applied sciences

MDPI

Article

Supporting Argumentation Dialogues in Group Decision Support Systems: An Approach Based on Dynamic Clustering

Luís Conceição [1,2,*], Vasco Rodrigues [1], Jorge Meira [1], Goreti Marreiros [1] and Paulo Novais [2]

[1] GECAD—Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development, LASI—Intelligent Systems Associate Laboratory, Institute of Engineering, Polytechnic of Porto, 4200-072 Porto, Portugal
[2] ALGORITMI Centre, LASI—Intelligent Systems Associate Laboratory, University of Minho, 4800-058 Guimarães, Portugal
* Correspondence: msc@isep.ipp.pt

Abstract: Group decision support systems (GDSSs) have been widely studied over the recent decades. The Web-based group decision support systems appeared to support the group decision-making process by creating the conditions for it to be effective, allowing the management and participation in the process to be carried out from any place and at any time. In GDSS, argumentation is ideal, since it makes it easier to use justifications and explanations in interactions between decision-makers so they can sustain their opinions. Aspect-based sentiment analysis (ABSA) intends to classify opinions at the aspect level and identify the elements of an opinion. Intelligent reports for GDSS provide decision makers with accurate information about each decision-making round. Applying ABSA techniques to group decision making context results in the automatic identification of alternatives and criteria, for instance. This automatic identification is essential to reduce the time decision makers take to step themselves up on group decision support systems and to offer them various insights and knowledge on the discussion they are participating in. In this work, we propose and implement a methodology that uses an unsupervised technique and clustering to group arguments on topics around a specific alternative, for example, or a discussion comparing two alternatives. We experimented with several combinations of word embedding, dimensionality reduction techniques, and different clustering algorithms to achieve the best approach. The best method consisted of applying the KMeans++ clustering technique, using SBERT as a word embedder with UMAP dimensionality reduction. These experiments achieved a silhouette score of 0.63 with eight clusters on the baseball dataset, which wielded good cluster results based on their manual review and word clouds. We obtained a silhouette score of 0.59 with 16 clusters on the car brand dataset, which we used as an approach validation dataset. With the results of this work, intelligent reports for GDSS become even more helpful, since they can dynamically organize the conversations taking place by grouping them on the arguments used.

Keywords: group decision making; dynamic clustering; natural language processing; argumentation

Citation: Conceição, L.; Rodrigues, V.; Meira, J.; Marreiros, G.; Novais, P. Supporting Argumentation Dialogues in Group Decision Support Systems: An Approach Based on Dynamic Clustering. *Appl. Sci.* **2022**, *12*, 10893. https://doi.org/10.3390/app122110893

Academic Editors: Kuo-Ping Lin, Chien-Chih Wang, Chieh-Liang Wu and Liang Dong

Received: 2 October 2022
Accepted: 22 October 2022
Published: 27 October 2022

1. Introduction

Currently, most decisions made by higher-ups in organizations are made in groups [1]. Group decision making (GDM) is a process where a group of people, usually called decision makers, select one or more alternatives to solve a specific problem they are discussing. Typically, this is a procedure where the decision makers discuss their viewpoints and opinions to achieve a consensus. There are several advantages associated with group decision-making processes, such as improving the quality of the decision made or sharing the workload. Nevertheless, the right conditions need to be acquired to take advantage of this process, such as the possibility of interaction between decision makers, allowing them to exchange ideas and the ability to understand the reasoning behind different preferences [2–4].

Group decision support systems (GDSSs) have been widely studied over the recent decades to create these conditions, so that decision makers can decide effectively. Due to globalization, higher-ups have been dispersed throughout the globe with different time zones, which led to the creation of GDSSs that could solve these location and time issues, the web-based GDSS. The purpose of these systems is to support the group decision-making process by creating the necessary conditions for it to be effective and by allowing the process to be carried out from any place and at any time [5]. These systems provide a way to solve problems, one of the problems being multi-criteria problems.

Multi-criteria problems consist of a finite number of alternatives, which are known initially before solving the problem [6], and a limited set of criteria that enable comparing alternatives. These criteria should be accurate, coherent with the decision and its domain, feasible, independent of each other, and measurable [7]. When the definition of alternatives and criteria are completed, a multi-criteria decision-making problem is created, and decision makers can express their opinion, arguing on the available alternatives, valuable through the criteria set.

In GDSS, argumentation is ideal, since it makes it easier to use justifications and explanations in interactions between decision makers, allowing them to express their ideas clearly. In addition to that, argumentation can be used to influence their preferences and, subsequently, the outcome of the decision-making process, as well as aiding the creation of higher quality agreements and, at the same time, decreasing the number of unsuccessful negotiations [8]. It is essential to notice that these systems based on argumentation dialogues can generate vast amounts of information, since a group decision-making process usually spans several iterations (rounds), making it difficult for the decision makers to analyze and follow the decision-making process. In addition to that, these systems are not widely accepted in organizations due to several factors, such as the resistance to change from organizations and the fear of losing the value obtained from physical meetings, but mainly due to the lack of explanations on how the system is proposing such solutions [9,10].

To make GDSSs more appealing to organizations, machine learning is beginning to gradually be used in GDSS to enhance their capabilities, for example, through argument mining (AM). AM consists of the automatic identification and extraction of the structure of inference and reasoning expressed as arguments presented in the natural language [11]. AM is helping GDSS to become more attractive to organizations by automatically obtaining meaningful information from unstructured text, such as aspect terms, aspect categories, and polarity detection field [12]. AM can automatically extract data from the discussion's unstructured text natural language and then present those data to decision makers or, for instance, highlight the relevant messages. These improvements decrease the time participants must take to set up their preferences in a GDSS.

AM combines different fields of natural language processing, such as information extraction, knowledge representation, and discourse analysis [13]. In addition to those fields, sentiment analysis can also be performed in AM, be it a document, sentence, or aspect level with different outputs, binary (positive or negative), or multi-level [12]. Sentiment analysis performed at the aspect level is called aspect-based sentiment analysis (ABSA) and intends to classify opinions at the aspect level and identify the elements of an opinion [12].

To make GDSS more accessible to its users, the automatic identification of elements of an opinion is a step needed to reduce the time necessary for decision makers to set themselves up on these systems. For example, the automatic identification of alternatives and criteria on natural language text used in discussions should be a feature in future GDSSs to make them more acceptable to organizations. Some work has been done in the GDM context to achieve this solution. For example, machine learning classifiers can automatically classify the direction (relation) between two arguments [14] or create intelligent reports where an algorithm selects which information topics should be reported to decision makers. These features improve decision makers' perception of the problem they are deciding on through the ability to present accurate and relevant information [14]. All these advancements in AM applied to the GDM context will lead to a better understanding

of the conversations. They will result in a more intelligent way to organize and display the conversation to the decision maker, enhancing the capabilities of the GDSSs.

This work intends to apply unsupervised techniques, most concretely clustering, to group arguments to organize the discussion dynamically. This work's output will enhance the capabilities of the GDSS by offering an ML-based dynamic organization of ideas that could outperform a standard filter, since these standard filters need annotations to be viable. Unsupervised techniques ditch the need for annotated data. They can detect comparisons being made and group them, as well as detecting micro-discussions about a small group of alternatives and group them together.

The rest of the paper is organized in the following order: Section 2 presents some related works on clustering in natural language processing, Section 3 presents the methodology to solve the problem addressed in this article, Section 4 presents the obtained experiments results, and Section 5 presents a discussion about the obtained results. In the last section, some conclusions are presented, alongside suggestions for work to be done afterward.

2. Related Work

This section intends to provide insight into what has been done in terms of clustering with NLP data. Since the context of this work is particular, approaches from multiple fields will be explored, and a critical analysis of each one will be performed to understand if it could be applied to the GDM context. Many approaches were made before clustering on natural language text to achieve multiple objectives.

Kim et al. [15] applied clustering with NLP in the biology field by extracting data from two diverse sources, microarray gene expression data and gene co-occurrences in the scientific literature from bioRxiv using NLP. After normalizing the microarray data and applying dimensionality reduction with principal component analysis (PCA), they grouped this data into clusters using the K-means technique. The resulting clusters were compared to the extracted gene co-occurrences pairs in the NLP data to evaluate the results of the steps taken. The evaluation was done using entropy analysis on the combined data, comparing it to the maximum entropy from the sole clusters. Their results approve the usage of NLP in this field to extract gene co-occurrences from the literature in which the use of clustering helped confirm this claim. Although the approach shows an excellent combination of both areas, it is not what it is intended with this project, since the goal is to cluster unstructured text and not structured data, that was, in their case, the microarray gene expression data.

Sarkar et al. [16] applied clustering with NLP as an intermediary step in creating a model to predict occupational accident risk. After extracting the data from an integrated steel plant's safety management system database, pre-processing is done where duplicates, missing data, and inconsistent data are removed. The authors used EM-based text clustering to build clusters with categorical attributes while using the silhouette coefficient to determine the optimal number of clusters. These data are then fed to a deep neural network (DNN) model, with a structure comprised of a stacked autoencoder (SAE) with an autoencoder (AE) and a SoftMax classifier. The AE is a feed-forward artificial neural network (ANN) comprising one input layer, one hidden layer, and one output layer. Usually, it is trained to copy its input to its output so that the errors become minimum. Therefore, the dimension of the input must be the same as that of the output. Support vector machine and random forest was used to compare this approach. For DNN, the grid search technique was used to find the best hyperparameters. This approach shows one usage of clustering to categorize unstructured data, finding hidden connections between them.

