Automated Employee Objective Matching Using Pre-trained Word Embeddings

Mohab Ghanem[*], Ahmed Elnaggar[*]
Technische Universität München, Germany
{*mohab.ghanem, ahmed.elnaggar*}*@tum.de*

Adam Mckinnon
QBE Insurance, Australia
adam.mckinnon@qbe.com

Christian Debes, Olivier Boisard
Merck, Germany
{*christian.debes, olivier.boisard*}*@merckgroup.com*

Florian Matthes
Technische Universität München, Germany
matthes@in.tum.de

[*]Both authors contributed equally to this work

***Abstract*—Employee performance objectives are recurrent short-term goals established between an organization and its employees. These objectives are typically aligned with the strategic goals of the company. Human Resources professionals are responsible for supporting employees in achieving their objectives, which in turn mediates enterprise success. One form of such support is grouping employees who have similar objectives in teams so that they can cooperate with one another. This process becomes difficult in large organizations with thousands of employees.**

In this paper, we present a system to assist Human Resources professionals in finding employees having similar objectives. The system automates the process of matching employees based on their common objectives using pre-trained word embeddings and semantic similarity metrics. Even though word embeddings are used to solve a variety of business problems, their effectiveness in searching for similar employee objectives is not sufficiently studied. In accordance with the feedback received from the Human Resources department of a partner company, the system proved to be effective in reducing both the time and effort consumed in the matching process.

***Index Terms*—Employee Objectives, Text Similarity, Document Embedding.**

I. Introduction

Employee performance objectives are personal performance goals that are the focus of an employee during a period of time (e.g., improving teamwork skills). These objectives aim to improve the performance of employees by raising their awareness of their weaknesses and encouraging them to improve their skills, thereby making them more efficient in their core duties. Employees need to understand both the organization's strategic goals and the expertise required to achieve these goals, then, they should formulate their own personal performance objectives in alignment with their organization's objectives [1]. The organization, in turn, should foster an enabling environment for employees to work on their personal performance objectives, and provide them with adequate assistance [2]. It is the responsibility of Human Resources specialists to maintain this environment and assist employees in defining and achieving their personal performance objectives. This assistance can take many forms, including enabling employees to work in teams on their common objectives. Teamwork has proven to magnify the outputs of collaborating members in various studies [3], [4]. It also deepens the acquired skills, and creates a sense of ownership and shared responsibility [5], [6].

Employee objectives are typically represented freely in natural language. As a result, the objectives vary in length and content. This makes the process of matching employees based on their objectives a challenge as the number of employees in an organization grows. Moreover, because of the personal nature of the objectives, employees within the same team can have different goals for their performance. HR specialists will need to match employees from different teams to work on their shared goals. This procedure needs to be repeated frequently because personal employee objectives are inherently periodic. For example, employees in a partner company define new objectives twice a year. This motivates the need for a system that automates this time-consuming and laborious process for HR specialists.

Pre-trained word embeddings are word embeddings trained on large and diverse datasets, such as all the English pages on Wikipedia [7] and BookCorpus [8]. Since a vast amount of knowledge is encoded in the embeddings, they can be directly applied to other tasks without being retrained, while remaining generalizable. This is a form of Transfer Learning, whereby the knowledge acquired during training on one task, can be used in a related task [9]. This paper demonstrates how the process of matching employee objectives can be automated using pre-trained word embeddings.

The general flow of the proposed system consists of two steps: Indexing, and Querying. At indexing time, the system uses pre-trained word embedding models to generate numerical representations of employee objectives, calculates the pairwise similarity between employee objectives, and saves the result in an offline index. At querying time, the system suggests K ***candidate work partners*** for a given ***query employee***, where K is a number chosen by the end-user of the system (typically an HR professional). In an attempt to explain the system suggestions, the system highlights important and common keywords in both the ***query employee***'s objective and the ***candidate work partners'*** objectives.

The system represents an abstract architecture for embedding, indexing, and querying datasets of employee objectives. We evaluate the system based on the use case of a partner company. However, the presented architecture can be used in other organizations with minor modifications. During the evaluation, the system was introduced to the Human Resources department and integrated into the process of searching for work partners based on the commonality of their performance objectives. Eight HR specialists participated in filling out a survey by which we get feedback on the system. They were asked to use the system and respond to a questionnaire where they express their agreement or disagreement with evaluation statements. They were also asked to give open-ended critiques of the system.

The rest of the paper is organized as follows. In Section II, we present background information about word embeddings and their applications in solving business problems. In Section III, we discuss the methodology used to evaluate word embedding models and explain design choices. Section IV provides details on the implementation of system components. In Section V we discuss the conducted evaluation, and in Section VI we portray the results and discuss limitations. Finally, Section VII presents a conclusion to the paper.

II. Background

In Natural Language Processing, word embeddings have been shown to perform better than other methods in various tasks, such as document search, language translation, and sentiment analysis [10], [11]. Among the tasks related to our use case, is the task of estimating the semantic similarity between texts. Word embedding models are inspired by the distributional hypothesis. The premise of the distributional hypothesis is that word representations can be inferred from the context in which those words appear [12]. However, word embeddings have not been studied in the context of finding similar employee objectives yet. In this section, we will discuss works that use word embeddings to solve other business problems. Following that, we will go over word embedding models, from their conception to their evolution.

