Automated Fact Checking Using A Knowledge Graph-based Model

Arghya Kundu and Uyen Trang Nguyen
Department of Electrical Engineering and Computer Science
York University, Toronto, Canada
{arghyak, unguyen}@yorku.ca

***Abstract*—Misinformation is a growing threat to the economy, social stability, public health, democracy, and national security. One of the most effective methods to combat misinformation is fact checking. Fact checking is the process of verifying the factual accuracy of a statement or claim. Fact checkers employ rigorous methodologies to scrutinize claims, verify sources, and expose falsehoods. However, the huge volume of content circulating online makes it challenging for humans to identify misinformation manually. Automated tools can analyze large datasets to detect patterns in misinformation content, scaling up fact checking efforts. This paper proposes a knowledge graph-based fact checking model that uses two separate knowledge graphs, one containing true claims and the other, false claims. The model uses knowledge graph embeddings which are based on convolutional neural networks. The deep learning model is trained on the above two knowledge graphs to learn distinguishing patterns between true and false claims. Additionally, we employ explainable artificial intelligence (XAI) techniques to provide explanations for the model's classification, reducing cost of errors and increasing transparency and user trust in the system.**

***Index Terms*—Misinformation, disinformation, fake news, fact checking, knowledge graphs, machine learning, natural language processing.**

I. Introduction

Misinformation is a pervasive and serious problem with far-reaching consequences. Misinformation poses a threat to democratic processes by manipulating public opinion and influencing elections [1]. When false information is deliberately spread, it erodes trust in institutions, leading to social divisions and polarization. Furthermore, the impact of misinformation extends to public health and safety. Misinformation about health conditions and treatments can result in harmful behavior and a decline in vaccination rates [2]. This issue has been particularly evident during the COVID-19 pandemic, where misinformation has hindered public health efforts and caused confusion among the general public. For example, a rumor caused the death of 800 people after the consummation of alcohol-based cleaning products as a cure for Covid-19 [3].

Thus, misinformation identification is very important. Misinformation identification, specifically through fact checking, plays a crucial role in mitigating the spread of false information. Fact checking is the process of verifying the factual accuracy of a statement or claim. Fact checkers help combat the spread of misinformation and promote informed decision making among the public [4]. Fact checking organizations and independent researchers employ rigorous methodologies to scrutinize claims, and expose falsehoods.

Recognizing the time consuming and difficult nature of manual fact checking, automated tools using advancements in machine learning and natural language processing (NLP) have emerged as valuable resources [5]. These tools can analyze large datasets, detect patterns, and identify misinformation at a much faster pace, assisting humans in the process of fact checking.

Existing methods for fact checking employ a range of techniques [6], including language analysis, evidence retrieval techniques and knowledge graphs. These approaches use different sources of evidence for fact checking.

Among these methods, one approach that has gained significant attention is the use of knowledge graphs (KGs) [7], [8]. Knowledge graphs are structured representations of facts that capture relationships between entities and their attributes. In the context of knowledge graphs, a triplet is a fundamental unit of information, consisting of three components ('h', 'r', 't'): a head entity ('h'), a tail entity ('t'), and 'r' signifies the relationship between them.

These methods use graph embedding methods to distinguish between true and false claims [9]. One advantage of the knowledge graph approach is its ability to capture complex relationships and dependencies between entities [9]. Furthermore, knowledge graphs can be continuously updated with new information, ensuring the system remains adaptable.

In this paper, we develop a fact checking system that uses knowledge graphs to verify the veracity of claims. To achieve this, we propose a model consisting of two KGs: one created using true claims and the other created using false claims. Two separate models are trained on these two KGs using convolutional neural networks (CNN)-based knowledge graph embedding (KGE), enabling the system to distinguish between of true and false claims.

In this study we limit fact checking to the following: given a claim, classify it as true or false. For instance, given a claim, *"Earth has not warmed for the last 17 years."*, our fact checking model classifies the claim as false. Additionally, it points out the false and true triplets in the claim in a colour coded manner and provides the confidence scores for each triplet. Figure 1 shows a sample output of our model given a claim along with the confidence scores.

