1 Summary

Knowledge graph uses the graph structure to capture the knowledge of the world. Recently, proposed methods show that incorporating time information into knowledge graphs can further improve the predicted results, namely, temporal knowledge graph embedding (TKGE). For instance, DE-SimplE extends SimplE method to embed time information into entities, thereby improving the performance. There are some downstream tasks of knowledge graphs that are widely used, such as recommender systems and Question-Answering systems. To improve the reliability of these systems, high accuracy, and robustness are required. Moreover, ensemble learning is a technique widely used in machine learning to improve overall performance by combining the predictions of multiple models. And deep learning methods achieved remarkable success in various domains. Thus, this project aims to combine the technique of ensemble learning and deep learning to further improve the performance of TKGE models doing link prediction.

The knowledge graph consists of facts, represented as (h, r, t) where h is the head of the fact, r is the relation of the fact, and t is the tail of the fact. In temporal knowledge graphs, the time information is normally represented as τ, thereby, the fact in temporal knowledge graph is represented as $(h, r, ?, \tau)$. Link prediction utilizes the embedding methods to predict the new relations in knowledge graphs or temporal knowledge graphs, generally, it can predict the head, tail, and timestamp as well. The process of link prediction can be illustrated by an example. For instance, a query (h, r, t, τ) as input feed into TKGE model. Firstly, the TKGE model generates a number of simulated facts by taking each entity into the query to fill out the query. Then, the model learns a score for each simulated fact which is named the simulated score. Finally, all the simulated facts are ranked by the simulated score.

Ensemble learning can be used to further improve the rank of the correct answer. The principle of ensemble learning technique is to combine the results from distinct models to enhance the final prediction. Generally, it can be classified into three main categories, namely, bagging, boosting, and stacking. The ensemble method used in this project is inspired by stacking. The idea behind stacking is that the meta-model utilizes the outputs of base models to learn further and predict the final results. In this project, several distinct TKGE models are trained individually to capture the different aspects of the data, then the neural network technique is utilized for further learning. However, instead of outputting the final prediction directly, the neural network model predicts the weights for each base model. Then the ensemble score is obtained by calculating the sum of the product of the weight and the initial prediction. Based on the ensemble score, the simulated facts are re-rank, thus, the correct answer is assigned a new rank which is highly chance better than the original rank. The input of the neural network is the simulated score for top predictions which is the predicted answer from each individual TKGE model. The target of the neural network is the rank of the correct answer. Before training, the normalization technique is used to help the network learn better from the dataset. In addition, the hyperparameter technique such as early stopping is used for finding the best neural network model.

Three experiments are conducted in this project.

- Each individual TKGE model is re-predicted based on the same dataset as other experiments, namely DE-SimplE, DE-DistMult, DE-TransE, ATiSE, and TERO.
- To demonstrate that the neural network can learn better weights than other methods, conducted the grid search experiment as the baseline method.
- Conducted two neural network experiments that have different numbers of inputs, for 5 inputs, it takes the simulated score of the top 1 prediction for each TKGE model, and for 25 inputs, it takes the simulated score of the top 5 predictions for each TKGE model.

The results show that the neural network-based ensemble model not only outperforms all the individual models but also from grid search-based ensemble model. In addition, a different number of features as input for neural networks has limited effects on the performance. Moreover, in the case that only predicts one element of a fact (predicting only head/relation/tail/timestamp), the neural network-based ensemble model improves better when predicting tail and timestamp elements.

Most ensemble methods can only learn weights for individual models, however, relying on the benefit of the neural network, the neural network-based ensemble model predicts weight for each query of each model. As a result, the weight assignment is more accurate, also, within a relatively short time. In addition, this research improves ensemble learning by combining the deep learning technique with the ensemble technique, to the best of our knowledge, it is used for the first time in TKGE model ensemble.

Since the result shows that neural network-based ensemble models are able to improve the prediction of tail and timestamp elements, it is worth having further research on the elementwise neural network architecture. In addition, in the experiment of this project, the number of input features has limited effects on the performance of ensemble model, thus, it could explore a more complex deep learning method, such as Long short-term memory which can capture the sequence information. On the

other hand, different inputs are also worth exploring, such as the embedding expression of the query, which contains more information than simulated scores.

Deep learning-based ensemble for temporal knowledge graph embedding in link prediction

Ning An
nan21@student.aau.dk
Aalborg University
Aalborg Øst, Denmark

Abstract

Question Answering (QA) has become a hot topic since the proposal of chatGPT. The QA system based on Knowledge Graph (KG)s or Temporal Knowledge Graph (TKG)s relies heavily on the graph structure and link prediction. To improve the performance of link prediction, this paper focuses on the investigation of ensemble learning techniques. Since deep learning-based methods can dynamically assign weights for each model and provide more accurate weights than other methods, it is investigated in this paper to predict weights as the hyperparameters of ensemble models, specific to TKGs. Two neural network architectures are proposed to predict ensemble weights and evaluated based on six metrics. To analyze the results, it is found that the neural network-based ensemble model not only outperforms all the individual models but also the grid search-based ensemble model. ATiSE has the best performance across all base models. However, after the grid search-based ensemble, the ensemble model performed 33% better than ATiSE in HITS@1. Moreover, after the neural-network-based ensemble, the ensemble model performed 8% better than the grid search-based ensemble model in HITS@1. Moreover, it is worth noticing that in the case that only predicts one element of a fact (predicting only head/relation/tail/timestamp), the neural network-based ensemble model improves better when predicting tail and timestamp elements.

Keywords: Ensemble Learning, Temporal Knowledge Graph Embedding, Neural Networks, Grid Search

2 Introduction

KG is a multi-relational graph composed of entities and relations, where entities are represented as nodes and relations are represented as edges [30]. They have been widely researched and used for some downstream tasks during the last decade, such as link prediction which typically refers to the task of predicting an entity that has a specific relation with another given entity [30]. TKG [20] is KG with time information annotated. It is a relatively new field that draws the attention of researchers in recent years. The widely used TKG datasets are Wikidata [20], YAGO [28], FreeBase [3], and ICEWS [10].

Typically, the information and structure of KG and TKG are represented as embedding for further applied to downstream tasks. Knowledge Graph Embedding (KGE) [30] is the technique to embed components of a KG, including entities and relations, into continuous vector spaces. This technique simplifies the manipulation and preserves the inherent structure of the KG. However, the relation between entities will change over time, Temporal Knowledge Graph Embedding (TKGE) [10], which utilizes TKG as datasets, embeds time information into embeddings in order to improve the accuracy of prediction for downstream tasks.

