International Journal of Computational Intelligence Systems (2023) 16:3
https://doi.org/10.1007/s44196-022-00164-8

RESEARCH ARTICLE

# WordNet Semantic Relations Based Enhancement of KNN Model for Implicit Aspect Identification in Sentiment Analysis

Halima Benarafa[1] · Mohammed Benkhalifa[1] · Moulay Akhloufi[2]

Received: 1 December 2021 / Accepted: 13 November 2022
© The Author(s) 2023

**Abstract**
Opinion mining or sentiment analysis (SA) is a key component of real-world applications for e-commerce organizations, manufacturers, and customers. SA deals with the computational evaluation of people's views, thoughts, and feelings in the text, whether they are visible or concealed. The Aspect based SA level is becoming one of the most active phases in this area. In this paper, we propose an approach to enrich K-Nearest Neighbors (KNN) to deal with Implicit Aspect Identification task (IAI). Through the use of WordNet semantic relations, we propose an enhancement for KNN distance computation to support the IAI task. For a conclusive empirical evaluation, experiments are conducted on two datasets of electronic products and restaurant reviews and the effects of our approach are examined and analyzed according to three criteria: KNN distance used to compute the similarity, the number of nearest neighbors (K) and the KNN behavior towards Overfitting and Underfitting. The experimental results show that our approach helps KNN improve its performance and better deal with Overfitting and Underfitting for Implicit Aspect Identification.

**Keywords** Implicit aspect-based sentiment analysis · Machine learning · Supervised approaches · WordNet · K-nearest neighbors · Lesk algorithm

**Abbreviations**

| | |
|---|---|
| ABSA | Aspect based sentiment analysis |
| ACD | Aspect category detection |
| ATE | Aspect term extraction |
| IAI | Implicit aspect identification |
| IAT | Implicit aspect term |
| IR | Improvement rate |
| JMTS | Joint multi-grain topic sentiment |
| KNN | K-nearest neighbors |
| LDA | Latent Dirichlet allocation |
| LSTM | Long short term memory |
| PMI | Pointwise mutual information |
| POS | Part of speech |
| RNN | Recurrent neural network |
| SA | Sentiment analysis |
| WN | WordNet |
| WSD | Word sense disambiguation |

✉ Halima Benarafa
halima_benarafa@um5.ac.ma

Mohammed Benkhalifa
m.benkhalifa@um5r.ac.ma

Moulay Akhloufi
moulay.akhloufi@umoncton.ca

[1] Algorithms, Networks, Intelligent Systems and Software Engineering (ANISSE), Department of Computer Science, Faculty of Sciences, Mohammed V University in Rabat, Rabat, Morocco

[2] Perception, Robotics, and Intelligent Machines Research Group (PRIME), Department of Computer Science, Moncton University, Moncton, NB, Canada

## 1 Introduction

SA studies have been done at three levels of granularity: document, sentence, and aspect level. Aspect-based sentiment analysis is concerned with collecting opinions about each aspect of existing entities in the text.

The most important task in aspect-level SA is aspect identification, that is why most researchers are interested in it. There are two types of aspects: implicit and explicit. Extracting explicit aspects and implicit aspects are both part of the aspect identification phase. There has been a great deal of effort put towards extracting explicit aspects, although implicit aspects have received little attention. Specific terms that appear expressly in the document are referred to as explicit aspects, they can be represented by noun or noun phrase. An implicit aspect, on the other hand, is not clearly stated in the document. It can appear in the form of verb, adjective, and adverb as they are used in [4–6]. Implicit aspects are essential as they can reflect the opinions implied in the text and contribute in the improvement of Opinion Mining Task.

In this study, we suggest a method for enriching KNN algorithm model by combining its basic distance with similarity function inspired from the Lesk algorithm [7] when applied to Word Sense Disambiguation (WSD) first introduced by [8]. As defined in [9–11], WSD is a process that automatically assigns a meaning to ambiguous words in context. The original Lesk algorithm defines the correct meaning of a word in given context as a sense with the maximum overlap between word dictionary definition and the given context.

In this article, we use the underlying idea of Lesk Algorithm for word sense disambiguation, however, our work is different and its originality is established at two levels: (i) The idea rationale: We inspire from Lesk algorithm to create a similarity function between words using WordNet dictionary (WN), created in [12], and employ this function to create a novel KNN distance that assigns higher weights to semantically similar words in nearest neighbors calculation. Based on this reasoning, we look to theoretically promote KNN search for nearest neighbors and therefore to empirically improve KNN performance. (ii) Model formulation: our similarity function, which will be formulated in details later, amplifies original score between words by squaring it then adding.

1. On the one hand, this new formulation guarantees much higher similarity scores for semantically similar words and consequently much smaller distances between them (since similarity and distances are defined to be inversely correlated), and on the other hand it keeps the same basic KNN distance for dissimilar words.

Several experiments are prepared according to protocols that are defined to serve our investigation objectives. The main outcomes of our research can be summarized as follows: (i) our approach aids KNN improve its performance for all considered distances without changing KNN performance evolution with respect to K, (ii) our technique supports KNN to better deal with Overfitting and Underfitting by reducing their impacts on KNN and therefore improving its performance and (iii) for both models (BasicKNN and KNN based on our method), sensitivity to Underfitting and Overfitting is inversely proportional to training data size.

The following is a breakdown of paper's structure. The second section discusses related works on Implicit Aspect based SA. Our technique is outlined in the third section. The experimental setting is presented in Sect. 4, proceeded by the findings and discussion section. Finally, the last section contains the conclusion.

## 2 Related Works

Implicit Aspect Identification task includes two subtasks, Aspect Term Extraction (ATE) and Aspect Category Detection (ACD). Earlier methods for ATE and ACD, like the one proposed by Hu et al. [13], rely on the frequency of nouns and noun phrases in the documents, with the hypothesis that aspect terms are more likely to be repeated. As cited in [14], the dependence on the frequency of particular word categories, nouns and noun phrases, is a limitation of this strategy, which may perform efficiently if the terms are frequent but may fail if terms are rare. In [15], Popescu and Etzioni enhance the precision of their approach by calculating Pointwise Mutual Information (PMI) for each aspect and excluding those aspects that do not match the supplied PMI score.

In recent years, dependency parser based methods have been actively used, like in [16, 17]. They use opinion-target relations to accomplish ATE and ACD tasks. Authors in [16] proposed a novel rule-based technique for detecting both explicit and implicit aspects that uses common-sense knowledge and phrase dependency trees. In [17], authors study rule-based linguistic patterns. They assume that detecting sentiment is easier than detecting aspect words. They proposed a group of opinion rules to detect sentiment words as a first step. In the second step, they employ grammatical dependencies to construct sentences' grammatical structure and detect aspect words. The final step entails adding infrequent words and removing irrelevant aspects. The most important factor in this method is the aspect-sentiment word lexical relation, which is able to identify low-frequency aspects.

Two of the most known co-occurrence based methods are introduced in [18, 19]. In [18], the authors predict implicit aspects depending on the co-occurrence frequency between the explicit aspects and the opinion words. Potential implicit aspects are based on a threshold calculated value. In [19], training data are enriched by the use of semantic relations from WordNet and the co-occurrence score is calculated for each extracted implicit aspect and its WordNet synsets.

The techniques that use WordNet or any other dictionary semantic relations are referred to as Dictionary-based methods. Fei et al. [6] is an earlier work that proposes a dictionary-based technique to identify aspects implied by adjectives. In [20] authors use synonym and definition relations extracted from WN to perform Implicit Aspect Identification task for adjectives and verbs. In [21], authors provide a novel hybrid model for Implicit Aspect Identification that combines semantic relations and frequency-based approach with supervised classifiers.

To perform ACD, topic modeling, which is an unsupervised machine learning technique, has been frequently used. Latent Dirichlet Allocation (LDA), that is very popular topic modeling algorithm, is used in [22–24]. In [22], authors propose a system called W2VLDA that is practically unsupervised. It combines LDA with continuous word embeddings and a Maximum Entropy classifier to perform Aspect Category Detection and sentiment classification. To extract semantic aspects, Alam et al. [23]. propose a domain-independent topic sentiment model named Joint Multi-grain Topic Sentiment (JMTS). JMTS efficiently extracts high-quality semantic aspects automatically, removing the need for manual probing. In [24], authors propose a novel LDA method to cluster aspect terms according to their aspect category. It employs semantic similarity between two words to improve the clustering process. In [25], authors propose an unsupervised model for aspect extraction and sentiment classification using LDA combined with linguistic rules. They ranked aspects based on their probability distribution values, and then clustered them into predetermined categories using domain knowledge with frequent terms.

