DSI DATA SCIENCE AND INFORMETRICS
VOL.3, NO.2, MAY. 2023

RESEARCH ARTICLE

An up –to –date comparative analysis of the KNN classifier distance metrics for text categorization

Onder Coban

Ataturk University, Faculty of Engineering, Department of Computer Engineering, Erzurum, Turkiye

ABSTRACT

Text categorization (TC) is one of the widely studied branches of text mining and has many applications in different domains. It tries to automatically assign a text document to one of the predefined categories often by using machine learning (ML) techniques. Choosing the best classifier in this task is the most important step in which k-Nearest Neighbor (KNN) is widely employed as a classifier as well as several other well-known ones such as Support Vector Machine, Multinomial Naive Bayes, Logistic Regression, and so on. The KNN has been extensively used for TC tasks and is one of the oldest and simplest methods for pattern classification. Its performance crucially relies on the distance metric used to identify nearest neighbors such that the most frequently observed label among these neighbors is used to classify an unseen test instance. Hence, in this paper, a comparative analysis of the KNN classifier is performed on a subset (i.e., R8) of the Reuters-21578 benchmark dataset for TC. Experimental results are obtained by using different distance metrics as well as recently proposed distance learning metrics under different cases where the feature model and term weighting scheme are different. Our comparative evaluation of the results shows that Bray-Curtis and Linear Discriminant Analysis (LDA) are often superior to the other metrics and work well with raw term frequency weights.

KEYWORDS

Text categorization; k-nearest neighbor; distance metric; distance learning algorithms

1 Introduction

Text categorization (TC) is a supervised Machine Learning (ML) task where a learning algorithm is trained on training documents at first and then the learning algorithm is expected to assign a label to the input test document (Dhar et al., 2021; Ghawi & Pfeffer, 2019). Due to the availability of a huge volume of text data, TC task has become one of the key ML applications in the fields of information retrieval, knowledge mining, text summarization, and so on (Chen, 2018; Dhar et al., 2021). At present, there exists great progress on this topic which tries to categorize text data using well-known classifiers as well as deep learning algorithms (Chen, 2018). On the other hand, the task of TC has now many applications in different do-

* Corresponding Author: onder.coban@atauni.edu.tr

mains in which there exists a need to automatically organize, manage, and categorize texts. News categorization, spam e-mail or SMS detection, sentiment analysis or opinion mining from reviews, and hate speech detection from user-generated content are examples of TC applications (Chen, 2018). On the traditional side, the TC task involves using hand-crafted features used to feed well-known classifiers such as support vector machines, multinomial naive Bayes, random forests, and so on. Among these algorithms, the KNN classifier has been widely employed in TC tasks and is one of the top-performing methods (Ghawi & Pfeffer, 2019).

The traditional KNN method has higher computational complexity (Ghawi & Pfeffer, 2019) and therefore it is often called a lazy learner which needs to calculate the similarity or distance between the test document and all the documents in the training set to detect nearest documents. The Euclidean distance is the most common distance metric (Chen et al.,2020) used to measure the distance between two vectorized text instances, but there exist many other metrics such as Chebyshev, Minkowski, Manhattan, and so on. Even though KNN is suffering from its complexity, it has several advantages as well to employ in classification problems. Some of these advantages are as follows (Bishnoi & Hisar, 2022): (i) KNN is highly unbiased in nature and does not take any assumption into consideration about underlying data, (ii) it produces output with less calculation time and easy interpretation, (iii) easy to implement and no re-training is required if new data is included into the training set.

KNN algorithm is therefore a popular classifier for classification tasks including TC and there exist many research efforts to improve its performance considering different aspects. Among the numerous studies, distance metric learning (DML) (Yang & Jin, 2006) is just an example of these efforts aiming to enable the KNN to significantly improve classification accuracy compared to the standard Euclidean distance. All of these efforts show that KNN as a conventional algorithm is very popular in classification tasks including TC due to its advantages, especially on small and medium-sized datasets. The majority of the previous TC tasks involving the KNN classifier use the standard Euclidean distance metric with a varying number of neighbors. In the literature, comparative analysis of distance metrics for KNN classification on general data has been studied many times before (Abu Alfeilat et al., 2019; Dhar et al., 2021).