Since to apply the downstream tasks, such as link prediction and QA [35] systems, into reality, a highly accurate prediction is demanded. Thus, the ensemble technique is utilized for improving the performance of individual TKGE models. In this paper, different approaches are examined to learn the weights for the ensemble model to improve the overall performance. Especially, neural network architecture is built to learn the weights dynamically for ensemble learning. In addition, Diachronic Embedding methods [11] (including DE-TransE, DE-DistMult, and DE-SimplE), ATiSE [32] and TERO [33] are chosen to capture different features of data as the base model for ensemble learning.

Although deep ensemble learning model has better generalization performance by combining ensemble learning and deep learning [9], to the best of our knowledge, it is the first time using the deep learning method to learn weights for the TKGE models.

Section 4 illustrated the methodologies in this paper, which are ensemble learning, grid search, and neural network model. Section 5 demonstrated the datasets, metrics, and experiments. Section 6 presents and analyses the results of individual TKGE models, grid search ensemble model, and neural network ensemble model.

3 Related Work

3.1 Temporal Knowledge Graph Embedding

Diachronic Embedding [11] is a TKGE method that provides hidden representations for the entities of TKGs at any time point. It can be applied to any existing method, this is called model-agnostic. In the original paper, there are 3 KGEs extended into TKGEs by using this method, which are TransE

Conference'17, July 2017, Washington, DC, USA
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

[4], SimplE [16], and DistMult [34]. TransE is a translation-based method that utilizes the score function to evaluate the quality of embedding based on distance. DistMult and SimplE are tensor factorization-based methods that utilize the score functions to evaluate the quality of embeddings based on similarity. In TransE, the relation is represented as a translation vector that connects the embedded entities, and it cannot model one-to-many, many-to-one, and many-to-many relations. DistMult utilizes neural network to learn the multi-relational representation, and it is unable to model the anti-symmetric relations. DE-SimplE has fully expressive competence which can model both symmetric and anti-symmetric relations.

ATiSE [32] incorporates time information into entity/relation representations using Additive Time Series decomposition. Additive Time Series decomposition decomposes a time series into trend, seasonality, and residuals. Trend captures the overall pattern or direction of the data. Seasonality captures the repetitive, periodic patterns in the data that occur over fixed intervals, such as daily, weekly, or yearly. Residuals captures the unexplained variation or noise in the time series. TERO [33] defines the temporal evolution of entity embedding as a rotation from the start time to the end time in the complex vector space. And each relation is represented as a pair of dual complex embeddings to handle the start and the end of the relation, respectively.

3.2 Ensemble Learning

Ensemble learning is a widely used methodology in the machine learning field. It captures diverse attributes from multiple models by combining them together to make more accurate predictions. Deep ensemble learning models combine the advantages of both the deep learning models as well as the ensemble learning such that the final model has better generalization performance [9]. Generally, these methods are categorized into bagging, boosting, and stacking.

3.2.1 Bagging. Bagging [24], also known as bootstrap aggregating, splits the training dataset into several different subsets of equal size. Each subset is utilized as input data to train a model. Finally, the predictions from those distinct models are combined. Therefore the final prediction is better than any of the individual models. Random Forest [5] is a typical ensemble learning method based on bagging. The most common base model for Random Forest is decision trees. Since an individual decision tree is allowed to grow very deep which can cause the overfitting problem, Random Forest combines multiple shallow decision trees to improve the predictions.

3.2.2 Boosting. Boosting is an ensemble learning method that transforms a weak learning model into a strong learning model with better generalization [9]. As ensemble learning models, both bagging and boosting combine the prediction of multiple models, although boosting is trained sequentially and bagging is trained independently. To apply boosting, each training sample in the dataset is assigned a weight based on the performance of the previous models. The misclassified samples are given higher weights so that they can have a greater impact on the current training. In terms of dataset utilization, boosting utilizes the entire training dataset rather than splitting the training dataset into several subsets to train the models. Gradient Boosting [8] and Adaboost [7] are typical examples of boosting. Both algorithms learn from the errors that the previous model made to generate a better model, although they are applied in different ways. Adaboost first assigns the same weights for each model and then adjusts weights through each iteration. Gradient Boosting introduces an objective function, the smaller the value of the objective function is, the better the model is. XGBoost [6] is one of the tree boosting algorithms that is widely used in machine learning. XGBoost builds trees by utilizing second-order Taylor approximation to minimize the objective function.

3.2.3 Stacking. Stacking [31], also known as stacked generalization, is the technique that leverages meta-learning algorithms, specifically for ensemble learning, to achieve outstanding performance. The process involves two steps that are similar to the meta-learning algorithm. The dataset is equally divided into J parts, and $(J-1)$th-folds are utilized for base learner training. In this step, the learners learned a variety of generalizable patterns from the dataset. Then, using the outputs from the base learners as input to further learn a meta learner to aggregate the predictions of base learners. Finally, the outputs from the meta learner provide the final prediction. This exemplifies the idea of learning from experience and it is also known as the winner-takes-all strategy.

This paper utilized stacking to ensemble TKGE models. First, the TKGE models are trained individually. Then, the meta learner combines the outputs of each individual TKGE model to predict the final prediction.

3.3 Hyperparameter Optimization

Grid Search [21] defines a range of discrete values as candidates for each hyperparameter, and then exhaustively evaluates the performance of each combination. However, it can be computationally expensive when the number of hyperparameters or the range of each hyperparameter is large, since as the number of hyperparameters or the range of each hyperparameter increase, the number of combinations grows exponentially. On the other hand, the ranges of candidates are in discrete numbers, even though the search is exhaustive, it may fail to capture the important regions of the space that contain optimal hyperparameters.

The main difference between Random Search [2] and Grid Search is that, in Random Search, the hyperparameter sets are chosen randomly. Since it does not try every combination,

it is not guaranteed to return the best-performing values, however, it does return a relatively good-performing model in a significantly shorter time. [15]

As with Grid Search and Random Search, the objective of Bayesian Optimization [27] is also to minimize the objective function to obtain the optimized hyperparameters, although Bayesian Optimization is more efficient. Apart from the objective function, it has the acquisition function, which is used to guide the search direction based on the performance of previous hyperparameters. Thus, the search frequency of hyperparameter combinations can be reduced. By using Bayesian Optimization, the search space grows more slowly than others as the dimension increases. In addition, since the hyperparameter space is not searched exhaustively, the distribution of the objective function is unclear. Therefore, a surrogate model initialized by the Gaussian process is built. The Gaussian process brings prior knowledge about the objective function into the optimization process, which can help lead the search to a promising region of the search space.

There are also other hyperparameter optimization algorithms. For instance, Evolutionary Optimization [26] uses evolutionary algorithms to search the hyperparameter space. Gradient-based Optimization [19] tunes the hyperparameters by computing the gradient using gradient descent or other optimization algorithms.

In this project, Grid Search are used as baseline method to learn weights for ensemble model. In addition, we propose neural network methods to learning the weights for ensemble models for link prediction.

