electronics

MDPI

Article

Recommending Words Using a Bayesian Network

Pedro Santos [1,*,†], Matilde Pato [1,2,3,*,†], Nuno Datia [1,3,4], José Sobral [1,5], Noel Leitão [6], Manuel Ramos Ferreira [6] and Nuno Gomes [1]

1 ISEL, Lisbon School of Engineering, Instituto Politécnico de Lisboa, 1959-007 Lisboa, Portugal; datia@isel.ipl.pt (N.D.)
2 LASIGE, FCUL, Faculty of Sciences of the University of Lisbon, 1600-277 Lisboa, Portugal
3 FIT-ISEL, 1959-007 Lisboa, Portugal
4 NOVA LINCS, NOVA School of Science and Technology, 2829-516 Monte da Caparica, Portugal
5 CENTEC, IST, Technical University of Lisbon, University of Lisbon, 1049-001 Lisboa, Portugal
6 TDGI—Property Management Technology, 2740-265 Porto Salvo, Portugal
* Correspondence: pedro.santos@deetc.isel.ipl.pt (P.S.); matilde.pato@isel.pt (M.P.)
† These authors contributed equally to this work.

Abstract: Asset management involves the coordinated activities of an organisation to derive value from assets, which may include physical assets. It encompasses activities related to design, construction, installation, operation, maintenance, renewal, and asset disposal. Asset management ensures the coordination of all activities, resources, and data related to physical assets. Recording and monitoring all maintenance activities is a key part of asset management, often done using work orders (WOs). Technicians typically create WOs using "free text", which can result in missing or ungrammatical words, making it difficult to identify trends and analyse information. To standardise the terminology used for the same asset maintenance operation, this paper proposes a method that suggests words to technicians as they complete WOs. The word suggestion algorithm is based on past maintenance records, and a Bayesian network-based recommender system adapts to present needs verified by technicians using implicit user feedback. Implementing this system aims to normalise the terms used by technicians when filling in a WO. The corpus for this work comes from asset management records collected in a health facility in Portugal operated by a private company.

Keywords: tag recommender system; NLP; word embeddings; asset management; Bayesian network

check for updates

Citation: Santos, P. ; Pato, M.; Datia, N.; Sobral, J.; Leitão, N.; Ramos Ferreira, M.; Gomes, N. Recommending Words Using a Bayesian Network. *Electronics* **2023**, *12*, 2218. https://doi.org/10.3390/electronics12102218

Academic Editor: Ruifeng Xu

Received: 27 March 2023
Revised: 4 May 2023
Accepted: 11 May 2023
Published: 12 May 2023

1. Introduction

Asset management involves effectively managing physical assets to maximise their value and ensure their continued operation over their entire life cycle [1]. The FWHA [2] defines asset management as: "(· · ·) a business process and a decision-making framework that covers an extended time horizon, draws from economics as well as engineering, and considers a broad range of assets (· · ·) and uses the information to help make cost-effective investment decisions". According to ISO/CD 55000, "An asset is an item, thing or entity that has potential or actual value to an organisation". Asset management produces a variety of documents throughout the asset life cycle. These documents are essential for the effective management of assets and include: (1) asset register, (2) maintenance schedule, (3) work orders, (4) inspection reports, and (5) asset disposal reports. Work orders (WOs) detail the work to be carried out on an asset, including the tasks to be performed, the materials required, and the time required, and track maintenance activities and ensure they are completed promptly and effectively. WOs in asset management can be written manually or generated by a computerised maintenance management system (CMMS). However, with the advent of computer technology, many organisations now use a CMMS to create and manage WOs automatically. Although manual WO generation can be less efficient than using a CMMS, it can still be effective for managing maintenance activities on a smaller scale or for one-off maintenance tasks. Our paper presents a solution in a real-world scenario, where asset management occurs in a healthcare facility in Portugal. Commonly, a technician fills out a WO whenever a non-compliance event happens with a piece of equipment.

Human-generated data is prone to errors, such as quality issues or inconsistencies, which can be problematic. These errors are frequently semantic. In addition, unless special precautions are taken, manual errors typically occur at a rate of 2-sigma, or 5–10%, much higher than many other types of error [3]. Manually entered data is widespread in maintenance but has traditionally been under-utilised or excluded from many computer analysis methods due to its complex processing. Even a small plant will have thousands of maintenance WOs, covering tasks from the routine to the unexpected and unique. The most common errors are misspellings, non-existent words or a lack of consistency in words, inconsistent formats, and missing fields [4]. Such issues have traditionally hindered the computer-assisted extraction of explicit knowledge. Despite this, if this data can be processed, it can provide valuable information regarding which machines require frequent maintenance, the severity of breakdowns, and the most commonly cited parts or procedures [5]. Computer-assisted extraction of explicit knowledge typically involves using natural language processing (NLP) techniques to identify and extract key information from text documents. However, it is essential to normalise the terms and words used by technicians. Subjective phrases can be excluded to make them easier to understand.

This normalisation can be achieved by recommending words (called tags) while filling out a description in the WO. The aim of this work is to achieve consistency in the words used when considering the same maintenance operation. A tag recommender system (RS) provides users with a list of relevant tags or keywords for a particular item. In this context, a tag is defined as a word freely added to an object by a user. This can help users to organise and find content more efficiently and improve search result accuracy [6,7]. There are two main sub-problems in the tag RS methods: (1) the object-centred problem, and (2) the personalised problem [8]. Tag RS can be combined into a more complex hybrid tag RS, as well as the regular hybrid RS.

The object-centred problem is what will be treated in this paper in order to normalise the words, and can be formulated as follows (adapted from [8]): "Let us consider a WO $w_i \in W$. Given a set of input tags I_{w_i} associated with the target object w_i, generate a list of candidates $C_{w_i,u}$, such as $C_{w_i,u} \cap I_{w_i} = \emptyset$, sorted according to their relevance to object w_i, and recommend the k candidates in the top positions of $C_{w_i,u}$."

The main contributions of this work are: (1) explore the potential of how tag RS can improve the way WOs are described in a language that is commonly used in a specific context—a Portuguese health facility; (2) normalise the lexicon used by technicians when describing a WO, preventing the same asset management situation being described in different ways; (3) creation of a tag RS based on a Bayesian network, which is capable of adapting to the present needs of the technicians.

This paper is organised as follows: Section 2 presents the research background prior to this research; Section 3 presents a description of the data used; Section 4 describes the methodology taken to address the problem and the steps taken towards the objective; Section 5 presents the evaluation and results of the proposed solution, alongside a discussion about the results obtained; Section 6 indicates some conclusions and future work.

2. State of the Art

Various state-of-the-art algorithms can be used for sentence completion and word suggestion tasks. Several methods of tagging have been used from different perspectives [8]. Most research has focused on personalising tag RS for bookmarking new resources, which suggests tags based on the user's preferences. Users, resources, and tags are often filtered together using CF. Related to the CF tag RS, to improve the search Krestel et al. [9] suggested an approach based on latent Dirichlet allocation (LDA) for recommending tags of resources to overcome the cold start problem for tagging new resources. The annotation process is used to uncover latent topics for which sparsely labelled resources are available.

Song et al. [6] proposed two document-centric approaches using a machine learning (ML) method. The first, graph-based, approach uses bipartite graphs to represent tagged data to find document topics. The other approach is prototype-based, which sees the most representative documents from the collected data and uses a Gaussian process classifier to recommend multiple tags simultaneously. Both approaches use a ranking method to rank the tags based on the current popularity of the tags. Wu et al. [7] presented a generative model for CB tag RS. The model makes use of the tag-content co-occurrence observation. Tag2Word builds on the LDA model to generate words for documents, following the labelled LDA model, assuming each tag is associated with one topic and the topic number of a record is the same as its tag number and, further, adds a latent variable to indicate the probability that the tag itself generates the word. To recommend hashtags for microblogging sites, Ding et al. [10] proposed a translation model combining the advantages of topic modelling and translation from content to tags. Godin et al. [11] developed a method that recommends tweets using the naive-Bayes (NB) technique. This method is based on a binary classifier discriminating between non-English and English tweets. Concerning finished sentences, Goulart et al. [12] introduced a hybrid model that combines the capabilities of the latent semantic analysis (LSA) and NB models. The former can infer the next word from a set of words, while the latter is focused on language semantics. The optimisation process uses the gradient descent (GD) technique and latent information, which involves considering an error function. Lei et al. [13] presented a system that automatically tags a piece of content by text classification. The system is called the capsule network and can be used for recommendation tasks. The capsule network learns from the data collected and updates its recommendations with updated perspectives.

The most recent models commonly used to solve problems such as the one presented here are those based on transformer architectures [14]. For instance, generative pre-trained transformer (GTP-3) [15], bidirectional encoder representations from transformers (BERT) [16], or robustly optimised BERT pretraining approach (RoBERTa) [17]. Despite the outstanding performance of these algorithms, it is safe to say that the costs of using them are much higher than the benefits they provide in this research case. One of the main reasons is that they require a significant amount of data, which in this domain is difficult to obtain since WOs are created according to the purpose of documenting an asset repair or malfunction.