Text categorization of Marathi news is performed by using ML techniques including KNN classifier in Lade & Dhore (2021). A narrow comparison of six distance metrics on a news articles text dataset of 10 categories showed that cosine is superior to others in terms of the number of wrongly categorized documents (Cheng et al., 2012). A comparison of several well-known distance metrics with the newly proposed one is performed on a dataset of Turkish news (Eminagaoglu, 2022). Excluding the proposed metric, the results of this study show that Bray-Curtis dissimilarity often provides better results for KNN classification of term frequency-inverse document frequency (tf*idf) weighted bag-of-words (bow) features. Another narrow-scope evaluation of distance metrics on a Bangle news text corpora is performed in Dhar et al., (2019), which shows that Mahalanobis is superior to other metrics including Minkowski, Canberra, Squared Euclidean, Manhattan, and Chebyshev. Comparison of well-known distance metrics for KNN classification on a subset of the Reuters-21578 data as well as Turkish Twitter feeds dataset is studied in Çoban et al. (2015) which shows that euclidean and manhattan metrics are often superior to other metrics including cosine, dice, extended-Jaccard, and so on. A comparison of euclidean, manhattan, and cosine distance metrics is performed on a text dataset devoted to detecting hate speech (Abu Alfeilat et al.,

2019). The results of this study show that KNN classification with euclidean distance obtains better accuracy. Another study evaluates distance metrics on a dataset of e-commerce reviews for KNN classification (Jain et al., 2020). This study shows that Manhattan provides better results than other well-known seven ones such as Euclidean, Chebyshev, Cosine, Dice, Jaccard, and so on.

Our review of the literature, however, shows that there exist only some narrow-scope evaluations (Abu Alfeilat et al., 2019; Cheng et al., 2012; Çoban et al., 2015; Dhar et al., 2019; Eminagaoglu, 2022; Lade & Dhore, 2021) of distance metrics for TC task (or its applications) and we could not detect any up-to-date and comprehensive study devoted to this purpose in recent years. Hence, in this study, we perform an up-to-date comparative evaluation of 17 distance metrics as well as 3 DML algorithms for the KNN classifier employed on a benchmark dataset. Please note that the reason behind selecting three DML algorithms is the suitability of these algorithms for TC and not requiring to handle some challenges stemming from the nature of both data and task. We investigate the performance of the classifier under different cases, where distance metric, feature set, and weighting scheme are different. The results of our extensive experiments show that Bray-Curtis and LDA are often superior to the other metrics and work well with raw term frequency weights.

We believe that the findings of this study will contribute to the literature by providing insight for ML practitioners who study TC applications involving the KNN classifier, especially on small and medium-sized datasets.

2 Material and Methods

This section presents the details of the material and methods used in this study under the following sub-headings.

2.1 Dataset

The dataset underlying this study is a subset of the Reuters-21578 which is the most widely used text collection for text categorization research. The data (Reuters-21578, 2022) has a standard ModApte train/test split of manually labeled documents which are composed of the Reuters newswire texts in 1987. Since the data has a very high skewness of documents across categories, several sub-collections (e.g., R10, R52, R90) are usually used in text categorization tasks.

One of these subsets is R8 which includes documents only with a single topic and observed at least one of the most frequent eight categories for both training and test set. The quantitative description of the R8 dataset is given in Figure 1.

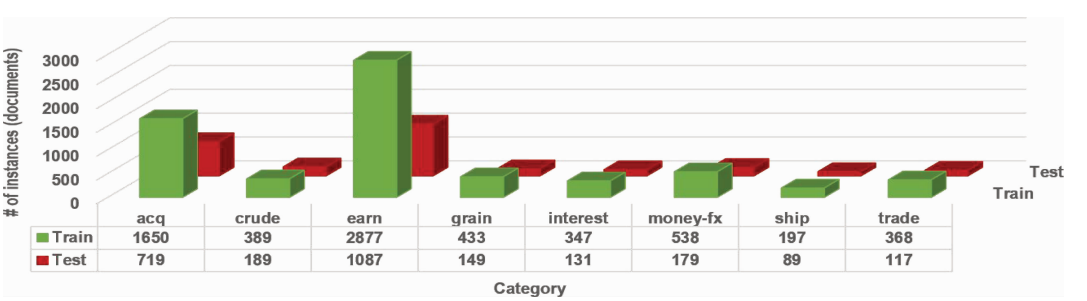

Figure 1 The number of documents within the train and test sets of the R8 data.

2.2 Methods

This section gives details on our methods including preprocessing, feature extraction, term weighting, classification, and performance evaluation under the following sub-headings respectively.

2.2.1 Preprocessing and feature extraction

In this step, we applied two different preprocessing depending on the feature set extracted from texts. We use the bow and character level trigram features (Dhar et al., 2021) to convert texts into numerical instance vectors. We remove punctuations, digits, multiple white spaces, and terms with lengths lower than two from texts before extracting trigram features. In the bow model, we apply the same steps along with stop word removal and stemming.

Note that we use the built-in English stop word list of NLTK (Bird, 2006) library to remove stopwords and Porter Stemmer (Porter, 2001) to reduce words into their roots. Please note that we only keep features observed in at least five documents in the collection (i.e., minimum document frequency filter) for both of the feature models.

