Enhancing Suggestion Detection in Online User Reviews through Integrated Information Retrieval and Deep Learning Approaches

Zahra Hadizadeh, Amin Nazari and Muharram Mansoorizadeh

Computer Engineering Department, Bu-Ali Sina University, Iran
E-mail: z.hadizadeh73@gmail.com; aminnazari91@gmail.com; mansoorm@basu.ac.ir
**Corresponding Author*

Received 02 October 2023; Accepted 17 April 2024

Abstract

In the aftermath of the COVID-19 pandemic, using web platforms as a communication medium and decision-making tool in online commerce has become widely acknowledged. User-generated comments, reflecting positive and negative sentiments towards specific items, serve as invaluable indicators, offering recommendations for product and organizational improvements. Consequently, the extraction of suggestions from mined opinions can enhance the efficacy of companies and organizations in this domain. Prevailing research in suggestion mining predominantly employs rule-based methodologies and statistical classifiers, relying on manually identified features. However, a recent trend has emerged wherein researchers explore solutions grounded in deep learning tools and techniques. This study aims to employ information retrieval techniques for the automated identification of suggestions. To this end, various methodologies, including distance measurement approaches, multilayer perceptron neural networks, support vector machines,

Journal of Web Engineering, Vol. 23_3, 431–464.
doi: 10.13052/jwe1540-9589.2335

regression logistics, convolutional neural networks utilizing TF-IDF, Bag of Words (BOW), and Word2Vec vectors, along with keyword extraction, have been integrated. The proposed approach is assessed using the *SemEval2019* dataset to extract suggestions from the textual content of online user reviews. The obtained results demonstrate a notable enhancement in the F_1 score, reaching 0.76 compared to prior research. The experiments further suggest that information retrieval-based approaches exhibit promising potential for this specific task.

Keywords: User generated content analysis, suggestion mining, information retrieval, information extraction, text mining, sentiment analysis, deep learning.

1 Introduction

E-commerce has a vital role in modern life. The number of online purchases is rising every day. Web2 allows users to share their comments, suggestions, and complaints about products and services. As a result, online user-generated text is becoming a popular source that can help other users make informed decisions. On the other hand, the content helps organizations utilize suggestions to improve their products and services [34]. The comments may include helpful suggestions to help companies better understand user preferences and requirements [12, 18]. To utilize comments on the web, opinion mining was introduced in 2002. This system mainly processes the text to summarize positive, negative, and neutral sentiments towards the particular item (product or service). However, opinionated texts contain descriptive information such as advice, hints, warnings, and suggestions, which are generally ignored in sentiment analysis tasks [23]. These more general classes of user feedback convey important information that is useful for vendors and dealers as well as the end-users who usually rely on customer tips and recommendations [21]. These insights motivate suggestion extraction as an elementary step in e-commerce decision-making and development processes.

While human experts extract various types of information from natural language texts quite effortlessly, manual processing of millions of opinion texts takes too long and is costly. Hence, automatic and efficient approaches in this regard are anticipated. The successful application of natural language processing (NLP) tools and techniques in neighboring tasks such as

text summarization further motivates the use of automatic approaches in suggestion extraction tasks [5, 21].

It is worth noting that suggestion mining and recommending systems are related concepts; however, they refer to different aspects of information extraction from textual content. The primary goal of suggestion mining is to find out what users suggest or recommend in their reviews and comments. These suggestions are often used by business managers to improve their products. On the other hand, the focus of recommendation systems is to help users find products and services based on their interests and preferences. The distinction is important since the former is a much newer field of study while the latter has rich and active literature [15, 16].

Suggestion mining is an emerging research area, and the problem definition is still in its early stages, with labeled datasets being scarce. Few studies have been conducted in the field, many of them being empirical [22]. However, suggestion mining can be employed in various applications such as product improvement ideas, customer-to-customer suggestions, suggestion summarization, sentiment recognition, and recommendation systems [20,32]. Therefore, the numerous applications and lack of research conducted are the primary motivations for this study.

Information retrieval (IR) methods are techniques and processes used to retrieve relevant information from large collections of data, typically in the form of text documents. The main goal is to find efficiently and present information that is most relevant to the questions or queries [30].

Adopting IR methods in suggestion mining facilitates the efficient retrieval of pertinent suggestions from extensive textual datasets, employing relevance ranking techniques like term frequency-inverse document frequency (TF-IDF) and BM25. The incorporation of NLP enhances contextual understanding, enabling the differentiation of sentiments and nuanced meanings within suggestions. Furthermore, these methods enhance the accuracy, relevance, and scalability of the suggestion mining systems. Likewise, deep learning provides advanced techniques to analyze complex patterns, induce more informative hidden representations (i.e. embeddings), and improve the overall performance of retrieval systems.

In this research, diverse approaches are utilized to extract feature vectors from opinion texts. Subsequently, supervised learning methods and deep neural networks are applied to detect suggestions within the text. Through evaluations conducted on the selected dataset, the results demonstrate that the proposed approach effectively and accurately extracts suggestions. The

findings suggest that this method holds potential for application in downstream tasks such as recommendation systems and decision support systems, showcasing its capability to identify suggestions with acceptable accuracy automatically.

The structure of the paper is organized as follows. In the second section, the related work is reviewed, and then in the third section, the proposed approach is presented with an emphasis on motivations, concepts, and terms. In the fourth section, the test and evaluation results of the proposed approach are reported, and in the fifth section, the conclusions, as well as a set of future directions are presented.

2 Related Work

In recent years, numerous studies have focused on text mining that extracts useful information from online user reviews. However, very few studies have been conducted to extract suggestions from online user reviews. The main reason is that natural language texts are largely unstructured, making the task inherently challenging. The few related works conclude that it is not straightforward to detect suggestions automatically; hence, this problem remains an open research question that needs to be explored. In the following, we review some of the available related studies.

As one of the earlier works, Goldberg et al. [9] employed rule-based approaches that use several linguistic techniques and extract standard expressions as suggestions. They focused on a corpus of product reviews and online political comments. Lacob and Harrison [11], studied similar approaches to identify language patterns and extract text with known patterns. They used linguistic resources such as vocabularies and ontologies, as well as NLP tools such as parsers and phrase extractors. Then, a set of patterns are manually composed as templates of suggestions. Since this approach relies on these patterns, it is classified as a domain-dependent approach. Brun and Hagege [2] extracted suggestions from product reviews using manually formulated rules. Jhamtani et al. [12] proposed a regulatory learning approach to identify suggestions from the comments to improve product features. They used several machine-learning approaches.

Negi et al. [20] used the convolution neural network (CNN) and long short-term memory (LSTM) for suggestion mining. In addition, their purpose is to eliminate the need for manual specification of feature types. The results (F_1 score) in the hotel domain in LSTM and CNN networks were 0.45 and 0.36, respectively, and in the electronic domain in LSTM and CNN

networks they were 0.51 and 0.39, respectively. Their findings show that LSTM networks perform better than other approaches. Ngo et al. [24] used machine learning approaches and deep learning with the CNN model on the Vietnamese media reviews and, reached the F_1 scores of 78.45 and 77.68.

Golchha et al. [8] proposed an approach to suggestion mining from customer-to-customer advice using a hybrid semi-supervised neural network model. This model uses semi-supervised learning to utilize useful information from large amounts of unlabeled data. Their F_1 scores on the hotel and electronics datasets were 65.6 and 65.5, respectively. Fatyanosa et al. [6] used machine learning approaches, such as logistic regression, random forest, multinomial naive Bayes, linear support vector classification, sublinear support vector classification, variable length chromosome genetic algorithm-naive Bayes, and CNN. They compared these approaches on the software and hotel datasets and obtained F_1 scores of 0.47 and 0.37, respectively.