In recent years deep learning techniques have started to be used for sentiment analysis after finding considerable success in a variety of application domains. Soni and Rambola [26] is an earlier work that proposes a hybrid approach that uses a Recurrent Neural Network (RNN) with similarity function from spaCy and similarity measures based on WordNet to detect implicit aspects. In [27], authors propose a topic-level model for sentiment analysis based on deep learning. They used a topic-level attention mechanism applied to a Long Short-Term Memory (LSTM) network to perform Aspect Category Detection and sentiment classification. In order to enhance the ATE task, authors in [28] suggest a two-step mixed unsupervised model that combines language patterns with deep learning methods. In the first step they use rule-based method to extract aspects, and then use fine-tuned word embedding to prune relevant aspects. In the second phase the attention-based deep learning model is trained using the first step's extracted aspects as labeled data.

Our study varies from all preceding works as it proposes WordNet semantic relations based enhancement of KNN classifier model to better deal with Implicit Aspect Identification task.

## 3 The Proposed Approach

To classify a test sample, KNN calculates distances between this test sample and each document in the training set. After that, the test sample is assigned to the most common class in its nearest neighbors. The most commonly used distance for KNN is Euclidean distance. It measures the real straight line between points in Euclidean space.

In this section, we present our approach, as detailed in Fig. 1, whose motivation is integrating external and relevant knowledge (namely semantic information extracted from WN lexical database) within KNN distance calculation. For this purpose, we introduce a novel distance function to KNN. Let $T_i$ and $T_j$ be two implicit aspect terms (IAT) in our data, and $Def_i$ and $Def_j$ be the respective sets of their definitions extracted from WordNet. If $Def_i$ contains n definitions and $Def_j$ contains m definitions then $Def_i$ and

$Def_j$ are defined as follows:

$$\mathrm{Def}_i = \{\mathrm{subset}_i 1, \ldots, \mathrm{subset}_{is}\}, s \in [1, n], \tag{1}$$

$$\mathrm{Def}_j = \{\mathrm{subset}_j 1, \ \ldots, \ \mathrm{subset}_{jt}\}, \ t \in [1, \ m], \tag{2}$$

where $\mathrm{subset}_{is}$ is the subset of words representing the $s$th definition of $\mathrm{Def}_i$, and $\mathrm{subset}_{jt}$ is the subset of words representing the $t$th definition of $\mathrm{Def}_j$. The new distance is computed according to the following formulas:

$$\mathrm{Score}\left(\mathrm{Def}_i, \mathrm{Def}_j\right) = \mathrm{maxNCW}_{ij}(s, t), s \in [1, n], t \in [1, m], \tag{3}$$

$$\mathrm{Similarity}\left(T_i, T_j\right) = \mathrm{score}^2\left(\mathrm{Def}_i, \mathrm{Def}_j\right) + 1, \tag{4}$$

$$\mathrm{DistNew}\left(T_i, T_j\right) = \mathrm{StandardDist}\left(T_i, T_j\right) / \mathrm{Similarity}(T_i, T_j). \tag{5}$$

The score is calculated trough the comparison of words definitions extracted from WordNet lexical database, since similar word senses are frequently defined by the same terms. We can make the following assumption: if the definitions of two terms contain similar words, they are similar. This score is inspired from the Lesk algorithm [7] which proposes using the number of common terms in the glosses of two concepts to compare them. First, the number of common words between each subset of $\mathrm{Def}_i$ and each subset of $\mathrm{Def}_j$ is computed.

Let's note this number as follows:

$\mathrm{NCW}_{ij}(s, \ t)$ = the number of common words between $\mathrm{subset}_{is}$ in $\mathrm{Def}_i$ and $\mathrm{subset}_{jt}$ in $\mathrm{Def}_j$.

As shown in Eq. (3), the score is then computed as the maximum of all these numbers $\mathrm{NCW}_{ij}(s, t)$ for $s \in [1, n]$ and $t \in [1, m]$.

Equation (4) demonstrates how Similarity($T_i$, $T_j$) is obtained. This latter is computed by adding 1 to the square of Score($\mathrm{Def}_i$, $\mathrm{Def}_j$). If $T_i$ and $T_j$ are dissimilar (Score($\mathrm{Def}_i$, $\mathrm{Def}_j$) = 0) then the new distance between them is set to the standard KNN distance since Similarity($T_i$, $T_j$) is equal to 1. The score is squared to provide higher similarity of terms having larger number of common words between subsets of their definitions and therefore smaller KNN distance between them.

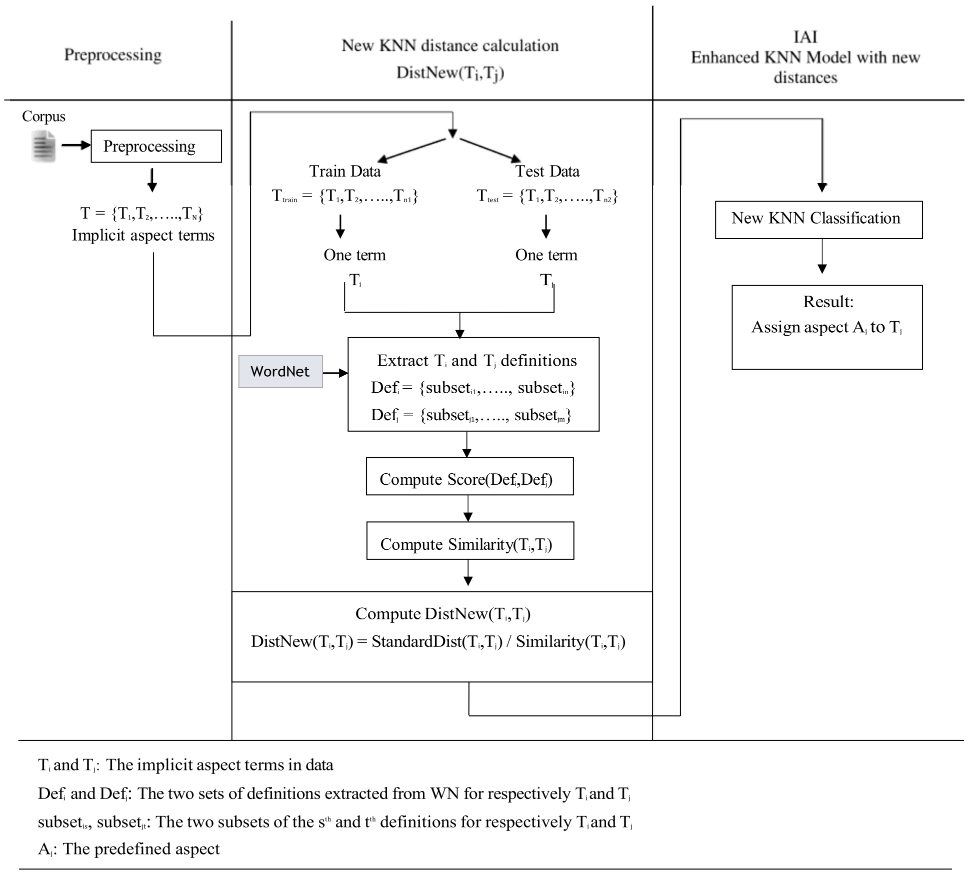
Preprocessing
New KNN distance calculation
DistNew(Ti,Tj)
IAI
Enhanced KNN Model with new distances
Corpus
Preprocessing
T = {T1,T2,…..,TN}
Implicit aspect terms
Train Data
Ttrain = {T1,T2,…..,Tn1}
Test Data
Ttest = {T1,T2,…..,Tn2}
One term Ti
One term Tj
WordNet
Extract Ti and Tj definitions
Defi = {subseti1,….., subsetin}
Defj = {subsetj1,….., subsetjm}
Compute Score(Defi,Defj)
Compute Similarity(Ti,Tj)
Compute DistNew(Ti,Tj)
DistNew(Ti,Tj) = StandardDist(Ti,Tj) / Similarity(Ti,Tj)
New KNN Classification
Result: Assign aspect Ai to Ti
Ti and Tj: The implicit aspect terms in data
Defi and Defj: The two sets of definitions extracted from WN for respectively Ti and Tj
subsetis, subsetjt: The two subsets of the sth and tth definitions for respectively Ti and Tj
Ai: The predefined aspect

**Fig. 1** Summary of our approach

In Eq. (5), the new KNN distance (DistNew($T_i$, $T_j$)), is calculated by dividing the standard distance (StandardDist($T_i$, $T_j$)) by the proposed similarity in Eq. (4).