4 Methodology

This section introduces the ensemble model and the methods employed to optimize the weights of ensemble learning. Section 4.1 includes the TKGE model chosen and the ensemble model architecture which illustrates how the model performs link prediction. Section 4.2 presents the grid search. Section 4.3 starts with the limitation of grid search, and then illustrates the inputs and outputs of the neural network. Finally, the architectures of learned neural networks for different inputs are presented.

4.1 Ensemble Model

4.1.1 TKGE Methods. Except for the quality of the datasets itself, the ability of TKGE models is an important factor decide the performance of ensemble learning. The principle of ensemble learning is to combine the advantage of each individual model. Thus, the model that is good at embedding different types of relations or times is chosen as the base model of ensemble learning. A fact in TKG consists of head, relation, tail, and time, represents as (h, r, t, τ). There are several different types of relations in a TKG:

- **Symmetric**: $(h, r, t, \tau) \wedge (t, r, h, \tau) == True$
- **Asymmetric**: $(h, r, t, \tau) \wedge \neg(t, r, h, \tau) == True$
- **Anti-symmetric**: $(h, r, t, \tau) \wedge \neg(t, r, h, \tau) == True$
- **Reflective**: $(h, r, h, \tau) \vee (t, r, t, \tau) == True$

There are 5 models chosen for the ensemble model, which are DE-TransE, DE-DistMult, DE-SimplE, ATiSE, and TERO. All the models can embed time information. All the DE models have the same way to embed time information, the embedding of entities and relations is inherited from the original model. Thus, DE-TransE is only able to capture asymmetric relations. DistMult is only able to capture the symmetric relation. DE-SimplE can capture both symmetric and asymmetric relations. TERO is designed for time intervals by the characteristic of complex numbers regarded as a rotation in a complex plane, capturing reflexive, asymmetric and symmetric relations. ATiSE is able to detect various patterns of time series by additive time series decomposition, such as seasoning, trend, and random patterns.

4.1.2 Ensemble Model Architecture. Ensemble learning utilizes the results from multiple models to predict the final output. The architecture of the ensemble model is illustrated in Figure 1. Link prediction is performed by this ensemble model with the aim to predict the most plausible heads/relations/tails/timestamps. Inspired by the idea of stacking, the ensemble model is separated into base model and meta model. Base model aims to predict the initial predictions. Then, the predictions of each base model are fed into the meta model to produce the final prediction.

As shown in Figure 1, the process of the base model is typically the process of link prediction. The input of the link prediction is a query that predicts one of the elements of a fact. For instance, if the query asks for the head of a fact, it will be represented as a corrupted quadruple $(?, r, t, \tau)$. Then, a number of simulated facts are generated. The simulated facts are potential facts that take every entity into a query. The number of simulated facts is equal to the number of entities. Next, the TKGE model will evaluate the plausibility of the simulated fact by score function. For translation-based TKGE models, the score function measures the distance between two entities, usually after a translation performed by the relation. Therefore, each simulated fact gets a simulated score. Then, a sequence of simulated facts can be ordered by the simulated scores. Finally, each simulated fact obtains a simulated rank. The sequence of simulated ranks for different TKGE models are represented as $P1$, $P2$, $P3$, $P4$, and $P5$.

The idea of the ensemble score is to take the sum of the product of the weights and the simulated ranks from each model. As illustrated in Figure 1, the meta model combine the predictions which is $P1$, $P2$, $P3$, $P4$, and $P5$ as the ensemble score. The calculation of the ensemble score is different in different methods of learning the weights. The grid search learns a set of weights that have 5 weights corresponding to 5 TKGE models, while the neural network learns a set of

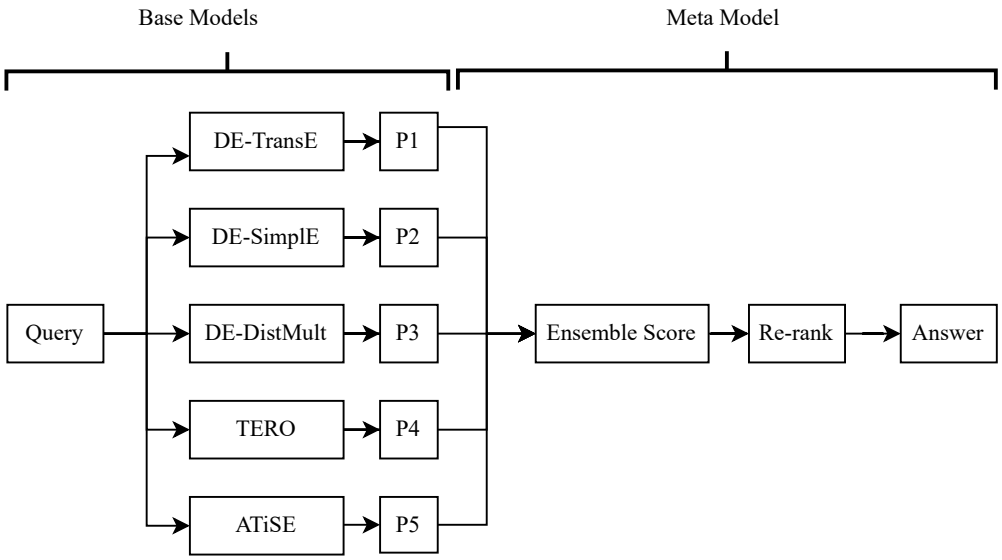

Figure 1. Ensemble Model.

weights that each query has a weight in each TKGE model. The ensemble score for each simulated fact when using grid search methods is shown in Eq. 1. w_i is the weight assigned to M_i. $Rank_i$ is the rank of simulated fact in M_i. The ensemble score for each simulated fact when using neural network methods is shown in Eq. 2.

$$ENS_{grid} = \sum_{i=1}^{|M|} w_i * Rank_i \tag{1}$$

$$ENS_{nn} = \sum_{i=1}^{|M|} \sum_{j=1}^{|Q|} w_{i,j} * Rank_{i,j} \tag{2}$$

Based on the ensemble score, the meta model re-ranks all the simulated facts. Now, the correct answer has a new rank which normally is smaller than the original rank. Thus, the performance of the model is improved. The approaches implemented to learn the weights for each base model are illustrated in Section 4.2 and Section 4.3.

4.2 Grid Search Method

In order to obtain the optimal weights for ensemble learning, grid search is utilized as a baseline method. After learning, the best-weight set can be utilized as hyperparameters for ensemble learning.