A. Related Works

In recent works, word embeddings were used by job boards to automatically evaluate whether applicants' profiles match job postings, and to predict the degree to which applicants will succeed at a position [13]–[16]. Similarly and inspired by the idea of embedding words in a vector space, other works extract skills from job descriptions and applicant profiles and map them to their own vector space called skill2vec [17], [18]. In software engineering, semantic text similarity is also applied in information retrieval tasks, whereby queries are expressed in traditional natural language, and results are present in natural or in programming languages [19]. Researchers in a related study [20] attempted to improve information retrieval in software engineering websites by using word embedding algorithms to create a vector space of software engineering documents. These documents can include API documentation, bug reports, and code snippets. The distance between the embedded representations of these documents in the vector space serves as a measure of similarity between their contents. This is used in an information retrieval system, for example in a community question-answering platform to suggest answers containing code to a question in natural language [19].

B. Text Embedding Models

Thanks to the advancements in deep neural networks, many word embedding algorithms are being developed at an accelerated rate. The goal of all embedding models is to project words onto a vector space, such that similar words get nearby vector representations (called embeddings). This is inspired by the distributional hypothesis, which states that the representation of a target word can be estimated from its context [21]. There are different ways in which *targets* and *contexts* can be defined based on the algorithm. In some cases, the *target* is a word and the *context* is the surrounding window [22]. In others, the *target* is a sentence and the *context* is the previous sentence [9]. Thus, we chose to divide the embedding algorithms into two categories: (1) word-based and (2) sentence-based. Word-based embedding algorithms are those trained to learn representation for a single token. While sentence-based algorithms are those trained to learn representation for multiple tokens. This section gives a brief description of the embedding algorithms supported by our system.

1) ***Word-Based:*** Word-based embedding algorithms learn representation for a single word during their training process. At inference time, these models produce vector representation for a single word. Getting a vector representation of a sentence requires aggregating the vectors of its constituent words.

The Neural Probabilistic Language Model [23] is one of the earliest algorithms for word embeddings. The model uses a neural network to estimate the probability of a sequence of words. A language model is created in such a way that likely word sequences get high probabilities, whereas unlikely word sequences get low probabilities [24]. The probability of a sequence of words is represented as the product of the conditional probability of each word in the sequence given its previous words as per equation 1. The resulting language model can be used to get numerical representations of words [23].

$$P(w_1^T) = \prod_{t=1}^{T} P(w_t | w_1^{t-1}) \tag{1}$$

Word2vec [22] is one breakthrough that paved the way for many other advancements. Word2vec comes in two flavors: a neural network that learns to predict a word given its context (Continuous Bag of Words) and a neural network that predicts the context given a word (Skip-gram). In both cases, it uses the parameters learned by the network as a vector representation for the target word.

The previous two models calculated only one numerical representation for a word regardless of its position within the sentence. However, a word can have different meanings as it appears in different contexts. ELMo [25] tackles this problem by introducing the concept of contextualized word embeddings, that is, a word can have multiple representations depending on its context. ELMo uses a bidirectional LSTM [26] to learn the embeddings of a word given both its previous and future contexts. However, for the LSTM to work, it has to pass through the text sequentially, which means that ELMo can not be trained on multiple processors in parallel. The research community then introduced the Transformer architecture. Transformers do not require the input to be fed sequentially because they are based on the attention model [27]. BERT [9] is a popular transformer architecture that takes into account left and right contexts, it is trained using masked language modeling, and next sentence prediction.

BERT suffers from some problems regarding allocated memory during training and rapid growth of training time complexity. ALBERT [28] is an improvement over BERT that addresses these problems by implementing parameter reduction techniques and allows for better scalability. DistilBERT [29] is another attempt to overcome the memory consumption and training time problems introduced by BERT. It uses knowledge distillation to reduce the number of parameters required in BERT by 40% while retaining 97% of BERT's performance, in addition to being 60% faster. Aiming for further improvement, the authors of RoBERTa [30] examine the usefulness of many of the design choices adopted by BERT. They remove training on next sentence prediction and train RoBERTa only on masked language models, they augment the masked language model of BERT with the idea of dynamic masking, where the masked tokens change during training. Finally, they report better performance on multiple downstream tasks.

OpenAI GPT [31] is a language model based on unidirectional transformers, contrary to BERT which employed bidirectional transformers. GPT2 [32] is a direct scale-up of GPT. It is also based on a unidirectional transformer, but trained with more than 10 times the number of parameters in GPT, and trained on more than 10 times the size of data.

XLNet [33] is another transformer model that aims to overcome the drawbacks of BERT's masked language model training, and introduces the idea of permutation language modeling, where all tokens are masked and predicted in random order. The authors of XLNet report that their work improves upon BERT in 20 NLP tasks. XLM [34] is a different transformer that is trained by masked language modeling. It uses corresponding sentences from different languages to learn relations between words in those languages. This also allows XLM to share vocabulary and context between languages and learn new inherent relations.

The Reformer [35] is a transformer architecture that focuses on modeling long sequences of text while remaining memory efficient. The Reformer supports up to 1M tokens in the input while reducing space complexity from $O(L^2)$ to $O(L\log(L))$, where L is the length of the sequence.

Previous efforts have mainly focused on increasing the number of data points, increasing the size of the model, or training for longer durations. On the contrary, ELECTRA [36] improves performance using the same model size and the same amount of data by adding a discriminator layer after the masked language model generator. An adversarial game is played between the generator and the discriminator, where the goal of the generator is to replace the masked tokens with the most likely token values, and the goal of the discriminator is to validate the quality of the replaced values. The authors of ELECTRA report that they outperform BERT, GPT, RoBERTa, and XLNet while training using the same model size, amount of data, and training duration.

The T5 model [37] - short for Text-To-Text Transfer Transformer - is another transformer model that differs from all the previously discussed transformer models in the way it learns. The model treats a wide variety of NLP tasks as simply taking input text and predicting output text. For example, for the task of sentiment analysis, the input text is a sentence, and the output text is the semantic polarity expressed as text. Another example is machine translation, the input text is a sentence in the source language, and the output text is the translated sentence in the target language. This approach enables the T5 model to perform many NLP tasks. It also enables the model to combine the information it learns from all the NLP tasks it is trained on.