## 4 Experiments and Results

The details of the experiments conducted to evaluate our approach are presented in this section. The experimental design is explained, including the pre-processing techniques employed, the classifier utilized, the datasets selected, the performance evaluation measures employed and the experimental protocols adopted.

### 4.1 Experimental Setup and Protocols

#### 4.1.1 Pre-processing

The first step in pre-processing is parsing the corpus to come up with a list of verbs and adjectives. All corpus terms are labeled using the Part of Speech Tagger (POS), and a list of extracted adjectives and verbs is created. The final list is made by removing all stop words from the first one.

#### 4.1.2 Classifier Used

One of the most widely used distance-based algorithms is the K-Nearest Neighbors classifier (KNN). KNN is a lazy supervised machine learning algorithm which assumes that elements in close proximity are similar. It is the simplest model since it non parametric because it makes no assumption on data distribution. It is lazy learning since the generalization of training data is delayed till a test data is submitted to the system. It does no training on training data, because there is no model to build. In particular, it just places the labeled data in some metric space. This algorithm classifies data by measuring the distance between the test sample and each element of the training set. After that, the test sample is assigned to the most common class in its nearest neighbors. In the case of $K = 1$, the test sample is assigned to its nearest neighbor's class.

#### 4.1.3 Datasets

To evaluate our method, we used Products and Restaurant datasets. Cruz-Garcia et al. [1]. created the Products dataset in which each IAT was manually labeled. This corpus is based on the Customer Review Datasets corpus described in [13]. The corpus includes reviews for five different electronic products: the “Apex AD2600 Progressive-scan DVD player”, the “Canon G3 digital camera”, the “Creative Labs Nomad Jukebox Zen Xtra 40 GB”, the “digital camera Nikon Coolpix 4300”, and the “Nokia6610 Phone”. Functionality, performance, appearance, price, quality, weight, and size, are the primary implicit aspects addressed.

The second corpus is Restaurant dataset. This dataset is distributed for aspect based sentiment analysis (ABSA), Task 4 of SemEval-2014 [3]. It consists of 3044 English sentences from the restaurant reviews of Ganu et al. [2] with five preset implicit aspects, ambiance, food, price, service, and anecdotes/miscellaneous. This latter is not taken into account in our research because it does not properly describe an implicit aspect.

#### 4.1.4 Evaluation Measures

The most commonly utilized evaluation measures for evaluating the performance of the sentiment analysis model are accuracy, precision, recall, and F1-score. The percentage of accurately predicted samples is known as accuracy. However, if the dataset is imbalanced, accuracy is not enough, thus precision, recall, and the F1-score are used. The F1-score is the equally weighted average of precision and recall [29].

#### 4.1.5 Experimental Protocols

We prepare our experiments so that our method can be empirically analyzed according to three KNN structural aspects:

- Distance metric used,
- Number of nearest neighbors (K),
- Overfitting and Underfitting.

All experiments are conducted using the tenfold and fivefold cross-validation to reduce the uncertainty of data split between training and test data and have two different training data sizes for testing. In the next four subsections, our experimental protocols will be described in details with a focus on the goal to be attained by each protocol and how this latter is prepared to meet its goal.

**4.1.5.1 The Distance Metric Used** KNN calculates the distance of every test data from all training data then finds the K-Nearest Neighbors of it. Hence, the core of KNN classifier depends mainly on distance or similarity between test data and training data and its performance depends significantly on this distance. For these reasons, we consider to empirically investigate the impact of our new similarity function on KNN performance when using different distances. In this work, we consider three different kinds of distances that are appropriately chosen:

*Euclidean distance* it is the most widely used distance that measures the real straight line between points (terms) in Euclidean space.

*Cosine distance* it calculates similarity between two term vectors to determine whether they are pointing to the same direction. It provides hidden information in the data that Euclidean distance does not capture. It is independent from the length of term vectors. The cosine distance is defined as the cosine similarity subtracted from 1.

*Jaccard distance* it is based on Jaccard index which determines the similarity between two finite sets based on their union and intersection and is defined as the size of the intersection divided by the size of the union of the sets. It neither captures the data set size nor considers the term frequency in the data sets. The Jaccard distance is then defined as the Jaccard index subtracted from 1.

**4.1.5.2 Number of Nearest Neighbors (K)** In general, very large values of $K$ lead to global class boundary smoothing. This latter, even if it reduces the overall noise, it induces Underfitting of KNN algorithm that becomes very generalized and performs badly on both training and test data. On the other hand, when K has very small values KNN is very specific (due to local smoothing), does not generalize and becomes very sensitive to noise. It performs well on training data and badly on test data. This refers to KNN Overfitting. $K$, which is related KNN model error, is then a core deciding factor for KNN algorithm and consequently has a powerful effect on its performance. Hence, our study considers an experimental investigation of the impact of our proposed similarity function-based distance on KNN performance for IAI task with respect to different values for $K$. In this work, we consider for $K$ different values appropriately chosen (very small and very large values) so that the impact of our new distance can be studied under the above mentioned KNN biasing conditions.

The choice of small and large values for $K$ depends on dataset used. For each dataset, we define these values for $K$ with respect to training data size. The following table (Table 1) provides detailed information on the two datasets used.

To deal with the issues (a) and (b), we conduct our experiments according to the following protocol:

**Table 1** Detailed information for the two datasets

| Dataset | Number of implicit aspect terms | Training data size for tenfold | Training data size for fivefold |
|---|---|---|---|
| Restaurant | 121 | 109 | 97 |
| Products | 552 | 497 | 442 |

#### 4.1.5.3 Protocol1: Experiments for Issues (a) and (b)

```
For each Distance (Dist) from {Euclidean, Cosine, Jaccard} :
  For each dataset (DS) from {Products, Restaurant} :
        Let K-Set = K-Set(DS)
        Execute BasicKNN and NewKNN for every K from K-Set
        Report results for F1 performances of both BasicKNN
        and NewKNN
        Where BasicKNN is KNN based on standard distance
        and NewKNN is KNN based on our proposed distance.
K-set(Restaurant)=[1,2,3,4,5,10,15,20,25,30,35,40,45,50,60,70,80,90,100]
K-set(Products)=[1,2,3,5,10,20,22,50,100,150,200,250,300,350,400,450,
                 470,480,490]
```

**4.1.5.4 Overfitting/Underfitting** To experimentally address issue (c), we prepare a protocol that aims at studying the impact of our similarity function-based distance on KNN performance for IAI task under Overfitting/Underfitting conditions and using three different distances. To achieve this goal, our protocol (protocol 2) shown below should:

1. Be based on conditions leading to KNN Overfitting and Underfitting. In general, Overfitting and Underfitting are caused by respectively small value(s) of K and large value(s) of K and they can be experimentally identified by cross-validation. In this protocol, we conduct our experiments using both fivefold and tenfold cross-validations in order to test two different training data sizes.
2. Offer a measure for the effect of Overfitting and Underfitting on KNN performance. Therefore, comparison can be made between BasicKNN and KNN based on our approach with respect to their behavior towards Overfitting and Underfitting. In Overfitting, KNN performs well on training data and badly on test data. Whereas, in Underfitting, KNN performs badly on both training and test data. We define four indicators for measuring sensitivities of both models to Overfitting and Underfitting. These indicators are introduced and explained in details in Sect. 4.2.2 of part ”Results and Discussion”.

**4.1.5.5 Protocol 2: Experiments for Issue (c)** Let $K$ be the number of nearest neighbors of KNN and $k$ is $k$-fold parameter for cross-validation. K≠k, where $K$ gets 1 if Overfitting and $K$ gets values from the set $K$-Large Values of a dataset, and $k$ gets values from the set $\{5, 10\}$.

```
For each dataset (DS) from {Products, Restaurant} :
  For each Distance (Dist) from {Euclidean, Cosine, Jaccard} :
    For k= {5,10} //for k-fold// :
          IF Overfitting :
                {Let K=1
                Execute BasicKNN and NewKNN for each fold-i
                for i=1 to k}
          ELSE //Underfitting// :
        {Execute BasicKNN and NewKNN for each K
        from K Large Values(DS) and for each fold-i
        for i=1 to k}
Report results for F1-scores of both BasicKNN and NewKNN
```

Where Dist and DistNew are defined in protocol 1. $K$ Large Values(DS) is the set of large values that $K$ takes. They depend on dataset and distance used.