2.2.2 Term weighting

Upon completion of the preprocessing and feature extraction, we used three well-known term weighting schemes (Çoban, 2016), namely, term frequency (tf), binary, and tf*idf to convert texts into term-document matrices. Note that the binary scheme assigns a weight value of 1 for any feature observed in a document at hand, otherwise, the feature's weight value gets zero. The tf scheme assigns the raw observed frequency of a feature as its weight value. On the other hand, the tf*idf scheme assigns feature weights considering both their local raw frequencies and the inverse of their document frequencies. The reader is advised to (Çoban, 2016; Dhar et al., 2019) for more detailed information on term weighting schemes.

2.2.3 Feature Selection

Feature selection is an important step in TC and aims to discover a subset of features that effectively represent the original dataset (Deng et al., 2019). In this study, we use one of the widely used and performant filter-based feature selectors namely chi-square (Zhai et al., 2018) to perform our analysis on a lower dimension of discriminative features. Please note that the chi-square method computes feature goodness scores by considering statistics between features and categories.

2.2.4 Classification

Upon completion of the feature extraction and selection, we perform classification with the help of the KNN classifier (Bishnoi & Hisar, 2022; Chen et al., 2020; Zhou et al., 2021) which is one of the simplest supervised ML algorithms. This algorithm tries to predict the label of a test instance based on the distances of each test instance from the training dataset. Even though the distance is often computed by the Euclidean function, it can also be computed by using several other alternative functions. In this study, we use different distance metrics to provide an up-to-date and comprehensive analysis of the effect of distance metrics on the performance of the KNN classifier employed in text categorization. Considering the data type, we use the following types of distance metrics as well as some of the recently proposed distance learning metrics.

Table 1 Functions and abbreviated (abbr.) names of metrics intended for real, integer, and boolean-valued vector spaces.

Metric	Abbr.	Function	Metric	Abbr.	Function
Euclidean	euc	$sqrt(sum((x-y)^2))$	Jaccard	jac	$NNEQ / NNZ$
Manhattan	man	$sum(\|x-y\|)$	Matching	mtc	$NNEQ / N$
Chebyshev	che	$\max(\|x-y\|)$	Dice	dic	$NNEQ / (NTT + NNZ)$
Minkowski	min	$sum(w*\|x-y\|^p)^{1/p}$	Kulsinski	kul	$(NNEQ + N - NTT) / (NNEQ + N)$
Squared Euclidean	seuc	$sqrt(sum((x-y)^2/V))$	Rogerstanimoto	rog	$2*NNEQ / (N + NNEQ)$
Mahalanobis	mah	$sprt((x-y)'^{V-1}(x-y))$	Russellrao	rus	$(N-NTT) / N$
Hamming	ham	$N_{unequal}(x,y)/N_{total}$	Sokalmichener	smic	$2*NNEQ / (N+NNEQ)$
Canberra	can	$sum(\frac{\|x-y\|}{\|x\|+\|y\|})$	Sokalsneath	ssne	$NNEQ / (NNEQ + 0.5*NTT)$
Bray–Curtis	bry	$\frac{sum(\|x-y\|)}{sum(\|x\|)+sum(\|y\|)}$			

2.2.4.1 Metrics for real– and integer–valued vector spaces

These metrics are intended to work with real and integer-valued vector spaces (VS). In this study, we use Euclidean, Manhattan, Chebyshev, Minkowski, Squared Euclidean, and Mahalanobis distance metrics from the scikit-learn (Pedregosa et al., 2011) package. The formal definitions of these metrics are given in Table 1, where their names are given in the first column.

2.2.4.2 Metrics for boolean–valued vector spaces

These metrics are intended for boolean-valued vector spaces where any non-zero entry is considered to be True. In this study, we use Jaccard, Matching, Dice, Kulsinski, Rogerstanimoto, Russellrao, Sokalmichener, and Sokalsneath metrics from the scikit-learn (Pedregosa et al., 2011) package. The formal definitions of these metrics are given in Table 1, where their names are given in the fourth column.

Please note that the abbreviated values for boolean metrics have the following definitions: N is the number of dimensions, NTT is the number of dimensions in which both values are True, NTF is the number of dimensions in which the first value is True and the second is False, NFT is the number of dimensions in which the first value is False and the second is True, and NFF is the number of dimensions in which both values are False. On the other hand, NNEQ is equal to NTF + NFT and NNZ is equal to NTF + NFT + NTT.

2.2.4.3 DML algorithms

DML aims at learning a distance to make the instances within the same category have a relatively close measure while the instances in different categories have a far away distance from others (Suarez et al., 2020). In the classification phase, the instances at hand are projected into the learned metric space and several of the instances are used as distance measures. In this study, we use information theoretic metric learning (ITML), LDA, and principal component analysis (PCA) methods. The reader is advised to (Zhou et al., 2021) for more detailed information and a comparative evaluation of well-known distance learning metrics on animals with attribute dataset.