The grid search method is used aim to minimize this objective function by minimizing the overall ensemble score. As shown in Eq. 3, the objective function is $f(W)$. Since there are 5 individual TKGE models, the best-weight set W contains 5 weights, each represented as w_i, and is optimized by this objective function. In Eq. 7, the RANK score of model i is defined as $RANK_i$.

$$f(W) = \sum_{i=1}^{|M|} w_i * RANK_i, w_i \in W \tag{3}$$

As illustrated in Section 5.1, RANK is not the best metric. Therefore, the objective function based on other metrics is additionally built for comparison. In Eq. 4, the MRR score is regarded as the score of the individual TKGE model to calculate the ensemble score, and the MRR score of model i is defined as MRR_i. The objective is then to maximize the objective function.

$$f(W) = \sum_{i=1}^{|M|} w_i * MRR_i, w_i \in W \tag{4}$$

In Eq. 5, the normalized MRR scores are directly served as the weight of the individual TKGE model. This experiment was designed to prove the effectiveness of ensemble learning.

$$w_i = \frac{MRR_i}{\sum_{i=1}^{|M|} MRR_i} \tag{5}$$

4.3 Neural Network Method

There are some limitations when using the grid search to learn the weights for the ensemble model. For instance, the grid search defines the search space consisting of discrete values of the weights. The fixed granularity restricts the ability to perform a fine-grained exploration of the hyperparameter space. Consequently, it potentially missing optimal values that exist between the predefined search space. However, neural networks have the ability to eliminate such limitations. In regression problems, the values of predictions are

continuous numbers, allowing for more precise and accurate learned weights. On the other hand, the weights learned from the grid search are only one set of weights. The weights that the neural network predicted are a set of weights for each query, which means the neural network can assign the weights dynamically to each query. Thus, this project introduces neural network to learn the weights for the ensemble model.

The input of the neural network model is the output of each individual TKGE model, since the rank is unobserved when the new instance is applied to this model, the simulated score is considered as the input of the neural network. The simulated score is the measure of rank, thus, to some extent, simulated scores reflect the embedding quality of the simulated facts. The simulated score is therefore as the input of the neural network. Then the rank of the correct answer is taken as the target of this neural network model. The targets are processed thus can be regarded as weights for each query, see Section 5.3.2. Overall, the neural network is trained to learn the relationship between the predicted values of each individual TKGE model and the rank of the correct answers.

The architecture of the neural network can be seen in Figure 2 and Figure 3.

4.3.1 Neural Network Architecture of ENN-5.

In Figure 2, ENN-5 model have 5 inputs, corresponding to the top 1 prediction of each TKGE model. The Rectified Linear Unit (ReLU) [25] which widely used in neural network is used for all the hidden layers here. The output layer has no activation function. The configuration of the model architecture is chosen by Optuna [1] through evaluating different combinations of hyperparameters. A more detailed description of the tuning process is presented in Section 5.6.1.

Both neural network models are trained in mini-batch, with Adam [17] as optimizer. The loss function is RMSE, as described in Eq 6, where p_j is the predicted value, and t_j is the target value. While both MSE and RMSE are used for regression tasks in machine learning, RMSE brings the evaluation metric back to the original scale of the data by taking the square root. This provides a straightforward interpretation of the average prediction error.

$$RMSE = \sqrt{\frac{1}{Q}\sum_{j=1}^{|Q|}(p_j - t_j)^2}, j \in Q \tag{6}$$

Both models are evaluated by calculating the RMSE on the final predicted values after training over a number of epochs. Early stopping is implemented in both models in order to prevent overfitting. It stops the training and saves the best-learned model if the model has not shown an improvement in the validation loss and the difference between training loss and validation loss has not decreased within 3 epochs. For a concrete design, see Section 5.6.2.

4.3.2 Neural Network Architecture of ENN-25.

In Figure 3, ENN-25 model have 25 inputs. The top 5 predictions of each TKGE model are taken as the inputs of ENN-25. The assumption is that the more information is fed into the neural network, the more hidden features will be learned through training. The activation function ReLU is also used here for the hidden layers. As ENN-5, the output layer has no activation function. The configuration of the model architecture is also chosen by a framework Optuna [1]. See Section 5.6.1 for further illustration.

5 Experiments

This section presents the metrics, datasets, dataset processing, hyperparameter tuning of the neural network, and the experimental setups employed in the experiments.

5.1 Metric

The evaluation of all experiments utilizes 6 metrics, which are HITS@1, HITS@3, HITS@10, MR, MRR, and RANK.

Link prediction outputs a sequence of simulated facts, where the rank is the index of the correct facts when ordered according to simulated scores. The process of link prediction is illustrated in Section 4.1.2. The metric RANK takes the sum of the ranks of all correct answers, as shown in Eq. 7, the RANK score of model i is represented as $RANK_i$. The higher the overall rank, the worse the performance of the model.

Since the further back the predicted answer is, the less plausible it is, although the ranking difference between 1 and 2, 1000 and 1001 are both 1, the performance difference is different. In addition, the sum of all ranks is likely a large number that is difficult to compare. It is therefore not a good choice to have RANK as the only metric.

$$RANK_i = \sum_{j=1}^{|Q|} Rank_{i,j}, i \in M \tag{7}$$

HITS@N metric takes the percentage of rank less or equal to n, as shown in Eq. 8, the HITS@N score of model i is represented as $HITS@N_i$. The higher score of the HITS@N is, the better the model performs. Since it only evaluates the top n correct answers, it allows the model that focuses especially on top n to stand out in specific link prediction tasks.

$$HITS@N_i = \frac{1}{|Q|}\sum_{j=1}^{|Q|} Rank_{i,j}, Rank_j \leq N, i \in M \tag{8}$$

MR is the abbreviation for mean rank. As the name suggests, it takes the mean value of the ranks of correct answers, as shown in Eq. 9, the MR score of model i is represented as MR_i. The lower the MR is, the better the performance of link prediction model is.

However, in the scenario where only one correct answer has a extremely high rank, even though other queries have