Hema and David [17] applied clustering with NLP in the medical field as an intermediary step in creating a model to predict diseases based on symptoms. The data are collected using medical forums about various stomach disease symptoms, and an OWL file is created. After data preprocessing, stopwords, stemming words, special characters, numbers, and white spaces were removed. Speech tagging is used to extract verbs, nouns,

subjective words, adverbs, etc., from the dataset, so afterward, Fuzzy c means can cluster the data into groups of common symptoms. RDF is then utilized for taxonomic relations, object relations, and data, while OWL is used for attribute relations. These relations are then mapped for the genetic algorithm to predict the disease of a customer based on the symptoms. This approach uses many steps that could be utilized in this project. However, having an intra-sentence segmentation step in our project, the usage of fuzzy c-means becomes less needed. Most of the data will be treated so that it can only be part of one cluster, removing the need for soft clustering techniques.

Dragos and Schmeelk [18] applied clustering with NLP in education to obtain meaningful information from student surveys. They receive open-text survey answers from five cybersecurity courses and cluster the answers to each question on the survey. They first select the cluster number based on heterogeneity. This measure represents the sum of all squared distances between data points in a cluster and the centroids. After the number of clusters is decided for each question, they use TF-IDF as word embedding to cluster the data with k-means and obtain categories based on the top keywords made manually. With this, they aim to fill the gap in identifying valid interpretations of student feedback in the literature. This approach was applied to education and student surveys. However, it seems like it can be adapted into any other field. Both techniques used are not domain-specific, and the categorization done afterward was manual according to the top keywords, meeting the objectives of our work in terms of clustering.

Gupta and Tripathy [19] applied clustering with NLP by creating a methodology that could be used in any domain. They tested it in a zoo dataset. The method consists of implementing a form of clustering that takes a non-numeric dataset and clusters it with the help of the word embeddings provided by the GloVe dataset by generating the vector representation for each of the sentences in the dataset of those words. Then, a dimensionality reduction is performed on the data set using t-distributed stochastic neighbour embedding (t-SNE) to obtain the accurate number of dimensions for proper cluster formation. The data are then clustered using k-means++. The only issue with this technique is that it chooses the number of clusters based on minimum inertia and the least number of clusters in total. They surpassed this difficulty by using the elbow method to decide the number of clusters formed by the algorithm. This methodology sounds interesting on paper, and the possibility of using it in any domain allows it to be adapted to this work.

Huang et al. [20] applied clustering with NLP in StackOverflow discussions to mine comparable technologies and opinions. They utilize tags in each discussion, considering the collection of technologies that a person would like to compare. To learn the tags, they compared two of the most used methods, the continuous skip-gram model and the CBOW model, where the first model outperforms the latter by a marginal difference. With this better model, they compared the difference between the number of dimensions and concluded that eight hundred was the one to use with the best accuracy. To obtain categorical knowledge, they run the tags against TagWiki to get its definition and then extract the tag category with a POS tagger. To mine comparative opinions, they extracted comparative sentences between both by using three steps for each pair of comparable technologies in the knowledge base. They first preprocessed the discussion considering only answers with a positive score and removing the punctuation and sentences that ended with question marks because they wanted to extract facts and not doubts. Finally, they lowercase everything to make tokens consistent with the technologies. Secondly, they locate candidate sentences using a large thesaurus of morphological forms of software-specific terms to match with tag names. In the last step, they select comparative sentences and develop a set of sentence patterns considering POS tags to obtain them. They use Word Mover's Distance to measure the similarity between sentences, which is helpful for short text comparison. This approach uses word embeddings to get a dense vector representation of each keyword from POS tags for comparisons, such as comparative adjectives and nouns, excluding the technologies under comparison. They then compute the minimal distance

between keywords between sentences and use those distances in a similarity score. If the similarity is superior to the threshold, they are considered similar. Finally, to cluster representative comparison aspects, they build a graph where each node is a sentence. They use TF-IDF to extract keywords from a comparative sentence in one community to represent the comparison aspect of this community, removing stop words and choosing the top three with the highest scores to represent the community. Each community is regarded as a document. This approach starts to be more in line with the objectives of our project, especially on comparative opinions mining, which is like our goal of reaching a consensus on the best alternative in a particular problem using criteria to describe the available alternatives.

Y. Liu et al. [21] applied clustering with NLP by collecting COVID-19-related data from Reddit in subreddits of North Carolina, utilizing data preprocessing techniques, such as POS tagging and stop word removal. After this step, GloVe and Word2Vec embedders were tested with the cosine similarity measure used to calculate the similarity between words. Topic modelling techniques and a BERT model were fine-tuned to find people's concerns and key points from the sentences typed on Reddit posts. With the results of the last step, K-Means were used to cluster the sentence vectors into three categories, concluding that reopening and spreading the virus were the most discussed topics during the time of the posts gathered. Some aspects of this approach were considered in our work, such as data preprocessing options and word embedders.

Reimers et al. [22] applied clustering with NLP by testing it in the context of open-domain argument search. To classify and cluster topic-dependent arguments, they measure the quality of contextualized word embeddings, ELMo and BERT. In terms of argument clustering, twenty-eight topics related to current issues about technology and society were picked. Since argument pairs addressing the same aspect should be assigned a high similarity score and arguments on various aspects a low score, they used a weak supervision approach to balance the selection of argument pairs regarding their similarity. After handling this issue, agglomerative hierarchical clustering with average linkage was used to cluster arguments. They also tested K-means and DBSCAN but agglomerative hierarchical clustering provided the best results in preliminary experiments.

Färber and Steyer [23] applied clustering with NLP on the argument search domain to identify arguments in natural language texts. To present aggregated arguments to users based on topic-aware argument clustering, they tried K-means and HDBSCAN, in addition to considering the argmax of the TF-IDF and LSA vectors to evaluate the results. Regarding word embeddings, TF-IDF, and BERT models, Bert-avg and Bert-cls were used as a pre-step for the clustering task. Another interesting remark is that they evaluated whether calculating TF-IDF within each topic separately is superior to computing the overall arguments in the document corpus. The dimensionality reduction technique, UMAP, was tested before clustering to verify its performance related to not using it in which HDBSCAN outperforms k-means on Bert-avg embeddings but using UMAP in combination with TF-IDF results in a slightly reduced performance. They found that Bert-avg embeddings result in marginally better scores than Bert-cls when using UMAP, concluding that this methodology can mine and search for arguments from an unstructured text on any given topic. Reimers et al. [22] and Färber and Steyer [23] contributed to the field of argument searching, which is similar to our work but not in the same context. Their approaches were used as an example for our project, using context-aware word embedding models (ELMo, BERT, and TF-IDF), the clustering techniques used, and their tested hyperparameters. The dimensionality reduction aspect brought by Färber and Steyer [23] is also interesting, as it helped obtain better results by reducing the number of features passed to the clustering techniques.

Dumani and Schenkel [24] applied clustering with NLP by creating a quality-aware ranking framework for arguments extracted from texts and represented in graphs. To achieve that, they used a (claim, premise) dataset based on debates taken on online portals in which they used SBERT instead of BERT, previously used on [25], to obtain the embeddings

of the claims and premises. With these embeddings, agglomerative clustering using Euclidian distance metric and average linkage method was applied to achieve the clustering task. Since the dataset was sizeable (400 k) with many dimensions from the embedder (1024 dimensions), to reduce the time it would take to cluster it with the agglomerative technique, they clustered the dataset with K-means for K = 4. Then, they used agglomerative clustering on the results of K-means. This approach brought to attention some interesting points, such as the size of the dataset used and which measures could be taken to overcome that. Instead of dimensionality reduction, they used K-means as a pre-clustering step to reduce the computational time.

Daxenberger et al. [26] applied clustering with NLP to the argument mining field by creating an argument classification and clustering project for generalized search scenarios. For that, the technology mines and clusters arguments from various textual sources for a broad range of topics and in multiple languages were used, generalizing to many different textual sources, ranging from news to reviews. After fine-tuning a BERT base model, since it outperforms the pre-trained variant by a good margin, the embeddings obtained by this model are sent to the agglomerative hierarchical clustering with a stopping threshold, aggregating all arguments retrieved for a topic into the clusters of aspects. This project, in terms of argument clustering, seems promising. They used a fine-tuned BERT model for the word embeddings and utilized agglomerative hierarchical clustering to obtain arguments divided by aspects, such as the one presented in our project.

3. Methodology

This section addresses the utilized datasets, the processing pipeline, and the definition of the experiments tested.

3.1. Datasets

The two datasets used for this work were created using the methodology for annotating aspect-based sentiment analysis datasets [27]. The baseball dataset discusses which player is the best of all time. The Cars dataset discusses which car brand is the best and why. Both were extracted from Reddit.

The baseball dataset offers 488 rows of annotated data with 20 features, while the car brands dataset offers 388 rows with 16 features. The baseball dataset is recent, which is why it has more features than the 16 categorical features of car brands. These new features consist of a categorical feature to tell from each discussion the row it came from and message upvotes, downvotes, and message score numerical features.

As explained in [27], from the features created, the following features will be used to achieve the proposed objective:

- Sentence text—message typed by a user divided into the sentence level;
- Alternative—list of values that contains the identifier of the alternative;
- Criterion—list of criteria present in the text;
- Aspect—indicates if a specific entity is indicated explicitly or not in a particular opinion, taking explicit or implicit values;
- Polarity—polarity of an opinion towards an entity–attribute pair in a phrase. It can be positive, negative, or neutral;
- OTE—an apparent reference to the entity present in an opinion.

3.2. Methods

This section addresses the processing pipeline: preprocessing tools, word embedders, dimensionality reduction techniques, and clustering techniques.

3.2.1. Preprocessing Steps

Since text documents are unstructured by nature, to properly use them in NLP, suitable preprocessing is needed to transform and represent those text documents in a more structured way so they can be used later on [28]. In addition to that, it increases the quality

of the final results when applied to classification problems, clustering, and other types of issues [29,30].

Online user-generated content, such as forums and social media discussions, is increasingly important, since it can provide essential knowledge to companies and organizations. However, this type of content has lots of noise, such as abbreviations, non-standard spelling, a specific lexicon of the platform, and no punctuation. These problems reduce the effectiveness of NLP tools, hence, why data preprocessing is needed [31].