### 4.2 Results and Discussion

The findings of the experiments are represented and analyzed in this section according to the following points.

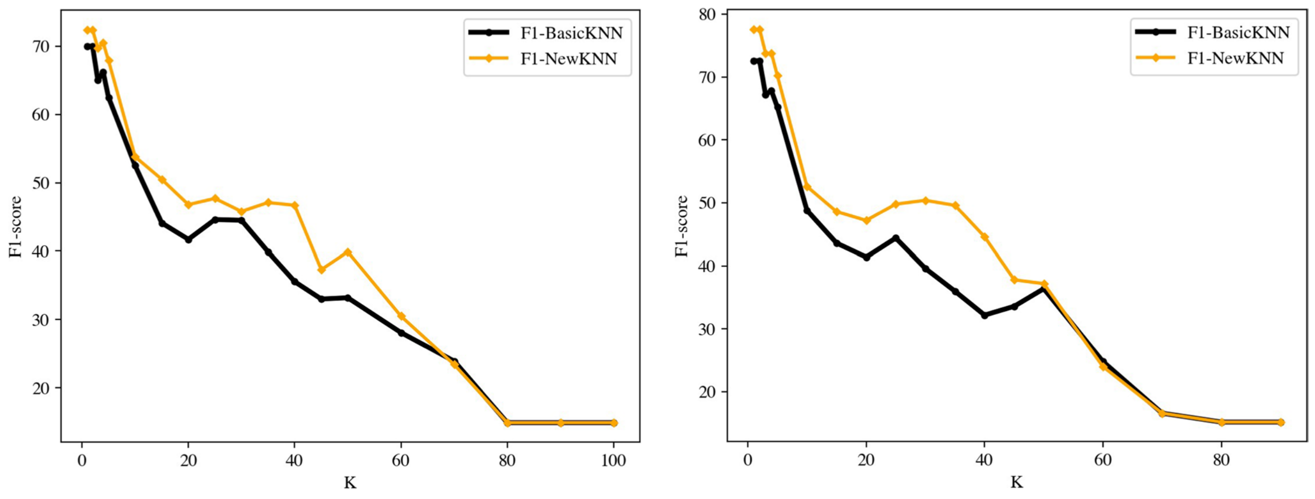

**Fig. 2** F1-score evolution for KNN using Euclidean distance with tenfold and fivefold cross-validation on Restaurant

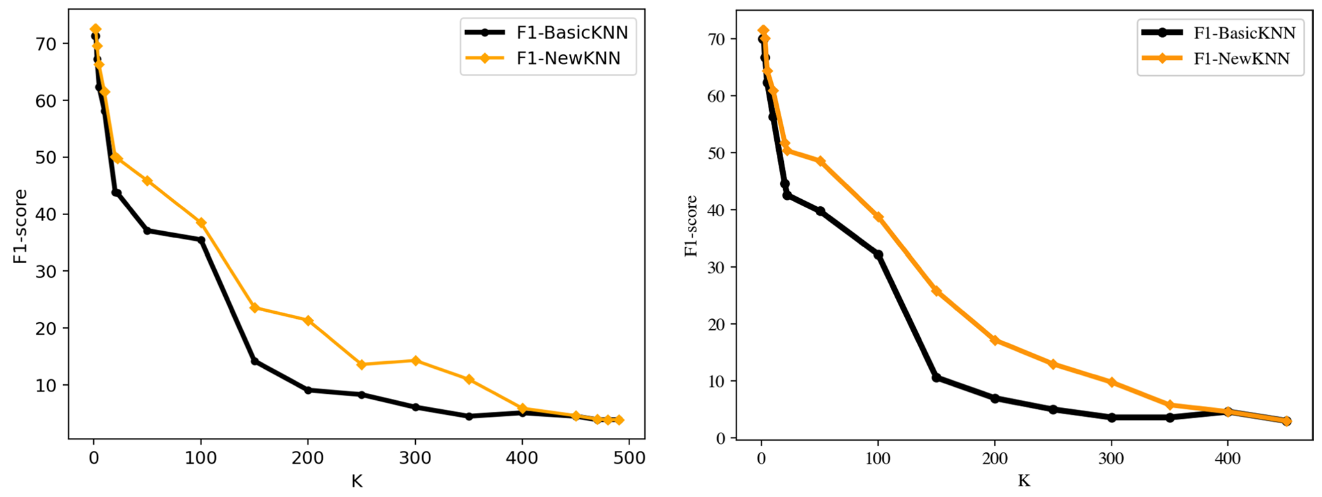

**Fig. 3** F1-score evolution for KNN using Euclidean distance with tenfold and fivefold cross-validation on Products

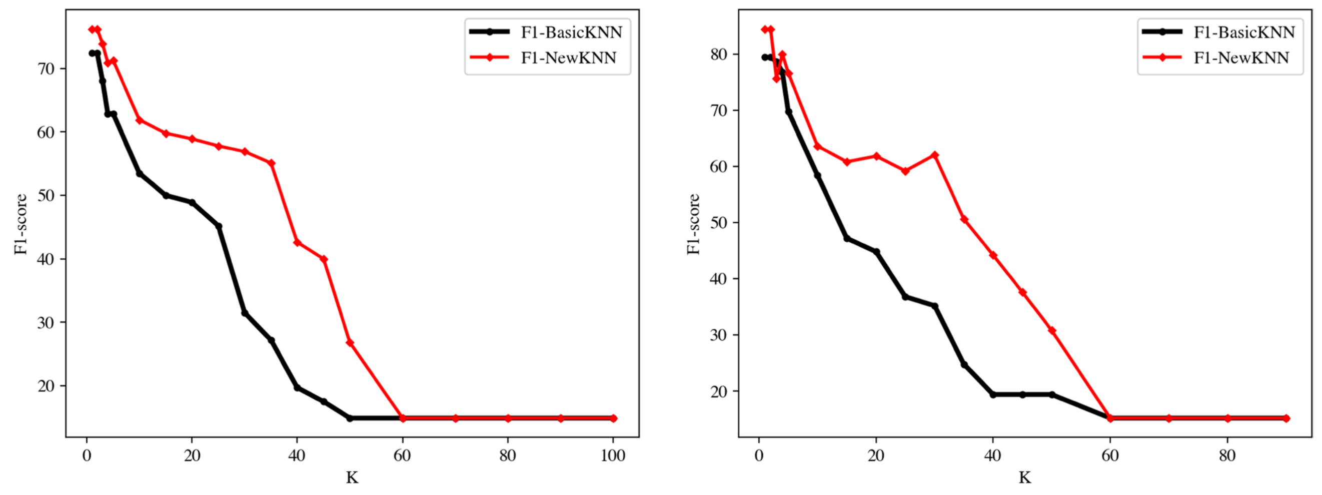

**Fig. 4** F1-score evolution for KNN using Jaccard distance with tenfold and fivefold cross-validation on Restaurant

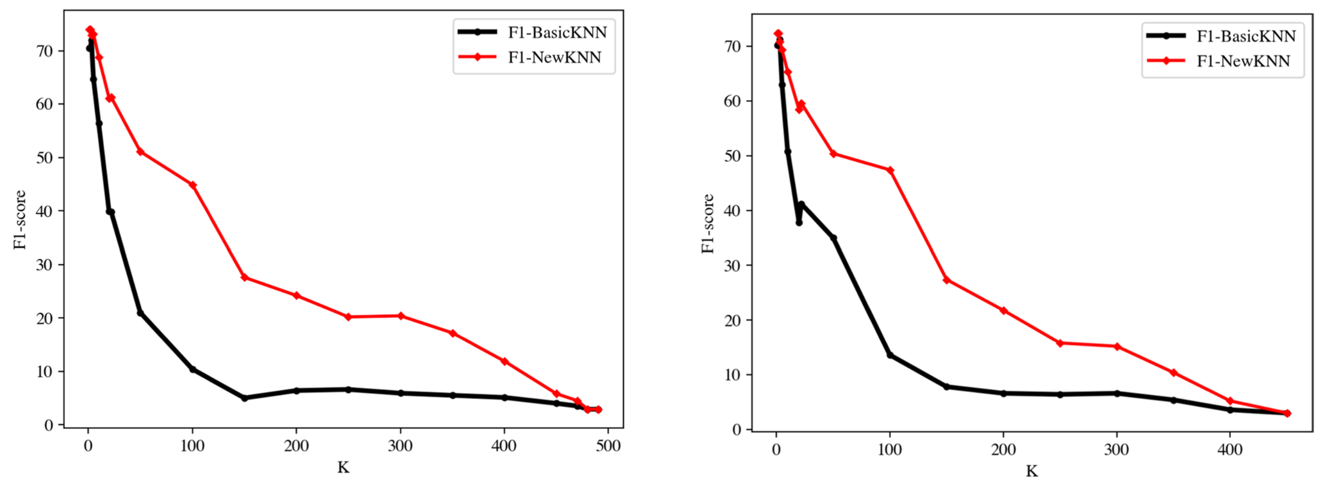

**Fig. 5** F1-score evolution for KNN using Jaccard distance with tenfold and fivefold cross-validation on Products

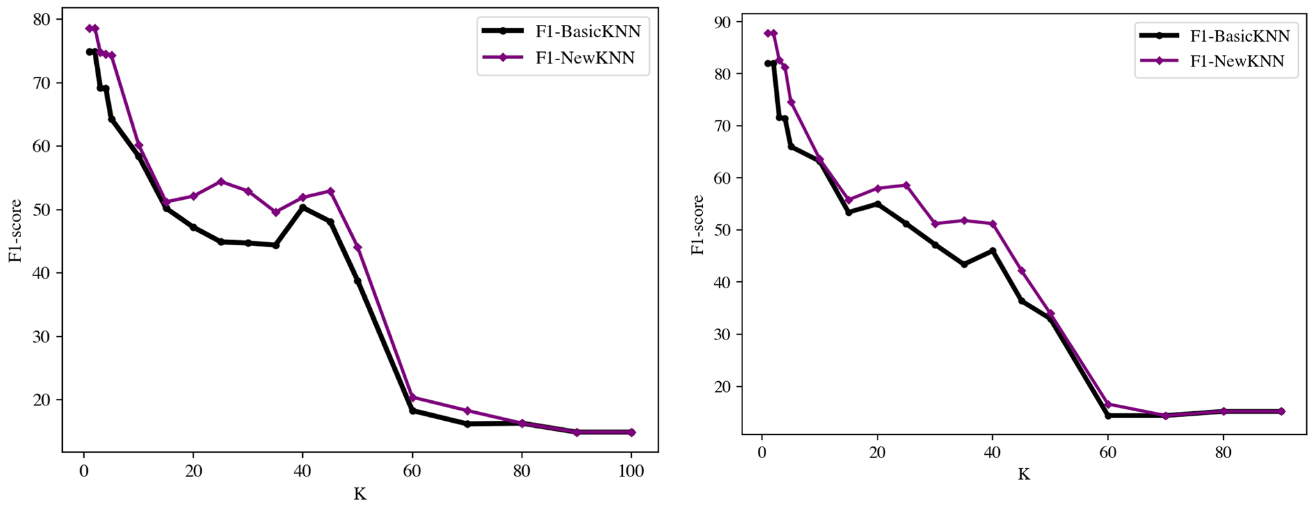

**Fig. 6** F1-score evolution for KNN using Cosine distance with tenfold and fivefold cross-validation on Restaurant

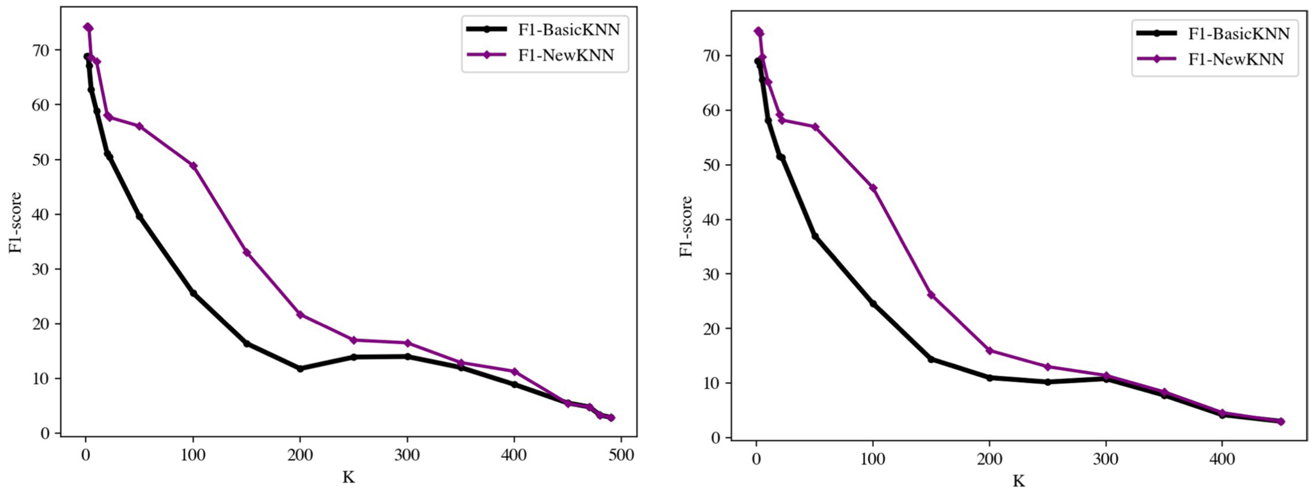

**Fig. 7** F1-score evolution for KNN using Cosine distance with tenfold and fivefold cross-validation on Products

### 4.2.1 The Distance and the Number of Neighbors Used

The figures presented in this section show the performance results for protocol 1. Figures 2, 3, 4, 5, 6, and 7 show the F1-score evolution for BasicKNN and NewKNN according to the distance used and to the value of *K*.

As shown in the six figures below, the novel distance function has enhanced KNN's performances for both datasets and the three utilized distances, without changing the performance behavior of the NewKNN compared to BasicKNN. For each dataset, the highest improvements achieved by NewKNN are observed within the same *K* value range for all used distances and for both fivefold and tenfold cross-validations. (Restaurant: *K* range ~ [20,45]; Products: *K* range ~ [50,350]). However, NewKNN has a relatively smaller superiority when *K* has small or large values and especially when it gets closer to the training data size.