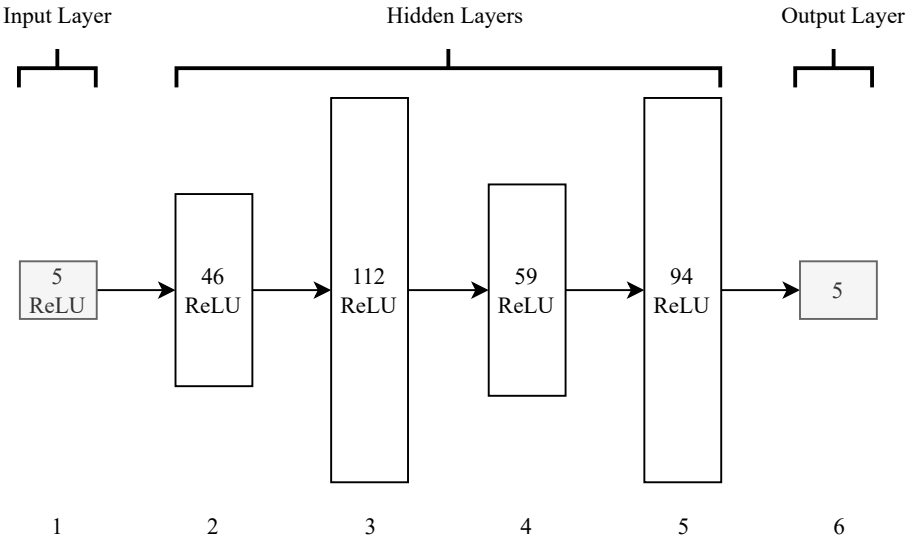

Figure 2. Neural Network Architecture with 5 inputs

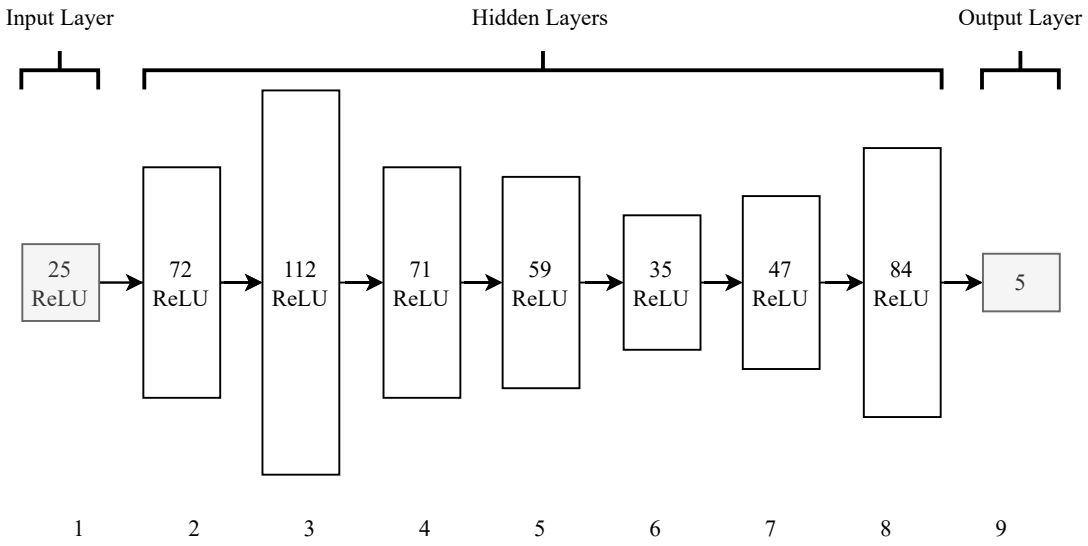

Figure 3. Neural Network Architecture with 25 inputs

low ranks, this can result in a high MR score overall. While the MR score is a good metric to evaluate the performance of the model, since the objective of link prediction is to find the most plausible answers, it is not the optimal metric to evaluate the performance of link prediction.

$$MR_i = \frac{1}{|Q|} \sum_{j=1}^{|Q|} Rank_{i,j}, i \in M \tag{9}$$

MRR stands for mean reciprocal rank and is presented as an equation in Eq. 10, the MRR score of model i is represented as MRR_i. It takes the average of reciprocal ranks of correct answers. The higher the MRR score is, the better the model performs. By taking the reciprocal value of ranks, it reduces the effect of a very large rank of correct answers. MRR is a good metric to use for both testing the TKGE and link prediction.

$$MRR_i = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{Rank_{i,j}}, i \in M \tag{10}$$

5.2 Dataset

The ICEWS14 dataset [10] is used for all the experiments in this project. The data is from the Integrated Crisis Early Warning System (ICEWS) in 2014. It contains 7128 entities, 230 relations, and 365 timestamps. The dataset is split into training, validation, and test datasets which have 72826 facts, 8941 facts, and 8963 facts respectively. The time information in ICEWS14 is represented as timestamp with the format yyyy-mm-dd. All individual TKGE models in this project are transductive models. Therefore, all the individual TKGE models are trained or predicted based on the existing entities, relations, and timestamps only, the models cannot generalize to new occurrences of entities, relations, or timestamps.

5.3 Dataset Processing

In all experiments involving individual TKGE models, the individual TKGE models is trained, validated, and tested on the same dataset.

For further training and testing of the ensemble learning models, the test dataset is split into *ens_train*, *ens_val*, and *ens_test* datasets. The proportion of each dataset is 0.7, 0.15, and 0.15, which have 6274, 1345, and 1344 facts, respectively. Based on the split, several data processing operations are applied to the baseline and neural network experiments.

5.3.1 Baseline Dataset Processing. In baseline experiments, only *ens_train* and *ens_test* are used. Each fact in the dataset generates 4 queries with answers. To further illustrate the idea of the query dataset, the query is represented as a five-element tuple in Eq. 11, where a is the answer to the query, and 0 is the target of the prediction. During implementation, since the data is difficult to manage in Txt format, a JSON format file of the dataset Q is created for link prediction. Then, in order to calculate the ensemble score in the weights-learning step, each query added another key-value pair the key *RANK* or *SIMU*, and the value is the rank or simulated score of the correct answer returned from each TKGE model.

$$Q = \{(0, r, t, \tau, a), (h, 0, t, \tau, a), (h, r, 0, \tau, a), (h, r, t, 0, a) \mid h, t \in \mathcal{E}, r \in \mathcal{R}, \tau \in \mathcal{T}, a \in \mathcal{E} \cup \mathcal{R} \cup \mathcal{T}\} \tag{11}$$

5.3.2 Neural Network Dataset Processing. In the experiments involving the neural network method, three distinct datasets are utilized. Firstly, *ens_train* dataset is employed to train the neural network model, enabling it to learn the hidden relation between the input and target values. Then, *ens_val* dataset is utilized to evaluate the performance of the learned model. During the evaluation, the optimal model is selected to serve as the final model. Finally, the optimal model chosen during evaluation utilizes *ens_test* to predict the optimal weights to serve to ensemble model.

Since different TKGE model has different score function resulting in different scale, the inputs should be normalized before feeding into the neural network. Min-max normalization [22] is utilized to mitigate the impact of different scales, as shown in Eq. 12. $Score_{min}$ and $Score_{max}$ are the minimum and maximum values over all the scores respectively.

$$Score_{\text{normalized}} = \frac{Score - Score_{min}}{Score_{max} - Score_{min}} \tag{12}$$

Furthermore, for DE-TransE, DE-SimplE, and DE-DistMult, a higher simulated score indicates a greater likelihood of a new relationship. Conversely, for TERO and ATiSE a lower simulated score suggests a higher possibility of a new relationship. For consistency, the score of DE-TransE, DE-SimplE, and DE-DistMult are reversed by calculating $1 - Score_{normalized}$ after min-max normalization.