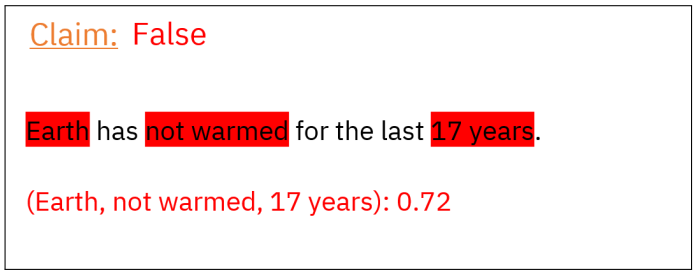

Figure 1. Sample output of our model given a claim

The remainder of this paper is structured as follows: Section III presents our methodology and the proposed fact checking model. Section IV discusses the XAI features. Section V describes the dataset. Section VI presents evaluation results and analysis. Section VII discusses limitations of the research presenyed in this paper. Section VIII concludes the paper.

II. Related Work

Previous research has demonstrated the potential of knowledge graphs in improving fact checking. For instance, Shiralkar [10] used a method that flows information from the subject to the object through different paths in the graph and compares the amount of information with a threshold to decide the truthfulness. Shi [8] proposed a method that finds different paths from the subject to the object in the graph and ranks them based on how well they can separate true and false claims. The paper then combines the ranked paths into a feature vector and feeds it to a binary classifier to predict the veracity of the content. Ciampaglia [7], proposed a weighted knowledge stream/segment based method to check the truthfulness of a given claim.

Nevertheless, existing models use knowledge graphs build from only true news/claims. However, by using additional knowledge graphs built from false content, a wider range of information and relationships can be captured. However, limited research has been done in this direction. Pan [11] used two knowledge graphs built from fake news and true news and trained two TransE models on the knowledge graphs for fake news detection. They used the max bias of triplets for detecting the veracity of news articles. However, their approach of using news articles to construct the knowledge graphs has limitations. Fake news may contain true claims, resulting in the inclusion of such claims in the fake news knowledge graph. This can create confusion and inaccuracies in the representation of knowledge. Additionally, their use of a translational embedding model (TEM) has several limitations that make it less suitable for fact checking. TEM struggles with modeling complex relationships beyond simple translations, and it is unable to handle one-to-many or many-to-one relations, as well as symmetric relations [12], [13]. Fact checking on the other hand, requires addressing inconsistencies, understanding complex relationships, and performing inference. For these purposes, more advanced deep learning-based embedding models that incorporate logical reasoning and semantic constraints are more appropriate for fact checking.

III. Methodology

In this section, we describe the methodology of the proposed model in detail. We present the steps used for creation and storage of the knowledge graphs, the classification models used and the training setup of the models.

A. Overview of ConvE

ConvE (convolutional 2D knowledge graph embedding) is a popular model in the field of knowledge graph embedding that uses convolutional neural networks (CNNs) to learn embeddings. Introduced by Dettmers [14], ConvE combines the strengths of CNN and knowledge graph embeddings to achieve competitive performance across various knowledge graph-related tasks, including link prediction and entity classification. The core idea revolves around using 2D convolutional layers to capture local patterns and interactions among entities and relations within a knowledge graph.

ConvE represents entities and relations in a knowledge graph using one-dimensional embeddings of size d. This compact embedding representation significantly reduces memory requirements and enables efficient processing of large-scale knowledge graphs. ConvE's architecture consists of three main components: an input layer, a convolutional layer, and a fully connected layer. The input layer reshapes the embeddings of the head entity, relation, and tail entity into two-dimensional matrices. The convolutional layer applies multiple filters to these matrices, capturing local patterns and interactions. The resulting feature maps are then flattened and fed into a fully connected layer, which generates the final embeddings.

ConvE strikes a favorable balance between embedding quality and computational complexity. By using significantly fewer parameters compared to fully connected neural networks, ConvE is well-suited for resource-constrained environments and handling large knowledge graphs. This is particularly advantageous when dealing with limited computational resources or processing large-scale datasets [14].

B. Knowledge Graph Creation

For creating the knowledge graphs, we use true and false claims seperately. From the true claims we create the true claims knowledge graph (TCKG) which is later used to train the true claims ConvE model (TCCM), and the false claims are used to create the false claims knowledge graph (FCKG) which is later used to train the false claims ConvE model (FCCM).

For creating the knowledge graphs, we first pre-process the claims, then we extract triplets. We use these triplets to create the knowledge graphs. The following sections details the individual processes.