Multiple steps can be taken in the preprocessing task, such as sentence segmentation, also known as sentence boundary detection and sentence boundary disambiguation, which consists of segmenting large paragraphs and documents into the fundamental unit of text processing, that is, a sentence [32]. Other operations, such as lowercasing text data [33,34] and stop word removal, are usually applied in preprocessing step. Stopwords are a type of word that does not have any linguistic value. Since they are considered low information, removing them allows for focusing on the essential terms of a text document [33]. This task requires a list of stopwords to remove specific words for each natural language, and this list is already compiled [34]. Stemming consists of reducing inflection in words to their base form, which can help deal with sparsity issues and standardizing the text document's vocabulary [33]. Lemmatization works similarly to stemming, in terms of reducing inflected words into their root form but varies in the fact that it tries to do it correctly without crude heuristics, making sure the word that resulted from the lemmatization (lemma) belongs to the language [33,34].

Furthermore, tokenization and normalization are used to break a text document into tokens, commonly words, for more accessible text manipulation [29]. It includes all sorts of lexical analysis steps, such as removing punctuation, number, accents, extra spacing, removing or converting emojis and emoticons, spelling correction, removal of URLs and HMTL characters, etc. [28,33,34]. In this work, a custom-designed intra-sentence segmentation tool was used. In addition to the standard sentence segmentation tool features, this tool works inside the sentence level. It detects comparisons, using the annotations to improve the results when assigning them back to the row. It then applies lowercasing of the text data, tokenization, removal of punctuation, and stopwords. Lemmatization of the tokens is the last preprocessing step taken.

3.2.2. Word Embedders

Word Embeddings transform text data into numerical representation, the so-called vectorization. According to Goldberg [35], word embedding, also known as distributed representations of words, is the term used to represent the technique where individual words are represented into real-value vectors. These vectors often have a dimension number in the tens or even thousands scale. Each word is mapped to one vector, representing a sentence in a list of these vectors. The mapping of a word to a vector can be done through dictionaries. This is better than using sparse word representations on the scale of thousands or even millions of dimensions [36].

TfidfVectorizer (scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer, accessed on 27 September 2022) is a method in the scikit-learn framework that enables the conversion of raw documents into a matrix of TF-IDF features. TF-IDF is the combination of the term frequency (TF) metric that represents the number of times a term occurs in a document versus the total number of terms in a document. In contrast, the inverse document frequency (IDF) represents the number of documents that contain the term [37]. The setting tested for this word embedder that works best for this work was **ngram_range = (1,1)**, where the first value indicates the minimum amount of grams to take into consideration and the second value the maximum, in which, by making them (1,1), we are solely utilizing unigrams.

Word2Vec (github.com/tmikolov/word2vec, accessed on 26 September 2022) is a popular word-embedding technique, developed by Tomas Mikolov [38]. It provides two methods to achieve this task, either by using a continuous bag-of-words (CBOW) [39] or

skip-gram model (SG) [38]. The CBOW method takes the context of each word as the input and tries to predict the word corresponding to the context, while the SG method inverts the CBOW method, using the target word as the input and trying to predict the context. According to the author, CBOW is faster and has better representations for more frequent words, while SG works well with a small amount of data and represents rare words well [40]. In addition, the original GitHub implementation, the Word2Vec method, is in a commonly used framework called Gensim (radimrehurek.com/gensim/models/word2vec, accessed on 26 September 2022). We used the **word2vec-google-news-300** (huggingface.co/fse/word2vec-google-news-300, accessed on 27 September 2022) pre-trained vectors with an activated binary mode max length equaling 200.

Global vectors for word representation (GloVe) is the unsupervised learning algorithm for this task created by Stanford (nlp.stanford.edu/projects/glove/, accessed on 26 September 2022) [41]. It is available on GitHub (github.com/stanfordnlp/GloVe, accessed on 26 September 2022), where they supply pre-trained models for the task, depending on the requirements. Using the pre-trained models means the GloVe model becomes a static dictionary, since we obtain the word and its vector representation by downloading a pre-trained model [42]. In our work, we used **glove.6B.200d** (nlp.stanford.edu/projects/glove/, accessed on 26 September 2022) with 6 B tokens, 400 K vocab, uncased, and 200 dimensions in which we maintained that dimensions preset (max length equaling 200).

FastText is a library for efficiently learning word representation and sentence classification created by Meta Research (opensource.fb.com, accessed on 26 September 2022; github.com/facebookresearch, accessed on 26 September 2022) [43]. It is available on GitHub (github.com/facebookresearch/fastText, accessed on 26 September 2022), where they offer their state-of-the-art model for English word vectors and word vectors for 157 additional languages. It diverges from Word2Vec by using subword information on word similarity tasks to improve its results. We used **crawl-300d-2M** (fasttext.cc/docs/en/english-vectors, accessed on 26 September 2022), which consists of 2-million-word vectors trained on common crawl with 600 B tokens, and we used a max length equaling 200.

Bidirectional encoder representations from transformers (BERT) is a language representation model released by Google. It considers the context when creating word and sentence-embedding vectors, where the exact two words can have two different vectors, [44,45]. We utilized the **BERT-Base** pre-trained model with 12 layers, 768 hidden states, 12 heads, and 110 M parameters found on their GitHub page (github.com/google-research/bert, accessed on 26 September 2022).

Sentence-BERT (SBERT) (github.com/UKPLab/sentence-transformers, accessed on 26 September 2022) is a modification of the original pre-trained BERT network that uses Siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine similarity [46]. We utilized the **all-MiniLM-L6-v2** (sbert.net/docs/pretrained_models, accessed on 26 September 2022) pre-trained model with 6 layers and 384 hidden states totaling 1 billion training pairs.

Embeddings from language models (ELMo) is a state-of-the-art NLP framework developed by AllenNLP (allenai.org/allennlp/software/elmo, accessed on 26 September 2022). ELMo's representations differ from traditional ones because each token is assigned a representation that is a function of the entire input sentence. This way, word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus [47]. We utilized this model's third version (**v3**) (tfhub.dev/google/elmo/3, accessed on 26 September 2022).

3.2.3. Dimensionality Reduction Techniques

In addition to word embedding, dimensionality reduction techniques are also advised to improve the accuracy of the clustering when handling data with a high number of features, making it advantageous in terms of computational efficiency [48,49].

Principal component analysis (PCA) was initially invented by Pearson [50] and later independently developed and named by Hotelling [51,52]. It is a statistical process that converts a group of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. All principal components are orthogonal to each other. Each one is chosen in a way that represents most of the available variance, with the first component having the maximum variance in a way that it selects a subset of variables from a more extensive set, based on which original variables have the highest correlation with the principal amount [53,54]. PCA can be named differently depending on the field of application, whereas in the ML field, it is called PCA and uses singular value decomposition (SVD) (scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA, accessed on 26 September 2022) [55].

The t-distributed stochastic neighbor embedding (t-SNE) was initially developed by Roweis and Hinton [56]. They created the concept of stochastic neighbor embedding, and later Van Der Maaten and Hinton [57] proposed the t-distributed variant. This variant is a nonlinear technique that converts similarities between data points to joint probabilities and tries to minimize the Kullback–Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. The t-SNE has a cost function that is not convex, meaning that with different initializations, we can get other results [57]. This technique has implementations in multiple technologies, making it widely available (lvdmaaten.github.io/tsne, accessed on 26 September 2022). The t-SNE implementation in scikit-learn uses the Barnes–Hut approximation algorithm, which relies on quad-trees or octa-tree, which makes the maximum number of dimensions that can be used with t-SNE three.

Uniform manifold approximation and projection (UMAP) was developed by McInnes et al. [58] with a theoretical framework based on Riemannian geometry and algebraic topology. It is based on three assumptions, the data are uniformly distributed on a Riemannian manifold, the Riemannian metric is locally constant (or can be approximated as such), and the manifold is locally connected. This way, it is possible to model the manifold with a fuzzy topological structure. The embedding is found by searching for a low-dimensional data projection with the closest possible equivalent fuzzy topological structure (umap-learn.readthedocs.io/en/latest/, accessed on 26 September 2022) [58].

In addition to these three techniques, a hybrid approach applies PCA and t-SNE. PCA reduced to fifty dimensions, followed by t-SNE, will suppress some noise and speed up the computation of pairwise distances between samples (scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE, accessed on 26 September 2022).

3.2.4. Clustering Techniques

Clustering is a decomposition of an entity set into "natural groups" in which these groups capture the natural structure of the data. There are two significant points to clustering, the first being the algorithmic issues on how to find such data decomposition and the second being the quality of the computed decomposition [59]. Initially introduced in data mining research as an unsupervised classification method to transform patterns into groups [59]. Later, it was expanded into other fields, such as information retrieval [60] and text summarization [61]. Its concern is to group a set of entities that are similar to each other and dissimilar from entities that belong to other groups [62]. In the case of intra-cluster density versus inter-cluster sparsity [59], the objective is to minimize intra-cluster distances and maximize inter-cluster distances. Other paradigms exist, such as the density-based paradigm, which is similar to human perception, since we are used to grouping things into categories in our daily life [59].

K-means is the most known clustering technique widely used in multiple fields. It is a partitional type of clustering published by Forgy [63], and then a more efficient version was proposed and published by Hartigan [64]. This method works with a distance function between data points to decide the number of clusters needed (k). Since this technique depends on the selection of the initial centroids for its results and it is a hard clustering

technique (one data point can only be in one cluster), in some cases, this can be seen as a problem. However, it is an effective technique used widely in multiple fields, becoming an excellent all-around clustering technique [65]. We used this technique in a 100-run experiment, with different centroid seeds, as a baseline technique.