Alternatively, generating synthetic data could help to increase the amount of processed data. The problem with synthetic data generation is that, in this specific asset management domain, there would be the need to verify if the generated data makes sense, usually by a technician of the domain. Therefore, it can be said that synthetic data generation is dependent on domain experts' validation, which can be challenging to obtain promptly. Furthermore, each WO occurs at a specific time and asset, which means synthetic data generation is time and location dependent. Consequently, it is more difficult to generate reliable synthetic data. In conclusion, the available data is insufficient to support these state-of-the-art algorithms. In addition to that, these models are also complex and require a lot of time and space to train compared to traditional ML [18] models. Several studies have shown that ML can improve the asset management process, but they did not find a way to implement a specialised word suggestion algorithm based on the data collected from these activities.

3. Data Description

As mentioned above, our data collection consists of 38,445 WOs related to asset management in a healthcare facility in Portugal. Each WO is composed of several fields, and the technician fills in the information that best describes the given situation (e.g., asset malfunction, routine inspection, component replacement, etc.). Of all the seventy-five fields that compose a WO, the ones with the most important information are presented hereafter, with an example for each field, taken from a real WO, as shown in Table 1.

Table 1. Example of a work order, with 11 considered fields out of 75.

Field	Value
Work order ID	16,715
Job number for the given WO—WOs can be completed through several jobs	1
Asset complete identifier	06001MCTEEE
Asset ID	6001
Work order description	Periodic review of leaks
Date when the WO began	2014-06-04T09:00:00
Date when the WO ended	2014-06-04T11:00:00
Description of the work performed	Periodic review of leaks
Asset designation	Chiller 1
Asset family	CT
Asset subfamily	CH

4. Tag RS Algorithm

A previous analysis of word embedding models (*fasttext* and *word2vec*), along with visualisation techniques of tag clouds [19], enabled us to see how the words are used and the relationships between them in the WOs. Our solution consists of a Bayesian network (BN) that can adapt to the needs of the technicians according to implicit user feedback.

A BN is a probabilistic model that uses Bayesian inference to perform probability computations [20]. Its purpose is to model conditional dependence and causality. In it, every node represents a unique random variable in a directed acyclic graph (DAG). When implementing these concepts to a problem, the model considers the event's significance as the occurrence of a word in a sentence. The goal of the BN is to compute the probability of a given word's occurrence.

Let us consider the DAG is represented as $G = (V, E)$, where V is a set of nodes representing random variables and E is a set of directed edges representing the probabilistic dependencies between the variables. The conditional probability distribution represents the probability of each variable given in its parent variables in the DAG. Each node in the DAG has a conditional probability associated with it, denoted as $P(X_i|Pa(X_i))$, where X_i is the current node and $Pa(X_i)$ is the set of parent nodes of X_i. Let $E = \{e_1, e_2, \cdots, e_n\}$ be a set of observations, where each e_i is an observed value of a variable in a network, the posterior probabilities represents the probabilities of each variable given an observation: $P(X_i|E)$. The probability of each variable given the evidence can be calculated using Bayes' rule:

$$P(X_i|E) = \alpha P(E|X_i)P(X_i|Pa(X_i))$$

where α is the normalisation constant, and $P(E|X_i)$ is the likelihood of the evidence given the value of X_i.

For instance, consider the sentence "fan assembly". $P(fan)$ predicts the likelihood of the word "fan" being used in a sentence, and $P(assembly)$ computes the likelihood of the word "assembly" occurring in a sentence. The probability is a number in the interval [0, 1]. So, $P(assembly|fan)$ gives the conditional probability of "assembly", assuming that the "fan" occurs. Using a frequency analysis of the words in the *corpus* allows the network to compute the conditional probabilities.

The probabilities found in each network state are calculated by analysing a set of WO descriptions. These descriptions serve as a reference for the recommendation of words. In other words, the capacity for the network to suggest words will be tailored according to the sentence constructions verified in the historical file that contains the WO—these can be altered by providing a different file with different descriptions when constructing the network. The frequencies found for each word and the other words' neighbours enable us to model the probability distributions.

Figure 1a depicts the structure of the BN. For instance, the first word is associated with the first state (`State 1`), while the second word is used to refer to the second state (`State 2`), and so on. Generally, the links between states are words that appear together in WOs. `State 1` of the network is composed of states that represent the likelihood of the word, represented by P, appearing as the first word in a sentence. `State 2`, on the other hand, is composed of states that represent the current and previous words in the sentence. The frequency of bigrams is used to model P(`State 2`). A bigram is a combination of two words from the first and second states of the sentence [21]. For instance, "substitution" has position 2 in the sentence "valve substitution". The probability table to model P(`State 2`) provides a list of all the possible word combinations of these two positions. Inputted bigrams will be associated with their respective probabilities of occurrence.

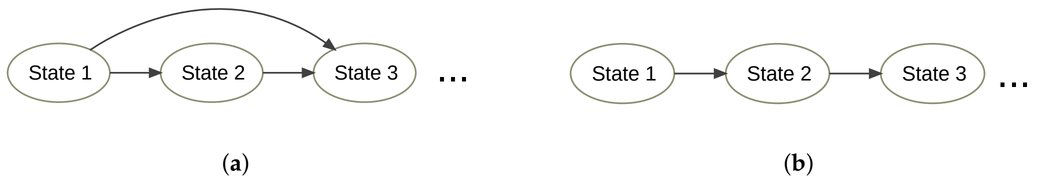


Figure 1. A fully dependent BN (**a**) and a BN respecting the Markov property (**b**).

The remaining states of the network are represented by two possible implementations, shown in Figure 1. The first does not respect the Markov chain property (Figure 1a). The second, on the other hand, does respect the Markov chain property (Figure 1b). That is, the first implementation considers each state as dependent on the previous ones, unlike the second implementation where each state is only dependent on the previous one. This concept is called the "Markov property" [22], which states that a given state has all the necessary information to move on to its next neighbour and state. The first implementation, where a given state depends on all previous states, is described below, to understand better which of the two options is more appropriate.

Now, consider `State 3` in Figure 1a. The user has all the necessary information about the previous states to recommend the most appropriate words. The problem with this assumption is that all of the combinations of words found in the earlier states must be considered when considering all of the states above. This means that the probability table of any given state would have to express all the combinations between the words found in earlier states. This can be achieved by making a list of words that appear in position one and another list of words that appear in position two. Finally, all the words are combined to form all possible sentences. The need to perform this combination of words leads to exponential growth as the number of states increases. This is due to the need to consider all previous words in all previous states, which requires much memory. Depending on the number of states, this approach may not be feasible. The complexity of this task may be unpredictable, depending on the number of states in the network. This computation would require a lot of computational resources and would not provide a significant gain.

The exponential growth problem is the reason why the BN is created using the second implementation of the Markov property, as shown in Figure 1b. The concept of this method significantly reduces the number of joint probability tables associated with each state. Since only the previous state is considered in the current state's probability table, the network's structure is simplified. Reducing memory consumption by considering the Markov property does not affect the network's ability to perform effective inference and answer queries [22]. Furthermore, the construction of a WO usually consists of up to a maximum of eleven words, which is rare. The average for such sentences is about three words per sentence, which makes the second implementation more appropriate.

4.1. Bayesian Inference

Inference in the network is conducted in two possible ways: (1) infer the next word, or (2) infer the remaining words that allow the sentence to be completed. For the first case of inference (1), the methodology is to analyse how many words are already inputted and check the state in the same position as the word to be suggested. Since the BN respects the Markov property, the only factors that influence the network's response to an inference of the next word are found in the current state and its previous neighbour.

For instance, when the user inputs the words "*substituição correia*" ("belt substitution"), the state that is consulted to assess the candidates for the next word is `State 3`, due to the next word being in position 3, as depicted in Figure 2. In this state, the algorithm looks for all the bigrams that begin with the word "*correia*", and sorts them according to their probabilities, placing the highest first. The probability of occurrence of each bigram is initially calculated through a frequency analysis of the inputted *corpus* of text during network creation. Word position is respected. Regarding `State 3`, it only contains bigrams composed of words that are in positions 2 and 3 of a sentence. In the example given in Figure 2, the candidates returned are types of belts, and the suggested words are the last words in each bigram (spz, xpz, and spa).

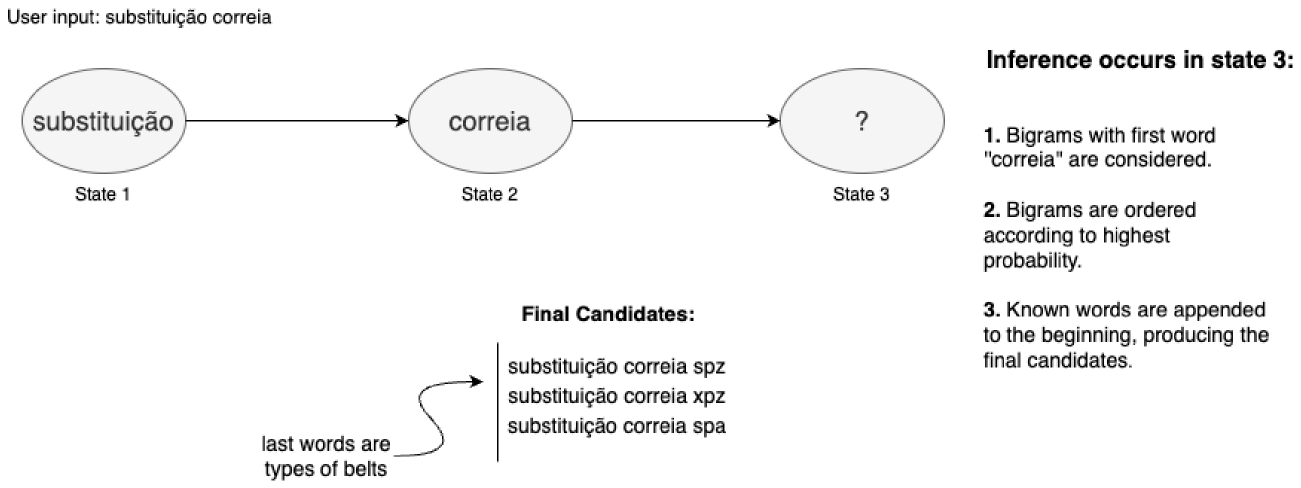

User input: substituição correia
substituição
correia
?
State 1
State 2
State 3
Inference occurs in state 3:
1. Bigrams with first word "correia" are considered.
2. Bigrams are ordered according to highest probability.
3. Known words are appended to the beginning, producing the final candidates.
Final Candidates:
substituição correia spz
substituição correia xpz
substituição correia spa
last words are types of belts

Figure 2. Illustration of the algorithm for predicting the next word.

