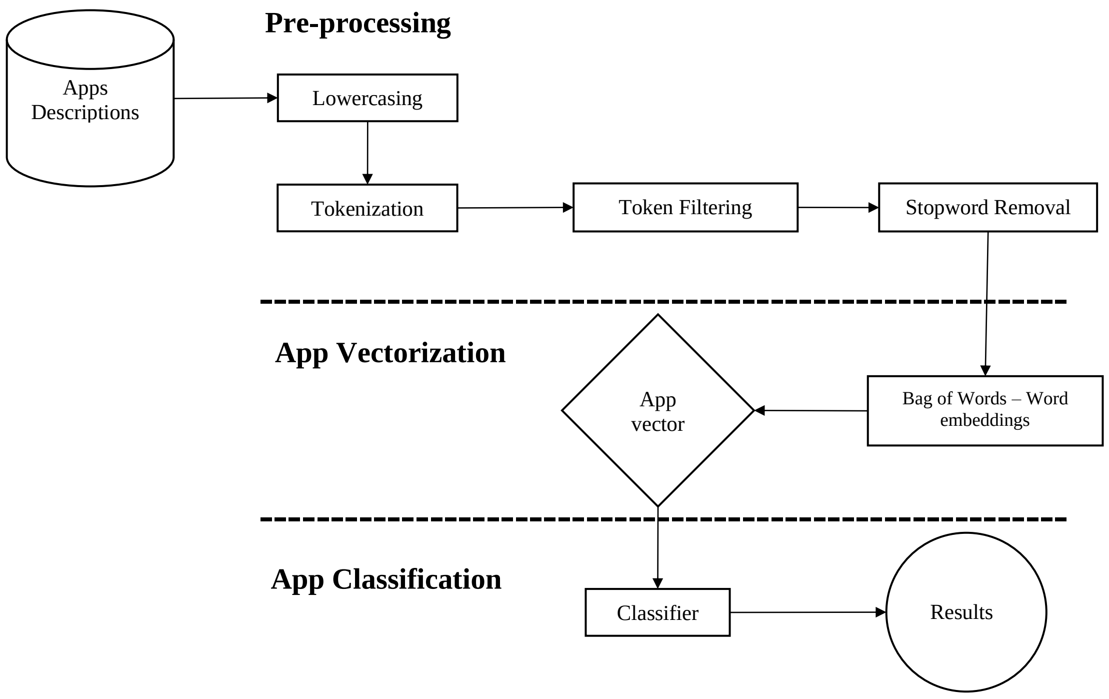

Figure 5.1. Essential steps of the recommended strategy

5.3. Experimental Baselines

We produce one extra vector for app descriptions in addition to our techniques. This vector will serve as the experimental baseline against which we will assess the performance of our approaches. The following is a description of this representation:

Latent Dirichlet Allocation: Blei et al. [3] established Latent Dirichlet Allocation (LDA) as a popular generative statistical model for topic modeling in 2003. It is intended to find hidden themes or subjects within a collection of papers without prior knowledge of the issues.

The basic assumption of LDA is that each document in the corpus is a mix of different themes, and each subject is a probability distribution over a set lexicon of words. The model infers the topic distribution in the corpus as well as the word distribution within each topic.

The training process of LDA involves two main steps: topic assignment and parameter estimation. In the topic assignment step, LDA randomly assigns topics to words in each document. The assigned topics are then refined iteratively based on the statistical properties of the corpus. In the parameter estimation step, LDA estimates the topic proportions in each document and the word distributions for each topic, using statistical inference techniques such as variational inference or Gibbs sampling.

We use Gensim, an open-source Python toolbox, for vector space modeling and topic modeling in our research. We utilize LDA to extract topics from our app description dataset. The hyperparameters of LDA, α and β, are calibrated using commonly used heuristics in text analysis. Specifically, α is determined by the corpus and β is set to 1 divided by the number of subjects. The sample method is iterated for 1000 iterations using Gibbs sampling to ensure subject stability. The number of topics (k) that LDA will uncover is set to be equal to the number of categorization labels.