*2) **Sentence-Based:*** Sentence-based algorithms are designed to produce embeddings that encode the semantics of a complete sentence or a document. Even though the number of sentence-based embeddings is less than that of word-based embeddings, they appear to be on the rise because they are easier to use for embedding multiple tokens. The Universal Sentence Encoder (USE) [38] is trained to model word sequences. It is available in two different implementations, either as a Transformer Encoder or as a Deep Averaging Network Encoder.

Lastly, the most recent work at the time of writing this paper to the best of our knowledge is Sentence transformers [39], [40] which enhances the previously mentioned transformer models, such as BERT and DistilBERT, to model word sequences instead of single words. The models are fine-tuned in such a way that sentences of similar meanings get close embeddings in vector space. The authors of Sentence Transformers report achieving state-of-the-art performance in multiple NLP tasks including semantic text similarity.

III. Methodology

The purpose of this paper is to present a system that assists HR specialists in searching for employees with similar performance objectives. The system uses pre-trained embedding models to project employee objectives onto an embedding space, and uses the distance between the projected objectives to estimate the semantic relatedness between the textual objectives. In order to provide our end-users with a seamless experience, we compare the models mentioned in Section II-B and select the best performer as the default settings in our system. We begin this section by discussing the properties of employee objectives data that is obtained from a partner company. Then, we present a comparison of the most popular pre-trained word embedding models and justify our choice of default settings.

A. Employee Objectives Data

For the purpose of understanding the properties of employee objectives data, we obtained anonymized historical employee objectives from a partner company. The dataset was received in *csv* format, and went through the following preprocessing steps: (1) Removing missing values and duplicate rows. (2) Removing useless columns and combining multiple columns that constitute the objective into a single column. (3) There existed multiple objectives for the same employee in the original data we received. However, we only chose the longest objective as the unique objective of an employee, because the current implementation of our system requires a 1:1 mapping between an employee and their objective. The processed data consisted of 512 employees along with their corresponding objectives, and had an average number of 90 words per objective.

B. Comparison of Pre-trained Models

TensorFlow Hub[1], Hugging Face Transformers[2], and Sentence Transformers[3] are three portals that provide pre-trained text embedding models with various configurations. Table I provides an overview of the models included in our comparison. Models are listed in the table along with their type (word-based or sentence-based), source, and the length of the embedding vector generated by the model. We compare the performance of these models on two tasks: (1) Semantic representation ability of the model i.e. projecting semantically similar sentences to nearby vectors in the embedding space. (2) Inference time i.e. the time taken by the model to generate a vector representation of a given sentence.

Semantic Representation Ability: To the best of our knowledge, there are no publicly available datasets with employee objectives labeled by their semantic relatedness. Nevertheless, there are labeled datasets for evaluating the semantic similarity of text in general, among those is SICK dataset [41]. The dataset contains 9927 English sentence pairs, along with their semantic relatedness scores ranging from 1 to 5, with 1 being strongly unrelated and 5 being strongly related[4]. The dataset was designed to serve as a benchmark for semantic similarity estimation and was indeed used in Task 1 of SemEval-2014 [5].

However, SICK has an average number of 9 words per sentence, which is 10 times less than the average for the employee objectives dataset. As a result, we will only use SICK to choose the default settings of our system. Furthermore, we will conduct an end-user evaluation with real-world employee objectives in order to manually validate the results. This is discussed more in Section V.

We use SICK for our evaluation and evaluate each pre-trained model in table I against the true scores from the dataset. The evaluation process for a single pre-trained model is described in algorithm 1. Namely, we use the model to project each sentence pair onto an embedding space (getting the embedding vector of a sentence using a word-based embedding model is done by averaging the individual embeddings of its constituent words). The semantic relatedness between the two sentences is then estimated by calculating the cosine similarity between their embedding vectors (Equation 2).

Algorithm 1 Evaluating a Pre-trrained Model

```
1: $true \leftarrow$ true relatedness scores list. Items $\in [1,5]$
2: $predicted \leftarrow$ empty list
3: for each $sentence\_pair$ do
4:     $u \leftarrow$ embedding of first sentence in $sentence\_pair$
5:     $v \leftarrow$ embedding of second sentence in $sentence\_pair$
6:     $relatedness \leftarrow Cosine\_Similarity(u, v)$
7:     $predicted.Insert(relatedness)$
8: end for
9: scale $predicted$ such that items $\in [1,5]$
10: $model\_score \leftarrow Correlation(true, predicted)$
```

$$Cosine\_Similarity(u, v) = \frac{u \cdot v}{\|u\|_2 \|v\|_2} \tag{2}$$

After repeating the previous for all sentence pairs in the dataset, we obtain a list of predicted relatedness scores, this list is then scaled to have the same limits of the true scores.