2.2.5 Performance evaluation

As our dataset has a default train-test split, we do not use an additional evaluation strate-

gy but we measure the performance of the KNN by using the f-macro score that calculates metrics for each label and finds their unweighted mean. Note that we intentionally preferred this type of f1-score on our imbalanced R8 dataset since the f-macro does not take label imbalance into account.

3 Results and Discussion

This section gives the details on our experimental setup as well as our findings under the following sub-headings respectively.

3.1 Experimental Setup

In this study, we applied the preprocessing, feature extraction, feature selection, classification, and performance evaluation steps with the help of the scikit-learn (Pedregosa et al., 2011) Python package. We use the NLTK (Bird, 2006) Python package for fetching and creating the R8 dataset as well as removing stopwords and applying Porter Stemmer on bow features in preprocessing phase. We also use pyDML (Suarez et al., 2020) Python package to employ DML algorithms. We would like to note that we feed all required methods and packages (e.g., NumPy) with a random seed value of 42 to make our results reproducible. In the classification phase, we left all parameters of the KNN algorithm untouched except for "n_neighbors" and "metric" which are used to compute the distance between two instance vectors.

3.2 Results

After completing the experimental setup, we preprocessed and converted the document into term-document matrices using both bow and trigram features. This process yielded 4,606 unique bow features and 10,668 unique trigram features. First of all, we performed KNN classification configured to run with euclidean distance of 5 nearest neighbors. Accordingly, we obtained f-scores of 0.582, 0.663, and 0.760 on binary, tf, and tf*idf weighted bow features respectively. On the other hand, we obtained f-scores of 0.631, 0.683, and 0.742 on binary, tf, and tf*idf weighted trigram features respectively. These results show that using binary and tf weighting schemes on trigram features provides better results than the results of bow features. However, tf*idf weighting is superior to binary and tf schemes for both of the feature models, and the highest f-score is obtained as an f-score of 0.760 by using 4,606 bow features.

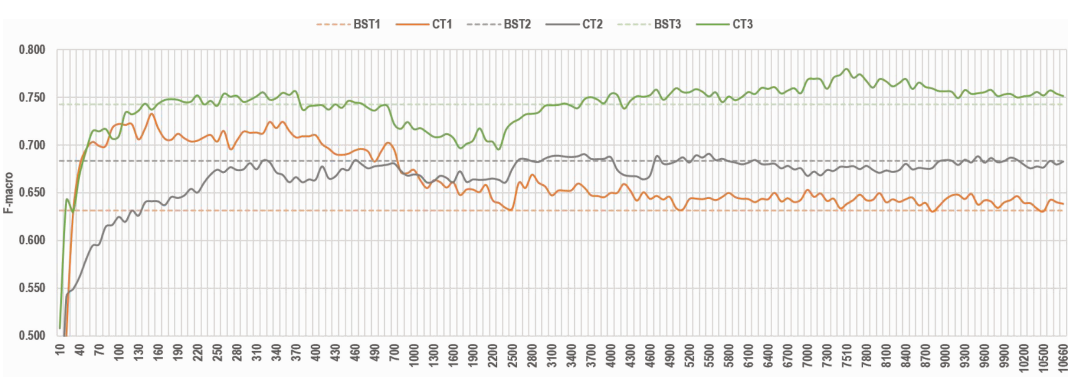

Figure 2 Obtained f-scores on trigram features with respect to different weighting schemes.

Next, we obtained the results by using the most discriminative features selected by the chi-square feature selector in a step-wise procedure. Please note that in the feature selection phase, we increased the number of features to be selected by 10 at each step and considered the cases when the KNN reaches or goes beyond the respective baseline results obtained with the complete set of features. Figure 2 shows the obtained f-scores with respect to the number of features selected by chi-square on trigram features. In Figure 2, the three baseline results obtained on trigram features considering binary, tf, and tf*idf schemes are represented by BST1, BST2, and BST3 respectively. Similarly, results obtained with chi-square feature selector on binary, tf, and tf*idf weighted trigram features are shown with CT1, CT2, and CT3 respectively.

As seen in Figure 2, feature selection improves the classification f-score in all three cases. The KNN classifier goes beyond the respective baseline f-score (i.e., 0.631) of binary weighting just by using the 150 most discriminative features. This is also observed for the other two cases on trigram features.