Turning our attention towards the second case of inference (2), we start by defining that the sentence is deemed to have reached its end when the network does not have more information. This can happen because there are no more states in the network, or no bigrams exist in a given state that permit word recommendation. Moreover, inferring a whole sentence requires searching the entire network for possible word combinations. The implemented inference algorithm uses recursive programming [23] to traverse the network; its modus operandi initiates with the initialisation of a matrix M, which saves possible sentence constructions as rows. When the algorithm reaches its end, this matrix M will have all the sentence constructions ordered by highest probability of occurrence. A matrix cell yields a tuple formed by the word and the likelihood of occurrence given by the Bayes theorem, in the form $(word, probability)$. The algorithm is illustrated in Figure 3, representing a BN with four states.

Furthermore, this algorithm for inference of a whole sentence initiates by searching the network, making use of the first kind of inference—inference of the next word—and having as input the last word entered by the user (step ① in Figure 3, word *a* is representative of any given word that the user could input); the five words that will most likely succeed as the next word are returned. The algorithm works by searching in depth. Therefore, the next step is to look at the first word returned from the group of five, the most probable successor, and infer the next possible successors to this word in the next state using inference (step ②). In the next state, only two suggestions are given, and the recursion depth is now at level 1. From this new group of two possible candidates, the first one is chosen again to infer the following word in the following state (step ③). The recursion depth is now at level 2, and no candidates are verified in `State 4`. When there are no more candidates, the words present on the path traversed during the search are appended to the matrix to form a sentence. Information about previously chosen words is kept so the matrix line can be filled with the resulting sentence when there are no more candidates. In the next step (step ④), the algorithm goes back to the previous state (`State 3`) and infers the next word of the second candidate. The procedure continues, always searching in depth through recursion, until no more words exist. Another sentence construction is added to the matrix every time it is verified.

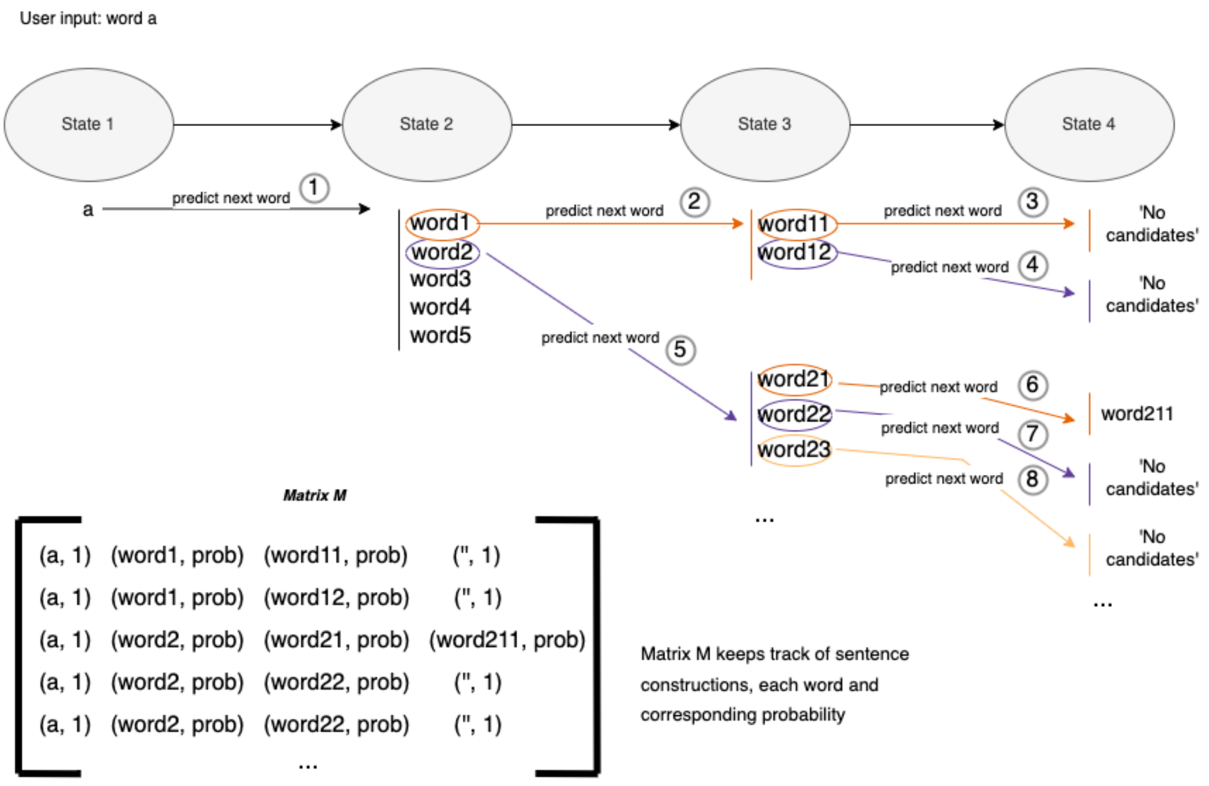


Figure 3. Illustration of the algorithm for inference of a whole sentence. The algorithm actions are numbered to indicate their order, as this is a depth-first search algorithm. The colours are used to represent the order of candidates in the current state, with orange indicating the first candidate, purple for the second, and yellow for the third.

The algorithm chooses a word in a given state and searches for the next group of candidates to complement this word in the next state. If the following state has candidates, the same procedure is applied to each candidate. Suppose the following state does not have information about possible candidates for the given word. The sentence is considered constructed. In that case, it is appended to the matrix, and the focus shifts again to the previous state. Then, the following word from the list of possible suggestions is considered, and the process repeats itself until the network is covered.

In summary, the algorithm analyses the suggestions given by the first type of inference, and recursively searches for the following five recommendations for each of the words in the next state until there are no more possible candidates. At the end of the inference of the whole sentence algorithm, the matrix M created in the first step has the sentence constructions sorted accordingly after the network's traversing is completed. The sorting procedure is achieved by multiplying the probability values associated with each word with each other, in each sentence (line in matrix).

Because the sorting is performed by multiplying the probabilities for each word, the final probability for each sentence construction is small. Sentences with the highest likelihood are considered first, and shorter sentences are given priority, by considering the tuple (" ", 1)—empty string with probability one—for cases where no candidates were found. The network structure presents four states in the modelling given in Figure 3. Knowledge from the domain specifies that WO descriptions should be small and straight to the point. Hence, the cases with long sentences, that would require support for more extensive networks, are rare in most subfamilies, allowing them to avoid the problem of the network occupying a lot of space. The pseudo-code for the algorithm of sentence completion can be found in Algorithms 1 and 2. It is divided into two algorithms for the sake of simplicity and readability, where Algorithm 1 has as its input parameter the words inputted by the technician ("known_words"), and Algorithm 2 is the recursive algorithm in which the traversing of the network occurs.

4.2. User Feedback in the Bayesian Network

Asset management is known to be a volatile domain. Domain knowledge lets us understand that different assets malfunction at different times and that preventive operations are specific to each asset, creating distinct WOs. This consideration allows the understanding that the network's recommendations cannot be static. They need to be adapted according to the situations that occur in the present. Since the probability distributions of the network are created using a specific set of historic WOs from an input file, the validity of those probabilities, in terms of how close they are to the present reality, expire over time. Hence, to deal with this necessity for adaptation, the user's (technician) feedback is included by analysing the suggestions they choose. This idea focuses on "reward asset management situations that have happened recently for the last time". For example, if the operation "fan disassembly" was verified two times on the same day, it makes sense to assume that the technicians might be doing maintenance operations that include this situation, giving it a higher probability of being suggested by the network.

Network adaptation is achieved by analysing the chosen suggestions (creating user feedback) to adapt to the current needs verified by the technicians. Figure 4 outlines the reward function implemented to support this user feedback mechanism. As illustrated, a decay function decreases as time passes, assuming that the highest rewards are allocated to situations that occur often. The values given for the function seem to be out of context. However, an empirical study on the most appropriate values for the function is conducted in the next section.