Ding et al. [4] used the stacked bidirectional LSTM(SBiLSTM) on software developer reviews. They used the pre-trained Word2Vec model to learn distributed sentence representations and reached the $F_1 Score$ of 0.57. Pecar et al. [25] used the recurrent neural network approach with BiLSTM layers and the attention mechanism. They tried to encrypt the words using the Embeddings from Language Model (ElMo) and reached the F_1 score of 0.68. Zhou et al. [36] used the convolutional neural network method and the pre-trained BERT model. They achieved the F_1 score of 0.70. Ping Yue et al. [33] used a combination of neural networks with layers of BiLSTM, gated recurrent unit (GRU), and CNN. They used the BERT pre-trained model and voting in the final classification stage. They finally reached the F_1 score of 0.73. Ezen-Can and Can [5] proposed a domain-independent suggestion mining system in which they analyze information sources in two ways: first, using external features in the recurrent neural network (RNN) and second, by collecting the RNN result with rule-based features. They trained their model on the electronics dataset and achieved an F_1 score of 77.70%. The trained model also achieved the F_1 score of 74.49% in the hotel reviews dataset.

Reddy et al. [28] applied various deep learning techniques such as RNN, LSTM, attention-based LSTM, and GRU for suggestion mining. The findings indicate that attention-based LSTM achieved the best accuracies for suggestion mining.

Recently, Laskari et al. [17] presented an explainable system to identify patterns and understand the significant components of suggestions. The

experimental results on the standard datasets demonstrate that the attention model can overtake other models. Zhao et al. [35] studied the performance of several classification models for Chinese suggestion mining. They used both traditional machine learning approaches (feature engineering-based models) and deep learning models in the Chinese reviews. Ramesh et al. [27] proposed the feature selection-based approach for suggestion mining. The method involves feature selection algorithms such as chi-square and multivariate relative discrimination criterion (MRDC), as well as learner models like support vector machine (SVM) and random forest (RF) as classifiers.

In summary, while text mining from user reviews has received considerable attention, suggestion mining remains a challenging and open research question. The spectrum of methodologies employed in existing studies ranges from rule-based approaches and machine-learning techniques to advanced deep-learning models. Major contributions include the application of LSTM and CNN networks, hybrid semi-supervised neural network models, and explainable systems utilizing attention models.

3 The Proposed Approach

The suggestion mining problem presented in *SemEval2019-Task9* has been raised as a binary classification problem. We also presented this problem as information retrieval. In this section, we discuss the various information retrieval approaches for text preprocessing and then the classification algorithms for extracting the suggestions from reviews.

3.1 Motivation

There are several gaps and limitations in the application of IR methods to suggestion mining. Notably, there is a predominant focus on machine learning and deep learning approaches, with insufficient exploration of traditional or advanced IR techniques. The studies often lack attention to query-document relevance, restricting the potential benefits of IR models that excel in capturing such relevance. Integration of semantic search capabilities, exploration of cross-domain applicability, and addressing real-time retrieval aspects are notably scarce. Furthermore, there is a need for a deeper understanding of user queries and the utilization of diverse retrieval models. Mitigating these gaps has the potential to develop a more comprehensive and deeper utilization of IR methods, thereby improving suggestion-mining processes.

Table 1 The opinions include suggestions for software developers

#	Suggestion
1	It would be nice to give developers the option to put a payment on hold when they reach the appropriate threshold, and have the option to put it on top of the next due payment when they reach the threshold after that.
2	Include an API to add, remove, deactivate tracking protection lists (TPL), to further drive down data usage, enable selective protection against rogue sites tracking, and just have a sense of control.

3.2 Basic Concepts

- *Reviews*: The opinions users write to provide feedback on products. Users express their reviews to share and communicate their thoughts and experiences as product consumers.
- *Suggestion*: The idea that a user expresses about a particular item. The suggestion can be in the form of advice; for those who want to buy a product, or can be a complaint to the producer, supplier, or vendor of the product. Table 1 shows instances of suggestions.
- *Information retrieval*: IR involves the organized and efficient retrieval of relevant information from large and diverse collections of data. Commonly used in search engines, databases, and digital libraries, IR includes methods such as indexing, ranking, and natural language processing to match user queries with relevant documents, enabling users to access information efficiently. The goal of information retrieval is to assist the user in finding the information they are seeking from an unstructured dataset.
- *Domain*: We refer to the term domain as the source of the text. Some resources can be software developer reviews, hotels, electronics reviews, and Tweets.

3.3 The Proposed Model

Figure 1 shows the block diagram of the proposed model. The model contains specialized components for input gathering, preprocessing, feature extraction, embedding, classification, and evaluation.

3.3.1 Input and preprocessing

Input to the model are review texts, in which each review is a sequence of a few sentences. The *input* component is a pipeline of text import operations.

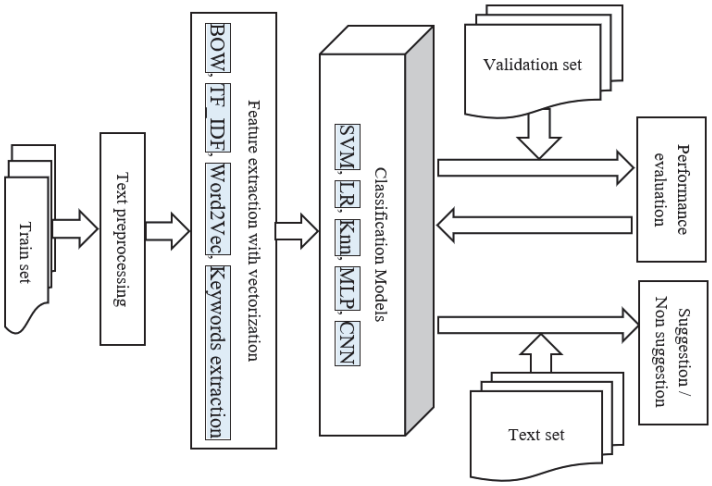

Figure 1 Block diagram of the proposed method.

It is usually realized as import utilities from raw texts or structured *XML* and *JSON* files.

Since normal web users write the texts; they usually follow varying writing styles and contain spelling errors and non-vocabulary terms. The preprocessing step cleans up and normalizes the texts in a few regular steps as follows:

1. **Spelling correction:** Makes context-aware spelling corrections by matching the words against a generic English dictionary.
2. **Emoji normalization:** Replaces emojis with text descriptions.
3. **Hyperlink removal:** Removes URLs using a regular expression.
4. **Non-alphanumeric removal:** Excludes punctuation and special characters.
5. **Whitespace removal:** Collapses multiple spaces into one.
6. **Stopword removal:** Removes common words using NLTK.
7. **Lowercasing:** Converts all words to lowercase.
8. **Numeric word removal:** Excludes words consisting only of digits.
9. **Short text removal:** Filters out texts shorter than the specified minimum length

The preprocessing operations are mainly conducted by *NLTK* [1] and *PyChant* toolkits.[2]

[1] https://www.nltk.org/

[2] https://pypi.org/project/pyenchant/

3.3.2 Feature extraction

At the end of preprocessing, each review is a set of tokens. Next, this set is represented by a feature vector that distinguishes those containing suggestions from the others. The following feature extraction and embedding methods are adopted in this study.