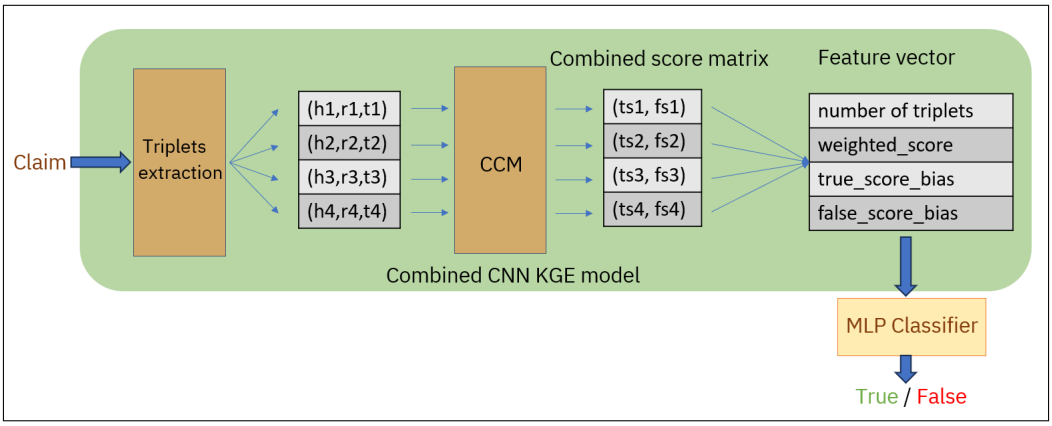

Figure 2. The overall architecture of the classification model

1) Data pre-processing: Pre-processing is an essential step in building a knowledge graph from a corpus. The following are the pre-processing steps involved,

- *Tokenization*: Split each sentence into individual words or tokens.
- *Co-reference resolution*: Resolve co-references in the text to determine which pronouns or noun phrases refer to the same entity. This step helps in establishing consistent references and connections between entities.
- *Lemmatization*: Convert each token to its base or dictionary form (lemma) to normalize the words. This step reduces the inflectional forms of words to their common base form, which helps in entity matching and reducing redundancy.
- *Stop word removal*: Remove common words, known as stop words, that do not carry significant semantic meaning and are not useful for extracting information. Examples of stop words include “and”, “the”, “is” etc.

2) Triplet extraction: For triplet extractions from the pre-processed data, we use the following steps:

- *Named entity recognition (NER)*: NER is a crucial task that identifies mentions of named entities in each text, such as people, organizations, locations, and other types. The entities recognized by NER serve two main purposes within the knowledge graph construction process. Firstly, they can be used to generate new candidate nodes, allowing the incorporation of the identified entities into the graph. Secondly, these entities can be linked to existing nodes through the named entity linking task, establishing connections and enriching the knowledge graph.
- *Named entity linking (NEL)*: NEL associates mentions of entities in a text with the existing nodes of a target knowledge graph. This task involves resolving and disambiguating entity mentions and linking them to their respective entities in a knowledge base. Additionally, as part of the NEL, entity disambiguation (ED) is also performed to determine the correct entity for each mention. To do this, we use the Wikification process, where Wikipedia is used as a knowledge base to disambiguate entities. However, due to the potential presence of multiple entities in Wikipedia with similar titles, we include both the Wikipedia entity title and their unique ID in the resulting linkage. In cases where a Wikipedia entry is not available, such as for very recent entities, a randomy generated unique ID is provided to represent the entity in the knowledge graph.
- *Relation extraction (RE)*: RE identifies and extracts relationships between entities mentioned in the text. This task involves two subtasks, parts-of-speech (POS) tagging and dependency parsing. POS tagging assigns a grammatical category (e.g., noun, verb, adjective) to each word or token, providing contextual information about their functions within a sentence. Dependency parsing analyzes the sentence’s grammatical structure, revealing the relationships and syntactic dependencies between words. By combining the outputs of POS tagging and dependency parsing, the relation extraction process accurately identifies and extract meaningful relationships between entities.
- *End-to-end triplet extraction* Triplet extraction is a challenging task that requires dealing with complex linguistic phenomena, such as long-distance dependencies and syntactic variations. Along with the previous mentioned steps, we use REBEL [15], an end-to-end sequence-to-sequence model that generates relation triplets from raw text, without relying on predefined relation schemas or hand-crafted features. REBEL is based on BART, a pre-trained language model that can both encode and decode natural language. REBEL can handle more than 200 different relation types and has been pre-trained on various RE benchmarks, including a large-scale distantly supervised dataset curated by the authors of REBEL. We merge the triplets extracted earlier with those obtained using REBEL, retaining only the distinct triplets.