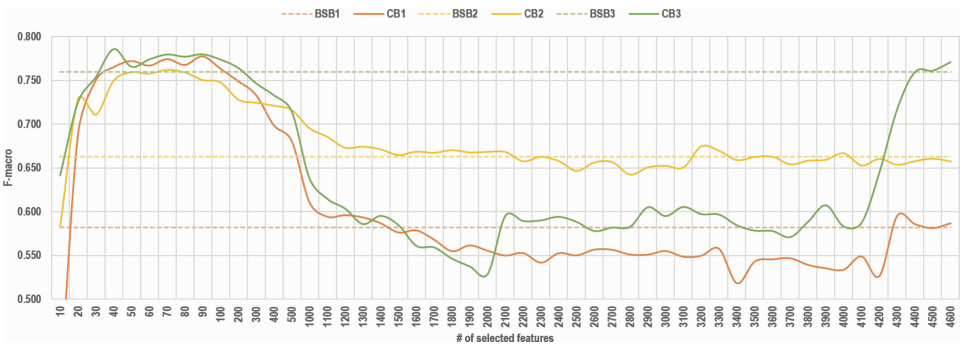

Figure 3 Obtained f-scores on bow features with respect to different weighting schemes.

The results of features selection on bow features considering three different weighting schemes are depicted in Figure 3, in which BSB1, BSB2, and BSB3 stand for the respective baseline scores of binary, tf, and tf*idf schemes respectively. The CB1, CB2, and CB3 again represent the results obtained with features selection on binary, tf, and tf*idf weighted bow features respectively. As seen in Figure 3, feature selection again improves the classification f-scores for three cases.

Please note that on trigram features (see Figure 2), the f-score of the classifier is akin to increasing when the number of features gets approximately halfway. This behavior seems to be true for tf weighting as well. This is because binary weighting ignores raw feature frequency, while tf and tf*idf consider local frequencies of features. The main reason behind climbing after halfway (especially for tf*idf) is due to added features after this step has more discriminative power across categories. Their power enables us to obtain discriminative feature weights and consequently including these features into the dataset increases f-score. A very similar behavior is also observed for bow features (see Figure 3) especially for tf*idf weighted ones due to the same reason.

To give a more easy-to-follow understanding, we summarized our results obtained both with feature selection and without feature selection in Table 2.

Table 2 F-scores and the number of unique features obtained with feature selection and without feature selection considering different feature sets and weighting schemes.

With feature selection							Complete set of features (no feature selection)						
Result	bow			trigram			Result	bow			trigram		
	binary	tf	tf*idf	binary	tf	tf*idf		binary	tf	tf*idf	binary	tf	tf*idf
f–score	0.777	0.761	0.785	0.733	0.684	0.743	f–score	0.582	0.663	0.760	0.631	0.683	0.742
# of features	90	70	40	150	320	140	# of features	4,606			10,668		

As seen in Table 2, feature selection improves the f-score of the KNN in all cases by using a lower number of features. For example, we obtain an f-score of 0.760 by using the complete set of 4,606 tf*idf weighted bow features, while we obtain a higher f-score of 0.785 just by using the selected most discriminative 40 features. As the KNN is a lazy learner, we use the selected set of features for each of the respective feature sets and weighting scheme combinations in the rest of our experiments. Note that the purpose of this preference is to run the KNN on a low-dimensional feature space and our new baseline results are now turning into the f-scores obtained with feature selection.

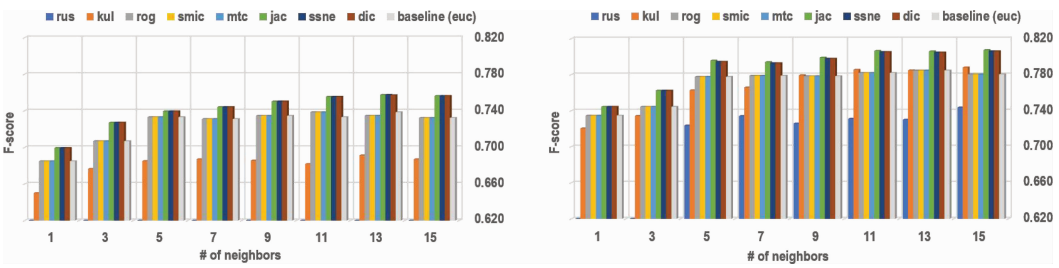

Figure 4 F-scores obtained with different boolean distance metrics employed on binary weighted (a) trigram, (b) bow features.

After detecting the best set of features for each scenario, we performed classification experiments using different distance metrics. In the first step, we used distance metrics intended for boolean vector spaces on the binary-weighted bow and trigram features selected by chi-square. The results of these experiments are depicted in Figure 4 which shows that the classification f-score is often increasing when the number of neighbors increases. In both cases, metrics intended for binary vector spaces have similar behavior, but the Russellrao and Kulsinski metrics are not often able to reach their respective baseline f-score. The highest results are obtained with Jaccard, Sokalsneath, and dice metrics in all cases. However, as seen in Figure 4a, the best f-score of 0.7574 is obtained by both Jaccard and Sokalsneath metrics on trigram features when the number of neighbors is 13. As seen in Figure 4b, on the other hand, the best f-score of 0.8073 is obtained by the Jaccard metric on bow features when the number of neighbors is set to 15. These results show that using Jaccard, Dice, or Sokalsneath metrics on binary-weighted term-document matrix provides better results than using the euclidean and other metrics intended for binary vector spaces.