- *BOW*: The *BOW* method represents a document as a lexicographically ordered set of words and their respective frequencies.
- *TF-IDF*: In this method, words are given weights based on their frequency in the documents. In an information retrieval system, this weight is a statistical measure used to express the importance of a word in a document. The *TF* specifies the frequency of a word in the document, and *IDF* is inversely proportional to the number of documents that contain that word. Both of the metrics are usually measured in the *log* scale [30].
- *Word2Vec*: This method induces continuous representations for the words, called word embeddings. Word2vec [19] has been efficiently adopted in several language understanding tasks, encouraging its application in new areas. In this research, We trained Gensim's[3] Word2vec over the entire set of documents. We also trained FastText [14] from the same module and also used several pre-trained models.
- *Filtering too-frequent and too-rare words*: In this step, too-frequent and too-rare words are removed from the extracted features, as the former group does not convey discrimination power and the latter has no reasonable support to consider in analyses. Since we used the sci-kit learn module, this step is implemented as setting parameter values in the respective functions.
- *Feature selection using the t-test*: The t-test can be utilized to assess the significance of individual features in distinguishing between different classes or categories. By calculating the t-statistic for each feature and selecting those with significant differences in means between classes (i.e. low p-values), one can identify the most informative features for classification or regression tasks.

  The t-test compares the means of two independent samples to estimate the t-statistic, which represents the difference between the sample means normalized by the standard error of the difference. The general formula

[3] https://radimrehurek.com/gensim/models/word2vec.html

for the t-statistic in a two-sample t-test is given in Equation (1):

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{1}$$

where:

- $\bar{X}_1$ and $\bar{X}_2$ are the sample means of the two groups
- s_1 and s_2 are the sample standard deviations of the two groups
- n_1 and n_2 are the sample sizes of the two groups.

The t-test is used to test the null hypothesis (H_0) that there is no significant difference between the means of the two groups against the alternative hypothesis (H_1) that there is a significant difference. The decision to reject or fail to reject the null hypothesis is based on the calculated t-statistic and the significance level (α) chosen for the test. In this research, we use an independent samples t-test and reject the null hypothesis with p-values lower than a pre-defined threshold. The threshold values used in the literature are 0.01, 0.05, and 0.1; however, to be more exhaustive, we used a much larger value of 0.25.

- *Principal component analysis (PCA)*: PCA [1] is adopted as a statistical method for dimensionality reduction. It transforms high-dimensional data into a lower-dimensional space while retaining most of the original information. PCA identifies orthogonal axes, called principal components, that capture the maximum variance in the data. By projecting the data onto these components, PCA reveals underlying patterns and simplifies analysis.
- *Keywords extraction*: Keywords are terms and phrases whose frequency is significantly different in the two classes, namely suggestion containers, and others. Essentially, these keyword frequencies are a subset of BOW vectors.

3.3.3 Data augmentation

IR often relies on large datasets of documents and queries for training retrieval models. However, in specific domains or for niche topics, acquiring enough labeled data can be challenging. Data augmentation techniques like back-translation, synonym replacements, or paraphrase generation can artificially enlarge the training data, improving model performance.

Since there are fewer instances of the suggestion class, we have replaced the synonymous words in the documents and generated new reviews using the

existing instances. Data augmentation is applied only to the train data. The approach is similar to the query expansion methods in search engines [13].

3.4 Classification and retrieval

The learning models used in this research range from classical machine learning algorithms to modern deep learning architectures. The models are categorized into four main groups based on their underlying principles and techniques. We present the formulation and main equations of each model, However, since we used implementations from *scikit-learn*[4] and similar standard Python libraries, we do not dive into the details of the learning and optimization considerations. For in-depth and comprehensive information please refer to generic textbooks such as [1, 10].

The first group contains basic machine learning models including linear discrimination analysis (LDA), logistic regression (LR), various flavors of SVM, decision trees, and kNN. The second group includes random forests and a few types of well-known boosting methods. The third group includes classic and modern feed-forward neural networks. Finally, the fourth group contains a single neural network with the BERT as its feature extraction module.

3.4.1 Basic machine learning models

Basic machine learning models form the foundation of many predictive analytics tasks. They are mainly statistical or algebraic models with a few trainable parameters. The parameters are usually optimized using analytic or iterative solutions [1].

Naive Bayes: The naive Bayes classifier is a probabilistic machine learning algorithm based on Bayes' theorem with an assumption of independence between features. The classifier predicts the probability of each class given a set of features using Equation (2):

$$P(y|x_1, x_2, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i|y)}{P(x_1, x_2, \ldots, x_n)} \tag{2}$$

where $P(y|x_1, x_2, \ldots, x_n)$ is the probability of class y given features $x_1, x_2, \ldots, x_n$, $P(y)$ is the prior probability of class y, $P(x_i|y)$ is the conditional probability of feature x_i given class y, and $P(x_1, x_2, \ldots, x_n)$ is the probability of the features.

[4] https://scikit-learn.org/stable/

The naive Bayes classifier works well with well-defined class distributions such as text classification tasks. Its simplicity and efficiency make it a popular baseline for classification tasks, specifically where interpretability and speed are important factors.

Bayesian ridge regression: Bayesian ridge regression combines Bayesian principles with linear regression. It involves assuming a Gaussian prior distribution over the regression coefficients and then updating these beliefs using Bayes' theorem to obtain the posterior distribution. The model is formulated as in Equation (3):

$$\hat{w} = \arg\min_{w} \left\{ \|y - Xw\|_2^2 + \alpha \|w\|_2^2 \right\} \tag{3}$$

where $\hat{w}$ is the estimated regression coefficients, y is the target variable, X is the design matrix of predictors, and α is a hyperparameter controlling the regularization strength. The regularization term, $\|w\|_2^2$, imposes a penalty on the size of the coefficients, to prevent overfitting.

Linear discriminant analysis: Linear discriminant analysis (LDA) aims to find a linear combination of features that best separates two or more classes. It assumes that the features are normally distributed and that the classes have identical covariance matrices. The decision function is defined in Equation (**??**):

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k) \tag{4}$$

where $\delta_k(x)$ represents the discriminant function for class k, x is the input feature vector, μ_k is the mean vector of class k, Σ is the covariance matrix, and π_k is the prior probability of class k.

Logistic regression: As a binary classification algorithm, logistic regression (LR) estimates the probability that a given input belongs to a particular class. The model is realized by Equation (5):

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p)}} \tag{5}$$

where:

$P(Y = 1|X)$ represents the probability of class membership,
X denotes the input features, and
$\beta_0, \beta_1, \ldots, \beta_p$ are the coefficients to be estimated.

Support vector machines (SVMs): SVMs are supervised learning models used for classification and regression tasks. They find the hyperplane that best separates classes in the feature space. The goal function is defined in Equation (6):

$$\text{minimize} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \tag{6}$$

subject to:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \tag{7}$$

$$\xi_i \geq 0 \tag{8}$$

where:

w is the weight vector,
b is the bias term,
ξ_i are slack variables,
C is a regularization parameter.

Decision trees: Decision trees recursively partition the feature space into smaller subsets based on the values of input features, facilitating both classification and regression tasks. The partitioning criteria is usually defined as the purity estimated such as the Gini index, Equation (9):

$$J(s,t) = \frac{m_{\text{left}}}{m}\text{Gini}_{\text{left}} + \frac{m_{\text{right}}}{m}\text{Gini}_{\text{right}} \tag{9}$$

where:

$J(s,t)$ represents the cost function at split s and threshold t,
m_{left} and m_{right} are the number of samples in the left and right subsets, and
$\text{Gini}_{\text{left}}$ and $\text{Gini}_{\text{right}}$ are the Gini impurities of the left and right subsets.

k-nearest neighbors (kNN): kNN is a non-parametric algorithm that makes predictions based on the majority class of its k nearest neighbors in the feature space.

$$\hat{y}(x) = \text{mode}(y_i | x_i \in N_k(x)) \tag{10}$$

where:

$\hat{y}(x)$ represents the predicted class for input x, and
$N_k(x)$ denotes the k nearest neighbors of x.