The score which reflects the model's ability to project similar sentences to nearby vectors in the embedding space is obtained by calculating the correlation between the true scores and the predicted scores, where correlation is defined per equation 3. This algorithm is repeated for all pre-trained models mentioned previously in table I. The results of the comparison are portrayed in the upper part of Fig. 1 where models achieving correlation close to 1 perform better. We can observe that sentence-based models are achieving a higher correlation with the truth.

$$Correlation(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3}$$

Inference Time: In Fig. 1 we present a comparison of the average time taken by each model to generate embedding vectors for batches of 50 sentences from SICK dataset. Choosing a pre-trained model with adequate inference time is essential to make our system scalable. As the number of objectives grows, there will be a significant difference between the time taken

[1]https://www.tensorflow.org/hub
[2]https://huggingface.co/transformers/
[3]https://github.com/UKPLab/sentence-transformers
[4]https://zenodo.org/record/2787612#.YMTEqPkzZAg
[5]https://alt.qcri.org/semeval2014/

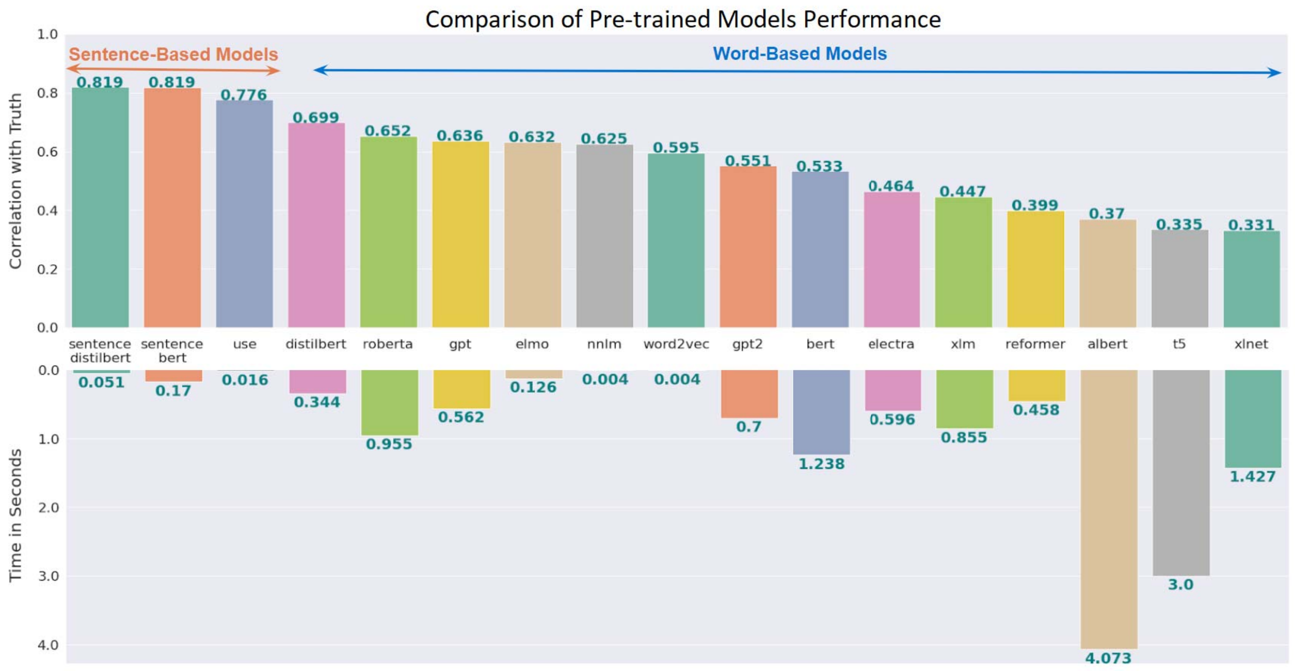


Fig. 1. Evaluation of pre-trained models in terms of correlation with true labels of SICK (Top), and average time consumed - in seconds - to generate embeddings for batches of 50 sentences from SICK (Bottom).

TABLE I
PRE-TRAINED WORD EMBEDDING MODELS SUPPORTED BY OUR SYSTEM

Model	Type	Source	Vector Length
NNLM	word-based	TensorFlow Hub	128
Word2vec	word-based	TensorFlow Hub	500
ELMo	word-based	TensorFlow Hub	1024
BERT	word-based	HuggingFace Transformers	1024
ALBERT	word-based	HuggingFace Transformers	4096
DistilBERT	word-based	HuggingFace Transformers	768
RoBERTa	word-based	HuggingFace Transformers	1024
GPT	word-based	HuggingFace Transformers	768
GPT2	word-based	HuggingFace Transformers	768
XLNet	word-based	HuggingFace Transformers	1024
XLM	word-based	HuggingFace Transformers	2048
Reformer	word-based	HuggingFace Transformers	256
ELECTRA	word-based	HuggingFace Transformers	256
T5	word-based	HuggingFace Transformers	512
Universal Sentence Encoder (USE)	sentence-based	TensorFlow Hub	512
Sentence BERT	sentence-based	Sentence Transformers	768
Sentence DistilBERT	sentence-based	Sentence Transformers	768

by a fast-inference and a slow-inference model. For example, processing 10K batches of SICK with *Sentence DistilBERT* takes 8.5 minutes, compared to 11 hours with *ALBERT*.

The performance of the pre-trained models on the previous two tasks leads us to select *Sentence DistilBERT* as the default embedding model in our system as it achieves the highest correlation with the true labels - in conjunction with *Sentence BERT*. However, *Sentence DistilBERT* has a faster inference time.

IV. Implementation

This section discusses the implementation of system components and provides a high-level workflow of how the system is used by HR personnel. The system consists mainly of three components: (A) Embeddings core (B) API server (C) User interface. The general system flow, as depicted in Fig. 3 can be summarized in two main paths: Indexing path, and Querying path.

During **Indexing** (red in Fig. 3) the HR specialist uses the user interface to upload *employee objectives*, along with a number N. The objectives file is a two-column *csv* file, the first column being *employee id* and the second being *employee objective*. As for N, it represents the maximum number of employees that the system can suggest for a given employee. The Embeddings core receives the objectives, and projects them onto the embedding space. It then calculates the pairwise similarity between those embeddings, and creates an *offline index* to be used at query time. The offline index is basically a map between an employee ID and the IDs of the most similar N employees. Since this computation adds latency to the system, creating embeddings and similarity indices is not performed on the fly.