The targets of the neural network are obtained from the rank of the correct answer. First, the min-max normalization is applied to the ranks. Then, align to the convention that higher output represents the higher probability of the TKGE model, the normalized score is reversed by calculating $1 - Score_{normalized}$

In ENN-5 model, it takes the simulated score of rank 1 prediction of each model as inputs. In the experiment ENN-25, to incorporate the order information into training, the top 5 rank predictions of each model are served as inputs. Both experiments have 5 targets calculated from rank which are regarded as weights for each TKGE model. The dataset is manipulated into a CSV file to feed into the neural network model.

5.4 Training of Ensemble Model

Before the ensemble model is employed in link prediction tasks, two steps of training are required. Firstly, each TKGE model is trained as a part of the base model, following all the setups from the original paper. Secondly, grid search and neural networks are trained to obtain the optimal weights. The optimal weights are used for calculating the ensemble score. Then, the ensemble model can be used for link prediction.

The process of obtaining best-weight sets for grid search is illustrated in Section 4.2. For neural networks, hyperparameter tuning is applied to obtain an appropriate neural network architecture for the specific dataset. The tuning process of hyperparameters is illustrated in Section 5.6. After hyperparameter tuning, the neural network architecture with the relatively lower loss on validation dataset is regarded as the neural network architecture to predict weights for ensemble learning. The neural network architectures are present in The neural network architectures are presented in Section 4.3.

There are several metrics in Section 5.1 to evaluate the performance of each TKGE model. The performance evaluation of the ensemble model is to assess the accuracy of the final

prediction. In addition, the improvement in performance reflects the quality of the learned weights.

5.5 Training of Grid Search Model

The initial weights for each individual TKGE model are all set to 0.2. The range of weights for each TKGE model is set to a list $W_{range} = [0.1, 0.2, 0.3, ..., 1.0]$, containing 10 numbers. As the weight range is the same for all TKGE models, the search space for the experiment is 10^5. Since learning the hyperparameter with such a large search space is time-consuming, a constraint is added to ensure that the sum of the weights is equal to 1.

5.6 Training of Neural Network Model

This subsection provides a description of the hyperparameters utilized in the neural network experiments, along with the tuning process employed for these hyperparameters. Additionally, present the implementation of the early stopping technique

5.6.1 Hyperparameter Tuning. Optuna [1] is a framework tuning the hyperparameter for neural networks automatically. It treats the hyperparameter optimization problem as a sequential decision process. It finds the most promising set of hyperparameters based on previous evaluation results, then suggests a new set of hyperparameters to evaluate. This process continues iteratively until met the stopping criterion, such as a maximum number of trials, a time limit, or a convergence threshold.

In the experiments of ENN-5 and ENN-25, the hyperparameters evaluated by Optuna are the number of hidden layers, the number of neurons for each layer, the dropout ratio, and the learning rate. The ENN-5 model is allowed to have 1 to 6 hidden layers, consisting of the number of neurons within the range 4 to 128. The ENN-25 model is allowed to have the number of hidden layers and the number of neurons in the range 1 to 10 and 4 to 128 respectively. Both ENN-5 and ENN-25 models have the dropout ratio from 0.1 to 0.5, the learning rate from 10^{-5} to 10^{-1}. Those combinations are evaluated on the validation dataset for each epoch using the RMSE loss function.

The architecture of ENN-5 and ENN-25 is presented in Section 4.3.1 and Section 4.3.2 respectively. ENN-5 result in an learning rate of 0.004 and a batch size of 128. ENN-25 has a learning rate of 0.002 and a batch size of 128.

5.6.2 Early Stopping. Early stopping [23] is a widely used technique for preventing overfitting and determine the optimal number of epochs for training neural networks. It monitors the performance of the model during training by considering both the training and evaluation processes each epoch.

In the implementation, a parameter known as *patience* is defined which is set to 3 in this paper. It is serves as a threshold for the oscillation of the validation loss. If the validation loss fails to improve over the subsequent 3 epochs compared to the current minimum loss, meanwhile, the difference between training loss and validation loss is not improved, which means the model has a highly chance to be overfitting, the training will be terminated, and the model with the best performance so far will be saved.

The model is validated immediately after training for each epoch. The training loss is the sum of the RMSE loss for all instances of training dataset during training. The validation loss is the sum of the RMSE loss for all instances of validation dataset during validation.

6 Results

6.1 Baseline Ensemble Learning Results

In order to have a comparable baseline for deep learning-based ensemble learning models, grid search technique is utilized to learn the optimal weights for TKGE models. By minimize the objective function based on different metrics scores, several best-weight set are learned, as it is shown in Table 1. The EG-RANK row is the optimal weights learned by the grid search when using RANK as the score of each TKGE model. The EG-MRR row is the optimal weights learned by utilizing the MRR score as the score of each TKGE model to feed into the objective function. In the last column, the weights of E-N are not learned from grid search, it is the score that normalizes the MRR score of each TKGE model directly. Evaluating the best-weight set of E-N provide evidence of the effectiveness of ensemble learning.

The best-weight sets obtained based on EG-MRR and EG-N metric scores are aligned with the performance of each model from the original paper [33] which the weight for ATiSE and TERO is higher than any other models. On the other hand, EG-RANK assigns the highest weight to DE-TransE, as the results from the original paper [11] show that DE-TransE performs the worst compared to DE-DistMult and DE-SimplE across all metrics and datasets, the result seems unreasonable. However, as demonstrated in Section 5.1, under the large dataset and uneven distribution of ranks, the efficacy of the model cannot be accurately evaluated solely based on the value of RANK. Thus, it is reasonable to get this result when using RANK as the score of each TKGE model to calculate the ensemble score for ensemble model.

In E-N, when rounding to 1 decimal place, the weight set is limited to either 0.1 or 0.2. While the weight assigned to each model has no considerable difference, it still indicates the difference in the performance of each model. This is because the weights are the normalization of MRR scores in each TKGE model directly, and the MRR score enables to represent of the performance of each model ideally. All the grid search results have one decimal place. Given that the sum of the weights is equal to 1, i.e. the weight for each model is less than 1, the one decimal place value is considered

Table 1. Best weights for baseline experiments.

MODEL	$W_{DE-TransE}$	$W_{DE-SimplE}$	$W_{DE-DistMult}$	W_{TERO}	W_{ATiSE}
EG-RANK	0.5	0.1	0.2	0.1	0.1
EG-MRR	0.1	0.1	0.1	0.3	0.4
E-N	0.14	0.21	0.19	0.24	0.22

insufficiently precise to effectively demonstrate the benefits of each TKGE model. However, as increasing the choice of weights will expand the search space, the sets of weights are kept one decimal place precision. It is therefore necessary to further explore other techniques such as neural network-based learning to provide a more accurate weights.