5.4. Classification and Evaluation

In the second stage, we use the vectorization techniques we discussed before to classify apps in our dataset. The following are our classification configurations:

- Classification algorithms: We experiment with several classification methods to categorize our data:
    - Support Vector Machines (SVM): Support Vector Machines are machine learning models for classification and regression. They find a line or surface (hyperplane) that best separates data into classes. SVMs focus on maximizing the margin between classes, and they can handle non-linear data using the kernel trick. They're effective in high-dimensional spaces, but their complexity and parameter tuning can be challenges.
    - Multi-layer Perceptron (MLP): A Multi-layer Perceptron (MLP) is a type of neural network with input, hidden, and output layers. Neurons process data through activation functions. It's used for tasks like classification and regression, learned through backpropagation. It's a key part of deep learning, capable of approximating complex functions.
    - Random Forests (RF): A Random Forest is an ensemble algorithm combining multiple decision trees. It reduces overfitting by using random subsets of data and features. It's used for classification and regression, providing robustness and feature importance insights.

  - Nearest Neighbor (KNN): The k-Nearest Neighbors (KNN) algorithm predicts by comparing a new data point to its nearest neighbors (based on distance). It's used for classification or regression, has a "k" parameter for the number of neighbors to consider, and is sensitive to feature scaling. It's simple but can be computationally expensive for large datasets.
  - AdaBoost: AdaBoost is an ensemble algorithm that combines weak learners (like simple decision trees) to create a strong model. It focuses on correcting mistakes made by previous models and gives more weight to difficult instances. The final prediction is a combination of weak learners' votes.
  - Decision Trees (DT): Decision Trees are predictive models that use a tree-like structure of decisions to make predictions based on features. They're easy to understand, handle different data types, and are building blocks for more complex algorithms like Random Forests. However, they can overfit and might not capture intricate patterns.
  - Logistic Regression (LR): Logistic Regression is used for binary classification. It estimates probabilities using a logistic function and creates a linear decision boundary in feature space. It's interpretable and commonly used for understanding feature contributions to outcomes.

Scikit-learn, a Python module that integrates a wide range of cutting-edge machine learning techniques for supervised and unsupervised classification issues, is used for our investigation. The predicted class is determined by the class that receives the most votes. Hyperparameter tweaking is used to guarantee that each classifier produces the best possible prediction given the data. **Table 5.1** displays our exact set of hyperparameters.

- Classification features: We use vectorization approaches, as explained in Section 5.2, to derive app categorization features from their descriptions. Using word embedding methods, the description of each app is converted into a feature vector. The chosen approach determines the dimensionality of the feature vector. The default dimensionality for Word2Vec and fastText is 300. On the other hand, GloVe allows for different vector sizes, with options such as 50, 100, 200, or 300. In the case of the VSM (Vector Space Model), the feature vector size is determined by the number of words in the description ($|V|$), where each word $wi \in V$ is represented as a dimension in the vector. Additionally, LDA is utilized to generate feature vectors for each app. Using LDA, app descriptions are converted into feature vectors of size k, which represent the description's probability distribution over the k LDA topics. We investigate the influence of combining current meta-data elements such as the average rating, app size, and pricing to further improve categorization accuracy. The retrieved meta-data is attached to the vectorized representation of each app for each vectorization technique, adding new information into the feature vectors.

- Training settings: We use the train, test, split method. The dataset is divided into two parts using this method: the training set and the test set. The function normally accepts as input the input features (X) and the matching target variables (y). The most common split ratio is 80% for training and 20% for testing, which is what we utilize.
- Validation metrics: To assess the effectiveness of our classification algorithms, we use the standard metrics of accuracy, recall (R), precision (P) and F-measure (F). For each classification label, these measurements are determined independently.

Table 5.1. Hyperparameter configuration for each algorithm.