In the second step, we performed KNN classification experiments by using different distance metrics intended for real and integer-valued vector spaces as well as three DML algorithms (i.e., ITML, LDA, and PCA). In this step, we again obtained results for different cases

where the weighting scheme, distance metric, and the number of neighbors are different.

As seen in Figure 5a, metrics except for Mahalanobis and ITML reach or go beyond the respective baseline f-scores on binary-weighted bow features. The highest results are often obtained by Bray-Curtis and LDA metrics of which the latter improves the highest respective baseline f-score of 0.7804 to an f-score of 0.8064 when k is 15. As seen in Figure 5b, Chebyshev, Mahalanobis, Hamming, and Manhattan metrics are not able to reach the respective baseline results almost in all cases.

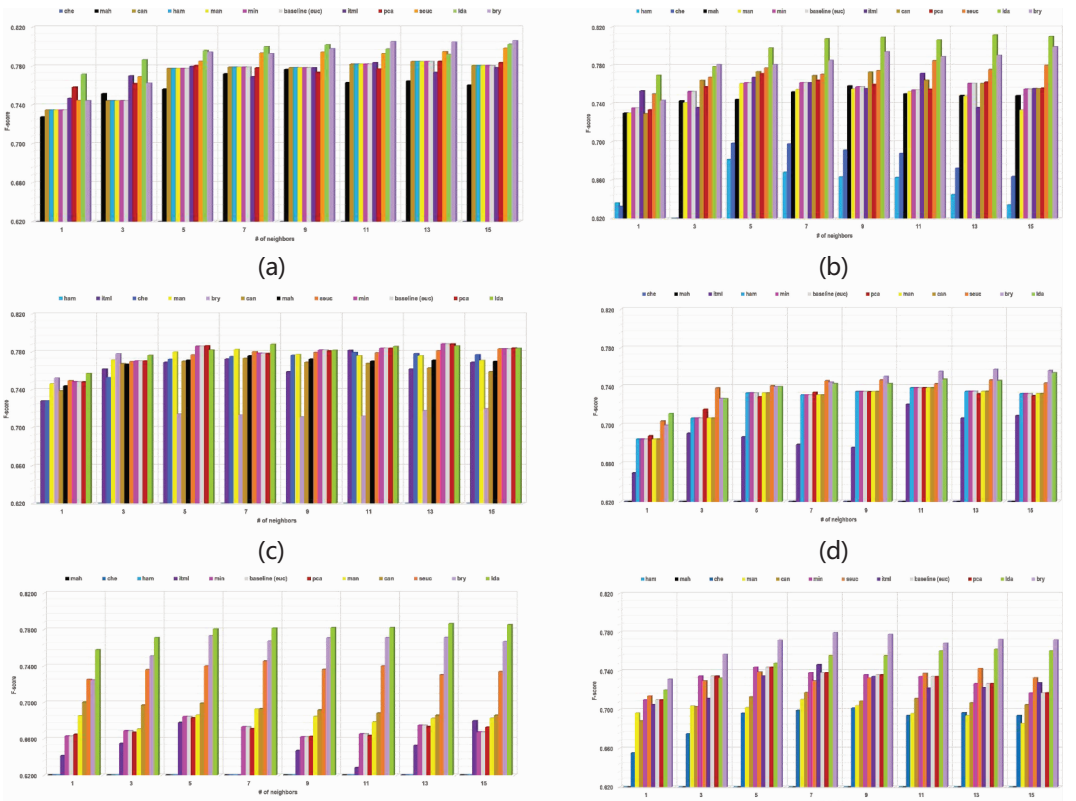

Figure 5 F-scores obtained with distance metrics employed on (a) binary weighted bow, (b) tf weighted bow, (c) tf*idf weighted bow, (d) binary weighted trigram, (e) tf weighted trigram, and (f) tf*idf weighted trigram features.

The most successful metrics are again Bray-Curtis and LDA metrics of which the latter again improves the respective baseline f-score of 0.7614 to an f-score of 0.8118 which is the best result on the tf-weighted bow features. As seen in Figure 5c, only PCA, LDA, and Minkowski metrics reach the respective f-scores in all cases. Interestingly, the Bray-Curtis metric is starting to fall behind the other metrics on tf*idf weighted bow features when the number of neighbors increases. In this case, PCS and Minkowski metrics obtain an equal f-score of 0.7881 which is the best score for this case and equal to the respective baseline f-score when the number of neighbors is 13. On the other hand, LDA outperforms all of the other metrics (including the baseline) and provides the second-best f-score of 0.7811 when the number of neighbors is 7.