The results of evaluation on the ensemble model that assigns those weights to TKGE models are shown in Table 2, as well as the results for evaluating individual TKGE models. Every ensemble model in this table is evaluated in the *ens_test* dataset. All the results are evaluated to do link prediction on head, relation, tail, and time. Other than MR and RANK, all metrics tend to have better performance when the scores are higher. The highest results are in boldface. The second-highest results are underlined.

For TKGE models, DE-TransE has the lowest score in all metrics which indicates DE-TransE is the worst TKGE model compared to others. TERO achieves the highest score across both MRR and all HITS metrics, while ATiSE has the second-highest score in those metrics.

From ensemble models, it is clear that EG-MRR where TERO is assigned the highest weight 0.4, has the best result across all metrics, except MR and RANK. The performance after ensemble learning improved 33% in HITS@1, 15% in HITS@3, 11% in HITS@10, and 20% in MRR than ATiSE which performs best across all individual TKGE models. On the other hand, EG-RANK has the best result on MR and RANK. It gets the lowest score in HITS@1, HITS@3, HITS@10, and MRR which are 20%, 7%, 5%, and 11% compared to other ensemble models. Even the lowest performance in ensemble models achieves improvement suggesting that ensemble learning is effective. Comparing the result from EG-RANK and EG-MRR with considering the principle behind metric RANK and MRR indicates that, in the case where the top answer is more important, such as QA systems where only the correctness of the rank 1 answer is valuable, the weights learned based on the score of HIT@1, HIT@3, HIT@10, and MRR are more reliable to learn the optimal weights for ensemble model, also to evaluate the performance of TKGE model. In the case where the overall ranks for the query are more important, the weights learned based on the MR and RANK scores are more appropriate. Among all the learned best-weight sets, E-N gives the most stable results across all metrics, it gets the second-highest results across all metrics.

6.2 Deep Learning-based Ensemble Learning Results

The neural network method is used as the deep learning-based solution for ensemble learning. The results of the evaluation of different neural networks architectures are shown in Table 2. ENN-5 is the results when the neural network has 5 inputs, and ENN-25 is the results when the neural network has 25 inputs. By minimizing the RMSE loss, the neural networks are able to predict weights for each query and get relatively good results than baseline ensemble models. The highest result is shown in boldface. The second-highest result is underlined.

As indicated in Table 2, compare the neural network methods to baseline methods used as a part of ensemble model, neural network methods achieved the highest and second-highest scores across all metrics. This validates the assumption that neural networks are capable of learning the weights for ensemble models and yield superior results compared to grid search. Moreover, it confirmed the assumption that the neural network has learned the relationship between the simulated scores and the rank of corrected answers. In addition, since the neural network method performs best across all the baseline models, it is also better than all the individual TKGE models. EG-MRR has the overall good performance of the baseline methods on most metrics, and ENN-5 has a better performance of deep learning-based methods on all metrics, except HITS@10. Compare EG-MRR to ENN-5, the results improved 8% in HITS@1, 3% in HITS@3, 2% in HITS@10, and 5% in MRR.

Generally speaking, more information enhances the learning ability of a neural network. However, the results of ENN-25 is worse than ENN-5 across all metrics, except HITS@10. This is understandable, since the more input feature a neural network has, the more complex the neural network architecture is, which means more difficulty in learning the hidden information from the data. This could be reflexed in the architecture of neural networks in Figure 2 and Figure 3 where ENN-5 has 4 hidden layers and ENN-25 have 7 hidden layers. In addition, the quality of the input data is also important for learning a neural network model. Based on the analyzes of the ranking list, the prediction of the TKGE model barely shares the same entities at the top 5 simulated scores, which means the top 5 scores correspond to different entities, relations, or timestamps. Although neural networks are able to learn the order of simulated facts, different TKGE models have different simulated facts on the same rank. Moreover,

Table 2. Evaluation for Baseline and Neural Network Experiments.

MODEL	HITS@1	HITS@3	HITS@10	MR	MRR	RANK
DE-TransE	0.0958	0.3134	0.4913	108.4182	0.2382	582856
DE-DistMult	0.2254	0.3650	0.5065	142.0130	0.3223	763462
DE-SimplE	0.2435	0.3943	0.5407	121.0993	0.3461	651030
ATiSE	0.2600	0.4200	0.5943	162.8296	0.3719	875372
TERO	0.2799	0.4589	0.6040	126.7958	0.3945	681654
EG-RANK	0.3248	0.4926	0.6352	93.0255	0.4334	500105
EG-MRR	0.3597	0.5314	0.6689	106.0714	0.4683	570240
E-N	0.3575	0.5210	0.6620	95.6097	0.4627	513998
ENN-5	**0.3901**	**0.5476**	0.6819	**52.4985**	**0.4925**	**282232**
ENN-25	0.3888	0.5471	**0.6821**	52.5061	0.4917	282273

the largest difference between ENN-5 and ENN-25 is the HITS@1 score, however, which is a relatively small number 0.0013. Thus, the more information is not providing a big improvement on results, it is worse more research in the future.

For further analysis, Table 3 is built to analyze the performance of each component of the query. Since the evaluation score of MR and RANK are relatively unstable in the results above, only HITS@1, HITS@3, HITS@10, and MRR scores are compared. As it is shown, tail always has a better performance than head across all metrics in both models. Then, relation and time have worse performance, while relation is better than time. Considering the results from [14], the performance of individual TKGE models following the order: tail > head > relation > tail in all evaluated TKGE models. This shows that even though ensemble learning improves the overall performance, it still follows the pattern of individual TKGE performance, as the performance of individual TKGE models is one of the reasons that determine the upper boundary of ensemble models. The better the individual TKGE model performs, the greater the potential of the ensemble model.

Table 4 is obtained by calculating the difference between the results of ENN-5 and EG-MRR in Table 3, then converting the numbers into percentages. The negative percentage represents that EG-MRR has a better performance on specific components. Inversely, the positive percentage indicates that ENN-5 has a better performance on specific components. As it is shown in Table 4, EG-MRR is slightly better at predicting head and relation than ENN-5, although the difference is not obverse, highest at 4%. ENN-5 is good at predicting tail and time than EG-MRR, and the difference is larger than the difference of head and relation, lowest at 4%. This suggests that the weights learned by neural network methods can better improve the prediction of tail and time than grid search. Since the objective function of grid search utilizes the metric score as the model performance score, it cannot capture the feature elementwise, however, the neural networks utilize simulated scores of each query can.