K-means++ was created by Arthur and Vassilvitskii [66]. The goal is to disperse the initial centroid by assigning the first centroid randomly and then choosing the rest of the centroids based on the maximum squared distance, pushing the centroids as far as possible from one another [65]. We used this technique in a 100-run experiment with different centroid seeds.

Ckmeans, Consensus K-Means, was created by Monti et al. [67]. It consists of an unsupervised ensemble clustering algorithm, combining multiple K-Means clustering executions. Each K-Means is trained on a random subset of the data and a random subset of the features. The predicted cluster memberships of each single clustering execution are then combined into a consensus matrix, determining the number of times each pair of samples was clustered over all clustering execution [67]. We used this technique in a 100-run experiment with different centroid seeds drawing 92% of the samples and 92% of features for each run. These last two values were obtained by performing preliminary testing.

According to Sonagara and Badheka [68], hierarchical clustering involves building a cluster hierarchy using a tree of clusters, commonly known as a dendrogram. There are two basic approaches to hierarchical clustering:

- Agglomerative—Understood as a bottom-up approach, it begins with points as individual clusters and, at every step, merges the most similar or nearest pair of clusters, needing a definition of cluster similarity or distance.
- Divisive—Understood as a top-down approach, it begins with one cluster gathering all the data. At every step, it splits the cluster until singleton clusters of individual points stay, needing, at every step, a decision on which cluster to separate and how to perform the split.

We utilized the agglomerative hierarchical clustering technique with linkage equaling average, since it was the best linkage method in terms of performance in preliminary tests and backed by state-of-the-art research.

3.3. Proposed Approach

We tested several combinations of techniques and analyzed the impact on the results. Different embedders were tested considering the context (or not), different clustering techniques (partitional and hierarchical-based), and dimension reduction techniques. Their impact on metrics was analyzed. Figure 1 illustrates the pipeline we developed to run these experiments and Table 1 presents a brief overview of the used combinations for the experiments.

The same preprocessing steps were used in every approach testing. They consisted of applying the intra-phrase segmentation algorithm, followed by lowercasing, tokenization, lemmatization, removal of punctuation, and stop words. Whenever additional input data were sent to the clustering method, the min–max method was used to normalize the categorical classes.

The models' outputs will not be changed in terms of dimension reduction. Using that as a baseline: from 200 to 100 dimensions in steps of 50, from 100 to 25 in steps of 25, from 25 to 5 in steps of 5, and from 5 to 1 in steps of 1. In terms of datasets, the baseball dataset, since it is more recent, will be used as the primary dataset to test approaches. In contrast, the Cars dataset will be used as a validation dataset.

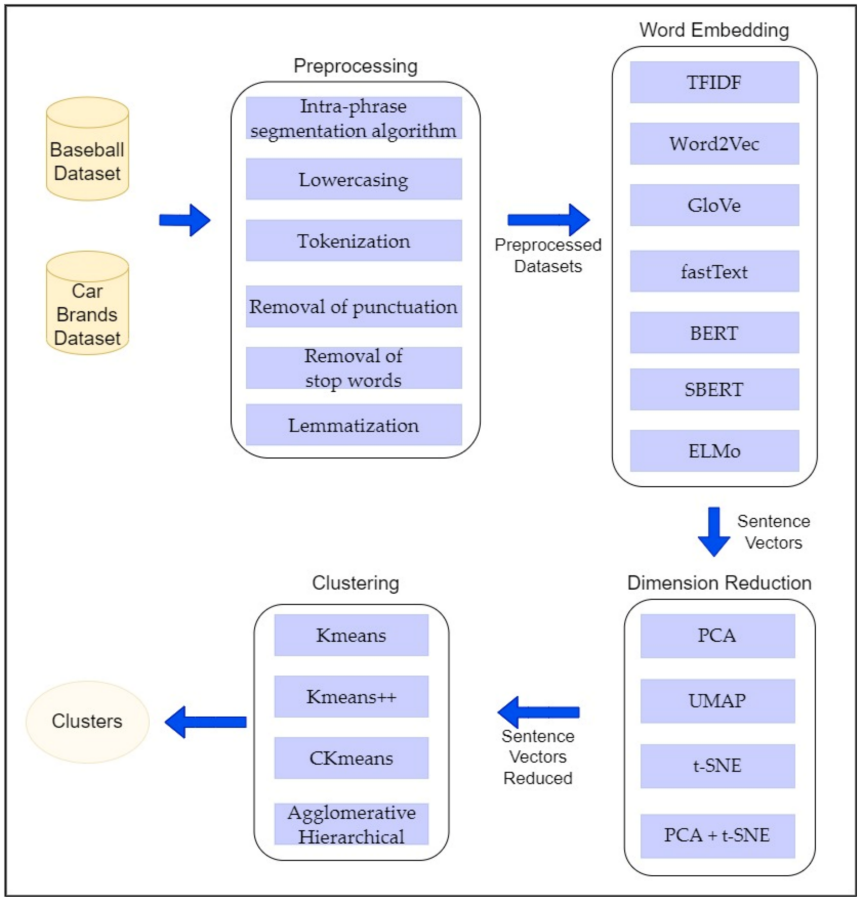


Figure 1. Methodology overview, adapted from [69].

Table 1. Quick visualization of the approach's definition.

	Technique	Implementation	Parameters
Word Embedding	TF-IDF	scikit-learn	ngram_range = (1,1)
	Word2Vec	word2vec-google-news-300	Binary mode = True max length = 200
	GloVe	glove.6B.200d	max length = 200
	fastText	crawl-300d-2M	max length = 200
	BERT	12/768 (BERT-Base)	-
	SBERT	all-MiniLM-L6-v2	-
	ELMo	v3	-
Dimensionality Reduction	PCA	scikit-learn	No change
	t-SNE	scikit-learn	200 to 100 in steps of 50 100 to 25 in steps of 25 25 to 5 in steps of 5 5 to 1 in steps of 1
	PCA + t-SNE	PCA to 50 dimensions and then t-SNE application	
	UMAP	umap-learn	
Clustering	Kmeans	scikit-learn	n_init = 100
	Kmeans++	scikit-learn	n_init = 100
	Agglomerative Hierarchical	scikit-learn	linkage = average
	CKmeans	pyckmeans	n_rep = 100 p_samp = 0.92 p_feat = 0.92

4. Experiments

In this section, the utilized metrics and the obtained experiment results are exposed to allow replication and further discussion in the article.

4.1. Metrics

The results of a clustering technique can be evaluated through metrics that might consider the ground truth labels if they are available. Ground truth labels are humanly provided classifications of the data on which the algorithms are trained or against which they are evaluated [70]. Some metrics will be addressed ahead.

4.1.1. Intrinsic Metrics

When ground truth labels are unavailable, only a few metrics are available to evaluate the performance of a clustering technique [71]. Silhouette is a method that provides a concise measure of how similar an object is to its cluster, compared to other clusters through the usage of distance metrics. Any metric can be used; typically, Euclidian is used to calculate the silhouette coefficient, and the results range from -1 to 1, where a high value means the clusters are well separated, minimizing the distance intra-cluster and maximizing the distance inter-cluster [72].

4.1.2. Extrinsic Metrics

When ground truth labels are available, some metrics exist to evaluate the performance of a clustering technique [71]. Mutual information functions are based on entropy, and entropy decreases as the uncertainty decreases. This way, mutual information reduces the entropy of class labels when we are given the cluster labels, allowing us to know how much the uncertainty about class labels decreases when we know the cluster labels, being similar to the information gathered in decision trees [73].

4.2. Sentences as Only Input Data

In this subset of experiments, only the sentences typed by the participants of the Reddit discussion were used as input data for each clustering technique. Initially, to evaluate which word embedders we should use moving forward, we decided to fixate the k (number of clusters hyperparameter) to the number of alternatives on the baseball dataset. Since the baseball dataset was manually annotated, the ground truth labels were available, allowing us to use the mutual information (MI) metric to evaluate the performance of the approaches. We decided to use MI, since other available metrics that use ground truth labels, such as homogeneity and completeness, have MI as part of their calculations. Furthermore, MI compares the ideal clustering results through the ground truth labels and the obtained clustering results and determines how similar both are.

4.2.1. Word Embedders Variation

As we can see in Figure 2, using K-means as a baseline clustering technique, Word2Vec, fastText, and GloVe all had similar results. Since all of them are static word vectors, GloVe was decided to be used from those three, since it was the fastest computationally wise. SBERT performed better than BERT and was much better computational-wise, hence, why BERT embeddings were dropped moving forward. Furthermore, ELMo is outperformed by SBERT as well, and since both embedders consider the context, SBERT was chosen to move forward as that type of embedder. This way, from these preliminary experiments, TF-IDF, GloVe, and SBERT were the chosen embedders to be used in the following experiments.

4.2.2. Dimensionality Reduction Techniques Variation

The process of converting raw text into word vectors used in clustering techniques produces vectors of variable size. Depending on the size of input data and the word embedding technique applied, the output vectors can reach a size in the order of the hundreds or even thousands per sentence, a length of 200 for the case of GloVe, and a length of 32,768 when we applied SBERT. This high number of features per sentence requires very high processing conditions in terms of RAM memory and processing cores, which are sometimes impossible to have, becoming computationally inefficient generally. In clustering, this is intensified because it makes it harder for a clustering technique to

find similarities in the data to cluster them together when this vast number of features are supplied as input.