Classifier	Parameter set
Random Forest	‘max_depth’ ∈ {10, 30, 50, 100, None}
	‘min_samples_split’ ∈ {2, 5, 10}
	‘n_estimators’ ∈ {10, 50, 100, 200}
Naïve Bayes	Gaussian NB: ‘var_smoothing’ ∈ {10^{-10}, 10^{1}}
	Multinomial NB: ‘alpha’ ∈ {0, 1}
Decision Trees	‘max_depth’ ∈ {10, 30, 50, 100, None}
	‘criterion’ ∈ {‘qini’, ‘entropy’}
	‘min _split’ ∈ {2, 5, 10}
KNN	‘n_neighbors’ ∈ {2, 5, 7, 10}
Multi-layer Perceptron	‘solver’ ∈ {‘adam’, ‘sgd’, ‘lbfgs’}
	‘hidden_layers_sizes’ ∈ {(100,100,100), (300,300,300)}
	‘max_iter’ ∈ {1000, 3600, 7000}
SVM	‘kernel’ ∈ {‘linear’, ‘rbf’, ‘poly’, ‘sigmoid’}
	‘C’ ∈ {0.1, 1, 10, 100}
	‘gamma’ ∈ {1/(n_features * X.var()), 1/n_features}
Adaboost	‘n_estimators’ ∈ {10, 50, 100, 200}
	‘learning_rate’ ∈ {0.1, 1, 10}
Logistic Regression	‘solver’ ∈ {‘newton-cg’, ‘sag’, ‘saga’, ‘lbfgs’}
	‘penalty ’ ∈ {‘l1’, ‘l2’}
	‘C’ ∈ {0.1, 1, 10, 100}

6. RESULTS

The results of classifying our dataset are shown in **Table 6.1.** The effectiveness of various classification algorithms while employing various suggested app description vectorization strategies. Our classification techniques scored best when app descriptions were vectorized utilizing VSM. This means that adopting VSM gave the best accurate app description form for our classification tests. MLP (adam solver) achieved the greatest results in categorizing the apps, with an F2 of **0.95**, whereas Decision Trees performed comparably (F2 = **0.89**).

A comparison of the performance based on the different size vectors generated by GloVe is shown in **Figure 6.1.** MLP classification results using different GloVe size vectors.. In general, GloVe achieved its best results at vector size 300. Increasing the size of vectors resulted in more expressive vectors that capture all word relations.

Our findings align with the results obtained by Berardi et al. [2] , Similar to their work, we preprocessed the app descriptions by, stop-word removal, tokenization and stemming, and then vectorized them. Additionally, we employed a mutual information-based feature selection method, as described in Berardi et al. [2] , to identify the most informative set of app features. The results from this analysis demonstrated that the inclusion of meta-data attributes, such as ratings, size, category, and price, did not contribute to enhanced classification results. This outcome was anticipated, as these meta-data attributes typically lack significant functionality-related information compared to the app descriptions. Consequently, the classifiers were not able to leverage the meta-data effectively for improved prediction accuracy.

Table 6.1. The effectiveness of various classification algorithms while employing various suggested app description vectorization strategies

Approach	*Classifier*	*P*	*R*	*F2*
VSM	MLP	**0.95**	**0.95**	**0.95**
	AdaBoost	0.78	0.83	0.78
	Random Forest	0.86	0.86	0.81
	KNN	0.84	0.86	0.83
	SVM	0.84	0.84	0.76
	Decision Trees	**0.89**	**0.9**	**0.89**
	Logistic Regrission	0.81	0.85	0.8
GloVe 300	MLP	0.82	0.85	0.83
	AdaBoost	0.77	0.83	0.78
	Random Forest	0.85	0.85	0.8
	KNN	0.83	0.85	0.84
	SVM	0.69	0.83	0.76
	Decision Trees	0.76	0.74	0.75
	Logistic Regrission	0.78	0.84	0.79
Word2Vec	MLP	0.82	0.85	0.82
	AdaBoost	0.78	0.84	0.78
	Random Forest	0.85	0.86	0.81
	KNN	0.83	0.85	0.83
	SVM	0.82	0.84	0.78
	Decision Trees	0.77	0.76	0.76
	Logistic Regrission	0.78	0.84	0.78
fastText	MLP	0.82	0.85	0.83
	AdaBoost	0.77	0.83	0.78
	Random Forest	0.85	0.86	0.8
	KNN	0.83	0.85	0.82
	SVM	0.71	0.81	0.77
	Decision Trees	0.75	0.73	0.74
	Logistic Regrission	0.78	0.84	0.77