Algorithm 1 Predict Sentence

```
Require: $self \neq NULL; known\_words \neq NULL;$
  if network does not know any of the words in $known\_words$ then
    return []
  end if
  $known\_words\_tupled \leftarrow list(map(lambda$ x: $(x, 1), known\_words))$
  $suggestions\_matrix \leftarrow$
        $self.predict\_sentence\_possibilities(known\_words,$
              $known\_words\_tupled, [])$                      ▷ call to algorithm 2
  $self.calculate\_final\_probabilities(suggestions\_matrix)$   ▷ multiply all the probabilities in each word
  for each found sentence
  $suggestions\_matrix.sort()$                                 ▷ sort by highest probabilities first
  return $suggestions\_matrix$
```

Algorithm 2 Predict Sentence Possibilities

```
Require: $self \neq NULL; known\_words \neq NULL; known\_words\_tupled \neq NULL;$
  $matrix \neq NULL;$
  $known\_words\_level \leftarrow len(known\_words)$    ▷ last state with known word
  if $known\_words\_level >= len(self.number\_of\_states)$ then
      return $matrix$
  end if
  $level\_bigrams \leftarrow$
          $self.states\_distributions[known\_words\_level]$    ▷ states_distributions has the probability distributions of each state
  $candidates \leftarrow$
          $self.predict\_next\_word(known\_words\_tupled, level\_bigrams)$    ▷ predict_next_word returns a list of tuples, where each tuple is a combination of the candidate to be the next word and the associated probability
  if $len(candidates) == 0$ then    ▷ if there were no candidates
      if has not reached last state then
          fill remaining states with $('', 1)$ in $known\_words\_tupled$
      end if
      $matrix.append(known\_words\_tupled)$
      return $matrix$
  end if
  for $candidate$ in $candidates$ do    ▷ when there are candidates
      if $candidate[0]$ not in $known\_words$ then    ▷ to prevent suggesting words already in the sentence
          $known\_words\_tupled\_plus\_candidate \leftarrow known\_words\_tupled + candidate$
          if has not reached last state then
              fill remaining states with $('', 1)$ in $known\_words\_tupled\_plus\_candidate$
          end if
          $matrix.append(known\_words\_tupled\_plus\_candidate)$
          $self.predict\_sentence\_possibilities(known\_words,$
          $known\_words\_tupled\_plus\_candidate, matrix)$    ▷ recursive call to find candidates from the new known word (current candidate)
      end if
  end for
  return $matrix$
```

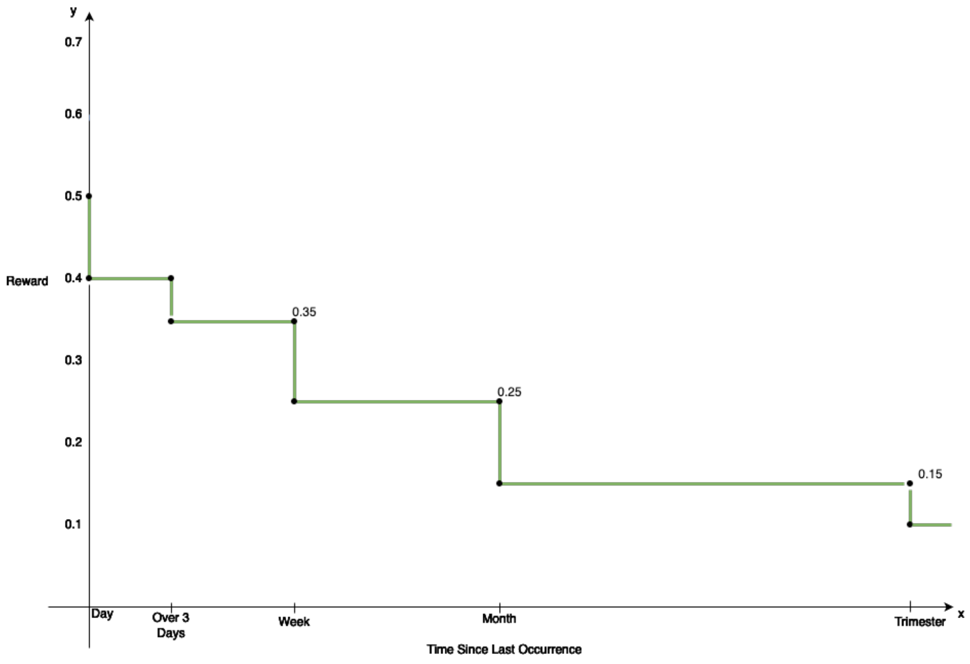


Figure 4. Sketch of the reward function that guides the network's domain adaptation.

The values on the x-axis increase at a significant rate—day, week, month, and quarter. These values in the reward function can be justified by the fact that preventive measures in asset management require users to schedule WOs at specific times (preventive WOs). This occurs to analyse assets and prevent malfunctions before they occur. Since these components do not degrade with a daily frequency, it is essential to consider that for most WOs, weeks, months, or even years may pass before the given malfunctions occur again. Over time, the reward function's temporal specificity grows exponentially, and reward values decrease. Similarly, lower values on the x-axis, such as day and week, have higher reward values, because a given asset management operation occurs multiple times in a short period. This means that technicians may be performing preventive maintenance operations on various assets of the same type, creating similar multiple WOs.

Consequently, the reward function prioritises situations that happen more frequently, making them appear as suggestions with higher rankings. Therefore, the recommendation system can adapt to technicians' needs by analysing the suggestions. The reward function assigns a reward based on how much time has passed since a given asset management situation occurred, prioritising more frequent conditions. The thought process behind the formulation of the reward function is supported through all of the previous considerations. The known literature does not present anything similar, so this reward function is idealised through understanding domain specifications, alongside the domain experts.

Moreover, the idea is that through the reward obtained using the decay function, it is possible to adapt the network's suggestions to the needs of the technicians present. Additionally, the methodology is to observe the choice made by the technician, e.g., "valve substitution", and determine the reward value by identifying the last time this situation occurred. The BN keeps records of the dates and times when a problem was last chosen. Considering that there was another valve substitution in the present day, the reward is 0.5. Using this reward value, the updated frequency for the word "valve" is calculated in `State 1`, and for the bigram "valve substitution" in `State 2`, by applying Equation (1).

$$NewFrequency = OldFrequency + (Reward \times MaxStateFrequency) \tag{1}$$

Looking at Equation (1), it can be understood that the new frequency for the word "valve" is obtained by taking the sum of its old frequency with the maximum frequency found in this first state, times the reward given, 0.5 in this case. The equation uses the maximum frequency found in the given state as a weight. This ensures that the resulting value for the new frequency still respects the probability distribution. This is to prevent the changes from being out of proportion (too large) to the previous configuration of the probability distribution. To end this example, let us assume the word "valve" had a previous frequency value of forty, and the maximum found is sixty-five. The new frequency of this word is given by $40 + (0.5 \times 65)$, which is 73. The frequencies are kept as integers. `State 2` is then calculated by repeating the procedure for the bigram "valve substitution", and calculating the new frequency. Moreover, this formula does not define the maximum value the new frequency can take because the maximum state frequency has no established maximum. The minimum value it can take is one, because zero would mean that the given bigram never appeared in the *corpus*. Hence, it would not make sense that it was included in the network.

Suppose a technician does not choose any suggestion and finishes the WO creation. In that case, the network is updated with the written description and, in the case of being a description with unknown words, these are added to the RS model. Each word is added to the corresponding state with a frequency equal to the mean of the frequencies found in each state. The reason for each new word to be added in each state with the mean of the frequencies is that new words inserted in the model do not start with a frequency equal to one. New words could not appear as suggestions if this was the case, because the other candidate words had higher frequency values.

The pseudo-code for the algorithm for including user feedback in the network can be found in Algorithms 3 and 4. It is divided into two algorithms for the sake of simplicity and readability. Algorithm 3 is run every time a technician chooses a suggestion from the list presented by the RS model. The selected suggestion consists of the “chosen_sentence” parameter. Algorithm 4 is responsible for changing the probability distributions of each state.

Algorithm 3 Inclusion of User feedback

```
Require: $self \neq NULL; chosen\_sentence \neq NULL;$
  $chosen\_words \leftarrow chosen\_sentence.rstrip().split()$
  $reward \leftarrow self.calculate\_time\_based\_reward(chosen\_sentence)$   ▷ calculate time-base reward using reward function
  $self.reconfigure\_state\_distribution(chosen\_words[0], 1, reward)$   ▷ reconfigure first state distribution
  for $n$ in $remainingstates$ do
      $bigram \leftarrow$ (current state word and previous state word)
      $self.reconfigure\_state\_distribution(bigram, n, reward)$
  end for
```

Algorithm 4 Reconfigure State Distribution

```
Require: $self \neq NULL; bigram \neq NULL; level \neq NULL; reward \neq NULL;$
  $prob\_table \leftarrow self.states\_distributions[level - 1]$   ▷ states_distributions has the prob. distributions of each state
  if $level$ != 1 then
      $distribution \leftarrow$ conversion of $prob\_table$ to dictionary
      $distribution \leftarrow$ denormalisation of distribution   ▷ get frequencies instead of probabilities
      if $bigram$ not in current state distribution then
          $bigram\_value \leftarrow$ mean value of all the frequencies
          $distribution[bigram] = bigram\_value$
          $self.added\_frequencies[level] += int(bigram\_value)$   ▷ to keep record of the added frequencies, allowing to maintain the properties of the distribution
          add each word in $bigram$ to $self.all\_words\_distribution$
      else
          $max\_frequency \leftarrow$ maximum frequency value in the distribution
          $frequency\_to\_sum \leftarrow int(max\_frequency * reward)$
          $self.added\_frequencies[level] += int(frequency\_to\_sum)$   ▷ to keep record of the added frequencies, allowing to maintain the properties of the distribution
          $distribution[bigram] = normalized\_distribution[bigram] + frequency\_to\_sum$
      end if
      $distribution \leftarrow$ normalisation of distribution   ▷ get probabilities instead of frequencies
      $prob\_table \leftarrow$ conversion of $distribution$ to list
      $self.states\_distributions[level - 1] = prob\_table$   ▷ update state distribution
  else
      $self.alter\_first\_distribution(bigram, reward, level, prob\_table)$   ▷ first state has independent distribution, dealt in the same manner, treating single words instead of bigrams
  end if
```

5. Results and Evaluations

Understanding the problem domain, the associated data, and the most appropriate algorithm(s) to use in this situation, an ML model can solve a complex problem. When a model is successfully created, evaluating its performance is necessary. It is important to point out that the chosen case study, a Portuguese health facility, was proposed by the company due to the fact that it encompasses the most common asset management activities, and with the importance of healthcare being a sensitive sector in which to conduct asset management, where good practices, such as fast failure resolution or appropriate preventive measures, are crucial. This case study is considered as a representative of similar asset management processes that the company exerts in other asset management contracts that regard different sectors of the economy, in different geographical locations.