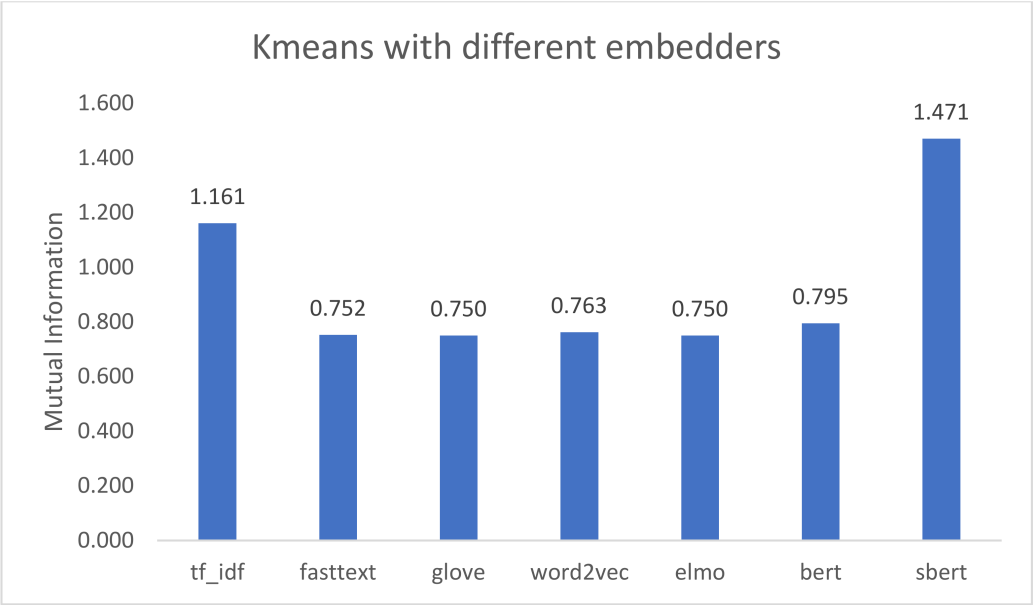


Figure 2. Kmeans performance with different word embedders on the baseball dataset.

Dimensionality reduction techniques, as stated in Section 3.2.2, enhance the methodology's computational efficiency and improve the clustering techniques' performance. Despite the known benefits of dimension reduction techniques, some disadvantages may come with their application. For instance, reducing features may lead to information loss. In addition to that, additional effort is needed to run several tests to find the optimal value for dimensionality reduction.

Having decided on word embeddings to be used in the experiments, we chose not to fixate the number of clusters (K) and test it out with multiple K ranging from 2 to 128 in exponentials of 2. With this change, our ground labels could not be used, since the number of clusters might not be the same as the number of unique ground labels, leading to only the silhouette score getting used. The combination of the silhouette score and the K value it maxes out for each approach will dictate the quality of the results.

Figure 3 shows the best silhouette score for each technique without applying dimensionality reduction techniques. In contrast, Figure 4 shows how the clustering results improve when putting all the embedders with the exact final dimensions, which are 200 from the GloVe word embedder technique. We can see a tendency where UMAP outperforms PCA in this number of dimensions.

Figure 5 shows how the silhouette score varies when reducing the number of dimensions of the embeddings; analyzing the tendencies of the approaches, since multiple approaches are overlapping, making it less readable, we can see that most approaches presented there show a minimal increase in performance until ten dimensions, where it starts to steadily increase as dimensions get reduced, except the agglomerative hierarchical clustering technique with GloVe word embedder and PCA dimensionality reduction that shows a stabilization until 15 dimensions and then a decrease in performance and joining the tendency of the remaining approaches on that graph after ten dimensions. The same pattern can be seen in Figure 6 with the TFIDF with PCA, which spiked in performance after ten dimensions. The GloVe with UMAP does not have a perceivable pattern where it spikes at specific dimensions. Based on most approach patterns, a decision was made to start experimenting from only ten until one dimension.

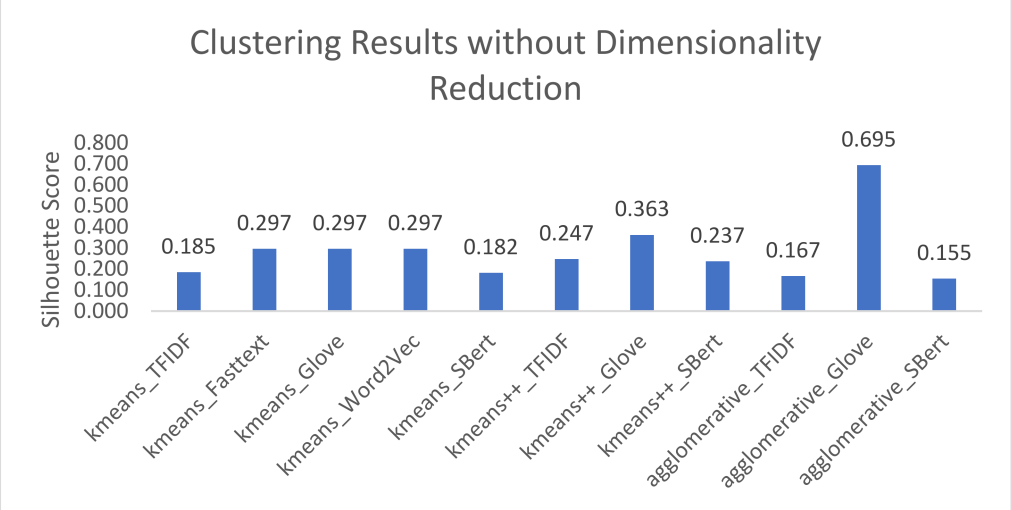


Figure 3. Clustering results in the new test settings on the baseball dataset.

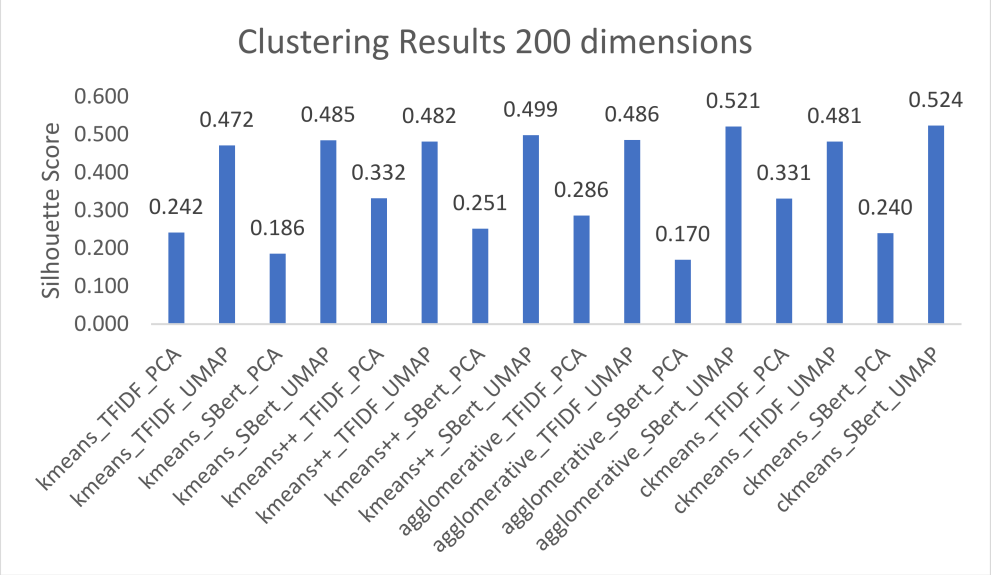


Figure 4. Clustering results with every approach at the 200 dimensions on the baseball dataset.

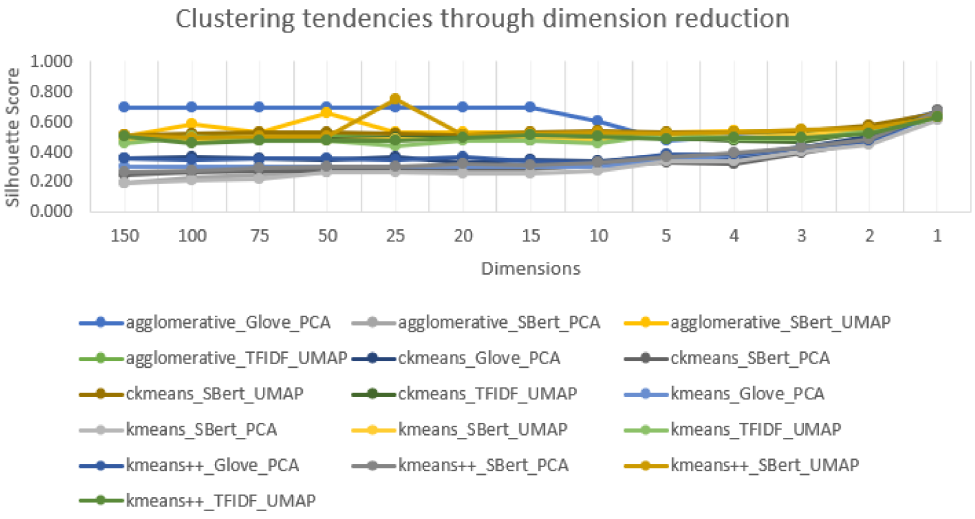


Figure 5. Clustering tendencies through dimension reduction on the baseball dataset.