The behavior of metrics seems not to have major changes on trigram features as well. As seen in Figure 5d, ITML, Canberra, Chebyshev, and Manhattan metrics are again not able to reach the baseline f-scores on binary-weighted trigram features. On the other hand, Bray-Curtis and LDA metrics provide the highest results almost in all cases. The best f-score of 0.7795 is obtained by the Bray-Curtis metric and this value is greater than the respective baseline f-score of 0.7382 when the number of neighbors is 7. As seen in Figure 5e, Mahalanobis, Chebyshev, ITML, and Manhattan again do not reach the respective baseline f-scores.

The most successful metrics are Bray-Curtis and LDA of which the LDA outperforms the others in all cases. The best result of the LDA on tf-weighted trigram features is a value of 0.7863 which is greater than the respective baseline f-score of 0.6735 for setting the number of neighbors to 13. Finally, as seen in Figure 5f, very similar behavior is observed such that ITML, Mahalanobis, and Chebyshev fall behind the baseline method (i.e., euclidean) in all cases. In contrast, Squared Euclidean, Bray-Curtis, and LDA outperform the baseline metrics in all cases on tf*idf weighted trigram features. In this case, the highest results are often obtained by Bray-Curtis which is providing the best f-score of 0.7574 which is higher than the respective baseline f-score of 0.7346 when the number of neighbors is 13.

Table 3 Summarized results of the experiments for different distance metrics considering the best and second best ones.

Feature Set	Weighting Scheme	Intended for ··· VS		The Best Metric(s)	F–score		# of Neighbors
		boolean	real and integer		Obtained	Respective Baseline	
Bow	Binary	×	√	Bray–Curtis, LDA	0.8060, 0.8024	0.7804	15
Bow	Binary	√	×	Jaccard, [Dice, Sokalsneath]	0.8073, 08060	0.7804	15
Bow	Tf	×	√	LDA, Bray–Curtis	0.8118, 0.7994	0.7614	13
Bow	Tf*idf	×	√	[PCA, Minkowski], LDA	0.7881, 0.7886	0.7881	13, 7
Trigram	Binary	×	√	Bray–Curtis, LDA	0.7574, 0.7464	0.7346	13
Trigram	Binary	√	×	[Jaccard, Sokalsneath]	0.7574	0.7384	13
Trigram	Tf	×	√	LDA, Bray–Curtis	0.7863, 0.7712	0.6748	13
Trigram	Tf*idf	×	√	Bray–Curtis, LDA	0.7795, 0.7560	0.7382	7

Summarizing all of the results obtained so far in Table 3 shows that there is a slight difference between using binary and non-binary intended metrics on binary term-document matrices. On the other hand, using tf or tf*idf weighting often provides better results compared to binary weighting.

Among the non-binary intended metrics, the most successful ones are often LDA and Bray-Curtis which work better on tf-weighted data compared to the tf*idf weighted one. The best result obtained so far is an f-score of 0.8118 produced by LDA on tf-weighted bow features.

4 Conclusion

In this study, we investigated the effect of the distance metrics on the performance of the KNN classifier employed for automatic TC task. For this purpose, we performed KNN classifi-

cation experiments involving different distance metrics (of which three are distance metric learning algorithms), feature sets, and weighting schemes. Our experimental results show that Bray-Curtis and LDA metrics are often providing better results than other ones including the standard Euclidean distance.

The success behind the LDA is due to the fact that it learns a new distance space and uses it to measure the similarity between two instances. Bray-Curtis is, on the other hand, often used in ecology and biology fields to measure dissimilarity between two samples. For TC, it tries to measure the distance between two vectorized texts and according to our results often produces better results. We believe that the reason behind this success is due to the fact that the Bray-Curtis is actually a normalization method that achieves this by dividing absolute difference into summation of two vectors. This enables it to capture the similarity between two instances in a better way.

It is possible to obtain better results using bow features on the R8 dataset, however, the performances of the distance metrics are not affected by the weighted scheme, and the most robust one seems to be LDA. Bray-Curtis algorithm only falls behind the other metrics only for tf*idf weighted bow features. This is possibly because the number of feature dimensions is very low (40, see Table 2) for this case.

Considering our findings, we conclude that ML practitioners can prefer Bray-Curtis or LDA distance metrics for their KNN classification of texts. However, LDA has higher computational complexity compared to Bray-Curtis since distance metric learning algorithms have an inner computing phase for learning new metric space. Hence, it seems that LDA or any other DML algorithms can be used on small-sized text datasets, while Bray-Curtis can also be employed on large-scale text datasets. In future work, we will try to include other DML algorithms in our evaluation and provide their challenges on text data.

Ethics in Publishing

There are no ethical issues regarding the publication of this study.

Author Contributions

All phases of this study including design, coding, evaluation, and writing are performed by the corresponding author.