By evaluating the RS model, it is possible to determine whether it can be used in technicians’ daily operations. Ideally, the most suitable way to evaluate the solution is through contact with independent external data, alongside the judgement of several domain experts. This procedure includes the word recommendation algorithm on the technicians’ tablets to create the WO. After experiencing the application that allows the creation of new WOs, with the RS providing its suggestions, the technicians would fill out a report where they would specify how appropriate the recommendations were and summarise the experience. This approach is challenging, because a significant amount of time would be necessary to evaluate many technicians and summarise their experiences. Since it is not possible to implement this ideal procedure, an alternative to evaluating the solution is considered. Moreover, the state of the art regarding the evaluation of ML models in asset management is limited, so no methodology is found for evaluating solutions without the participation of technicians. Consequently, the following subsection presents a proposed RS evaluation method.

5.1. Methodology for Recommendation System Evaluation

This subsection aims at explaining the evaluation methodology. By using the BN, this assessment methodology attempts to simulate technicians’ behaviour in creating WOs and observing the network’s recommendations. Therefore, an RS should be tested against a sequence of WOs that are invisible and representative of what technicians might enter. This sequence of WOs has been specially designed to simulate the asset management situations encountered by technicians. One important consideration is that the order of the sequence of WOs is essential, alongside the date of each, so that the evaluation is as close to reality as possible. This consideration allows for the reward function described in the previous section to be properly evaluated. Furthermore, the order in which these WOs appear in the sequence and the period between them is carefully considered, so that specific asset management situations that appear one after the other in real-life scenarios are simulated. This requires an entire set of WOs to test the RS. However, the RS has not seen this sequencing of WOs before; they are representative of the operations conducted in the present by the technicians. The designed methodology is illustrated in Figure 5.

Observation of Figure 5 allows us to understand that the methodology begins with a sequence of carefully chosen WOs; the RS has not “seen” this sequence before, that is, from the entirety of the WOs, a subset is taken for training and a different subset for testing. For the sake of simplicity in the methodology’s overview, let us consider a sequence that starts with the situations limpeza cilindro humidificador” (“humidifying cylinder cleaning”) and “*substituição filtro*” (“filter replacement”). In step ①, the procedure starts by taking the first sentence, looking at the first word, and querying the network for suggestions, as if the technician was inputting this WO. After querying the network for “*limpeza*”, the recommendations are observed in step ②. The performance indicators of this given case are calculated in step ③, following in what rank the network put the word “*cilindro*”, because this word corresponds to the reality, that is, what the technician intended to write. The metrics chosen to evaluate the network are discussed in the following subsection. Let us assume that there is one metric in the performance indicators, which ranges from zero to one hundred percent, named mean reciprocal rank (MRR) [24,25] and discussed in the next subsection. Let us also assume that the network ranked “*cilindro*” in the first place, which means 100% MRR for this case. After calculating the MRR for this word, the procedure is repeated, because the first sentence still has another word; step ④ takes the procedure back to the first sentence, and now the network is queried for suggestions for the case “*limpeza cilindro*”. The network’s recommendations are observed in step ②, and in step ③ the performance indicators for this new case are calculated . This time, the network ranked “*humidificador*” in place five out of ten, for example, so let us assume the MRR is 20% (the calculation of the MRR and other performance indicators is left for the following subsection, for simplicity of explanation of the methodology). Once the performance indicators are calculated for this case, and the sentence reaches its end, in step ⑤, it is possible to summarise the performance indicators for this first WO: considering only the MRR, for simplicity of explanation, the average between 100% from the first suggestion and 20% from the second gives a total of 60% MRR for this first WO. Step ⑥ evaluates whether the sequence of WOs has reached the end. Since there are more WOs in the sequence that have not been processed, the procedure shifts its focus to the next one in step ⑦, which is "*substituição filtro*", and saves the performance indicators of the previously tested WO. In this second WO, the procedure is repeated. Take the first word, "*substituição*", and query the network. Observe the rankings and compute the performance indicators based on what position the intended word "*filtro*". Since this WO only has two words, only one suggestion is made, and the final performance indicators of the sentence are the same as the suggestion.

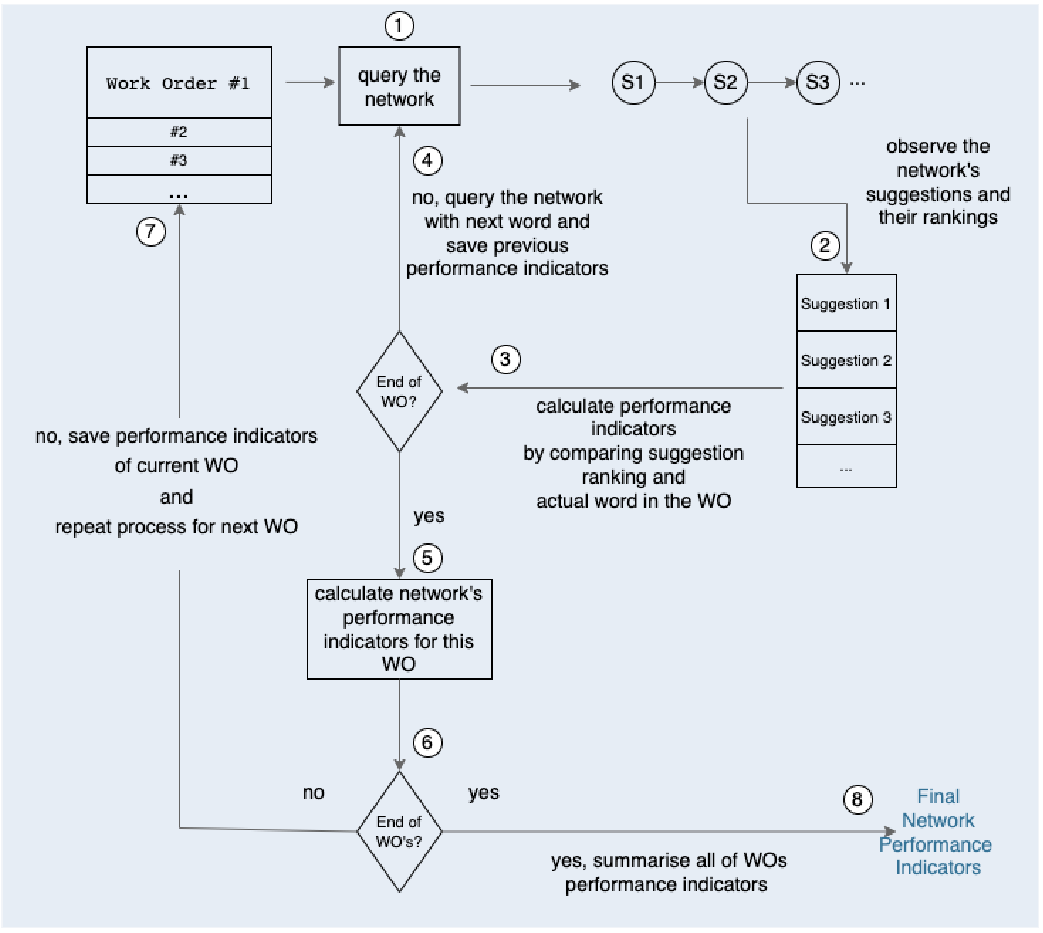

Work Order #1
#2
#3
...
query the network
S1
S2
S3
observe the network's suggestions and their rankings
Suggestion 1
Suggestion 2
Suggestion 3
no, query the network with next word and save previous performance indicators
End of WO?
calculate performance indicators by comparing suggestion ranking and actual word in the WO
no, save performance indicators of current WO and repeat process for next WO
yes
calculate network's performance indicators for this WO
no
yes
End of WO's?
yes, summarise all of WOs performance indicators
Final Network Performance Indicators

Figure 5. Methodology for evaluating the BN.

This methodology is applied for all of the sequences, saving each WO's performance indicators, and summarising all of the indicator's values into one to obtain the final RS's performance in step ⑧. After a given sentence is evaluated, the algorithm for including the user's feedback is run with the WO that was just analysed, allowing for the network's adaptation for future WOs.