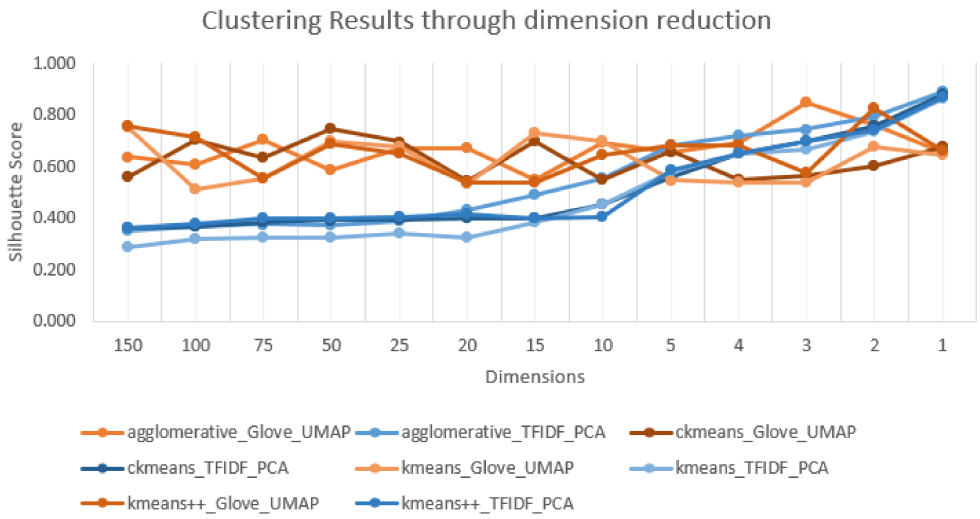
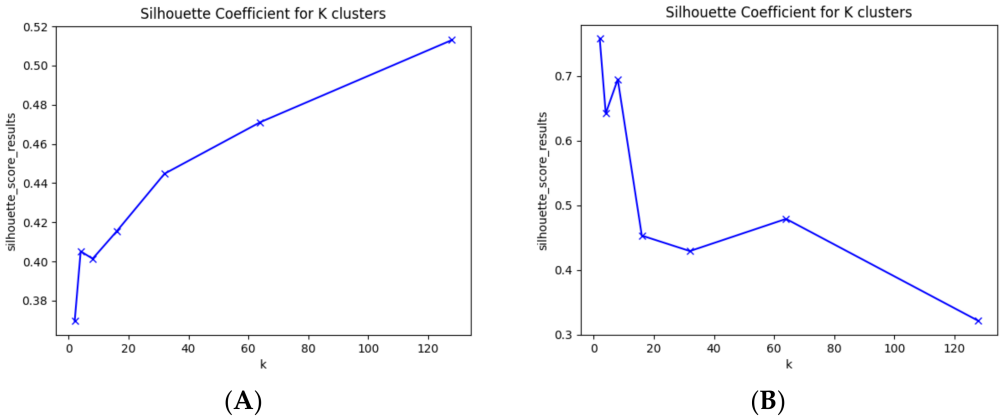


Figure 6. Clustering results through dimension reduction on the baseball dataset.

Even though it displays promising results silhouette score-wise, when analyzing the number of clusters used to reach those values, they either maximize at the 128 clusters (Figure 7A) or on 2 clusters (Figure 7B), which is not the expected result for this task, since 2 clusters group the data with no perceivable differences and 128 clusters are too many clusters with no interest for the solution of the problem that this works intends to solve. Therefore, a tradeoff between the silhouette score and the number of clusters at which a specific approach maxed its silhouette score is considered when evaluating the obtained results.



(**A**) (**B**)

Figure 7. Example of approaches performance through multiple Ks. (**A**) Maximizing in 128 clusters, (**B**) Maximizing in 2 clusters.

4.3. Addition of Polarity to the Input Data

Adding Polarity to the input data did not change the results silhouette score-wise nor on a brief analysis of the resulting clusters. Still, the number of clusters for the best silhouette score of each approach starts to show good values, with some methods maximizing at four and eight clusters, each being more in line with the expected number of clusters when they are not predefined. Since one sentence can have multiple annotations, brief experiments were done to minimize the number of duplicated sentences, no duplicates, and two dupes max, with no interesting results to appoint.

4.4. Addition of Alternative and Criterion to the Input Data

Adding alternative and criterion to the input data (which was already sentenced and polarity) provided some exciting results with one approach that maximized at four clusters showing that the kmeans++ clustering technique with TFIDF word embedder and the

PCA reducing to two dimensions divided the data into polarity and criterion. Whether the criterion part existed or not, it created clusters with positive polarity with criterion and created negative polarity with criterion, etc. Unfortunately, this was not the objective of the work. Still, it was interesting that an unsupervised technique made such a division, which led us to believe that the method put too much emphasis on those two features and was not equally distributed. Only adding alternative and criteria to the sentence without polarity did not achieve any exciting results.

4.5. Addition of Alternative to the Implicit Sentences Text

This way, we believed that going entirely for the sentence as the only input for the clustering technique would bring more desirable results. With just the input sentence, we adjusted some parameters, such as the range of K to be from 2 to 20 in steps of 1 and decided to add the alternative to the input sentence, resulting in high-quality clusters closely related to what was expected. We found that two dimensions was the best amount for reducing dimensions, since it is more in line with practices of the area where a reduction to two dimensions for visualization is made. Most approaches maximized their silhouette score at a sufficient number of K (not in the lowest value of 2 or the highest value of 20) with two dimensions. The best approaches did not show any improvements between the two and one dimensions.

Adding the alternative at the beginning or the end of the sentence did not wield any significant changes to the silhouette score.

5. Discussion

After obtaining the experiment's results, understanding and evaluating them is required to develop an approach to solve the problem.

As we can see in Figure 8, the best approach was the agglomerative hierarchical clustering technique with TFIDF word embedder and PCA reducing to two dimensions. However, this approach reached the best value at two clusters, which we previously discarded; since the goal is to organize conversations, splitting them into meaningful groups, only two clusters would be too reductive. Therefore, the accepted approach to solve the objective of this work was the kmeans++ clustering technique with SBERT word embedder and UMAP reducing to two dimensions, which resulted in eight clusters.

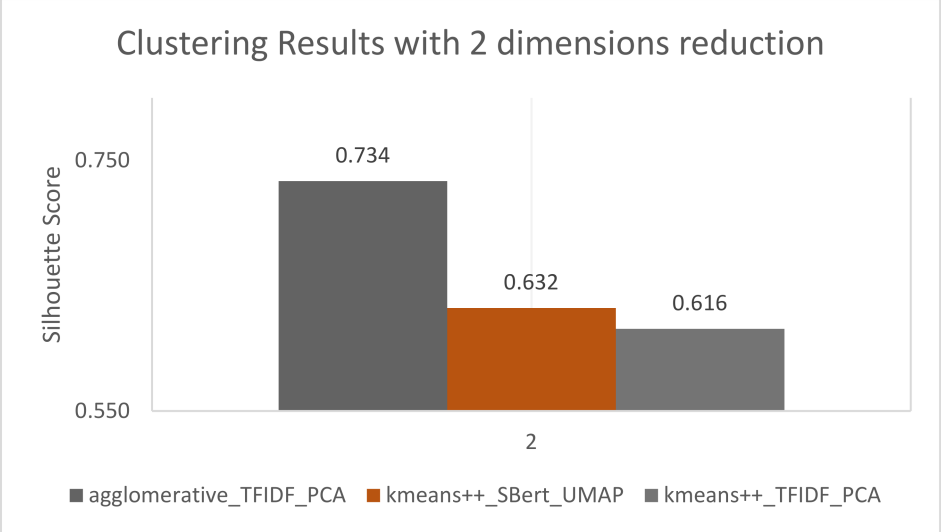


Figure 8. Best clustering results with two dimensions reduction on the baseball dataset.

Analyzing these clusters manually and through word clouds (Figure 9A–H) made us accept this approach because of the variety and the excellent division between them, even though it has a silhouette score of 0.63.

Figure 9. Word clouds with the clustering results of the baseball dataset.

Validating this approach on the other dataset we had, the car brands dataset, wielded good results but not as good as the baseball dataset results, with 16 clusters and 0.59 silhouette score. The word clouds (Figures 10A–H and 11A–H) show some variety, but some clusters could have been better divided, since they refer to different topics.

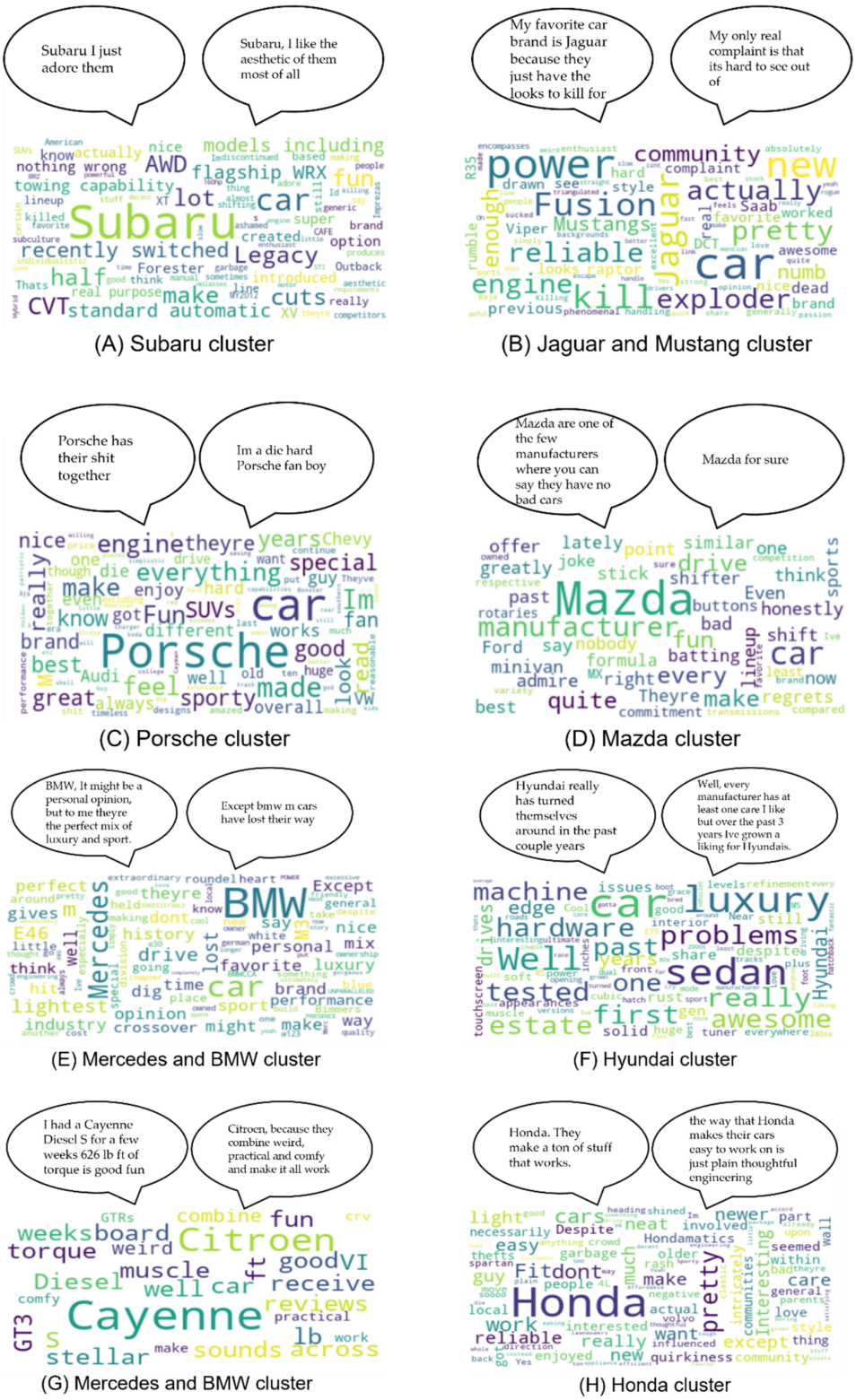


Figure 10. Word clouds with some clusters results of the car brands dataset.