3) Regular updating of knowledge graphs: By continuously incorporating new information, the knowledge graphs

remain up to date with the latest knowledge. To this end, we collect both true and false claims from three different fact-checking websites, namely "Polygraph.info", "Politfact.com" and "Snopes.com". We use the previously mentioned pre-processing steps and the triplet generation pipeline for updating the knowledge graphs.

- "Polygraph.info" [1] is a fact-checking website that focuses on debunking false or misleading claims with a particular emphasis on disinformation related to global affairs.
- "Politifact.com" [2] is a highly reputable fact-checking platform known for its thorough evaluation and rating of the accuracy of various claims, ensuring that readers have access to reliable information and aiding in the fight against misinformation.
- "Snopes.com" [3] is a well-known website that verifies and debunks urban legends, rumors, and various misinformation circulating online.

C. Overall Architecture of the Proposed Model

Figure 2 depicts the overall architecture of our proposed model. Given a new claim C, we follow a series of steps to process the claim and make a prediction. The steps are described as follows:

1) *Triplets extraction:* Firstly, we extract triplets from the claim. Each triplet is represented as (h, r, t), where 'h' represents the head entity or subject, 'r' represents the relation or predicate, and 't' represents the tail entity or object.
2) *Combined ConvE model:* These extracted triplets vector is then passed through the combined ConvE model (CCM) to produce the 2D score matrix. Where each record in the matrix denotes the scores of the corresponding triplet. For a given triplet (h,r,t), the score is represented as (ts, fs), where ts is the score generated by the true claims ConvE model (TCCM) and fs is the score generated by the false claims ConvE model (FCCM). The scores in the matrix are determined by the CCM's ability to learn the underlying patterns and relationships between the entities and relations, enabling it to assess the plausibility of each triplet based on the given claim.
3) *Feature vector generation:* From the 2D score matrix obtained from the CCM, we derive a 1D feature vector. This feature vector serves as a condensed representation of the extracted triplets and their respective scores. It captures the important information necessary for making predictions.
4) *Classifier:* Finally, we pass the feature vector to a trained multi-layer perceptron (MLP) classifier. The MLP classifier is a supervised learning algorithm used for classification tasks. It takes the feature vector as input and predicts the final outcome or label for the given claim. The MLP classifier has been trained on labeled data to learn the patterns and characteristics of true and false claims.

[1] www.polygraph.info
[2] www.politifact.com
[3] www.snopes.com

The following sections will provide detailed descriptions of each individual module in our model, shedding light on their inner workings and the techniques.

D. Combined ConvE Model

First, we create two separate ConvE models, one from true claims, referred to as the true claims ConvE model (TCCM) and the other from false claims, referred to as the false claims ConvE model (FCCM). Subsequently, we build the combined ConvE model (CCM) using the combination of TCCM and FCCM. The combined ConvE model (CCM) incorporates the true claims ConvE model (TCCM) and the false claims ConvE model (FCCM) to improve performance on knowledge graph-related tasks. By training these models separately on the true claims knowledge graph (TCKG) and the false claims knowledge graph (FCKG) respectively, we use the strengths of both models and achieve higher accuracy, precision, and recall. These models are trained independently and then combine TCCM and FCCM in parallel to form the CCM, refer Figure 3. The combined ConvE model (CCM) takes advantage of the complementary information learned from TCCM and FCCM.

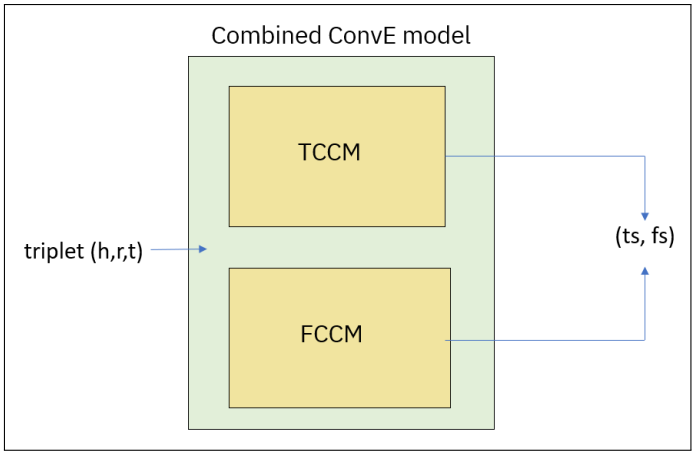

Figure 3. Combined ConvE model for one triplet (h,r,t), where ts and fs are the scores calculated for the triplet from TCCM and FCCM respectively.