5.2. Metrics for Evaluating the Network

The state of the art presents several metrics that can be used to evaluate RSs [26–28]. The most commonly used quality measures are the following: (1) prediction metrics, (2) set recommendation metrics, and (3) rank recommendation metrics. Therefore, the model in this study is evaluated for the relevance of the suggestions given to the technician while filling out a WO form. A model is estimated based on the ranking in which the correct response is located and whether the RS provides adequate guidance. The validation process is performed by employing the most common cross-validation (CV) techniques, such as random sub-sampling and k-fold CV [28].

Some algorithms aim to predict the rating a user would give an item, while others strive to recommend a ranked list of items, i.e., the top@K items, where K is the list size. When the algorithms return a ranked list of items, these may be evaluated for the number of relevant items [24,25]. For example: (1) precision at K (P@K)—this metric measures the proportion of recommended items that are relevant to the user out of the top K recommended items. It is a binary metric, where 1 indicates a recommended relevant item, and 0 indicates a non-recommended relevant item; (2) mean average precision (MAP)—this metric calculates the average precision at each position in the ranked list of recommendations and takes the average of these values. It rewards systems that have high precision across the entire list of recommendations; (3) mean reciprocal rank (MRR)—this metric measures the quality of the top-ranked item in the recommendation list. It calculates the reciprocal rank of the first relevant item in the list and averages these values across all users.

The MAP rewards lists of suggestions with many "correct" (relevant) recommendations and rewards those at the top with the most likely correct recommendations. The problem encountered with this metric when we tried to apply it to our problem, is that there can be several correct answers (suggestions), so the final value for this metric is given by the combination of how many accurate suggestions there are, taking into account the rankings as well. In our solution, only one suggestion is suitable, because only one can be chosen to form a WO's description. Hence, this resulted in labelling the MAP metric as inadequate for the problem at hand, since its value would be very low if it were considered with only one possible correct answer.

Once it was understood that the chosen metric can only consider one right answer from the set of suggestions, the metric chosen was the MRR. So, when a technician is presented with a list of suggestions and chooses the third one, for example, the MRR would be $^1/_3$. If none of the suggestions is what the technician intended and no suggestion is chosen, then the MRR is considered to be zero. In this case, the metric is appropriate, since it is calculated based on ranking the suggestions and only finds one correct answer.

Additionally, using this metric alone can be insufficient to understand the recommendation system's performance. The MRR can be high because there are suggestions where the technician picked the first ranked suggestion. Still, it is also essential to analyse the number of times that none of the suggestions was chosen, that is, the number of times the network produced a list of preliminary results and none of them was considered. Therefore, a second metric is regarded alongside the MRR, which is the *false negative rate*, also known as the *miss rate* [24,25]. This metric is obtained by how often the network presents a list of suggestions where no suggestion is the correct answer, divided by the total number of recommendations made. This is the probability that a relevant item is not recommended.

5.3. Experimental Results and Discussion

The experimental results are produced by applying the previously presented test methodology in carefully chosen scenarios. Three subfamilies with distinct characteristics are considered: *chillers*, *air treatment units*, and *general equipment*. The *chillers* subfamily has the characteristic of dealing only with assets of the chillers type. However, it registers a low quantity of data, only 168 WOs. The *air treatment units* subfamily represents the assets that are responsible for air treatment and holds a total of 2871 WOs. The *general equipment* subfamily is slightly different from the previous two, since it represents a broad spectrum of assets known as available equipment; the WOs present in this subfamily are very distinct from each other, and the malfunctions/situations that are found are particular. It holds 886 WOs.

The methodology consisted in holding a subset for testing and training, with the rest of the WOs used for each of the subfamilies. Three possible partitions for train/test percentages were chosen: 70/30%, 80/20%, and 50/50%. The considered metrics were the ones addressed in the previous subsection, MRR and *miss rate*, with information about the total number of suggestions made and how many of them did not have any relevant suggestion (*miss rate*). Tables 2–4 present the obtained results.

Table 2. Results obtained for the *chillers* subfamily.

Chillers				
Train/Test	**MRR**	**Total Suggestions**	**Missed Suggestions**	**Miss Rate**
70/30	0.5153	167	73	0.4300
80/20	0.5666	94	35	0.3700
50/50	0.4636	278	123	0.4400

Table 3. Results obtained for the UTA subfamily.

UTA				
Train/Test	**MRR**	**Total Suggestions**	**Missed Suggestions**	**Miss Rate**
70/30	0.5700	3548	928	0.2600
80/20	0.5070	2600	743	0.2800
50/50	0.5550	4762	1267	0.2600

Table 4. Results obtained for the general equipment subfamily.

General Equipment				
Train/Test	**MRR**	**Total Suggestions**	**Missed Suggestions**	**Miss Rate**
70/30	0.2787	750	481	0.6400
80/20	0.2751	491	317	0.6400
50/50	0.2610	1253	830	0.6600

From observation of these tables, it is possible to discuss the obtained results.

5.3.1. Chillers Subfamily Results

Starting with the chillers subfamily in Table 2, it is noticeable that the *miss rate* is relatively high for the three combinations of training and testing. This means that the network had difficulty understanding the suitable suggestions to provide to the user, failing to include relevant recommendations about 41% of the time (*miss rate* mean). Looking at the MRR, it is about 50% for the three cases. Remembering that the MRR is calculated by performing the multiplicative inverse of the rank of the first correct answer, that is, 1 for first place, $1/2$ for second place, $1/3$ for third place, and so on, an MRR of $1/2$ means that when there is a correct suggestion from the list presented, it usually sits in the top rank. Therefore, the conclusion is that the network missed a large number of the suggestions that it should make. Still, when it presented a list with a correct option, this right option was one of the top suggestions, which is as intended—correct answers being ranked first indicates a good functioning of the recommendation system. It is also noticeable that the best combination of train and test percentages is 80/20, which demonstrates that the more data the network has to train on, the better its suggestions will be.

By observing that this chillers subfamily has the characteristic of presenting a low number of data, the technique of ten-fold cross-validation (CV) can be applied, with the intent of obtaining better results. Table 5 presents the obtained results. It is possible to conclude that there are better results with a bigger k in the k-fold CV process but by a tiny margin. The case of the chiller subfamily allows us to understand that there are subfamilies with too low quantities of data to generate an optimal model, so there might be a benefit in using other techniques, such as synthetic data generation, with the help of domain experts.

This subfamily and its conclusions represent all other subfamilies presenting low quantities of data.

Table 5. Results obtained for the chillers subfamily with cross-validation.

Chillers with Cross-Validation				
Train/Test	**MRR**	**Total Suggestions**	**Missed Suggestions**	**Miss Rate**
8-fold CV	0.5470	518	176	0.3400
10-fold CV	0.5540	518	182	0.3400
12-fold CV	0.5570	518	173	0.3300
80/20	0.5666	94	35	0.3700

Following another topic of interest in this research, still considering the chillers subfamily, it is possible to include domain knowledge into the recommendation system to obtain more accurate results. The WOs used to train the network initially pass through a NLP pipeline, allowing us to better understand the words and the relationships between them and to remove words that do not present relevance to the domain. Therefore, domain knowledge can be included, using specific stopwords that allow only the relevant words to be kept. The words considered not to be included in the final sentences of the WOs are given by a stopwords file that can be altered. Specifying a subfamily-specific stopwords file, with information about the words that appear in the subfamily that do not present significant information, can allow the recommendation system to achieve better results. The idea is that a domain expert can point out words that do not show significant details. Therefore, domain knowledge that improves the network's adequacy to a given subfamily is included. A stopwords file was generated from a series of meetings with domain experts that was specific to the chillers subfamily. The results compare the best example without subfamily-specific stopwords (80/20) with the same subfamily-specific stopwords model. The results are presented in Table 6. In conclusion, including domain knowledge allows for more adequate results in terms of the MRR metric. However, the *miss rate* remains the same due to the amount of data present in this subfamily being low. For the same reason, the conclusions cannot be taken as absolute.

Table 6. Results obtained for the chillers subfamily with the application of a specific stopwords file.

Chillers with Specific Stopwords				
With/Without CH Stopwords	**MRR**	**Total Suggestions**	**Missed Suggestions**	**Miss Rate**
80/20 with	0.7083	27	10	0.3700
80/20 without	0.5666	94	35	0.3700

Legend: CH means *chillers*.

Further investigations are needed, with more significant quantities of data, considering different subfamilies. However, it is difficult to ask a domain expert to catalogue all unimportant words in a given subfamily, since it is a recurring task of considerable proportions. The main idea here is that including domain knowledge in the model makes it more adequate.

5.3.2. UTA Subfamily Results

The second subfamily on which the recommendation system is tested is "*unidades de tratamento de ar*" (air treatment units). The same conditions as Table 2 are considered, without any CV or inclusion of specific stopwords. The results can be observed in Table 3. The first noticeable thing is the number of evaluated suggestions, which is much higher than in the chillers subfamily. The UTA subfamily had a considerable number of data, 2871 WOs. The MRR is about the same as for the chillers subfamily, 50%, which allows us to conclude that when there are correct answers in the list of suggestions, they sit in the top ranks. However, there is a noticeable difference, which is the *miss rate*. It is a considerably lower value when compared to the chiller's 37%, with about a 27% *miss rate*. This allows us to conclude that, with a considerable amount of data, the recommendation system can be trained appropriately. Therefore, the results are more adequate. This UTA subfamily has a more significant amount of data than the chillers subfamily, which allowed it to conduct training and testing in better conditions. The best proportions of training and testing are 70/30. The subfamilies with the same characteristics as the UTA are considered the most appropriate for training the network since they present patterns that can be discovered and a considerable amount of data.