This discrepancy could be attributed to multiple factors, such as the annotation quality of the datasets, their size (since the baseball dataset is bigger than the cars dataset), the distribution of alternatives in each dataset (number of times they appear in messages), the quality of the discussion, and the arguments used. Nevertheless, we believe this approach can achieve the objective of dynamically organizing the conversation based on the arguments used. In a real setting, not fixating on the value of clusters and with even

more data, this approach manages to pick the correct number of clusters for the input data and group that data into perceivable clusters that can then be utilized in intelligent reports for the decision makers.

Figure 11. Word clouds with the remaining clusters results of the car brands dataset.

6. Conclusions

This work aimed to study the application of clustering techniques to the context of group decision making to dynamically generate clusters of the messages exchanged by decision makers.

To achieve the proposed goals, we studied and experimented with multiple clustering techniques, word embedders, and dimensionality reduction techniques to understand what configuration of these three techniques works best for the GDM context. From the tested experiments, the best approach consisted of applying the K-means++ clustering technique with SBERT word embedder and UMAP dimensionality reduction technique, reducing to two dimensions, which resulted in eight clusters with 0.63 silhouette score. Using the same approach on the validation dataset (car brands dataset) obtained satisfactory results but not as good as in the baseball dataset. This difference in results can be related to the small dimension of the car brands dataset and its higher dispersion concerning alternatives, leading to a higher number of formed clusters containing few observations. However, we believe this approach is a feasible solution for the problem we intend to tackle. In a real environment, this approach will automatically pick the correct number of clusters and group the data into perceivable clusters that can then be utilized in intelligent reports for decision makers.

This work contributed to the enhancement of our GDSS prototype, enabling a new feature capable of presenting clusters of messages to decision makers inside the intelligent reports. With this feature, intelligent reports for GDSS become even more helpful, since it can dynamically organize the conversations taking place by grouping them on the arguments used, allowing the decision makers to have a better perception of the direction of the conversation.

In future work, we intend to continue using these two datasets for other experiments, such as a model to detect criteria used in an argument or sentiment analysis models to predict the polarity of the argument in a discussion. Furthermore, the inclusion of this work on the fully fledged GDSS with strong AM models is planned.

Author Contributions: L.C.: Investigation, Conceptualization, Validation, Writing—review and editing. V.R.: Investigation, Software, Writing—original draft. J.M.: Validation, Writing—review and editing. G.M.: Supervision, Writing—review and editing. P.N.: Supervision, Writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Funds through the Portuguese FCT—Fundação para a Ciência e a Tecnologia under the R&D Units Project Scope UIDB/00319/2020, UIDB/00760/2020, UIDP/00760/2020, and by the Luís Conceição Ph.D. Grant with the reference SFRH/BD/137150/2018.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Datasets used in this work are available at https://cloud.gecad.isep.ipp.pt/s/mDMPSPbrAQ9jFge.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Liu, W.; Dong, Y.; Chiclana, F.; Cabrerizo, F.J.; Herrera-Viedma, E. Group decision-making based on heterogeneous preference relations with self-confidence. *Fuzzy Optim. Decis. Mak.* **2016**, *16*, 429–447. [CrossRef]
2. Carneiro, J.; Alves, P.; Marreiros, G.; Novais, P. Group decision support systems for current times: Overcoming the challenges of dispersed group decision-making. *Neurocomputing* **2020**, *423*, 735–746. [CrossRef]
3. Carneiro, J.; Martinho, D.; Alves, P.; Conceição, L.; Marreiros, G.; Novais, P. A Multiple Criteria Decision Analysis Framework for Dispersed Group Decision-Making Contexts. *Appl. Sci.* **2020**, *10*, 4614. [CrossRef]
4. Kaner, S. *Facilitator's Guide to Participatory Decision-Making*, 3rd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2014. Available online: https://www.wiley.com/en-us/Facilitator%27s+Guide+to+Participatory+Decision+Making%2C+3rd+Edition-p-9781118404959 (accessed on 18 March 2022).
5. Carneiro, J.; Martinho, D.; Marreiros, G.; Novais, P. Arguing with Behavior Influence: A Model for Web-Based Group Decision Support Systems. *Int. J. Inf. Technol. Decis. Mak.* **2019**, *18*, 517–553. [CrossRef]
6. Chen, C.-T. Extensions of the TOPSIS for group decision-making under fuzzy environment. *Fuzzy Sets Syst.* **2000**, *114*, 1–9. [CrossRef]

7. Majumder, M. Multi Criteria Decision Making. In *Impact of Urbanization on Water Shortage in Face of Climatic Aberrations*; Springer: Singapore, 2015; pp. 35–47. [CrossRef]
8. Carneiro, J.; Martinho, D.; Marreiros, G.; Jimenez, A.; Novais, P. Dynamic argumentation in UbiGDSS. *Knowl. Inf. Syst.* **2017**, *55*, 633–669. [CrossRef]
9. Tang, M.; Liao, H. From conventional group decision making to large-scale group decision making: What are the challenges and how to meet them in big data era? A state-of-the-art survey. *Omega* **2019**, *100*, 102141. [CrossRef]
10. Conceição, L.; Martinho, D.; Andrade, R.; Carneiro, J.; Martins, C.; Marreiros, G.; Novais, P. A web-based group decision support system for multicriteria problems. *Concurr. Comput. Pr. Exp.* **2019**, *33*, 499–534. [CrossRef]
11. Lawrence, J.; Reed, C. Argument Mining: A Survey. *Comput. Linguistics* **2020**, *45*, 765–818. [CrossRef]
12. Zuheros, C.; Martínez-Cámara, E.; Herrera-Viedma, E.; Herrera, F. Sentiment Analysis based Multi-Person Multi-criteria Decision Making methodology using natural language processing and deep learning for smarter decision aid. Case study of restaurant choice using TripAdvisor reviews. *Inf. Fusion* **2020**, *68*, 22–36. [CrossRef]
13. Lippi, M.; Torroni, P. Argument Mining: A Machine Learning Perspective. *Lect. Notes Comput. Sci.* **2015**, *9524*, 163–176. [CrossRef]
14. Conceição, L.; Carneiro, J.; Martinho, D.; Marreiros, G.; Novais, P. Generation of Intelligent Reports for Ubiquitous Group Decision Support Systems. In Proceedings of the 2016 Global Information Infrastructure and Networking Symposium, Porto, Portugal, 19–21 October 2016; pp. 1–6. [CrossRef]
15. Kim, C.; Yin, P.; Soto, C.; Blaby, I.; Yoo, S. Multimodal Biological Analysis Using NLP and Expression Profile. In Proceedings of the 2018 New York Scientific Data Summit (NYSDS), New York, NY, USA, 6–8 August 2018. [CrossRef]
16. Sarkar, S.; Lodhi, V.; Maiti, J. Text-Clustering Based Deep Neural Network for Prediction of Occupational Accident Risk: A Case Study. In Proceedings of the 2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), Pattaya, Thailand, 15–17 November 2018. [CrossRef]
17. Hema, D.; David, V. Fuzzy Clustering and Genetic Algorithm for Clinical Pratice Guideline Execution Engines. In Proceedings of the 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Coimbatore, India, 14–16 December 2017; pp. 13–16. [CrossRef]
18. Dragos, D.; Schmeelk, S. What are they Reporting? Examining Student Cybersecurity Course Surveys through the Lens of Machine Learning. In Proceedings of the 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 14–17 December 2020; pp. 961–964. [CrossRef]
19. Gupta, A.; Tripathy, B. Implementing GloVe for Context Based k-Means++ Clustering. In Proceedings of the International Conference on Intelligent Sustainable Systems, ICISS 2017, Palladam, India, 7–8 December 2017; pp. 1041–1046. [CrossRef]
20. Huang, Y.; Chen, C.; Xing, Z.; Lin, T.; Liu, Y. Tell them apart: Distilling technology differences from crowd-scale comparison discussions. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, Montpellier, France, 3–7 September 2018; pp. 214–224. [CrossRef]
21. Liu, Y.; Whitfield, C.; Zhang, T.; Hauser, A.; Reynolds, T.; Anwar, M. Monitoring COVID-19 pandemic through the lens of social media using natural language processing and machine learning. *Health Inf. Sci. Syst.* **2021**, *9*, 1–16. [CrossRef]
22. Reimers, N.; Schiller, B.; Beck, T.; Daxenberger, J.; Stab, C.; Gurevych, I. Classification and clustering of arguments with contextualized word embeddings. In Proceedings of the ACL 2019—57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 567–578. [CrossRef]
23. Färber, M.; Steyer, A. Towards Full-Fledged Argument Search: A Framework for Extracting and Clustering Arguments from Unstructured Text. 2021. Available online: http://arxiv.org/abs/2112.00160 (accessed on 28 September 2022).
24. Dumani, L.; Schenkel, R. Quality-Aware Ranking of Arguments. In Proceedings of the 29th ACM International Conference on Information and Knowledge, New York, NY, USA, 19–23 October 2022; pp. 335–344. [CrossRef]
25. Dumani, L.; Neumann, P.J.; Schenkel, R. A Framework for Argument Retrieval. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; Volume 12035, pp. 431–445. [CrossRef]
26. Daxenberger, J.; Schiller, B.; Stahlhut, C.; Kaiser, E.; Gurevych, I. ArgumenText: Argument Classification and Clustering in a Generalized Search Scenario. *Datenbank Spektrum* **2020**, *20*, 115–121. [CrossRef]
27. Cardoso, T.; Rodrigues, V.; Conceição, L.; Carneiro, J.; Marreiros, G.; Novais, P. Aspect Based Sentiment Analysis Annotation Methodology for Group Decision Making Problems: An Insight on the Baseball Domain. In *Information Systems and Technologies*; Springer: Cham, Switzerland, 2022; pp. 25–36.
28. Denny, M.J.; Spirling, A. Text Preprocessing For Unsupervised Learning: Why It Matters, When It Misleads, And What To Do About It. *Politi. Anal.* **2018**, *26*, 168–189. [CrossRef]
29. Weiss, S.; Indurkhya, N.; Zhang, T.; Damerau, F. *Text Mining: Predictive Methods for Analyzing Unstructured Information*; Springer: New York, NY, USA, 2004; pp. 1–237. [CrossRef]
30. Ramasubramanian, C.; Ramya, R. Effective Pre-Processing Activities in Text Mining using Improved Porter's Stemming Algorithm. *Int. J. Adv. Res. Comput. Commun. Eng.* **2013**, *2*, 4536–4538. Available online: www.ijarcce.com (accessed on 28 September 2022).
31. Čibej, J.; Fišer, D.; Erjavec, T. Normalisation, Tokenisation and Sentence Segmentation of Slovene Tweets. In *Normalisation and Analysis of Social Media Texts (NormSoMe)*; 2016; Volume 2016, pp. 5–10. Available online: http://www.lrec-conf.org/proceedings/lrec2016/workshops/LREC2016Workshop-NormSoMe_Proceedings.pdf#page=10 (accessed on 28 September 2022).