These improvements are achieved because the terms that are semantically similar, tend to belong to the same aspect category.

The new distance used for KNN, reorganizes and optimizes the distribution of the nearest neighbors, by prioritizing semantically similar words for the identification of a term nearest neighbors, and then improves the classification's performance. However, when K gets larger values both NewKNN and BasicKNN suffer more from Underfitting and therefore they perform poorly.

Table 2 exposes the averages of F1-score's improvement rates (IR) of NewKNN over BasicKNN for fivefold and tenfold cross-validations and for all considered K values.

**Table 2** Averages of improvement rates of NewKNN over BasicKNN on both datasets for three distances and with fivefold and tenfold cross-validation

| Cross-validation | Restaurant | | | | | | Products | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1-score averages | | | | | | F1-score averages | | | | | |
| | Euclidian | | Jaccard | | Cosine | | Euclidian | | Jaccard | | Cosine | |
| | Tenfold | Fivefold | Tenfold | Fivefold | Tenfold | Fivefold | Tenfold | Fivefold | Tenfold | Fivefold | Tenfold | Fivefold |
| F1-BasicKnn | 42.07 | 42.96 | 37.98 | 41.68 | 45.26 | 47.83 | 29.18 | 32.63 | 25.94 | 30.78 | 30.89 | 34.75 |
| F1-NewKnn | 45.63 | 47.88 | 47.52 | 51.8 | 49.21 | 52.32 | 33.32 | 37.98 | 37.83 | 41.56 | 37.31 | 41.31 |
| IR (%) | 8.46 | 11.43 | 25.12 | 24.28 | 8.73 | 9.39 | 14.69 | 16.40 | 45.84 | 35.02 | 20.78 | 18.88 |

From Table 2, it is noticed that, on the average and for both datasets, improvement rates are:

- Higher with fivefold than with tenfold cross-validation for Euclidean distance.
- Smaller with fivefold than with tenfold cross-validation for Jaccard distance.Almost equal for both tenfold and fivefold cross-validations for Cosine distance.