During **Querying** (blue in Fig. 3) the HR specialist uses the user interface to query the system by sending an employee ID (called ***query employee***), along with a number $K \leq N$. The system searches the offline index for the employee ID and returns a list of K employees who are similar to the one sent by the HR specialist (called ***candidate work partners***).

The next subsections describe the implementation of the *Embeddings Core* and the explainability features of the *User Interface*.

A. Embeddings Core

The embeddings core contains the main functionality of the system. It is responsible for the whole process of generating embeddings from objectives and finding ***candidate work partners*** with the most similar objectives to a ***query employee***. The default settings for the Embeddings Core is to use *Sentence DistilBERT* embeddings as motivated in Section III-B.

During indexing: The Embeddings Core reads the employee objectives data, generates numerical embeddings, and saves these embeddings to an HDFstore. The HDFstore is partitioned into chunks, each chunk contains a list of objective embeddings, along with the corresponding employees' IDs as depicted in Fig. 4. The HDF-Store and the chunking allow us to compute pairwise similarity without loading all of the embeddings in memory at once.

After populating the HDF-Store, the Embeddings Core calculates the pairwise similarity of the generated embeddings using algorithm 2. This is accomplished by loading two chunks in memory at the same time and calculating the pairwise similarity between all employees in those chunks. Meanwhile, a min-heap of size N is maintained for each employee, called the ***employee-specific heap***. The heap is used to save the most similar N ***candidate work partners*** to its specific employee along with their similarity score, this is shown in Fig. 5. As the system iterates over all two-chunk combinations, an ***employee-specific heap*** will be updated when the system finds a candidate that is more similar than the least similar candidate inside the heap (hence the min-heap). The system keeps track of all ***employee-specific heaps*** in a map that takes the form: $employee\ id \rightarrow employee\ specific\ heap$. This map is called the ***offline index***.

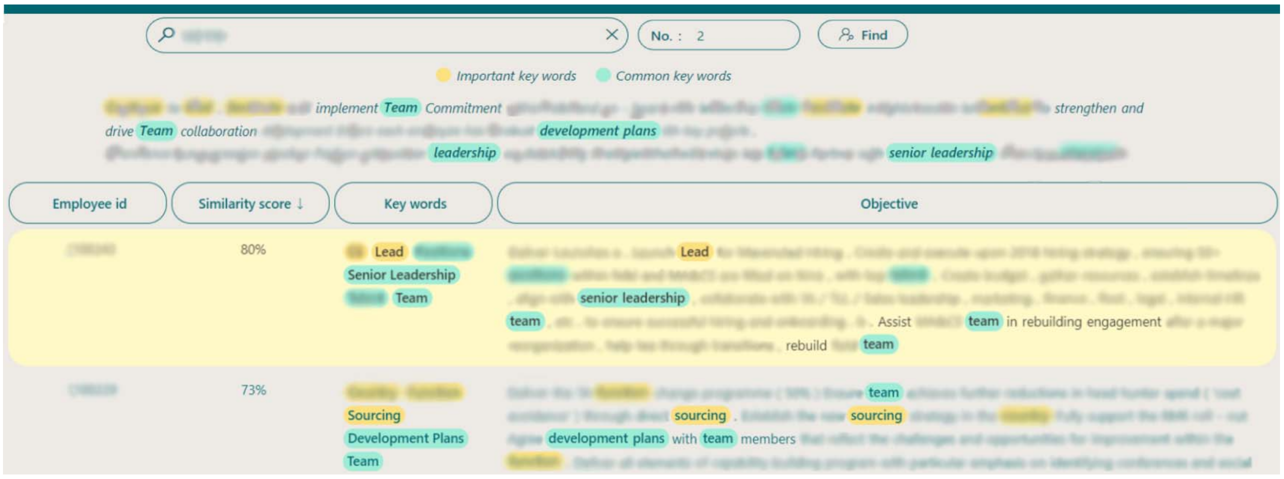
Algorithm 2 Pairwise Similarity Calculation

```
1: $heaps \leftarrow$ map of empty employee-specific heaps (offline
   index)
2: for each ($chunk_i$, $chunk_j$) combination in HDF-Store do
3:     for each embedding $u$ in $chunk_i$ do
4:         for each embedding $v$ in $chunk_j$ do
5:             $similarity \leftarrow CalculateSimilarity(u, v)$
6:             if $similarity > heaps$(employee $u$)$.min$ then
7:                 $heaps$(employee $u$)$.add(v)$
8:             end if
9:         end for
10:    end for
11: end for
```

During querying: The Embeddings Core receives a ***query employee*** ID and a number $K \leq N$ from the user. Using the ID, it loads the ***employee-specific heap*** from the ***offline index***, and returns the K employees having the highest similarity from the heap as ***candidate work partners***.

B. User Interface

The user interface allows the user to use the functionality of the system and to see the suggested ***candidate work partners*** along with their estimated similarity scores. Explainability features are presented along with the suggestions in order to help the HR specialist understand why the system is making these recommendations. These features include two types of highlighted words: common keywords, and important keywords. **(1) Common keywords** are simply some of the words that are common between the objective of the ***query employee***, and the objectives of the suggested ***candidate work partners***, these words are highlighted in cyan. **(2) Important keywords** are words that are essential to understanding a specific objective; technically, they are uni-grams and bi-grams with high TF-IDF scores, these are highlighted in yellow. In order to visually demonstrate these features, the employee objectives data cannot be directly used because of