3.4.2 Ensemble methods

Ensemble methods combine multiple learning models to enhance predictive performance, leveraging the diversity of individual models.

Random forests: Random forests build multiple decision trees during training and output the mode of the classes (classification) or the mean prediction (regression) of the individual trees.

$$\hat{y}(x) = \frac{1}{B} \sum_{i=1}^{B} \hat{y}_i(x) \tag{11}$$

where:

$\hat{y}(x)$ is the ensemble prediction for input x,
$\hat{y}_i(x)$ is the prediction of the ith decision tree, and
B is the number of trees in the forest.

Boosting methods: Boosting algorithms such as AdaBoost, Gradient Boosting, and XGBoost sequentially train weak learners to correct the errors of their predecessors, producing a strong ensemble model [7].

$$F(x) = \sum_{m=1}^{M} \beta_m h_m(x) \tag{12}$$

where:

$F(x)$ is the ensemble prediction for input x,
β_m is the weight assigned to the mth weak learner, and
$h_m(x)$ is the prediction of the mth weak learner.

Mixture of experts (MoE): An MoE is an ensemble model that combines multiple specialized models (experts) using a gating network. The gating network assigns weights to each expert's prediction based on the input data, resulting in a final prediction that benefits from the expertise of each model. This approach allows MoE models to capture complex patterns in the data and achieve high performance in various tasks.

Given an input x, the MoE model predicts the output y by combining the predictions of multiple expert models using Equation (13):

$$y = \sum_{i=1}^{N} \alpha_i \cdot f_i(x) \tag{13}$$

where:

N is the number of expert models,
α_i are the gating coefficients produced by the gating network, representing the contribution of each expert, and
$f_i(x)$ is the prediction of the ith expert model for the input x.

The gating coefficients α_i are computed using a softmax function applied to the output of the gating network, ensuring that they sum up to one and represent a probability distribution over the expert models:

$$\alpha_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

where z_i represents the output of the gating network for the ith expert model.

3.4.3 Neural networks

Neural networks consist of interconnected layers of nodes that process input data to produce output predictions.

Classic feed-forward neural networks: Classic feed-forward neural networks (Figure 2) learn complex mappings between inputs and outputs through layers of neurons.

$$\mathbf{y} = \sigma(\mathbf{W}^T\mathbf{X} + \mathbf{b}) \tag{14}$$

where:

$\mathbf{y}$ is the output vector,
$\mathbf{W}$ is the weight matrix,
$\mathbf{X}$ is the input vector,
$\mathbf{b}$ is the bias vector,
σ is the activation function.

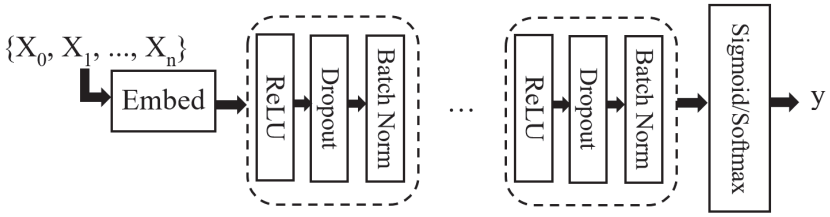

Figure 2 MLP architecture.

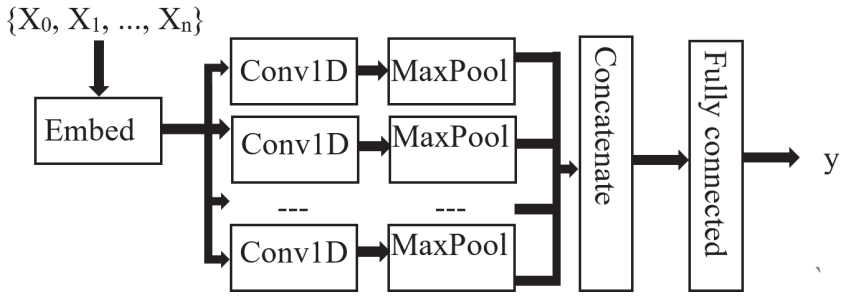

Figure 3 CNN architecture.

Convolutional neural networks (CNNs): CNNs are deep learning models specifically designed for processing structured grid data, such as images. They have revolutionized the field of computer vision and have been widely adopted in various domains.

As depicted in Figure 3, the architecture of a typical CNN consists of multiple convolutional layers followed by activation functions (such as ReLU), pooling layers, and fully connected layers for classification or regression tasks. The model is trained using backpropagation and optimization algorithms like stochastic gradient descent (SGD) or variants like Adam.

The effectiveness of CNNs stems from their ability to automatically learn hierarchical representations of data, making them well-suited for tasks such as image classification, object detection, and segmentation. Their success has also led to applications in other domains, including natural language processing and speech recognition.

Long short-term memory (LSTM): LSTM networks are a memory augmented recurrent neural network (RNN) designed to address the vanishing gradient problem and capture long-range dependencies in sequential data.

LSTMs are composed of memory cells with self-connections called gates, which control the flow of information. These gates include an input gate, a forget gate, and an output gate, which regulate the information flow into and out of the memory cell. This architecture allows LSTMs to selectively remember or forget information over long sequences, making them particularly effective for tasks involving sequential data.

The key feature of LSTMs is their ability to maintain a constant error flow over time, which helps mitigate the vanishing gradient problem encountered in traditional RNNs. This allows LSTMs to capture dependencies over longer time scales and learn more complex patterns in sequential data.

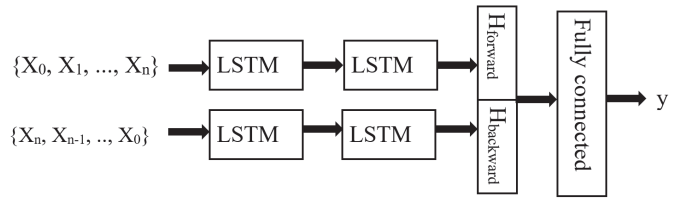

Figure 4 BiDi-LSTM network.

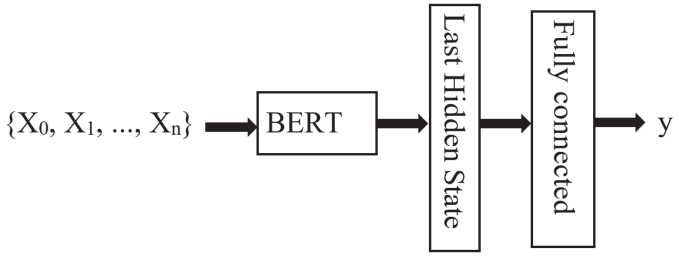

Figure 5 The BERT based neural network.

The bi-directional LSTM architecture adopted in this study (Figure 4) embeds the input sequence of words and their reverse. The embeddings are then concatenated and fed to generic fully connected layers to classify.

BERT-based neural network: BERT-based neural networks (Figure 5) integrate pre-trained language models like BERT [3] as feature extraction modules to leverage contextual word embeddings for various natural language processing tasks.

$$\text{BERToutput} = \text{BERTinput}(\text{Sentence}) \tag{15}$$

where:

BERToutput represents the contextual embeddings generated by BERT, and BERTinput(Sentence) is the input sentence processed by BERT.