This shows that the improvement rate is not related to the training data size since it varies by changing the underlying KNN distance.

Table 2 gives a global view on NewKNN performance superiority. To give clearer idea about this superiority, we introduce Table 3 that defines three ranges for $K$ values corresponding to structural change of KNN performance. Thus, Table 3 shows the improvement rates with respect to the three defined ranges of $K$ values where $K$ is the core KNN parameter representing the most impacting performance factor. These three ranges of $K$ values are $R_{\text{small}}$, $R_{\text{middle}}$ and $R_{\text{large}}$ that respectively represent the small $K$ values ($K < 20$ for Restaurant; $K < 50$ for Products), the medium $K$ values ($K \in [20, 45]$ for Restaurant; $K \in [50, 350]$ for Products), and the large $K$ values ($K > 45$ for Restaurant; $K > 350$ for Products). The $K$ values of three ranges are extracted from conducted experiments.

We can notice from this table that the improvement is considerable in the middle area. However, for small and large areas of $K$ values the improvement is less important, because these two areas correspond respectively to Overfitting and Underfitting situations, where KNN performance has a special behavior. As it will be detailed in next section, in Underfitting, KNN performance is negatively impacted, whereas in Overfitting KNN achieves its best performance. Moreover, the KNN superiority is globally maintained, but it is particularly highly significant in the middle range of $K$ for: the three distances, both 5 and tenfold cross-validations, and both datasets, that are also major impacting factors of KNN performance (For Restaurant: IR are { Euclidian: (tenfold: 14.1%, fivefold: 23.71%); Jaccard: (tenfold: 79.39%, fivefold: 83.44%); Cosine: (tenfold: 12.46%, fivefold: 12.49%)}). For Products: IR are {Euclidian: (tenfold: 82.32%, fivefold: 103.62%); Jaccard: (tenfold: 267.10%, fivefold: 163.41%); Cosine: (tenfold: 52.30%, fivefold: 44.05%)}).

**Table 3** Improvement rate averages with respect to three ranges of $K$ values for both datasets and three distances with tenfold and fivefold cross-validations

| Distance | Restaurant | | | Products | | |
|---|---|---|---|---|---|---|
| | $R_{\text{small}}$ ($K < 20$) (%) | $R_{\text{middle}}$ ($K \in [20, 45]$) (%) | $R_{\text{large}}$ ($K > 45$) (%) | $R_{\text{small}}$ ($K < 50$) (%) | $R_{\text{middle}}$ ($K \in [50, 350]$) (%) | $R_{\text{large}}$ ($K > 350$) (%) |
| Euclidian (tenfold) | 6.60 | 14.1 | 4.44 | 6.68 | 82.32 | 4.1 |
| Euclidian (fivefold) | 8.48 | 23.71 | − 0.2 | 7.93 | 103.62 | 0 |
| Jaccard (tenfold) | 11.48 | 79.39 | 13.31 | 21.9 | 267.10 | 41.38 |
| Jaccard (fivefold) | 8.63 | 83.44 | 11.75 | 20.53 | 163.41 | 22.22 |
| Cosine (tenfold) | 6.62 | 12.46 | 6.35 | 11.20 | 52.30 | 5.03 |
| Cosine (fivefold) | 8.77 | 12.49 | 3.66 | 10.16 | 44.05 | 4.76 |

**Table 4** F1-score average performances of NewKNN and BasicKNN under Overfitting for both datasets and using three distances

| Cross-validation | Restaurant | | | | | | Products | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1-score averages | | | | | | F1-score averages | | | | | |
| | Euclidian | | Jaccard | | Cosine | | Euclidian | | Jaccard | | Cosine | |
| | Tenfold | Fivefold | Tenfold | Fivefold | Tenfold | Fivefold | Tenfold | Fivefold | Tenfold | Fivefold | Tenfold | Fivefold |
| F1-test(BasicKnn) | 70 | 72.6 | 72.4 | 79.4 | 74.9 | 82 | 71.4 | 70 | 70.5 | 70.2 | 68.9 | 69 |
| F1-train(BasicKnn) | 100 | 100 | 100 | 100 | 100 | 100 | 96.7 | 95.4 | 93.7 | 95.8 | 93.6 | 94.6 |
| F1-test(NewKnn) | 72.4 | 77.6 | 76.2 | 84.4 | 78.6 | 87.8 | 72.6 | 71.6 | 74 | 72.4 | 74.3 | 74.6 |
| F1-train(NewKnn) | 100 | 100 | 100 | 100 | 100 | 100 | 96.0 | 96.4 | 96.2 | 96 | 94.8 | 94.4 |
| Delta-test | 2.4 | 5 | 3.8 | 5 | 3.7 | 5.8 | 1.2 | 1.6 | 3.5 | 2.2 | 5.4 | 5.6 |
| Delta-train | 0 | 0 | 0 | 0 | 0 | 0 | − 0.7 | 1 | 2.5 | 0.2 | 1.2 | − 0.2 |
| Delta-BasicKnn | 30 | 27.4 | 27.6 | 20.6 | 25.1 | 18 | 25.3 | 25.4 | 23.2 | 25.6 | 24.7 | 25.6 |
| Delta-NewKnn | 27.59 | 22.4 | 23.8 | 15.6 | 21.4 | 12.2 | 23.40 | 24.8 | 22.2 | 23.6 | 20.5 | 19.8 |
| Delta (%) | 2.41 | 5 | 3.8 | 5 | 3.7 | 5.8 | 1.9 | 0.6 | 1 | 2 | 4.2 | 5.8 |

### 4.2.2 Overfitting and Underfitting

To analyze the behavior of the original and new models under Overfitting, we performed the classification using $K = 1,2$.

Table 4 exposes F1-score averages (where each average is computed over different folds) for BasicKNN model and the proposed model and on Products and Restaurant datasets in Overfitting situation when both models perform well on training data and poorly on test data.

To analyze the sensitivity of NewKNN and BasicKNN to Overfitting, we introduce four indicators in Table 4 that are used to measure how much resilient both models are to Overfitting.

Table 4 shows that even under Overfitting conditions, NewKNN outperforms BasicKNN for all experiments (Delta-test > 0 in all experiments in Table 4).This fact is the first indicator to less Overfitting sensitivity of NewKNN compared to BasicKNN, because the first model outperforms the second one on test data.

It is clear from this table that the difference between F1-score averages for training and testing data are smaller in the NewKNN model. Delta- BasicKNN (Delta-NewKNN) is the difference between F1-test and F1-train of BasicKNN (NewKNN). (i.e. Delta-BasicKNN = F1-train(BasicKNN)—F1-test(BasicKNN)).

Delta-BasicKNN and Delta-NewKNN are two other indicators of Overfitting sensitivity of respectively BasicKNN and NewKNN. These two indicators measure how much performance losses are made by both NewKNN and BasicKNN between training and test data. When Delta-BasicKNN (Delta-NewKNN) gets higher the BasicKNN (NewKNN) achieves poorer performance on test data than on training data. This means that BasicKNN (NewKNN) is more sensitive to Overfitting. Delta (which is equal to Delta-BasicKNN—Delta-NewKNN) indicates which model is more sensitive to Overfitting amongst BasicKNN and NewKNN. When Delta is positive BasicKNN is more sensitive otherwise it is the NewKNN that is more sensitive. Also, when Delta gets higher BasicKNN gets more sensitive than NewKNN.

Table 4 shows that for both datasets and all used distances, all values of Delta are positive. This means that in NewKNN the difference between F1-test and F1-train is less important than in BasicKNN, and this signifies less performance loss between training and test data and therefore less Overfitting sensitivity.

Even if it is less pertinent, a fourth indicator Delta-train is added (Delta- train = F1-train(NewKNN) – F1-train(BasicKNN)). Table 4 shows that the values of Delta-train are either null (for Restaurant dataset) or small positive (for 4 out of 6 cases for Products dataset). This indicates that NewKNN is slightly more performant on training data than BasicKNN for most experiments on Products, whereas it is as performant as BasicKNN on training data for Restaurant dataset. This means that NewKNN shows towards Overfitting relatively more tolerance (for Products) and the same resilience (for Restaurant) compared to BasicKNN. This indicator is less pertinent than the three others because both models basically perform well on training data within Overfitting.