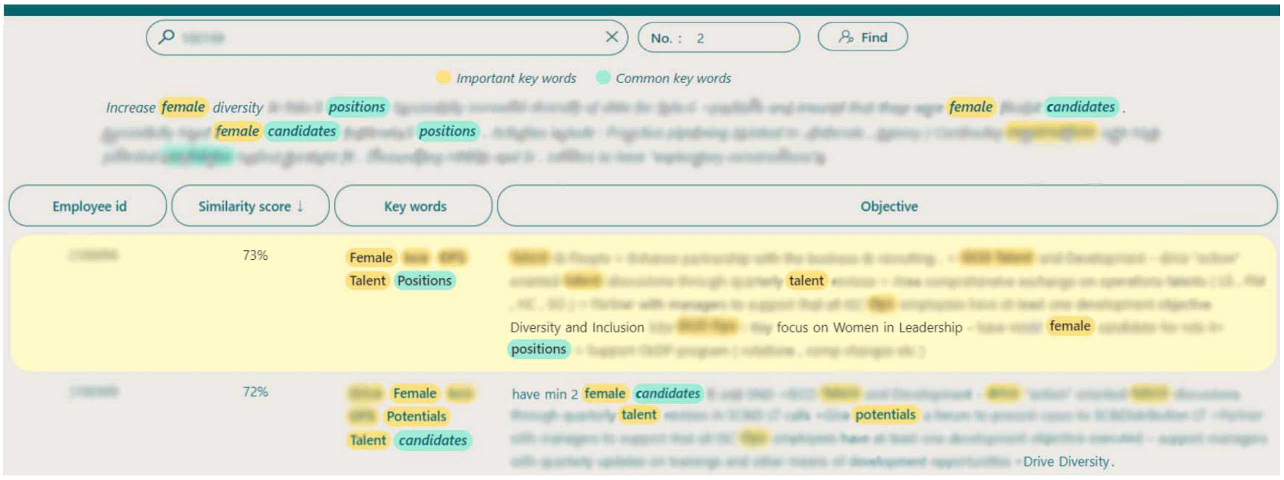
No. : 2 | Find

● Important key words ● Common key words

[illegible] implement Team Commitment [illegible] strengthen and drive Team collaboration [illegible] development plans [illegible] leadership [illegible] senior leadership [illegible]

Employee id	Similarity score ↓	Key words	Objective
[illegible]	80%	Lead Senior Leadership Team	[illegible] Lead [illegible] senior leadership [illegible] team [illegible] Assist [illegible] team in rebuilding engagement [illegible] rebuild [illegible] team
[illegible]	73%	Sourcing Development Plans Team	[illegible] team [illegible] sourcing [illegible] sourcing [illegible] development plans with team members [illegible]

(a)



No. : 2 | Find

● Important key words ● Common key words

Increase female diversity [illegible] positions [illegible] female [illegible] candidates . [illegible] female candidates [illegible] positions [illegible]

Employee id	Similarity score ↓	Key words	Objective
[illegible]	73%	Female Talent Positions	[illegible] talent [illegible] Diversity and Inclusion [illegible] focus on Women in Leadership [illegible] female [illegible] positions [illegible]
[illegible]	72%	Female Potentials Talent candidates	have min 2 female candidates [illegible] talent [illegible] potentials [illegible] Drive Diversity.

(b)

Fig. 2. Two examples of system output using blurred performance objectives. HR specialists enter ***query employee*** ID in the search box, along with the number of candidates to return, Three objectives appear in each example. The topmost represents the objective of the ***query employee***, the table below shows the objectives of two ***candidate employees***, along with their *employee id*, *similarity score*. Keywords are highlighted for explainability. Objectives in part (a) revolve around teams, they could belong to employees working on improving their teamwork and leadership skills. Objectives in part (b) could apply to recruiters and hiring managers working on increasing the diversity in their teams.

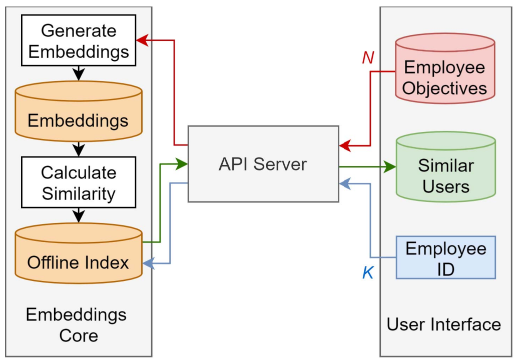


Fig. 3. General system flow and main components. Indexing path is shown in red, querying path in blue and system replies in green.

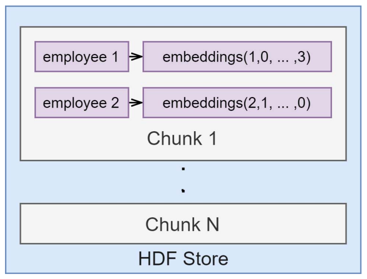


Fig. 4. Embeddings saved by the embeddings generator and organized into the HDF-Store.

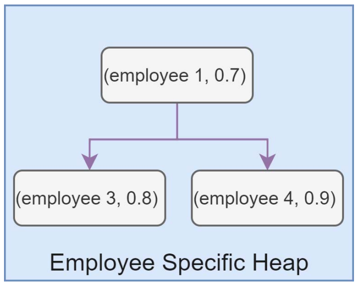


Fig. 5. Employee-specific heap. Keeps track of most similar N employees along with their similarity score.