5.3.3. General Equipment Subfamily Results

The third subfamily on which the network was evaluated was the "*equipamentos gerais*" (general equipment). It was tested with the same conditions as Tables 2 and 3, without any CV or inclusion of specific stopwords. The results can be observed in Table 4. Looking at it, the first noticeable thing is the low score on the MRR metric, about 27%. If the focus is shifted to the *miss rate*, the score is inadequate, with about 65% of missed suggestions. The explanation for these poor results is the fact that there is a high variance in the WOs that this subfamily encompasses. The name of the subfamily, general equipment, suggests that there is a lot of possible WOs on a considerable variety of different assets.

Additionally, the words used are not explicitly used to describe a given kind of asset, such as in the chillers or the UTA subfamily. In this case, there is a high diversity in words used and the assets considered, so the recommendation system cannot identify patterns that will lead to good suggestions. This finding results in the need for a different approach with this kind of subfamily encompassing a broad spectrum of asset management situations and assets.

5.4. Results Regarding Implicit User Feedback

This subsection is intended to evaluate the impact of the implicit user feedback through an ablation study [29], where the adequate models in each subfamily are considered, and an experiment testing the models with and without implicit user feedback is performed. The results can be observed in Table 7.

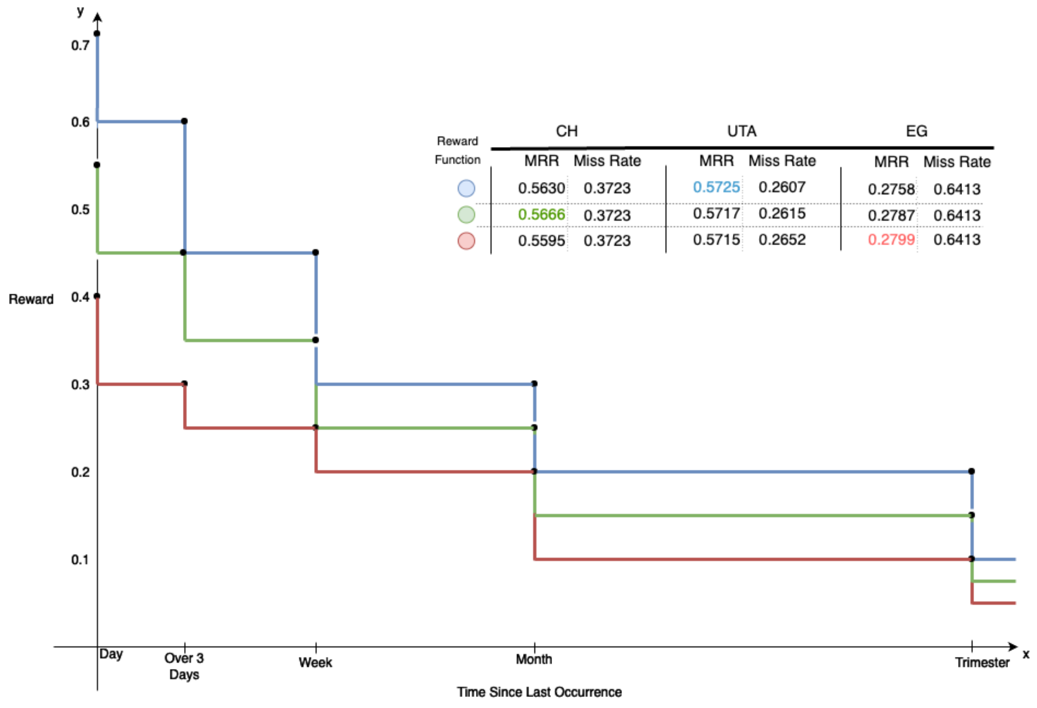
Table 7. Results obtained for evaluation of the user implicit feedback mechanism.

Implicit User Feedback					
Subfamily—Best Eval	**User Feedback**	**MRR**	**Total Suggestions**	**Missed Suggestions**	**Miss Rate**
CH 80/20	Yes	0.5666	94	35	0.3700
CH 80/20	No	0.3480	94	53	0.5600
UTA 70/30	Yes	0.5700	3548	928	0.2600
UTA 70/30	No	0.4576	3548	1307	0.3700
EG 70/30	Yes	0.2787	750	481	0.6400
EG 70/30	No	0.2700	750	513	0.6800

Legend: CH means *chillers*, UTA means *air treatment units* and EG means *general equipment*.

Observation of Table 7 allows us to understand that the recommendation system performs more adequately when the mechanism of implicit user feedback is on. The MRR for all of the subfamilies considered is higher when the mechanism is on, indicating that the correct suggestions are found higher in the list rankings. The *miss rate* is lower for cases where the mechanism is on. There is a noticeable loss in ability to suggest lists with a correct answer when the mechanism is off—in the chillers subfamily, it went from 36% to 56%, for example. The creation of the recommendation system without the mechanism of implicit user feedback results in the only knowledge that the network has being that obtained when historical WO files were presented. The probability distributions in each state are modelled according to the frequencies found in these historical WO files. If the mechanism is not considered, these probability distributions remain static across the lifetime of the recommendation system, resulting in a solution that cannot adapt to new sequences of WOs and learn from them. Alternatively, suppose the recommendation system is created with implicit user feedback. In that case, the probability distributions are altered at each suggestion selection, the technicians are presented with more relevant suggestions, and the network keeps on learning.

A critical component of the presented solution is the reward function, which provides the implicit user feedback mechanism with the notion of how much time has passed between the same asset management situation and how the suggestion that the technician selected should be rewarded. Figure 6 presents three different reward functions, coloured in blue, green, and red. The motivation for this study is to understand the impact of the values given to the reward function by shifting it in the reward axis. Hence, by shifting the importance of the reward associated with each period present on the x-axis, it is possible to produce three functions and study the effect of the reward values on the overall solution. Remember, these values are calculated based on the intricacies of the domain and how the management of work orders is performed. Work orders can occur due to planned preventive activities or corrective operations needed in a given asset. The reward values and periods are designed to meet how the work orders are created and managed in the asset management domain.



Reward Function	CH		UTA		EG	
	MRR	Miss Rate	MRR	Miss Rate	MRR	Miss Rate
🔵	0.5630	0.3723	0.5725	0.2607	0.2758	0.6413
🟢	0.5666	0.3723	0.5717	0.2615	0.2787	0.6413
🔴	0.5595	0.3723	0.5715	0.2652	0.2799	0.6413

Figure 6. Study of reward function values with three distinct functions.

The models with adequate performance indicators are chosen for each subfamily, that is, 80/20 (training and testing percentage) for *chillers*, and 70/30 for *air treatment units* and *general equipment*. The effect produced by the three functions can be observed in the table presented in Figure 6. The performance indicators do not change significantly when using different functions with different reward values in the 0.7 to 0.05 reward range. The function with the best results is different for each tested model. The explanation might be that each subfamily has a specific scheduling of WOs in its preventive plans, and the time that spans between these WOs is more akin to a given reward function. This affinity to a given reward function could be because it has the highest rewards in the most frequent periods of time verified in the maintenance operations of the given subfamily.

6. Conclusions

This research is intended to normalise the words used to describe asset management situations when creating WOs, so that the same situation is not described using similar words. By recommending words while a technician is filling out a WO, this normalisation is achieved, enabling better automatic identification of asset management situations. Statistical tendencies and overall knowledge can be derived from the normalisation of how these WOs are written, enabling large-scale analysis of specific asset management situations in several WOs, allowing the company to improve their decision-making strategies. By unambiguously identifying a given asset management situation in the WOs, asset managers can have a better insight into how asset management activities are being performed at a given facility. Performance indicators of the management process can be established, and preventive measures can be adapted according to the analysis conducted on the normalised descriptions.

The tag RS is implemented using a probabilistic algorithm, a BN, with a custom algorithm that allows it to learn by including implicit user feedback. The network's states model the position of a given word in a sentence. That is, the first state models the first word, the second state the second word, and so on. Each state includes a probability distribution from which the RS suggests words based on the probabilities of the bigrams present in each state. It is from the updating of these probability distributions that the adaptation of the network occurs. Since asset management is volatile, the same asset's functionality will produce different WOs at different times. This domain consideration requires that the network adapts itself to the needs that the technicians demonstrate through implicit feedback. This mechanism of implicit feedback uses a reward function which decays over time, so it works by prioritising asset management situations that have happened more recently for the last time, placing them in higher rankings of the suggestions. The goal of this RS is to influence the writing of the technicians. In contrast, the WO is being created so that the words used are the ones suggested by the RS and, consequently, the ones used in most cases, leading to the normalisation. The mechanism of implicit feedback is created to prevent the suggestion that the network has stopped being relevant due to the volatility of the asset management domain, enabling the network to learn with the recommendations that the technician chooses.