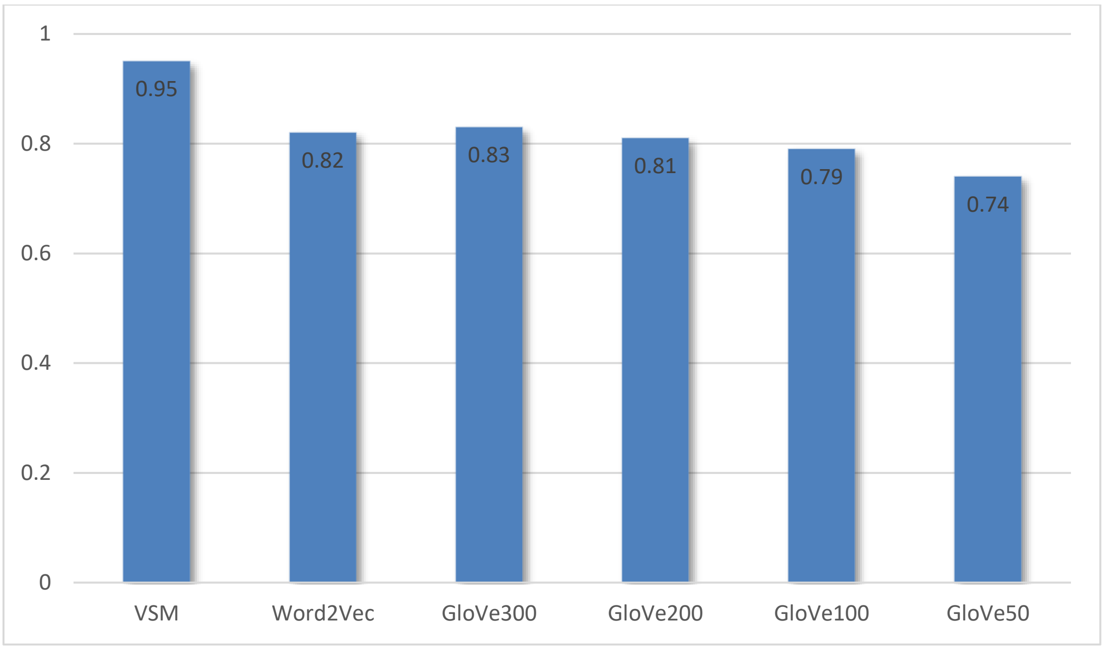

Figure 6.1. MLP classification results using different GloVe size vectors.

7. DISCUSSION

This section discusses our primary analytical findings and provides more insight on some of our results in this work.

7.1. Bag of Words vs. Topic Modeling

In discussion, we compare the performance of our techniques to that of LDA, a popular method for app categorization tasks. According to our findings, bag of words models outperformed LDA in correctly recognizing app age ratings. The performance disparity can be attributable to the poor quality of the topics created by LDA for our dataset. The generated themes were discovered to be of low quality and did not correspond well with the expert-provided categories. LDA's poor performance can be attributed in part to the short length of app descriptions.

Prior research has demonstrated that when working with short input documents, LDA does not function well. LDA is a data-intensive technique that necessitates a large volume of text to build meaningful topic distributions. However, because app descriptions are often brief, using traditional LDA to such data frequently results in incoherent and overlapping subjects. These findings illustrate LDA's limitations when dealing with brief text inputs, stressing the benefits of bag of words models in capturing the semantic meaning of app descriptions and attaining superior classification results.

7.2. GloVe vs. Word2Vec and fastText

GloVe defeated Word2Vec and fastText in our app categorization tests, according to our data. In general, context-free word embedding models such as GloVe, Word2Vec, and fastText produce equivalent results. Their performance, however, can vary slightly depending on the complexity of the text corpus on which they are trained. One possible explanation for GloVe's superior performance is that the language used to train the GloVe model was more broad than the vocabulary used to train the Word2Vec model.

GloVe can capture a better representation of word meanings and associations with a larger vocabulary, perhaps leading to increased classification accuracy.

In the case of fastText, our findings revealed that the embeddings generated for unusual words utilizing character n-grams did not significantly improve classification accuracy. This could be because unusual words, which are represented by character n-grams, were extremely infrequent in our sample. As a consequence, including character n-gram embeddings provided no significant gains for our app categorization tasks. Overall, our findings emphasize the necessity of taking into account the specific properties of the dataset as well as the training procedure when picking the best word embedding model for a given task.