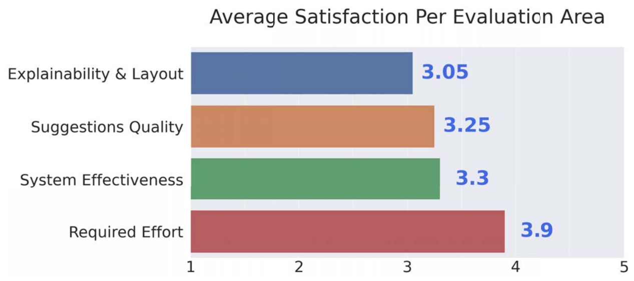


Fig. 6. Scores of the evaluation survey grouped by evaluation area.

its confidentiality. Nevertheless, to give readers a sense of what the results of the original data should look like, we provide two examples in Fig. 2 with blurred objective text except for some limited keywords. The figure shows two possible matches done by the system. The evaluation of the explainability features by the end-users is discussed in section V.

V. Evaluation

In this section, we review how the system evaluation was conducted and highlight the main outcomes and limitations according to the feedback we received. In light of the similarity between our system and recommender systems, and because we are seeking feedback from the end-users, we decide to evaluate the system from a user-centric perspective. The user-centric evaluation paradigm measures subjective feedback from users, such as how useful the system is, how good the suggestions are, and how inclined the users are to adopt the system [42]. This paradigm is emerging for recommender systems evaluation as it emphasizes how the users perceive the system's features, which would otherwise be rendered useless.

User-centric evaluations are mainly conducted using questionnaires. We designed our questionnaire following the guidelines of user-centric evaluation frameworks of recommender systems [42]–[44]. We adopted related elements from those frameworks and modified the questions when necessary to reflect the specifics of our system. The questionnaire (Appendix A) comprised 15 statements evaluating four areas of interest: (1) Explainability and Layout (2) Suggestions Quality (3) System Effectiveness (4) Required Effort. Participants were asked to read each statement and express their opinion on a five-point scale from strongly disagree to strongly agree, they were also required to give open-ended feedback.

VI. Results and Discussion

Eight randomly selected HR personnel from a partner company (possible end-users of the system) were asked to use the system and respond to the questionnaire (Appendix A) they were also asked to give general open-ended feedback. A summary of their responses to the questionnaire grouped by focus area is illustrated in Fig. 6. The evaluation area **Required Effort** received the highest average score and the feedback indicated the system was easy to use. As for the second and third scores, they pertain to **System Effectiveness** and **Suggestions Quality**, which both received relatively good feedback. Lastly, **Explainability and Layout** received the lowest satisfaction scores; feedback indicated that the color coding was confusing for users and the explainability features were not informative.

A. Findings

According to the feedback from HR specialists, the system is a promising first step towards automating the process of matching employees with work partners who share similar objectives. When the suggestions perfectly match the ***query employee***, the system performs the required task efficiently. Even when the best candidates aren't immediately suggested, it still minimizes the search space and reduces the amount of manual work HR specialists would have to do otherwise.

B. Limitations

The open-ended evaluations revealed some of the limitations of our system. Concerning the **common keywords** feature, the captured words were indeed common between the ***query employee***'s objective and the objectives of the ***candidate work partners***. However, they may or may not represent significant shared goals. The captured words as shown in Fig. 2 had general meanings like *team* and *female*, which requires further validation by the HR specialist to make sure they convey the meaning they are looking for. As for the **important keywords** feature, the users suggested going beyond uni-grams and bi-grams and emphasize larger n-grams that have high importance in understanding the objectives.

Another limitation of the system is that it only supports having one objective per employee. Occasionally, employees can have multiple objectives and can be part of several non-overlapping co-working groups at once. These scenarios are not possible with the current design of our system. It is possible to work around this problem by adding another entry with a different employee ID to the data, but this is difficult to maintain and can result in undesirable errors. Moreover, It burdens the end-user with additional manual work.

Finally, the choice of *Sentence DistilBERT* as the default model was made based on evaluating the accuracy of the models using SICK dataset, which differs in length and word distribution from employee objectives data. This could mean that *Sentence DistilBERT* may not be the best candidate for embedding employee objectives. However, it is still among the best candidates given the large gap between sentence-based embeddings and word-based embeddings illustrated in Fig. 1.

C. Future Work

An important missing feature of the system is the ability to search by objective text. Currently, the system can only index employees using their IDs and retrieve employees who have similar objectives to a given employee. The ability to search through objective texts for given keywords, and retrieve employees accordingly, will be a great addition to the tools at the HR professional's disposal. As a result, they will have a better understanding of how different objectives are distributed within the company, enabling them to check if the spectrum of employee objectives aligns with the goals of the organization.

VII. Conclusion

The ability of employees to achieve their objectives is among the factors contributing to an organization's success, therefore, organizations need to create an environment that empowers employees to accomplish their goals [2]. Teamwork is one of the most powerful tools for success, and organizations are encouraged to allow employees to work in teams [3], [5]. The aim of this paper is to present a system to assist HR specialists in grouping employees with similar objectives so that they can work together toward achieving their shared goals. The system utilizes recent advances in Natural Language Processing and Deep Learning to tackle this problem. Namely, the system uses pre-trained word embedding models to project the textual employee objectives to numerical vector embeddings, and defines the similarity of the objectives of two employees as the closeness of their embeddings in vector space. While word embeddings have been used in solving numerous business problems, studies on using word embeddings to find similar employee objectives are lacking.

The system allows HR personnel to load employee objectives data into the system, and later ask the system to suggest work partners for a specific employee. The system then provides a list of suggestions with their similarity scores, and highlights keywords in the objectives so the user will understand the reasoning behind these decisions.

By means of a questionnaire, eight members from the human resources department of a partner company evaluated the system. The questionnaire is created in accordance with guidelines of popular recommender system evaluation frameworks [42], [43]. In the evaluations, the system was determined to be of good help to the HR personnel. The system, however, has some limitations, for example, it currently allows an employee to have only one objective. Additionally, most users did not find the explainability features to be very informative. Another future improvement to the system is the ability to search through employee objectives given a custom set of keywords, which can provide HR professionals with more insights into the objectives of employees in their company. Finally, the system represents a promising start in automating the manual process of finding employees with shared objectives and highlights some of the open challenges.