4 Evaluation

In this section, the evaluation of the proposed method is discussed. For this purpose, we use the conventional evaluation metrics of IR such as precision, recall, accuracy, and F_1 score, as summarized in Table 2. Precision measures the purity of the retrieved set and recall measures its coverage of the related

Table 2 A confusion matrix and its IR metrics

	Suggestion	Non-suggestion
Suggestion	True-positive (TP)	False-negative (FN)
Non-suggestion	False-positive (FP)	True-Nnegative (TP)

Precision $(Pr) = \frac{TP}{TP+FP}$ Recall $(Re) = \frac{TP}{TP+FN}$ F_1 score $= \frac{2\times Pr\times Re}{Pr+Re}$

Accuracy $= \frac{TP+TN}{TP+FP+TN+FN}$

Table 3 The dataset and all of its components

	# non-suggestions	# suggestions
Train	6415	2085
Validation	296	296
Test	746	87

items. These two metrics are usually in a tradeoff such that higher recall comes with lower precision rates. Hence the F_1 score is utilized as a combined metric. Furthermore, in tasks such as suggestion mining the positive class is of special interest and its per-class metrics are used in comparison scenarios. However, we report the metrics for both of the classes. Accuracy is also reported as a classic measure of supervised learning performance.

4.1 Dataset

The text of user reviews about software developers is taken from the *UserVoice* [23]. It is a labeled dataset that is available on the *Kaggle* website and divided into *train*, *validation*, and *test* sets. This dataset contains user reviews in the software developers' forum, with only a few including suggestions. The primary objective is to identify reviews that offer suggestions for enhancing software. Table 3 provides the information about the dataset *SemEval2019-Task9* in detail.

4.2 Results and Discussion

4.2.1 Rule-based methods

Since the suggestions within the text are usually expressed using imperative speech and auxiliary verbs including *would, could, should, and must*, grammar-based pattern matching and rule generation methods may seem

Table 4 Top discriminative words and phrases

Source	Keywords
Human Expert	please, should, could, might, ought to, would, recommend, suggest, consider, better, allow, would you mind, could you please, I suggest, if you want to, be able to, it would be
BOW, ttest	be, please, would, should, would be, it would be, it would, add, should be, to, be nice, allow, would be nice, be great, would like, like, could, like to, nice, please add, would be great
TFIDF, ttest	be, should, it would be, please, it would, would be, would, add, should be, to, allow, be nice, would be nice, would like, nice, like to, be great, be able, it would be great, please add

helpful; however, experiments revealed that some of these methods cover only a small portion of the dataset, since in general, the suggestion is implicitly expressed. We conducted experiments using *keyword search*, *longest common subsequence (LCS)*, *dyanamic time warping (DTW)*, and *parts of speech (POS) tagging* techniques. In this group of experiments, LCS and DTW got lower results than the baseline methods and hence are not reported. However, keyword searches got competitive results to the average learning models.

Keyword search looks for specific words and phrases within the texts. Table 4 contains three groups of keywords. We induced the first group heuristically based on our knowledge of the English language. The second and third groups are filtered from BOW and TFIDF features using the *ttest* experiment. Since most of the words and phrases in all three groups are stopwords, we should not remove the stopwords in the preprocessing step.

Table 5 shows the results for keyword-based classification using the three groups. The best F_1 score for the suggestion class is achieved with the 17 manually crafted features. BOW and TFIDF keywords achieve significantly lower results. Besides the poor coverage of these methods, their F_1 score also were far below the lowest learning-based result. Hence, we report the results here. Interested readers may consult the GitHub repository of the project.[5]

4.2.2 Results of the learning models

We have conducted extensive experiments covering various hyperparameter configurations, pre-processing options, feature extraction methods, and classification algorithms. We tried to find the best setting of parameters and hyperparameters using the *training* and *validation* sets then we used

[5]https://github.com/cse-teacher/suggestion-mining

Table 5 Results of keyword-based classification

			Non-suggestion			Suggestion		
Group	N	Acc	Pr	Re	F1	Pr	Re	F1
bow	1	0.75	0.92	0.78	0.85	0.19	0.44	0.26
bow	5	0.74	0.95	0.75	0.84	0.24	0.68	0.35
bow	10	0.41	0.96	0.35	0.52	0.14	0.89	0.24
bow	15	0.41	0.96	0.35	0.52	0.14	0.89	0.24
bow	20	0.4	0.97	0.34	0.5	0.14	0.92	0.24
bow	50	0.24	0.96	0.16	0.27	0.12	0.94	0.21
tfidf	1	0.75	0.92	0.78	0.85	0.19	0.44	0.26
tfidf	5	0.75	0.94	0.77	0.85	0.23	0.61	0.34
tfidf	10	0.41	0.96	0.35	0.52	0.14	0.89	0.24
tfidf	20	0.26	0.94	0.19	0.31	0.11	0.90	0.20
tfidf	50	0.24	0.95	0.16	0.27	0.11	0.93	0.20
human	1	0.90	0.91	0.99	0.95	0.59	0.18	0.28
human	5	0.88	0.94	0.93	0.94	0.45	0.51	0.48
human	10	0.87	0.96	0.90	0.93	0.43	0.67	0.52
human	15	0.87	0.97	0.89	0.93	0.44	0.75	0.55
human	17	0.87	0.97	0.89	0.93	0.44	0.76	0.56

$training \cup validtion$ for training and used the *test* set for testing. For instance, after extensive experimentation with varying values of k in the kNN classifier, ranging from 1 to 100, we observed that the classifier demonstrated optimal results within the range of $k = 5 - 21$ with negligible variations.

Table 6 reports the highest scores for the suggestion class using each of the models in the four groups. In summary, F_1 score of the suggestion class varies from 27 for naive Bayes to 76 for the BERT-based neural network, revealing significant differences between various models and configurations.

The basic models focus on feature value distributions. The naive Bayes classifier makes a simple, interpretable, and robust baseline. Since it estimates univariate distributions, it may also handle missing data elegantly. However, for imbalanced and hard classification tasks, as in our case, its performance is lower than more complex models. The kNN and LDA achieve significantly higher $f_1 score$ than naive hover it is far below the nonlinear models in this group. the SVM with RBF kernel has the highest score in this group mentioning the importance of robustness in classification as well as the uniformity of class distributions in local and smooth neighborhoods.