Thus, our approach helps KNN dealing better with Overfitting. In other terms, the proposed model is less sensitive to Overfitting than the basic one for all the datasets and distances.

Like BasicKNN, we can also observe that NewKNN tolerance to Overfitting is higher with fivefold cross-validation than with tenfold. This means NewKNN tolerance to Overfitting is proportional to training data size. More training data helps reducing its sensitivity to Overfitting.

**Table 5** F1-score average performances of NewKNN and BasicKNN in Underfitting, for both datasets and using three distances with tenfold cross-validation

| Distance/Dataset | *K* | F1-test-basic | F1-train-basic | F1-test-new | F1-train-new | Delta-test | Delta-train | Delta-new | Delta-basic |
|---|---|---|---|---|---|---|---|---|---|
| Euclidian-rest | 40 | 35.55 | 39.85 | 46.7 | 51.9 | 11.15 | 12.05 | 5.2 | 4.3 |
| Euclidian-prod | 150 | 14.2 | 18.6 | 23.6 | 31 | 9.4 | 12.4 | 7.4 | 4.4 |
| Jaccard-rest | 40 | 19.7 | 19 | 42.6 | 56.5 | 22.9 | 37.5 | 13.9 | − 0.7 |
| Jaccard-prod | 50 | 21.0 | 29.3 | 51.1 | 62.8 | 30.1 | 33.5 | 11.3 | 8.3 |
| Cosine-rest | 60 | 18.3 | 28 | 20.4 | 33.8 | 2.1 | 5.8 | 13.4 | 9.7 |
| Cosine-prod | 100 | 25.6 | 35.2 | 48.9 | 57.9 | 23.3 | 22.7 | 9 | 9.6 |

**Table 6** F1-score average performances of NewKNN and BasicKNN in underfitting, for both datasets and using three distances with fivefold cross-validation

| Distance/Dataset | *K* | F1-test-basic | F1-train-basic | F1-test-new | F1-train-new | Delta-test | Delta-train | Delta-new | Delta-basic |
|---|---|---|---|---|---|---|---|---|---|
| Euclidian-rest | 40 | 33.6 | 37 | 37.8 | 43.4 | 4.2 | 6.4 | 5.6 | 3.4 |
| Euclidian-prod | 150 | 10.6 | 15.6 | 25.8 | 29.6 | 15 | 14 | 3.8 | 5 |
| Jaccard-rest | 35 | 24.8 | 29.6 | 50.6 | 55.2 | 25.8 | 25.6 | 4.6 | 4.8 |
| Jaccard-prod | 100 | 13.6 | 22.4 | 47.4 | 52 | 33.8 | 29.6 | 4.6 | 8.8 |
| Cosine-rest | 60 | 14.4 | 21.4 | 16.6 | 23.2 | 2.2 | 1.8 | 6.6 | 7 |
| Cosine-prod | 100 | 24.6 | 31 | 45.8 | 53 | 21.2 | 22 | 7 | 6.4 |

In order to analyze the behavior of BasicKNN and NewKNN during Underfitting, we extract the values of K leading to Underfitting from graphs shown in Fig. 2, 3, 4, 5, 6, and 7.

Tables 5 and 6 show the behavior of BasicKNN and NewKNN in Underfitting conditions when both models poorly perform on both training and testing data. To conduct a complete analysis of Underfitting sensitivity of both NewKNN and BasicKNN, we introduce in Tables 5 and 6 four indicators that are to be used all together in order to measure how tolerant both techniques are to Underfitting.

Tables 5 and 6 show that even within Underfitting conditions, NewKNN outperforms BasicKNN for all experiments (shown by positive values for delta-test in all experiments in both tables, where delta-test = F1-test(NewKNN) – F1-test(BasicKNN)). This fact is the first indicator to less Underfitting sensitivity of NewKNN compared to BasicKNN since the first model outperforms the second one on test data. The second indicator is delta-train (delta-train = F1-train(NewKNN) – F1-train(BasicKNN)). The two tables show that the values of delta-train are positive. This indicates that NewKNN is more per- formant on training data than BasicKNN. This means higher Underfitting tolerance of NewKNN compared to BasicKNN. The two other indicators to Underfitting sensitivity of respectively NewKNN and BasicKNN are delta-New and delta-Basic (delta-New = F1-train-New – F1-test-New and delta-Basic = F1-train-Basic – F1-test-Basic). These two indicators measure how much performance losses are made by both NewKNN and BasicKNN between training and test data. When delta-Basic (delta-New) gets higher the BasicKNN (NewKNN) achieves poorer performance on test data than on training data. This means that BasicKNN (NewKNN) is more sensitive to Underfitting.

Unlike Overfitting, the two tables show that the values of delta-New are smaller than their corresponding values of delta-Basic only for most of fivefold experiments and one tenfold experiment. Thus, NewKNN is less sensitive to Underfitting than BasicKNN in most of fivefold cross-validation experiments when the size of training data is bigger. Exceptionally, for all 10-fold cross-validation experiments (except one), NewKNN is more sensitive to Underfitting than BasicKNN.

Even though NewKNN does not show more tolerance to Underfitting than BasicKNN for tenfold cross-validation, it consistently keeps the same tolerance variation (like in Overfitting) with respect to number of folds in cross-validation. Indeed, the tolerance of NewKNN to Underfitting is higher with fivefold cross-validation than with tenfold cross-validation. This means that NewKNN tolerance to Underfitting is proportional to training data size. More training data helps reducing NewKNN sensitivity to Underfitting.

### 4.2.3 Comparison with Other Works

The suggested technique is compared to various methods from the literature in order to assess the proposed method's performance. Table 7 presents a comparison of traditional and deep learning approaches with our proposed method for

**Table 7** F1-score performances of selected traditional and deep learning approaches and our proposed methods for IAI and using Restaurant and Products datasets

| Method | Type | F1-score (Restaurant) (%) | F1-score (Products) (%) |
|---|---|---|---|
| W2VLDA [30] | Traditional | 72.00 | – |
| Schouten et al. Supervised [19] | Traditional | 83.80 | – |
| MNB + WN [20] | Traditional | 77.40 | 90.00 |
| BNB + WN [20] | Traditional | 78.40 | 93.30 |
| SVM + WN + frequency [21] | Traditional | 85.30 | 91.80 |
| KNN + WN + frequency [21] | Traditional | 85.30 | 91.80 |
| MNB + WN + frequency [21] | Traditional | 87.55 | 91.80 |
| LSTM + WN + frequency [21] | Deep learning | 85.20 | 89.09 |
| Att-LSTM + WN + frequency [21] | Deep learning | 87.83 | 94.36 |
| Proposed KNN with Cosine dist | Traditional | 87.80 | 74.60 |
| Proposed KNN with Jaccard dist | Traditional | 84.40 | 74.00 |
| Proposed KNN with Euclidian dist | Traditional | 77.60 | 72.60 |

Implicit Aspect Identification. It is very important to mention that all works utilize the same datasets however they do not operate at the same level. On one hand, W2VLDA[30], Schouten et al. Supervised method [19] and our proposed approaches (with 3 distances) operate on algorithmic level where they suggest adjustments or additions, on the other hand, the remaining techniques operate on training data level where they bring improvements to training data quality.

From Table 7, we observe that:

For Restaurant dataset, our proposed approach globally shows very competing performance level compared to other works. Four comparisons are worth mentioning: (i) Our proposed approach (especially, with Cosine and Jaccard distances) outperforms W2VLDA [30], Schouten et al. Supervised method [19], MNB + WN [20] and BNB + WN [20]. Even if when adjusting the core model that is very sensitive and challenging, our proposed approach succeeds to achieve better performances than the four other techniques. (ii) Our proposed approach with Cosine distance (best performing amongst three proposed methods) slightly outperforms KNN + WN + Frequency [21] even if this latter operates on data level which is less challenging. It is important to note that our proposed method uses Cosine distance whereas the other one uses Euclidian one. (iii) Our proposed approach with Euclidian distance is outperformed by KNN + WN + Frequency [21] that uses the same distance and operates on training data by enhancing its quality which allows to remedy to class unbalanced nature of Restaurant dataset. (iv) Our proposed approach with Cosine distance (best performing amongst three proposed methods) shows almost the same performance as Att-LSTM + WN + Frequency [21] which is not only a deep learning technique, which is generally reputed for higher performance on classification, but also operating on less challenging and less sensitive data level.