E. Score Generation for One Triplet

The high level ConvE architecture consists of an encoding component and a scoring component. Given an input triplet (h, r, t), the encoding component maps entities $h, t \in E$ to their distributed embedding representations $e_h, e_t \in R^d$, with d being the embedding dimension.

In the scoring component, the two entity embeddings e_h and e_t are scored by a function ψ_r. The score of a triple (h, r, t) is defined as $\psi(h, r, t) = \psi_r(e_h, e_t) \in R^d$.

Formally, the scoring function is defined as Equation 1:

$$\psi_r(e_h, e_t) = f(vec(f([\bar{e_h}, \bar{r_r}] * \omega))\mathbf{W})e_t \quad (1)$$

where $r_r \in R^d$ is a relation parameter depending on r, $\bar{e_h}$ and $\bar{r_r}$ denote a 2D reshaping of e_h and r_r, respectively: if $e_h, r_r \in R^d$, then $\bar{e_h}, \bar{r_r} \in R^{d_w \times d_h}$, where $d = d_w \times d_h$.

In the feed-forward pass, the model performs a row-vector look-up operation on two embedding matrices, one for entities, denoted $\mathbf{E}^{\varepsilon \times d}$ and one for relations, denoted $\mathbf{R}^{R \times d'}$, where d and d' are the entity and relation embedding dimensions, and $|\varepsilon|$ and $|R|$ denote the number of entities and relations.

The model then concatenates e_h and r_r, and uses it as an input for a 2D convolutional layer with filters ω. Such a layer returns a feature map tensor $\tau \in R^{c \times m \times n}$, where c is the number of 2D feature maps with dimensions m and n. The tensor τ is then reshaped into a vector $vec(\tau) \in R^{c \times m \times n}$, which is then projected into a d-dimensional space using a linear transformation parameterised by the matrix $\mathbf{W} \in \mathbf{R}^{c \times m \times n \times d}$ and matched with the object embedding e_t via an inner product. The parameters of the convolutional filters and the matrix $\mathbf{W}$ are independent of the parameters for the entities h and t and the relationship r. After which we apply the sigmoid function, $\sigma(\cdot)$ to the scores, that is $p = \sigma(\psi_r(e_h, e_t))$

It is important to note that the parameters for the convolutional filters and the matrix $\mathbf{W}$ are independent of the parameters for the entities and relations. This means that they are learned separately during the training process and do not directly depend on the entity and relation embeddings.

To generate the combined score, the CCM first passes the triplet through TCCM and FCCM independently, as shown in Figure 3. Each model calculates a score based on its learned embeddings, capturing the relationship between the head entity, relation, and tail entity in the triplet. These scores represent the models' confidence for the likelihood of the triplet, with higher scores indicating a higher likelihood of the triplet being true or false according to each model's perspective.

Next, scores from TCCM and FCCM are passed through an aggregator function, which produces a tuple (ts, fs), where ts represents the score from TCCM and fs represents the score from FCCM.

F. Score Generation for a Claim

Each claim may consists of multiple triplets. A given claim C can be expressed as a set of triplets $T = \{ t1, t2, t3, \ldots, tn \}$. Each of the triplets (t) are passed through the scoring module to produce the score of a claim (S), as shown in Equation 2.

$$S = \{\text{score}(t) : t \in T\} \tag{2}$$

Here, S is the 2D matrix of scores, and score(t) denotes the score generated for each individual triplet t within the claim as shown in Figure 4.

G. Feature Vector Extraction

From the 2D score matrix we extract feature vector $\mathbf{V}$, the input to the classifier.

1) Number of triplets: This denotes the number of triplets present in the claim.

$$num_{triplets} = |[S]| \tag{3}$$

where S is the 2D score matrix.