In the group of ensemble methods, boosting methods achieve results competing with the best basic classifier however the three tree-based classifiers

Table 6 Results of the experiments

				Not- Suggestion			Suggestion		
Group	ModelName	Feature	Accuracy	Pr	Re	F1	Pr	Re	F1
Keywords	keyword filter	tfidf: 5	0.75	0.94	0.77	0.85	0.23	0.61	0.34
Keywords	keyword filter	bow: 5	0.74	0.95	0.75	0.84	0.24	0.68	0.35
Keywords	keyword filter	human:17	0.87	0.97	0.89	0.93	0.44	0.76	0.56
Basic	Naïve Bayes	bow	0.91	0.91	1	0.95	0.88	0.16	0.27
Basic	LDA	bow	0.83	0.95	0.86	0.9	0.34	0.6	0.43
Basic	kNN	bow, PCA	0.9	0.93	0.96	0.95	0.54	0.41	0.47
Basic	Decision Tree	bow	0.9	0.96	0.93	0.94	0.52	0.67	0.58
Basic	BayesianRidge	bow	0.92	0.95	0.97	0.96	0.66	0.57	0.61
Basic	LR	bow	0.92	0.96	0.95	0.96	0.61	0.63	0.62
Basic	SVM(RBF)	bow	0.93	0.96	0.96	0.96	0.65	0.63	0.64
Ensemble	AdaBoost	bow	0.91	0.95	0.95	0.95	0.55	0.55	0.55
Ensemble	LightGBM	bow	0.92	0.96	0.95	0.95	0.59	0.64	0.62
Ensemble	balanced RF	bow	0.89	0.98	0.89	0.93	0.48	0.87	0.62
Ensemble	CatBoost	bow	0.92	0.96	0.96	0.96	0.63	0.63	0.63
Ensemble	Grad.Boost	bow	0.92	0.95	0.96	0.96	0.65	0.61	0.63
Ensemble	Extra Trees	bow	0.93	0.96	0.96	0.96	0.67	0.64	0.65
Ensemble	MoE	bow	0.93	0.97	0.95	0.96	0.62	0.75	0.68
Ensemble	Random Forest	bow	0.94	0.96	0.98	0.97	0.79	0.64	0.71
NN	LSTM	Sequence	0.9	0.96	0.92	0.94	0.5	0.68	0.58
NN	MLP	bow	0.91	0.96	0.94	0.95	0.55	0.67	0.60
NN	CNN	Sequence	0.86	0.87	0.96	0.91	0.84	0.57	0.67
NN	BERT(small)	small	0.94	0.98	0.95	0.96	0.66	0.82	0.73
NN	BERT(large)	large	0.95	0.98	0.96	0.97	0.71	0.83	0.76

achieve reasonably higher scores. The winner in this group is the random forest classifier.

An important case in this group is the balanced random forest that induces several random forest learners from balanced inputs (Algorithm 1). The balanced input consists of the minority class plus a random subset of the majority class. For instance, if the minority class has N instances and the majority class has $10 \times N$ instances, the balanced training set will contain N instances from each of the classes. For the test instances, the label is determined as the majority vote of the learners. This ensemble is developed in the hope that it mitigates the class imbalance problem. However, experiments revealed that it is not as good as expected.

While the relatively poor performance of this balanced ensemble can be attributed to the feature representation, model complexity, evaluation metrics, or hyperparameter tuning, we believe that intrinsic data complexity plays a major role. That is, the suggestion class is inherently difficult to distinguish

from non-suggestion, hence the task requires more advanced techniques or a deeper understanding of the domain to achieve significant improvements.

Algorithm 1 Balanced random forest

Require: Training dataset $D = X, y$ with minority ($y = 1$) and majority ($y = 0$) class instances
Require: Number of learners K
Ensure: Ensemble model consisting of K balanced random forest learners
$Ensemble = .$
Set N = Minority class size
Set $D_1 = X_{y=1}, y_{y=1}$
for $k = 1$ to K **do**
    Set $D_0 = RandomSubset(X_{y=0}, y_{y=0})$
    Set $D_k = D_0 \cup D_1$
    $RF_k = TrainRandomForest(D_k)$
    $Ensemble = Ensemble \cup RF_k$
end for
// Combine the predictions of all learners in $Ensemble$ using a voting mechanism:
for each x_j in the test dataset **do**
    $f_0 = 0, f_1 = 0$ // initialize vote frequency for the two classes
    for each RF in $Ensemble$ **do**
        $y_k = RF(x_j)$
        $f_{y_k} = f_{y_k} + 1$ //Increment winner classes votes
        $y_j = argmax(f_0, f_1)$
    end for

end for
Output: The ensemble model $Ensemble$.

The MoE in this group is a mixture of a random forest, gradient boosting, logistic regression, and MLP classifiers. The gating network averages the results of the members. This configuration ranked second in the ensembles, signifying the importance of the differences between the models. However, a slightly better score of the generic random forest classifier again supports the complexity hypothesis.

Within the group of neural networks, the BERT, a transformer-based large language model, achieves considerably better scores, as expected. However, there is a surprise with the results of the LSTM network, since it has been the state of the art for sequence learning, especially in NLP tasks. Besides larger and more complex architecture, the major difference between LSTM and BERT is that the latter employs the *attention mechanism*, which dynamically adjusts the weights of words in the sequence embedding.

4.2.3 Word2vec Analysis

Furthermore, in the hope of capturing the semantic content of the reviews, we used several pre-trained word2vec models along with the Gensim's *doc2vec* trained on the adopted dataset.

We conducted experiments utilizing pre-trained word2vec embedding models. These embedding models are widely used in various NLP tasks such as text classification, sentiment analysis, named entity recognition, and machine translation. The models are usually acknowledged as capturing semantic relationships and similarities between words or documents.

- **glove-wiki-gigaword-50, glove-wiki-gigaword-100, glove-wiki-gigaword-200, glove-wiki-gigaword-300:** This models are trained on a combination of Wikipedia and Gigaword corpus using the GloVe algorithm. They generate word embeddings of 50, 100, 200, and 300 dimensions to capture semantic relationships between words in different granularities.
- **glove-twitter-25, glove-twitter-50, glove-twitter-100, glove-twitter-200:** These models are trained on Twitter data using the GloVe algorithm. They provide word embeddings of different dimensions (25, 50, 100, 200) capturing semantic information from Twitter text.
- **fasttext-wiki-news-subwords-300:** This model is trained on Wikipedia and news data using the FastText algorithm [14]. It provides 300-dimensional word embeddings, including subword information, which can handle out-of-vocabulary words and capture morphological similarities .
- **gensim-fasttext-words:** This model is a FastText-based word embeddings model implemented in the Gensim library. It provides word embeddings trained on your own corpus using the FastText algorithm.
- **word2vec-google-news-300:** This model is trained on a large corpus of Google News articles using the Word2Vec algorithm [19]. It provides 300-dimensional word embeddings capturing semantic relationships from news text.
- **gensim-doc2vec:** This model is implemented in the Gensim library and is based on the Doc2Vec algorithm. It generates document embeddings, also known as paragraph embeddings or sentence embeddings, which represent the semantic meaning of documents.

We retrieved the models from the gensim library [6] [29]. The embedding for a document is computed as the mean of its constituent word embeddings.

[6] https://radimrehurek.com/gensim/

Table 7 Results with the different word2vec embeddings

			Not- Suggestion			Suggestion		
Model	Feature	Accuracy	Pr	Re	F1	Pr	Re	F1
kNN	glove-wiki-gigaword-50	0.84	0.92	0.91	0.91	0.27	0.29	0.28
kNN	glove-twitter-25	0.85	0.92	0.91	0.92	0.31	0.33	0.32
Decision Tree	glove-wiki-gigaword-100	0.75	0.94	0.77	0.85	0.23	0.6	0.34
LDA	glove-wiki-gigaword-200	0.88	0.92	0.95	0.93	0.4	0.31	0.35
kNN	glove-twitter-50	0.85	0.93	0.91	0.92	0.33	0.39	0.36
LDA	glove-wiki-gigaword-300	0.87	0.93	0.92	0.93	0.37	0.38	0.38
LR	glove-twitter-200	0.88	0.93	0.94	0.94	0.43	0.37	0.4
LR	glove-twitter-100	0.89	0.93	0.95	0.94	0.45	0.37	0.41
SVM	fasttext-wiki-news-subwords-300	0.90	0.93	0.97	0.95	0.55	0.37	0.44
SVM	gensim-fasttext-words	0.9	0.94	0.96	0.95	0.54	0.45	0.49
SVM	word2vec-google-news-300	0.89	0.94	0.94	0.94	0.49	0.53	0.51
SVM	gensim-doc2vec	0.91	0.95	0.96	0.95	0.6	0.54	0.57

We also trained the Gensim's *doc2vec* and Fasttext models on the adopted dataset. For each of the models, we report the best F_1 score of the suggestion class in table 7. The results indicate that incorporating word2vec features does not yield reasonable improvements in suggestion identification.