References

Abu Alfeilat, H. A., Hassanat, A. B. A., Lasassmeh, O., Tarawneh, A. S., Alhasanat, M. B., Eyal Salman, H. S., & Prasath, V. B. S. (2019). Effects of distance measure choice on K–Nearest Neighbor Classifier performance: A review. *Big Data, 7* (4), 221– 248. https://doi.org/10.1089/big.2018.0175

Bird, S. (2006). NLTK: The natural language toolkit. *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, 69– 72. https://doi.org/10.3115/1225403.1225421

Bishnoi, S., & Hisar, H. (2022). *k– Nearest Neighbor (k–NN) algorithm for classification. In Advances in Mathematical and Statistical Science*. 109.

Chen, S. (2018). K–Nearest Neighbor algorithm optimization in text categorization. *IOP Conference Series: Earth and Environmental Science, 108* (5), 052074. https://doi.org/10.1088/1755–1315/108/5/052074

Chen, Z., Zhou, L. J., Li, X. D., Zhang, J. N., & Huo, W. J. (2020). The Lao text classification method based on KNN. *Procedia Computer Science, 166*, 523– 528. https://doi.org/10.1016/j.procs.2020.02.053

Cheng, S., Shi, Y., & Qin, Q. (2012). Particle swarm optimization based semi–supervised learning on Chinese

text categorization. *2012 IEEE Congress on Evolutionary Computation*, 1– 8. https://doi.org/10.1109/CEC.2012.6252959

Çoban, Ö. (2016). *Metin sInIflandIrma teknikleri ile türkÇe twitter duygu analizi*. Atatürk üniversitesi.

Çoban, Ö., Özyer, B. Ö., & Tümüklü, G. (2015). A comparison of similarity metrics for sentiment analysis on Turkish Twitter feeds. 2015 *IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, 333– 338. https://doi.org/10.1109/SmartCity.2015.93

Deng, X., Li, Y., Weng, J., & Zhang, J. (2019). Feature selection for text classification: A review. *Multimedia Tools and Applications, 78* (3), 3797– 3816. https://doi.org/10.1007/s11042–018–6083–5

Dhar, A., Dash, N. S., & Roy, K. (2019). A study of Distance Metrics in document classification. In *Document Processing Using Machine Learning* (pp. 69 – 84). Chapman and Hall/CRC. https://doi.org/10.1201/9780429277573–6

Dhar, A., Mukherjee, H., Dash, N. S., & Roy, K. (2021). Text categorization: Past and present. *Artificial Intelligence Review, 54* (4), 3007– 3054. https://doi.org/10.1007/s10462–020–09919–1

Eminagaoglu, M. (2022). A new similarity measure for vector space models in text classification and information retrieval. *Journal of Information Science, 48* (4), 463– 476. https://doi.org/10.1177/0165551520968055

Ghawi, R., & Pfeffer, J. (2019). Efficient Hyperparameter Tuning with Grid Search for Text Categorization using kNN Approach with BM25 Similarity. *Open Computer Science, 9* (1), 160 – 180. https://doi.org/10.1515/comp–2019–0011

Jain, S., Jain, S. C., & Vishwakarma, S. (2020). A Proposed Similarity Measure for Text–Classification. *International Journal of Innovative Technology and Exploring Engineering, 9* (6), 2232 – 2235. https://doi.org/10.35940/ijitee.D1939.049620

Lade, S., & Dhore, M. L. (2021). Text categorization of Marathi news articles using machine learning. In V. H. Patil, N. Dey, P. N. Mahalle, M. Shafi Pathan, & Vinod. V. Kimbahune (Eds.), *Proceeding of First Doctoral Symposium on Natural Computing Research* (pp. 63– 72). Springer. https://doi.org/10.1007/978–981–33–4073–2_7

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., & Cournapeau, D. (2011). Scikit –learn: Machine Learning in Python. *MACHINE LEARNING IN PYTHON*.

Porter, M., F. (2001). *Snowball: A language for stemming algorithms*. http://snowball.tartarus.org/texts/introduction.html

Reuters–21578. (2022). *Document Collection*.

Suarez, J. L., Garcla, S., & Herrera, F. (2020). PyDML: A Python Library for Distance Metric Learning. *Journal of Machine Learning Research, 21* (96), 1–7.

Yang, L., & Jin, R. (2006). *Distance Metric Learning: A comprehensive survey*. https://www.cs.cmu.edu/~liuy/frame_survey_v2.pdf

Zhai, Y., Song, W., Liu, X., Liu, L., & Zhao, X. (2018). A Chi–square statistics based feature selection method in text classification. *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, 160– 163. https://doi.org/10.1109/ICSESS.2018.8663882

Zhou, Y., Liu, Y., Wang, Y., & Zheng, H. (2021). *Project 2: KNN Classification with Different Distance Metrics.*