32. Wicks, R.; Post, M. A unified approach to sentence segmentation of punctuated text in many languages. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Bangkok, Thailand, 1–6 August 2021; pp. 3995–4007. [CrossRef]
33. Ganesan, K. All you need to know about Text Preprocessing for Machine Learning & NLP. KDnuggets 2019. Available online: https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html (accessed on 27 January 2022).
34. Kumar, S. Getting started with Text Preprocessing. Kaggle. 2019. Available online: https://www.kaggle.com/sudalairajkumar/getting-started-with-text-preprocessing/notebook (accessed on 27 January 2022).
35. Goldberg, Y. Neural Network Methods for Natural Language Processing. *Synth. Lect. Hum. Lang. Technol.* **2017**, *10*, 92. [CrossRef]
36. Brownlee, J. Machine Learning Mastery: What Are Word Embeddings for Text? *Mach. Learn. Mastery* **2019**. Available online: https://machinelearningmastery.com/what-are-word-embeddings/ (accessed on 11 February 2022).
37. Leskovec, J.; Rajaraman, A.; Ullman, J. *Mining of Massive Datasets*; Cambridge University Press: Cambridge, UK, 2014. [CrossRef]
38. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed representations of words and phrases and their compositionality. *Adv. Neural. Inf. Process. Syst.* **2013**, *2*, 3111–3119.
39. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. In Proceedings of the 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, AZ, USA, 2–4 May 2013; pp. 1–12.
40. Karani, D. Introduction to Word Embedding and Word2Vec. *Towards Data Sci.* **2018**. Available online: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa (accessed on 11 February 2022).
41. Pennington, J.; Socher, R.; Manning, C. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543. [CrossRef]
42. Theiler, S. Basics of Using Pre-trained GloVe Vectors in Python. *Medium. Anal. Vidhya* **2019**. Available online: https://medium.com/analytics-vidhya/basics-of-using-pre-trained-glove-vectors-in-python-d38905f356db#:~{}:text=GlobalVectorsforWordRepresentation,inahigh-dimensionalspace (accessed on 11 February 2022).
43. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. *Trans. Assoc. Comput. Linguistics* **2017**, *5*, 135–146. [CrossRef]
44. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the NAACL HLT 2019–2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; Volume 1, pp. 4171–4186. [CrossRef]
45. McCormick, C.; Ryan, N. BERT Word Embeddings Tutorial. Available online: http://www.mccormickml.com (accessed on 16 August 2022).
46. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 3980–3990. [CrossRef]
47. Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. In Proceedings of the NAACL HLT 2018—2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 1–6 June 2018; Volume 1, pp. 2227–2237. [CrossRef]
48. Cunningham, P. Dimension reduction. In *Machine Learning Techniques for Multimedia*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 91–112. [CrossRef]
49. Allaoui, M.; Kherfi, M.; Cheriet, A. Considerably improving clustering algorithms using umap dimensionality reduction technique: A comparative study. *Lect. Notes Comput. Sci.* **2020**, *12119*, 317–325. [CrossRef]
50. Pearson, K. On lines and planes of closest fit to systems of points in space. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **1901**, *2*, 559–572. [CrossRef]
51. Hotelling, H. Analysis of a complex of statistical variables into principal components. *J. Educ. Psychol.* **1933**, *24*, 417–441. [CrossRef]
52. Hotelling, H. Relations Between Two Sets of Variates. *Biometrika* **1936**, *28*, 321. [CrossRef]
53. Kumar, A. Principal Component Analysis with Python. GeeksforGeeks. 2022. Available online: https://www.geeksforgeeks.org/principal-component-analysis-with-python/ (accessed on 17 August 2022).
54. Kambhatla, N.; Leen, T.K. Dimension Reduction by Local Principal Component Analysis. *Neural Comput.* **1997**, *9*, 1493–1516. [CrossRef]
55. Stewart, G.W. On the Early History of the Singular Value Decomposition. *SIAM Rev.* **1993**, *35*, 551–566. [CrossRef]
56. Hinton, G.; Roweis, S. Stochastic Neighbor Embedding. *Adv. Neural Inf. Process. Syst.* **2002**, *15*, 857–864. Available online: https://papers.nips.cc/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf (accessed on 17 August 2022).
57. van der Maaten, L.; Hinton, G. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
58. McInnes, L.; Healy, J.; Saul, N.; Großberger, L. UMAP: Uniform Manifold Approximation and Projection. *J. Open Source Softw.* **2018**, *3*, 851. [CrossRef]
59. Gaertler, M. Clustering. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Brandes, U., Erlebach, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3418, pp. 178–215. [CrossRef]
60. Bordogna, G.; Pasi, G. Soft clustering for information retrieval applications. *WIREs Data Min. Knowl. Discov.* **2011**, *1*, 138–146. [CrossRef]

61. Deshpande, A.; Lobo, L. Text Summarization using Clustering Technique. *Int. J. Eng. Trends Technol.* **2013**, *4*, 8. Available online: http://julio.staff.ipb.ac.id/files/2011/12/IJETT-V4I8P120.pdf (accessed on 28 September 2022).
62. Bramer, M. Clustering. In *Principles of Data Mining*; Springer: London, UK, 2007; pp. 221–238. [CrossRef]
63. Forgy, E. Cluster analysis of multivariate data: Efficiency versus interpretability of classifications. *Biometrics* **1965**, *21*, 768–769.
64. Hartigan, J. Clustering Algorithms. *J. Mark. Res.* **1997**, *14*, 1. [CrossRef]
65. Iliassich, L. Clustering Algorithms: From Start To State Of The Art. Toptal. 2016. Available online: https://www.toptal.com/machine-learning/clustering-algorithms (accessed on 10 February 2022).
66. Arthur, D.; Vassilvitskii, S. K-means++: The advantages of careful seeding. *Proc. Annu. ACM-SIAM Symp. Discret. Algorithms* **2007**, *7–9*, 1027–1035. [CrossRef]
67. Monti, S.; Tamayo, P.; Mesirov, J.; Golub, T. Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. *Mach. Learn.* **2003**, *52*, 91–118. [CrossRef]
68. Sonagara, D.; Badheka, S. Comparison of Basic Clustering Algorithms. *Int. J. Comput. Sci. Mob. Comput.* **2014**, *3*, 58–61.
69. Kang, Y.; Cai, Z.; Tan, C.-W.; Huang, Q.; Liu, H. Natural language processing (NLP) in management research: A literature review. *J. Manag. Anal.* **2020**, *7*, 139–172. [CrossRef]
70. Urumov, G. The Solid Facts of Ground Truth Annotations. UNDERSTAND.AI. 2021. Available online: https://understand.ai/blog/annotation/machine-learning/autonomous-driving/2021/05/31/ground-truth-annotations.html (accessed on 9 September 2022).
71. Han, J.; Kamber, M.; Pei, J. Cluster Analysis. In *Data Mining*; Elsevier: Amsterdam, The Netherlands, 2012; pp. 443–495. [CrossRef]
72. Rousseeuw, P.J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65. [CrossRef]
73. Yıldırım, S. Evaluation Metrics for Clustering Models. *Towards Data Sci.* **2021**. Available online: https://towardsdatascience.com/evaluation-metrics-for-clustering-models-5dde821dd6cd (accessed on 9 September 2022).