References

[1] Wendy Boswell. Aligning employees with the organization's strategic objectives: Out of 'line of sight', out of mind. *The International Journal of Human Resource Management*, 17(9):1489–1511, 2006.

[2] Radoslaw Nowak. Does employee understanding of strategic objectives matter? effects on culture and performance. *Journal of Strategy and Management*, 2020.

[3] Sheikh Raheel Manzoor, Hafiz Ullah, Murad Hussain, Zulqarnain Muhammad Ahmad, et al. Effect of teamwork on employee performance. *International Journal of Learning and Development*, 1(1):110–126, 2011.

[4] Peter J Sloane, Melanie K Jones, Richard J Jones, and Paul L Latreille. Training, job satisfaction and establishment performance. 2007.

[5] Nicolas Bacon and Paul Blyton. The impact of teamwork on skills: Employee perceptions of who gains and who loses. *Human Resource Management Journal*, 13(2):13–29, 2003.

[6] J Richard Hackman and Greg R Oldham. Motivation through the design of work: Test of a theory. *Organizational behavior and human performance*, 16(2):250–279, 1976.

[7] English wikipedia https://en.wikipedia.org/wiki/english_wikipedia. [Online; accessed 21-August-2021].

[8] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

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

[10] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, 2013.

[11] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[12] Thomas K Landauer and Susan T Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997.

[13] Jianbo Yuan, Walid Shalaby, Mohammed Korayem, David Lin, Khalifeh AlJadda, and Jiebo Luo. Solving cold-start problem in large-scale recommendation engines: A deep learning approach. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1901–1910. IEEE, 2016.

[14] José Luís Fava de Matos Pombo. *Landing on the right job: a machine learning approach to match candidates with jobs applying semantic embeddings.* PhD thesis, 2019.

[15] Steffen Schnitzer, Dominik Reis, Wael Alkhatib, Christoph Rensing, and Ralf Steinmetz. Preselection of documents for personalized recommendations of job postings based on word embeddings. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1683–1686, 2019.

[16] Francis C Fernández-Reyes and Suraj Shinde. Cv retrieval system based on job description matching using hybrid word embeddings. *Computer Speech & Language*, 56:73–79, 2019.

[17] Le Van-Duyet, Vo Minh Quan, and Dang Quang An. Skill2vec: Machine learning approach for determining the relevant skills from job description. *arXiv preprint arXiv:1707.09751*, 2017.

[18] Tak-Lam Wong, Haoran Xie, Fu Lee Wang, Chung Keung Poon, and Di Zou. An automatic approach for discovering skill relationship from learning data. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, pages 608–609, 2017.

[19] Xiaodong Gu, Hongyu Zhang, and Sunghun Kim. Deep code search. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 933–944, 2018.

[20] Xin Ye, Hui Shen, Xiao Ma, Razvan Bunescu, and Chang Liu. From word embeddings to document similarities for improved information retrieval in software engineering. In *Proceedings of the 38th international conference on software engineering*, pages 404–415, 2016.

[21] John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[23] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[24] Dengliang Shi. A study on neural network language modeling. *arXiv preprint arXiv:1708.07252*, 2017.

[25] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017.

[28] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[29] Victor Sanh. Smaller, faster, cheaper, lighter: Introducing distilbert, a distilled version of bert. *Medium (blog). August*, 28:2019, 2019.

[30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[31] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[32] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[33] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

[34] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

[35] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[36] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[38] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[39] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[40] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2020.

[41] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. A sick cure for the evaluation of compositional distributional semantic models. In *Lrec*, pages 216–223. Reykjavik, 2014.

[42] Pearl Pu, Li Chen, and Rong Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 157–164, 2011.

[43] Bart P Knijnenburg, Martijn C Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction*, 22(4):441–504, 2012.

[44] Pearl Pu, Li Chen, and Rong Hu. Evaluating recommender systems from the user's perspective: survey of the state of the art. *User Modeling and User-Adapted Interaction*, 22(4-5):317–355, 2012.

Appendix

A. Evaluation Questionnaire

The following points comprise the questionnaire used to evaluate our system. They are inspired by user-centric evaluation frameworks for recommender systems [42]–[44]. After using the system, participants were asked to rate their agreement with the statements below on a scale of 1-5, with 1 being strongly disagreed and 5 being strongly agreed.

*1) **Explainability and Layout**:* The purpose of this section is to evaluate the explainability features and the interpretability of system results.

- The explanation facilities of the system help me make better decisions.
- The explanation facilities helped increase my acceptance of the recommendations made by the system.
- The system provides information at the right level of detail.
- I was easily able to identify the work partner candidate who is most similar to my search.

*2) **Suggestions Quality**:* The purpose of this section is to evaluate the quality of the suggestions as perceived by the user. Quality is assessed in terms of relevance, variety, and user satisfaction.

- The recommended work partners were relevant to my search.
- The system showed too many bad suggestions.
- I am satisfied with the suggested work partners.
- The system educated me about the similarity and variety of employee objectives in the organization.

*3) **System Effectiveness**:* The purpose of this section is to evaluate the effectiveness of the system in the process of finding employees with similar objectives.

- The system saves me time looking for work partners.
- I can find better work partners without the help of the system.
- The system improves the process of matching employees with similar objectives.
- I would use the suggestions of the system to connect work partners.

*4) **Required Effort**:* The purpose of this section is to measure the level of effort required by users to use the system.

- I have to invest a lot of effort in the system.
- The system provides results quickly.
- I find the system easy to use.