4.2.4 Error analysis

To get an insight on the nature of the errors, we collected false rejects of the winner model and paraphrased it using chatGPT with a simple prompt as *make this sentence imperative*. The model accepted 13 out of 15 instances as positive cases. The rejected instances are instances 5 and 8 in the table. The rejection is because "open-source" and "refactor" are rarely used as verbs. By changing these two verbs to "Make open-source" and "Do refactor", the model accepts both of them. This experiment encourages paraphrasing as a preprocessing step. However, the paraphrasing task has its complexities [31]. We plan to expand this idea in further research.

4.2.5 Comparision with similar works

Table 9 compares the proposed approaches with some presented papers at the $SemEval2019$ conference in the metric of F_1 score.

Table 9 indicates that the proposed approach has a competitive performance to the other approaches. Since our approach applies keyword

Table 8 False negative instances for the BERT NN model and their parphrases

Original Sentence	Paraphrased Sentence
Having an API allowing getting the color of the pixels under a pointer (eventually throttled/async and only when having pointer capture on) would increase the expressiveness of the creation tools in UWP applications.	Implement an API that allows retrieval of pixel colors under a pointer
Unfortunately we can't create UWP apps because there is no support for the custom WCF binding - which is needed - to talk to these instruments.	Provide support for custom WCF binding to facilitate the creation of UWP apps for communication with instruments.
Just like an user have right for review so should developers be able to RESPOND TO EACH AND EVERY REVIEWS submitted to the Store.	Enable developers to respond to every review submitted to the Store
Read email access Locally is very important to create professional and amazing apps for UWP apps, please allow us to achieve this!	Allow local access to email for the development of professional and compelling UWP apps.
The solution is open-sourcing WinDbg and related tools (dbghlp WinDbg extensions etc.) which will allow the community to fix and improve what the rather small team at Microsoft doesn't manage to.	Open-source WinDbg and related tools to allow the community to contribute improvements beyond the capacity of the small Microsoft team.
It is frustrated for a devoloper then an app is'nt Show in any category and the app can only find per direct search by Name.	Ensure that apps are categorized properly to avoid frustration for developers and users.
If the Phone doesn't have an accurate location simply return what it does have.	Return available location information if the phone lacks accurate location data.
Task-based operations would make the code much more clean and readable.	Refactor code to utilize task-based operations for improved cleanliness and readability.
Maybe you should think about some way to reward those users who take their time to actually try an app and review it.	Consider implementing a reward system for users who test and review apps.
SharpDX proves that those key technologies can be made available to CSharp: I would suggest to either support the project or to provide your own interface.	Support SharpDX or provide a native interface to enable access to key technologies in C#.
Read/write accsess to email sync settings could enable applications to create custom sync profiles (based on week-days time position etc).	Grant read/write access to email sync settings for creating custom sync profiles based on various parameters.

(*Continued*)

Table 8 Continued

Original Sentence	Paraphrased Sentence
We need support special multitasking tasks like VOIP and IM!	Provide support for special multitasking tasks such as VOIP and IM.
why shouldn't I as user be able to see reviews made by people from other countries?	Enable users to view reviews from other countries for a broader perspective on app quality.
Thus this idea is actually for the developers to make scrolling faster after each subsequent scroll like every 5 scrolls the scrolling speed will increase..	Develop a feature to increase scrolling speed after every five scrolls
And a smart dialer can then be a full alternative to the built-in phone app since the app also can have a quick-button for goto the regular phone app whenever the user wants to do that.	Create a smart dialer app that serves as a full alternative to the built-in phone app

Table 9 All methods

#Ref	Method	F_1 Score(Average)	F_1 Score(Class 1)
[18]	BERT+Transfer learning	0.85	-
[4]	BiLSTM+Word2Vec	0.56	-
[26]	CNN + BERT	-	0.68
[23]	Ensemble(LR+CNN+GRU FFA)+BERT	-	0.78
Our best model	BERT(large)	0.87	0.76

extraction and also uses Word2Vec embeddings, it can exploit the semantic concept of the text. It is worth noting that keyword-based retrieval is one of the most effective IR approaches.

4.3 Practical Implications

Using the proposed approach, enhanced search and recommendation systems can be realized. By efficiently identifying relevant suggestions within online reviews, blogs, and social media, users can locate desired information and recommendations with greater ease. However, a key limitation arises from its inherent dependency on supervised learning approaches. These methods require large volumes of accurately labeled data for training.

As a potential solution, crowdsourcing has facilitated data acquisition and annotation in related NLP tasks like sentiment analysis and named entity recognition. However, the subjective nature of suggestions presents a unique

challenge. Consider a user complaining about the battery drain caused by phone widgets. One annotator might interpret this as purely negative sentiment, while another might infer a suggestion to limit the type or number of active widgets.

This inter-annotator variability, stemming from the subjective interpretation of suggestions, significantly impacts the quality of labeled data and subsequently, the performance of trained models. Accurately capturing suggestions requires an understanding of implicit meaning, context, and potential actions.

Exploring semi-supervised or weakly-supervised learning is another solution. Utilizing unlabeled data alongside a smaller set of labeled data can potentially address the data scarcity issue and incorporate valuable information from real-world scenarios.

5 Conclusion

Users generally express their opinions about items through online reviews, blogs, or social media platforms. Along with positive and negative sentiments, the opinions may also contain suggestions for improving the product or advice to active and prospective users. However, with the growing number of documents on the web and the large collection of texts, finding the required document among the mass of data and information retrieval has become very complicated. Therefore, improving this area is important. Suggestion mining is an emerging field in the IR domain.

In this paper, IR approaches are used for suggestion mining. At first, several approaches are used to induce meaningful feature vectors for the opinion texts, and then supervised learning methods and deep neural networks are employed to identify suggestions in the text. The evaluations are performed on the known datasets and the results show that the proposed approaches are capable of automatically extracting suggestions with acceptable accuracy. Therefore, the proposed method can be used in recommendation systems and decision support systems.

Finding the exact piece of text conveying the suggestion and identifying frequent suggestions, as well as more specialized tasks such as defect report recognition, are a few directions for further application-oriented research. On the other hand, recent developments in the language understanding domain such as transformer networks draw the second line of research that mainly focuses on the methods and procedures in the hope that the anticipated tasks are conducted more efficiently and accurately.

Declarations

Funding

The authors did not receive support from any organization for the submitted work.

Conflict of Interest/Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Authors' Contributions

All authors contributed to the study conception and design. Material preparation, data collection, and analysis were performed by *Zahra Hadizadeh*. The first draft of the manuscript was written by *Zahra Hadizadeh* and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Ethical Approval

This study does not involve human or animal participants.

Availability of Supporting Data

The datasets generated and analyzed during the current study are available from the original provider.[7] The code for current research along with the results and some of the trained models are also available on the corresponding author's GitHub.[8]

Acknowledgments

The authors would like to thank *Sapna Negi* for her helpful comments.