7 Discussion

All the information a neural network is able to utilize for training is the inputs and targets from the dataset. Therefore, it is crucial to ensure the quality of data. In this paper, the simulated score and rank information are used to build the neural network. The performance improvement is not significant when expanding the input size to 25. One of the possible reasons for this is that there is noisy data in the dataset, as the order information introduced into the neural network is incomplete, top 1 for ENN-5, top 5 for ENN-25. However, if the complete sequence of entities is used as inputs, it would be very time-consuming to train the neural network model or the ensemble model. According to the experiment, the number of hidden layers is increasing from 4 to 7 when the input increases from 5 to 25, thus, the trained model for a complete sequence will be very large. Although the problem can be solved by training the data in batches, it is not worthwhile. Thus, it is merit to examine the embeddings encoded by each individual TKGE model as inputs to train the network, since the embedding encodes more information than a single score. On the other hand, it could be the structure of the neural network is too simple to capture the hidden pattern of the data. The hidden layers of the neural network used fully connected layers in this project. However, there are other types of layers that are more expressive. For instance, the convolutional layer, as the main part of the convolutional neural network [18], it can learn features hierarchically by stacking multiple convolutional layers. As link prediction predicts a sequence of ordered entities, there are a number of neural networks specific for sequence data that can be applied to learn the information from the ordered sequence, such as Transformer [29], Long shortterm Memory [13].

The ensemble model in this project trained the base model part and meta model part separately which is not efficient

Table 3. Elementwise Performance Between ENN-5 and EG-MRR

MODEL	HITS@1	HITS@3	HITS@10	MRR
ENN-5_{HEAD}	0.4427	0.6332	0.7731	0.5580
ENN-$5_{RELATION}$	0.3356	0.5268	0.7292	0.4671
ENN-5_{TAIL}	0.5714	0.7478	0.8430	0.6732
ENN-5_{TIME}	0.2106	0.2827	0.3824	0.2717
EG-MRR_{HEAD}	0.4487	0.6473	0.7775	0.5658
EG-$MRR_{RELATION}$	0.3385	0.5625	0.7574	0.4828
EG-MRR_{TAIL}	0.5104	0.6935	0.8028	0.6165
EG-MRR_{TIME}	0.1414	0.2225	0.3378	0.2079

Table 4. Performance Difference Between ENN-5 and EG-MRR

MODEL	HITS@1	HITS@3	HITS@10	MRR
HEAD	-1%	-1%	0%	-1%
RELATION	0%	-4%	-3%	-2%
TAIL	6%	5%	4%	6%
TIME	7%	6%	4%	6%

if the number of base model is large. The hypernetwork [12] is an approach of using one network, also known as a hypernetwork, to generate the weights for another network dynamically. By using hypernetwork, base model part and meta model part can be trained simultaneously which enables the model to adapt to new tasks quickly.

In order to get a better performance result, the base model should be selected from those that have good performance themselves. In addition, the model chosen should be based on different characteristics of datasets. For example, if the dataset has no symmetric relation, then the advantage of DE-SimplE is not shown through evaluation. Moreover, the number of base models is worth researching to find the appropriate number of models to balance learning and performance.

8 Conclusion

This research proposes a neural network approach to predict the weights of each query at every TKGE model of the ensemble model. And has a thorough comparison between the evaluation results of the neural network ensemble model and the grid search ensemble model.

To conclude:

- MRR score-based objective function in the grid search obtained the best weight set than others.
- The elementwise results of all weight learning methods used in this paper, after ensemble learning, follow the same pattern as the original results of the individual TKGE model, which are tail > head > relation > time.
- Neural networks are able to learn the weights for the ensemble TKGE model and generate better results than grid search.
- The number of features input to the neural network has limited effects on the performance of the ensemble model.
- The neural networks-based ensemble model improve tail and time prediction better than grid search-based ensemble model.

9 Future Work

The project evaluate the result of link predciton on all element of a query. In Table 4, the difference of ENN-5 and EG-MRR suggest that neural network model have a potential to learn better at time information. It is therefore worth having further research on the elementwise neural network architecture.

Acknowledgments

I would like to thank my supervisors Daniele Dell'Aglio and Huan Li for their help and guidance.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2623–2631.

[2] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research* 13, 2 (2012).

[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1247–1250.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* 26 (2013).

[5] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.

[6] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[7] Yoav Freund and Robert E Schapire. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Second European Conference, EuroCOLT'95 Barcelona, Spain, March 13–15, 1995 Proceedings 2*. Springer, 23–37.

[8] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[9] Mudasir A Ganaie, Minghui Hu, AK Malik, M Tanveer, and PN Suganthan. 2022. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence* 115 (2022), 105151.

[10] Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. *arXiv preprint arXiv:1809.03202* (2018).

[11] Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 3988–3995.

[12] David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016).

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[14] Astrid Ipsen, Jeppe Lindberg W., Jonas Lindberg C., and Ning An. 2022. *Investigating Properties of Selected Knowledge Graph Embedding Methods*. Aalborg, Denmark. https://projekter.aau.dk/projekter/da/studentthesis/investigating-properties-of-selected-temporal-knowledge-graph-embedding-methods(4c090514-3055-461f-b357-863e2aed53f2).html%7D,howpublished={univeristyassignment},organization={AalborgUniversity},numpages={14}

[15] Idil Ismiguzel. 2021. Hyperparameter Tuning with Grid Search and Random Search. https://towardsdatascience.com/hyperparameter-tuning-with-grid-search-and-random-search-6e1b5e175144

[16] Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. *Advances in neural information processing systems* 31 (2018).

[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.

[19] Jan Larsen, Lars Kai Hansen, Claus Svarer, and M Ohlsson. 1996. Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*. IEEE, 62–71.

[20] Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion proceedings of the the web conference 2018*. 1771–1776.

[21] Petro Liashchynskyi and Pavlo Liashchynskyi. 2019. Grid search, random search, genetic algorithm: a big comparison for NAS. *arXiv preprint arXiv:1912.06059* (2019).

[22] SGOPAL Patro and Kishore Kumar Sahu. 2015. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462* (2015).

[23] Lutz Prechelt. 2002. Early stopping-but when? In *Neural Networks: Tricks of the trade*. Springer, 55–69.

[24] Omer Sagi and Lior Rokach. 2018. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 4 (2018), e1249.

[25] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. 2017. Activation functions in neural networks. *Towards Data Sci* 6, 12 (2017), 310–316.

[26] Dan Simon. 2013. *Evolutionary optimization algorithms*. John Wiley & Sons.

[27] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems* 25 (2012).

[28] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. 697–706.

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

[30] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[31] David H Wolpert. 1992. Stacked generalization. *Neural networks* 5, 2 (1992), 241–259.

[32] Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. 2019. Temporal knowledge graph embedding model based on additive time series decomposition. *arXiv preprint arXiv:1911.07893* (2019).

[33] Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. 2020. TeRo: A time-aware knowledge graph embedding via temporal rotation. *arXiv preprint arXiv:2010.01029* (2020).

[34] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).

[35] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.