The evaluation of the BN on three different subfamilies presented slightly distinct outcomes, and the conclusions drawn from each subfamily represent what happens in other subfamilies with the same characteristics. The chillers subfamily presents a low number of data, so the recommendation system had difficulty in conducting appropriate training. In future work, different techniques should be explored, such as the synthetic generation of data, with the help of domain experts, to increase the amount of data. The results were adequate for the *"unidades de tratamento de ar"* (air treatment units) subfamily since this subfamily had a considerable amount of data, and the network was able to identify the patterns present in it. For the *"equipamentos gerais"* (general equipment) subfamily, the network produced inaccurate results, because there exists a high variance in the WOs that this subfamily encompasses. There is a high diversity in words found in the WOs and the considered assets, so the network cannot identify patterns that will lead to adequate suggestions. In future work, it would be interesting to study how the patterns present in these broad-spectrum subfamilies can be discovered. One possibility is drilling down into the hierarchy and learning the WOs by asset instead of by subfamily.

One of the main components of this research is the mechanism of implicit user feedback. It is evaluated through an ablation study. The conclusion is that the suggestions of the recommendation system are improved significantly when this mechanism is considered. The only registered drawback regarding this mechanism is that the values given for the reward function appear out of domain assumptions since there is limited literature. This reward function was designed through a thorough understanding of the domain, how the work orders are created and managed, and how these details could be included in the recommendation system, making it able to learn the needs of the technicians. Consequently, a study which aims at understanding the effect of the reward values in the overall solution is conducted, where three different reward functions are used to test other models, and their performance indicators are observed. Concluding, the performance indicators do not present significant changes when using each of the different reward functions.

The next step in this research would be to test the entirety of the one hundred and eight (108) subfamilies one-by-one, observe the results, and understand the next steps that can be taken to improve the subfamilies in which the results were not the most adequate—whether because they encompassed too many distinct assets or the network could not identify patterns. Another interesting path to undergo with our investigation would be to consider a different case study regarding another facility managed by the company, such as a shopping mall. Training and evaluating the RS based on WOs produced in the context of asset management activities in a shopping mall could present significant indicators of how versatile our solution is, taking into account the different assets that are present in this kind of facility when compared with the healthcare one. To conclude, the implementation of a tag RS to normalise the terms used to create a WO was successful.

Author Contributions: Conceptualization, P.S., M.P. and N.D.; data curation, P.S.; investigation, P.S., M.P., N.D. and J.S.; methodology, P.S., M.P. and N.D.; resources, N.L. and M.R.F.; software, P.S.; supervision, M.P., N.D. and J.S.; validation, P.S., M.P., N.D., J.S., N.L., M.R.F. and N.G.; visualization, P.S.; writing—original draft, P.S., M.P. and N.D.; writing—review and editing, P.S., M.P., N.D., J.S. and N.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the REV@CONSTRUCTION mobiliser project, under the grant LISBOA-01-0247-FEDER-046123 from ANI (National Innovation Agency), and by NOVA LINCS (UIDB/04516/2020) and LASIGE (UIDB/00408/2020) with financial support from FCT (Fundação para a Ciência e a Tecnologia), through national funds. This work contributes to the Strategic Research Plan of the Centre for Marine Technology and Ocean Engineering (CENTEC), which is financed by FCT under contract UIDB/UIDP/00134/2020.

Data Availability Statement: The data is proprietary, confidential, and exclusively owned by TDGI.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BERT	Bidirectional encoder representations from transformers
BN	Bayesian network
CB	Content-based
CF	Collaborative filtering
CMMS	Computerised maintenance management system
CV	Cross-validation
DAG	Directed acyclic graph
GD	Gradient descent
GTP-3	Generative pre-trained transformer
LDA	Latent Dirichlet allocation
LSA	Latent semantic analysis
MAP	Mean average precision
ML	Machine learning
MRR	Mean reciprocal rank
NB	Naive-Bayes
NLP	Natural language processing
RoBERTa	Robustly optimised BERT pretraining approach
RS	Recommender system
WO	Work order

References

1. Gavrikova, E.; Volkova, I.; Burda, Y. Strategic aspects of asset management: An overview of current research. *Sustainability* **2020**, *12*, 5955. [CrossRef]
2. FHWA, F. *Asset Management Primer: Federal Highway Administration*; US Department of Transportation: Washington, DC, USA, 1999.
3. Becker, D.; King, T.D.; McMullen, B. Big data, big data quality problem. In Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 29 October–1 November 2015; pp. 2644–2653. [CrossRef]
4. Sexton, T.; Hodkiewicz, M.; Brundage, M.P. Categorization errors for data entry in maintenance work-orders. In Proceedings of the Annual Conference of the PHM Society, Scottsdale, AZ, USA, 21–26 September 2019; Volume 11. [CrossRef]
5. Navinchandran, M.; Sharp, M.E.; Brundage, M.P.; Sexton, T.B. Discovering critical KPI factors from natural language in maintenance work orders. *J. Intell. Manuf.* **2022**, *33*, 1859–1877. [CrossRef]
6. Song, Y.; Zhang, L.; Giles, C.L. Automatic tag recommendation algorithms for social recommender systems. *ACM Trans. Web (TWEB)* **2011**, *5*, 1–31. [CrossRef]
7. Wu, Y.; Yao, Y.; Xu, F.; Tong, H.; Lu, J. Tag2word: Using tags to generate words for content based tag recommendation. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, Indianapolis, IN, USA, 24–28 October 2016; pp. 2287–2292. [CrossRef]
8. Belém, F.M.; Almeida, J.M.; Gonçalves, M.A. A survey on tag recommendation methods. *J. Assoc. Inf. Sci. Technol.* **2017**, *68*, 830–844. [CrossRef]
9. Krestel, R.; Fankhauser, P.; Nejdl, W. Latent dirichlet allocation for tag recommendation. In Proceedings of the Third ACM Conference on Recommender Systems, New York, NY, USA, 23–25 October 2009; pp. 61–68. [CrossRef]
10. Ding, Z.; Qiu, X.; Zhang, Q.; Huang, X. Learning topical translation model for microblog hashtag suggestion. In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, Beijing, China, 3–9 August 2013.
11. Godin, F.; Slavkovikj, V.; De Neve, W.; Schrauwen, B.; Van de Walle, R. Using topic models for twitter hashtag recommendation. In Proceedings of the 22nd International Conference on World Wide Web, Rio de Janeiro, Brazil, 13–17 May 2013; pp. 593–596. [CrossRef]
12. Goulart, H.X.; Tosi, M.D.; Gonçalves, D.S.; Maia, R.F.; Wachs-Lopes, G.A. Hybrid model for word prediction using naive bayes and latent information. *arXiv* **2018**, arXiv:1803.00985. [CrossRef]
13. Lei, K.; Fu, Q.; Yang, M.; Liang, Y. Tag recommendation by text classification with attention-based capsule network. *Neurocomputing* **2020**, *391*, 65–73. [CrossRef]
14. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
15. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. 2018, *preprint*. Available online: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf (accessed on 26 March 2023).
16. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805. [CrossRef]
17. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692. [CrossRef]
18. Hu, X.; Chu, L.; Pei, J.; Liu, W.; Bian, J. Model complexity of deep learning: A survey. *Knowl. Inf. Syst.* **2021**, *63*, 2585–2619. [CrossRef]
19. Santos, P.; Datia, N.; Pato, M.; Sobral, J. Comparing Word Embeddings through Visualisation. In Proceedings of the IV2022, 26th International Conference Information Visualisation, Vienna, Austria, 19–22 July 2022. [CrossRef]
20. Liu, S.; McGree, J.; Ge, Z.; Xie, Y. Classification methods. In *Computational and Statistical Methods for Analysing Big Data with Applications*; Liu, S., McGree, J., Ge, Z., Xie, Y., Eds.; Academic Press: San Diego, CA, USA, 2016; pp. 7–28. [CrossRef]
21. Cavnar, W.B.; Trenkle, J.M. N-gram-based text categorization. In Proceedings of the SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, USA, 11–13 April 1994; Volume 161175.
22. Korb, K.B.; Nicholson, A.E. *Bayesian Artificial Intelligence*; CRC Press: Boca Raton, FL, USA, 2010; Chapters 3 and 4.
23. Dijkstra, E.W. Recursive programming. *Numer. Math.* **1960**, *2*, 312–318. [CrossRef]
24. Schröder, G.; Thiele, M.; Lehner, W. Setting goals and choosing metrics for recommender system evaluations. In Proceedings of the UCERSTI2 Workshop at the 5th ACM Conference on Recommender Systems, Chicago, IL, USA, 23–27 October 2011; Volume 23, p. 53.
25. Valcarce, D.; Bellogín, A.; Parapar, J.; Castells, P. Assessing ranking metrics in top-N recommendation. *Inf. Retr. J.* **2020**, *23*, 411–448. [CrossRef]
26. Shani, G.; Gunawardana, A. Evaluating Recommendation Systems. In *Recommender Systems Handbook*; Ricci, F., Rokach, L., Shapira, B., Kantor, P.B., Eds.; Springer: Boston, MA, USA, 2011; pp. 257–297. [CrossRef]
27. Silveira, T.; Zhang, M.; Lin, X.; Liu, Y.; Ma, S. How good your recommender system is? A survey on evaluations in recommendation. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 813–831. [CrossRef]

28. Bobadilla, J.; Ortega, F.; Hernando, A.; Gutiérrez, A. Recommender systems survey. *Knowl.-Based Syst.* **2013**, *46*, 109–132. [CrossRef]
29. Meyes, R.; Lu, M.; de Puiseau, C.W.; Meisen, T. Ablation studies in artificial neural networks. *arXiv* **2019**, arXiv.1901.08644. [CrossRef]