8. CONCLUSION

In this paper, we proposed an original method for categorizing mobile apps based on their app store descriptions. To construct vector representations of app descriptions, we used both Bag-of-Words and word embeddings models, which were subsequently used for app categorization and age rating prediction. To test our method, we ran trials with a dataset of apps drawn from various categories of the Apple App Store. Our study's findings have important practical consequences. Users can profit from our method by more successfully identifying mobile apps that match their unique interests. Our approach allows users to identify apps that are appropriate for their age group and preferences by precisely classifying and age-rating apps. Additionally, developers can leverage our approach to identify direct competitors within the app store and gain insights into market trends. For future work, we propose two main directions for further exploration. Firstly, conducting more extensive analysis using expert-generated categorizations across a wider range of application domains would provide deeper insights into the effectiveness and generalizability of our approach. Expanding the dataset and incorporating diverse expert opinions will enhance the robustness of our classification models. Secondly, we envision developing a working prototype that implements our findings. This prototype would ideally be implemented as a mobile app with a user-friendly interface, facilitating the discovery of apps based on age ratings and specific interests. Building such a tool would bring our research closer to real-world application and provide practical benefits to mobile app users. By pursuing these future directions, we aim to further enhance the accuracy and usability of our approach, ultimately improving the app discovery experience for users and supporting developers in making informed decisions in the competitive app market.

REFERENCES

[1] Afnan Al-Subaihin, Federica Sarro, Sue Black, and Licia Capra. 2019. Empirical comparison of text-based mobile apps similarity measurement techniques. In Empirical Software Engineering. 1–26.

[2] Giacomo Berardi, Andrea Esuli, Tiziano Fagni, and Fabrizio Sebastiani. 2015. Multi-store metadata-based supervised mobile app classification. In Annual ACM Symposium on Applied Computing. 585–588.

[3] David Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. Journal of Machine Learning Research 3 (2003), 993–1022.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics 5 (2017), 135–146.

[5] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In SIGIR ’07, pages 231–238, 2007.

[6] H. Cao, D. H. Hu, D. Shen, D. Jiang, J.-T. Sun, E. Chen, and Q. Yang. Context-aware query classification. In SIGIR’09, pages 3–10, 2009.

[7] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans. Feature model extraction from large collections of informal product descriptions. In FSE 2013, pages 290–300, Aug. 2013.

[8] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli. On-demand feature recommendations derived from mining public product descriptions. In ICSE ’11, pages 181–190, 2011.

[9] Fahimeh Ebrahimi, Miroslav Tushev and Anas Mahmoud. 2021. Classifying Mobile Applications Using Word Embeddings. In ACM Transactions on Software Engineering and Methodology.

[10] Anthony Finkelstein, Mark Harman, Yue Jia, William Martin, Federica Sarro, and Yuanyuan Zhang. 2017. Investigating the relationship between price, rating, and popularity in the Blackberry World App Store. Information and Software Technology 87 (2017), 119–139.

[11] Fuad A. and Al-Yahya M. Analysis and Classification of Mobile Apps Using Topic Modeling: A Case Study on Google Play Arabic Apps. Hindawi Complexity Volume 2021, Article ID 6677413, 12 pages https://doi.org/10.1155/2021/6677413

[12] Gállego A., Motger Q., Franch X. and Marco J. TransFeatEx: a NLP pipeline for feature extraction. Co-located with REFSQ 2023. Barcelona, Catalunya, Spain, April 17, 2023.

[13] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller. Checking app behavior against app descriptions. In Proceedings of the 36th International Conference on Software Engineering, pages 1025–1035. ACM, 2014.

[14] E. Guzman and W. Maalej. How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews. In IEEE 22nd International Requirements Engineering Conference (RE), pages 153–162, 2014.

[15] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher. Supporting Domain Analysis through Mining and Recommending Features from Online Product Listings. IEEE TSE, 39(12):1736–1752, 2013.