We acknowledge the use of ChatGPT, developed by OpenAI,[9] for assistance in the refinement of our text. The authors take full responsibility for the content and interpretation presented in this work

[7] https://github.com/Semeval2019Task9/Subtask-A

[8] https://github.com/cse-teacher/suggestion-mining

[9] https://chat.openai.com/

References

[1] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Caroline Brun and Caroline Hagege. Suggestion mining: Detecting suggestions for improvement in users' comments. *Res. Comput. Sci.*, 70(79):171–181, 2013.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Yunxia Ding, Xiaobing Zhou, and Xuejie Zhang. Ynu_dyx at semeval-2019 task 9: A stacked bilstm for suggestion mining classification. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1272–1276, 2019.

[5] Aysu Ezen-Can and Ethem F Can. Hybrid rnn at semeval-2019 task 9: blending information sources for domain-independent suggestion mining. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1199–1203, 2019.

[6] Tirana Fatyanosa, Al Hafiz Akbar Maulana Siagian, and Masayoshi Aritsugi. DBMS-KU at SemEval-2019 task 9: Exploring machine learning approaches in classifying text as suggestion or non-suggestion. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1185–1191, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics.

[7] Jerome H Friedman. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[8] Hitesh Golchha, Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. Helping each other: a framework for customer-to-customer suggestion mining using a semi-supervised deep neural network. *arXiv preprint arXiv:1811.00379*, 2018.

[9] Andrew B Goldberg, Nathanael Fillmore, David Andrzejewski, Zhiting Xu, Bryan Gibson, and Xiaojin Zhu. May all your wishes come true: A study of wishes and how to recognize them. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 263–271, 2009.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[11] Claudia Iacob and Rachel Harrison. Retrieving and analyzing mobile apps feature requests from online reviews. In *2013 10th working conference on mining software repositories (MSR)*, pages 41–44. IEEE, 2013.

[12] Harsh Jhamtani, Niyati Chhaya, Shweta Karwa, Devesh Varshney, Deepam Kedia, and Vineet Gupta. Identifying suggestions for improvement of product features from online product reviews. In *International Conference on Social Informatics*, pages 112–119. Springer, 2015.

[13] Rosie Jones and Fuchun Peng. *Web Search Query Rewriting*, pages 3491–3493. Springer US, Boston, MA, 2009.

[14] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of Tricks for Efficient Text Classification. 2016. Accessed: April 2024.

[15] Navid Khaledian, Amin Nazari, Keyhan Khamforoosh, Laith Abualigah, and Danial Javaheri. Trustdl: Use of trust-based dictionary learning to facilitate recommendation in social networks. *Expert Systems with Applications*, 228:120487, 2023.

[16] Mojtaba Kordabadi, Amin Nazari, and Muharram Mansoorizadeh. A movie recommender system based on topic modeling using machine learning methods. *International Journal of Web Research*, 5(2):19–28, 2022.

[17] Naveen Kumar Laskari and Suresh Kumar Sanampudi. Explainable system for suggestion mining using attention. In *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 1, pages 679–684. IEEE, 2022.

[18] Naveen Kumar Laskari and Suresh Kumar Sanampudi. Aspect-oriented suggestion mining from opinion reviews. *Journal of Theoretical and Applied Information Technology*, 101(12), 2023.

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. 2013. Accessed: April 2024.

[20] Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. A study of suggestions in opinionated texts and their automatic detection. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 170–178, 2016.

[21] Sapna Negi and P Buitelaar. Suggestion mining from opinionated text. *Sentiment Analysis in Social Networks*, pages 129–139, 2017.

[22] Sapna Negi and Paul Buitelaar. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015*

Conference on Empirical Methods in Natural Language Processing, pages 2159–2167, 2015.

[23] Sapna Negi, Tobias Daudert, and Paul Buitelaar. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th international workshop on semantic evaluation*, pages 877–887, 2019.

[24] Thi-Lan Ngo, Khac Linh Pham, Hideaki Takeda, Son Bao Pham, and Xuan Hieu Phan. On the identification of suggestion intents from vietnamese conversational texts. In *Proceedings of the 8th International Symposium on Information and Communication Technology*, pages 417–424, 2017.

[25] Samuel Pecar, Marian Simko, and Maria Bielikova. Nl-fiit at semeval-2019 task 9: Neural model ensemble for suggestion mining. *arXiv preprint arXiv:1904.02981*, 2019.

[26] Sai Prasanna and Sri Ananda Seelan. Zoho at semeval-2019 task 9: Semi-supervised domain adaptation using tri-training for suggestion mining. *arXiv preprint arXiv:1902.10623*, 2019.

[27] A Ramesh, K Pradeep Reddy, M Sreenivas, and Para Upendar. Feature selection technique-based approach for suggestion mining. In *Evolution in Computational Intelligence: Proceedings of the 9th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA 2021)*, pages 541–549. Springer, 2022.

[28] T Raghunadha Reddy, P Vijayapal Reddy, T Murali Mohan, and Raju Dara. An approach for suggestion mining based on deep learning techniques. In *IOP Conference Series: Materials Science and Engineering*, volume 1074, pages 1–10. IOP Publishing, 2021.

[29] Radim Rehurek and Petr Sojka. Gensim: Topic modelling for humans. https://radimrehurek.com/gensim/. Accessed: April 2024.

[30] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[31] Hassan Shahmohammadi, MirHossein Dezfoulian, and Muharram Mansoorizadeh. Paraphrase detection using lstm networks and handcrafted features. *Multimedia Tools and Applications*, 80(4):6479–6492, 2021.

[32] Jie Wu, Tong Yang, Zhiwei Zhou, and Narisa Zhao. Consumers' affective needs matter: Open innovation through mining luxury hotels' online reviews. *International Journal of Hospitality Management*, 114:103556, 2023.

[33] Ping Yue, Jin Wang, and Xuejie Zhang. Ynu-hpcc at semeval-2019 task 9: using a bert and cnn-bilstm-gru model for suggestion mining. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1277–1281, 2019.

[34] Min Zhang, Lin Sun, Yuzhuo Li, G Alan Wang, and Zhen He. Using supplementary reviews to improve customer requirement identification and product design development. *Journal of Management Science and Engineering*, 2023.

[35] Hua Zhao, PeiXin Zhang, Yue Liang, and Sitenda Amos. A comparative study on chinese open domain suggestion mining. *Journal of Intelligent & Fuzzy Systems*, 42(6):5385–5398, 2022.

[36] Qimin Zhou, Zhengxin Zhang, Hao Wu, and Linmao Wang. Zqm at semeval-2019 task9: a single layer cnn based on pre-trained model for suggestion mining. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1287–1291, 2019.

Biographies

Zahra Hadizadeh received her B.Sc. degree in Computer Software Engineering from Payame-Noor University in Germi, Ardabil, Iran in 2015, followed by an M.Sc. degree in Artificial Intelligence from Bu-Ali Sina University in 2022. Her research interests include text mining, information retrieval, and recommender systems.

Amin Nazari received his B.Sc. degree in Computer Software Engineering from Islamic Azad University in Hamedan in 2009, followed by an M.Sc. degree in Computer Software Engineering from Arak University in 2015. Currently, he is a Ph.D. candidate in artificial intelligence at Bu-Ali Sina University in Hamedan. His research interests include wireless sensor networks, the Internet of Things, IoT-fog networks, and recommender systems.

Muharram Mansoorizadeh is an associate professor at the Computer Engineering Department of Bu-Ali Sina University. He received his B.Sc. degree in software engineering from the University of Isfahan, Isfahan, Iran, in 2001, and his M.Sc. degree in software engineering and Ph.D. in computer engineering from Tarbiat Modares University, Tehran, Iran, in 2004 and 2010, respectively. His current research interests include machine learning, affective computing and information retrieval.