For Products dataset our three proposed approaches are largely outperformed by all other methods that apply to Products dataset. These latter are data level techniques, that enhance training data by adding semantic relations from WN, and this should reduce the high class unbalanced structure of Products dataset. Whereas our approaches operate on algorithm level without modifying the structure of training data.

## 5 Conclusion

In this work, we propose an approach to improve KNN algorithm to deal with Implicit Aspect Identification task. Through the use of WordNet semantic relations, we propose an enhancement for KNN distance computation to support the IAI task. For empirical evaluation, experiments are conducted on two datasets of electronic products and restaurant reviews, and the effects of our method are analyzed according to three different KNN aspects: (i) The KNN distance used to compute the similarity, (ii) The number of nearest neighbors (K) and (iii) The KNN behavior towards Overfitting and Underfitting. The main findings of our work can be summarized as follows:

1. Our approach helps KNN boost its performance for the three considered distances and for different values of K without changing KNN performance evolution with respect to K.
2. Our technique supports KNN to better deal with Overfitting and Underfitting by reducing their impacts on KNN and therefore improving its performances.
3. For both models (BasicKNN and KNN based on our method), sensitivity to Underfitting and Overfitting is inversely proportional to training data size.

In future work, we will consider investigating WordNet synonym semantic relation for KNN model enhancement for Implicit Aspect Identification task. We also plan to apply our technique to improve other machine learning classifiers for sentiment analysis, particularly deep learning classifiers.

**Author Contributions** All authors contributed to the study conception and design. Experimental protocol design are prepared by all authors. Experiments are conducted by BH. Results analysis and discussion were performed by all authors. The first draft of the manuscript was written by BH and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

**Funding** No funding was received for conducting this study.

**Availability of Data and Materials** The Restaurant Reviews dataset that supports the findings of this study is available in SemEval-2014 ABSA Restaurant Reviews, http://metashare.ilsp.gr:8080/repository/browse/semeval-2014-absa-restaurant-reviews-train-data/479d18c0625011e38685842b2b6a04d72cb57ba6c07743b-9879d1a04e72185b8/. The Products dataset that supports the findings of this study is available on request from the corresponding author I. Cruz [1]. The data were publicly available in Implicit Aspect Extraction Corpus, www.gelbukh.com/resources/ implicit-aspect-extraction-corpus, but the link is no longer accessible.

## Declarations

**Conflict of Interest** The authors have no competing interests to declare that are relevant to the content of this article.

**Ethics Approval and Consent to Participate** Not applicable.

**Consent for Publication** Not applicable.

## References

1. Cruz, I., Gelbukh, A., Sidorov, G.: Implicit aspect indicator extraction for aspect-based opinion mining. Int. J. Comput. Linguist. Appl. **5**, 135–152 (2014)
2. Gayatree, G., Noémie, E., and Amélie, M.: Beyond the stars: Improving rating predictions using review text content. In WebDB, 2009.
3. Maria, P., Dimitris, G., John, P., Harris, P., Ion, A., and Suresh, M.: SemEval-2014 task 4: Aspect based sentiment analysis. In Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 27–35, Dublin, Ireland, August 2014. Association for Computational Linguistics.
4. Liu, B., Minqing, H., and Junsheng, C.: Opinion observer: analyzing and comparing opinions on the web. In WWW '05, 2005.
5. Li, S., Sheng, L., Jiyun, L., and JuTao, L.: A novel context-based implicit feature extracting method. 2014 International Conference on Data Science and Advanced Analytics (DSAA), pages 420–424, 2014.
6. Geli, F., Liu, B., Meichun, H., Malú, C., and Riddhiman, G.: A dictionary-based approach to identifying aspects implied by adjectives for opinion mining. In COLING, 2012.
7. Michael, E.L.: Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In SIGDOC '86, 1986.
8. Weaver, W.: 1955. Translation. In William N. Locke and A. Donald Booth, editors, Machine Translation of Languages. John Wiley & Sons, New York, pages 15–23. (Reprint of mimeographed version, 1949.)
9. Ide, N., Véronis, J.: Word sense disambiguation: the state of the art. Comput. Linguist. **24**, 1–41 (1998)
10. Marco, M., Simone, C., Michele, B., and Roberto, N.: 2022. Nibbling at the Hard Core of Word Sense Disambiguation. In Pro- ceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 4724–4737, Dublin, Ireland. Association for Computational Linguistics.
11. Simone, C. and Roberto, N.; 2021. Framing word sense disambiguation as a multi-label problem for model-agnostic knowledge integration. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 3269–3275, Online. Association for Computational Linguistics.
12. Miller, G.A.: Wordnet: a lexical database for english. Commun. ACM **38**, 39–41 (1992)
13. Minqing, H. and Bing, L.: Mining and summarizing customer reviews. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, 2004.
14. Toqir, A.R., Yu-N, C.: Aspect extraction in sentiment analysis: comparative analysis and survey. Artif. Intell. Rev. **46**, 459–483 (2016)
15. Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05, page 339–346, USA, 2005. Association for Computational Linguistics.
16. Soujanya, P., Cambria, E., Lun-Wei, K., Chen, G., and Alexander, G.: A rule-based approach to aspect extraction from product reviews. In SocialNLP@COLING, 2014.
17. Rajesh, P., Vedika, G., Vivek, K.S.: Movie prism: a novel system for aspect-level sentiment profiling of movies. J. Intell. Fuzzy Syst. **32**, 3297–3311 (2017)
18. Kim, S. and Flavius, F.: Finding implicit features in consumer reviews for sentiment analysis. In ICWE, 2014.
19. Schouten, K., van der Weijde, O., Frasincar, F., Dekker, R.: Supervised and unsupervised aspect category detection for sentiment analysis with co-occurrence data. IEEE Trans. Cybern. **48**, 1263–1275 (2018)
20. Hajar, E.H. and Benkhalifa, M.: Using synonym and definition wordnet semantic relations for implicit aspect identification in sentiment analysis. In NISS19, 2019.
21. Hajar, E.H. and Benkhalifa, M.: A new semantic relations-based hybrid approach for implicit aspect identification in sentiment analysis. J. Inf. Knowl. Manag. 19:2050019:1– 2050019:37, 2020.
22. García-Pablos, A., Cuadros, M., Rigau, G.: W2vlda: almost unsupervised system for aspect based sentiment analysis. Expert Syst. Appl. **91**, 127–137 (2018)

23. Hijbul, A., Woo-Jong, R., SangKeun, L.: Joint multi-grain topic sentiment: modeling semantic aspects for online reviews. Inf. Sci. **339**, 206–223 (2016)
24. Soujanya, P., Iti, C., Cambria, E. and Federica, B.: Sentic LDA: Improving on LDA with semantic similarity for aspect-based sentiment analysis. 2016 International Joint Conference on Neural Networks (IJCNN), pages 4465–4473, 2016.
25. Pathik, N., Shukla, P.: Aspect based sentiment analysis of unlabeled reviews using linguistic rule based LDA. J. Cases Inf. Technol. (JCIT) **24**, 1–9 (2022)
26. Soni, P.K., Rambola, R.: Deep learning, WordNet, and spaCy based hybrid method for detection of implicit aspects for sentiment. Analysis (2021). https://doi.org/10.1109/conit51480.2021.9498372
27. Ajeet, R.P., Manjusha, P., Siddharth, R.: Topic- level sentiment analysis of social media data using deep learning. Appl. Soft Comput. **108**, 107440 (2021). https://doi.org/10.1016/j.asoc.2021.107440
28. Ganpat, S.C., Yogesh, K.M., Dinesh, G., Ravi, N.: A two-step hybrid unsupervised model with attention mechanism for aspect extraction. Expert Syst. Appl. **161**, 113673 (2020). https://doi.org/10.1016/j.eswa.2020.113673
29. Vickery, B.C.: Reviews: van rijsbergen, c. j. information retrieval. 2nd edn. London, butterworths, i978 208pp. J. Librariansh **11**(3), 237–237 (1979)
30. Aitor, G.-P., Montse, C., German, R.: W2VLDA: almost unsupervised system for aspect based sentiment analysis. Expert Syst. Appl. **91**, 127–137 (2018). https://doi.org/10.1016/j.eswa.2017.08.049