[16] Priya Kalaivani K and Arulanand N. Mobile App Categorization Based On App Descriptions And Api Calls. International Journal of Aquatic Science ISSN: 2008-8019 Vol 12, Issue 02, 2021

[17] S. Kawaguchi, P. K. Garg, M. Matsushita, and K. Inoue. MUDABlue: An automatic categorization system for Open-Source repositories. Journal of Systems and Software, 79(7):939–953, July 2006.

[18] David Lulu and Tsvi Kuflik. 2013. Functionality-based clustering using short textual description: Helping users to find apps installed on their mobile device. In International Conference on Intelligent User Interfaces. 297–306

[19] Y. S. Maarek, D. M. Berry, and G. E. Kaiser. An information retrieval approach for automatically constructing software libraries. IEEE TSE, 17(8):800–813, 1991.

[20] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang. The app sampling problem for app store mining. In 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, pages 123–133. IEEE, 2015.

[21] W. Martin, F. Sarro, and M. Harman. Causal Impact Analysis for App Releases in Google Play. In FSE'16, 2016.

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In Workshop of 1st International Conference on Learning Representations.

[23] Qiu H., Wu Z. and Zhang X. Exploring Multiple Genres Text Classification Classifying 61 genres of Mobile App Description based on Naïve Bayes and CountVectorizer. 2022 3rd International Conference on Electronic Communication and Artificial Intelligence (IWECAI)

[24] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In Conference on Empirical Methods in Natural Language Processing. 1532–1543.

[25] X.-H. Phan, C.-T. Nguyen, D.-T. Le, L.-M. Nguyen, S. Horiguchi, and Q.-T. Ha. A hidden topic-based framework toward building applications with short web documents. IEEE Transactions on Knowledge and Data Engineering, 23:961 – 976, 2010.

[26] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In WWW '06, pages 377–386, 2006.

[27] Borja Sanz, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, and Pablo Bringas. 2012. On the automatic categorisation of Android applications. In Consumer Communications and Networking Conference. 149–153.

[28] S. Seneviratne, A. Seneviratne, M. A. Kaafar, A. Mahanti, and P. Mohapatra. Early detection of spam mobile apps. WWW ’15, pages 949–959, 2015.

[29] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In SIGIR ’06, pages 131–138, 2006.

[30] Singla K., Bose J. and Mukherjee N. Multimodal Language Independent App Classification Using Images and Text. Conference: NLDB 2018, Paris

[31] K. Tian, M. Revelle, and D. Poshyvanyk. Using Latent Dirichlet Allocation for automatic categorization of software. In 6th IEEE International Working Conference on Mining Software Repositories MSR’09, pages 163–166. IEEE, 2009.

[32] S. Vakulenko, O. Muller, and J. Brocke. Enriching ¨ iTunes App Store Categories via Topic Modeling. In Proc. of the Thirty Fifth International Conference on Information Systems, ICIS’14, 2014.

[33] T. Wang, H. Wang, G. Yin, C. X. Ling, X. Li, and P. Zou. Mining Software Profile across Multiple Repositories for Hierarchical Categorization. In IEEE International Conference on Software Maintenance ICSE’13, pages 240–249. IEEE, Sept. 2013.

[34] W.-T. Yih and C. Meek. Improving similarity measures for short segments of text. In Proceedings of the 22nd national conference on Artificial intelligence - Volume 2, pages 1489–1494, 2007.

[35] https://www.kaggle.com/datasets/fentyforte/365k-ios-apps-dataset, Access Date 12.06.2022

[36] Information Technologies Use in Children Research, 2021, TUIK (Turkish Statistical Institute) 41132 – 22/12/2021 https://data.tuik.gov.tr/Bulten/Index?p=Survey-on-Information-and-Communication-Technology-Usage-by-Children-2021-41132&dil=2

CURRICULUM VITAE

Name Surname **:**Muhammed Emin IDELBI

EDUCATION:

- **Undergraduate** : 2020,Çukurova University, Faculty of Engineering, Department of Computer Engineering
- **Graduate**: Continue, Sakarya University, Department of Computer And Informatıon Engineering

PROFESSIONAL EXPERIENCE AND AWARDS:

- He worked as a software developer at Seraş Yazılım between 2023-Now.