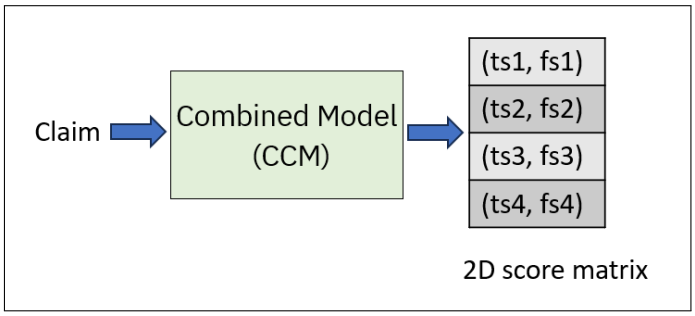

Figure 4. Combined score calculation of a claim

2) Weighted combined score: This feature represents the degree of veracity of the triplets within the claim. Additionally, the weight component factors in the importance of a subject in the triplet relative to the claim.

Consider a triplet (h, r, t). The combined score (cs) is first derived by:

$$cs = |ts - fs| \tag{4}$$

where (ts, fs) represents the tuple of scores of the triplet (h, r, t) obtained from the combined model CCM. Let W represent the frequency of the head entity of the triplet relative to the claim, such that $W \in \{|e_h|, |e_t|\}$. The vectors $\mathbf{W}$ and CS consist of the weights (W) and combined scores (cs) of all the triplets in the claim, i.e., $\mathbf{W} = [W]$ and $CS = [cs]$. Here, $|W| = |CS| = num_{triplets}$.

The weighted combined score (WCS) is computed as the dot product of $\mathbf{W}$ and CS, as depicted by the formulae:

$$WCS = \mathbf{W} \cdot CS \tag{5}$$

3) True score bias: This is defined as follows,

$$T_{bias} = max(|ts_i - ts_j|) \tag{6}$$

where $i \neq j$ and $1 \leq i, j \leq \text{num}_{\text{triplets}}$.

This represents the maximum score difference among all the triplets in respect to the true claims ConvE model (TCCM). This is used to denote the degree of variation of truthfulness of the triplets.

4) False score bias: This is defined as follows,

$$F_{bias} = max(|fs_i - fs_j|) \tag{7}$$

where $i \neq j$ and $1 \leq i, j \leq \text{num}_{\text{triplets}}$.

This represents the maximum score difference among all the triplets in respect to the false claims ConvE model (FCCM). This is used to denote the degree of variation of falsehood of the triplets.

H. Classification of Claims

The feature vector is then passed through a multi-layer perceptron (MLP) classifier, which predicts the veracity of the claim. The MLP classifier is a feed-forward neural network used in supervised learning for classification tasks. It comprises three or more layers of nodes: an input layer, one or more hidden layers, and an output layer. MLP is capable of modeling complex, non-linear relationships in data and is often utilized in a variety of classification problems. In this study, we use a MLP classifier with two hidden layers with (5,2) nodes respectively.

IV. Explainability of the Model

Given a claim, we represent the claim as a set of triplets. $T = \{t_1, t_2, t_3 \ldots, t_n\}$. For every triplet ($t_i$), we find the combined score tuple (ts, fs).

Given triplet t_i, the triplet veracity TV_i and confidence score CS_i of the model are calculated as folows:

$$TV_i = \begin{Bmatrix} \text{True}, & ts > fs \\ \text{False}, & fs > ts \end{Bmatrix} \tag{8}$$

$$CS_i = \begin{Bmatrix} ts, & ts > fs \\ fs, & fs > ts \end{Bmatrix} \tag{9}$$

Using TV_i we render the triplets as true or false in a color-coded scheme and use CS_i to provide the model's confidence score for each triplet, as illustrated in Figure 1.

V. Dataset

In this study we use the LIAR dataset. Liar dataset was created by Wang et al., (2017) [16], which is one of the largest fact checking datasets, containing 12,836 claims along with its veracity determined by human evaluators. The dataset is labelled with a discrete set of values from one to six, corresponding to pants-fire, false, mostly-false, half-true, mostly-true, and true. As our problem is based on binary class classification, we transform these into two labels. Pants-fire and false are contemplated as false and mostly-true, and true as true.

The dataset is split into 80:20 for training and testing. The training data contains 56% true and 44% false statements, and the testing data contains 56% true and 44% false statements. Thus, there is not much imbalance between the different labels.

The dataset provides some additional metadata like the subject, speaker, job, state, party, context, history. However, in the real-life scenario, we may not have this metadata always available. Therefore, we experiment only on the textual data of the dataset.

VI. Results and Discussion

All experiments are conducted on a machine with Intel(R) Core(TM) i7-8750H 2.20GHz CPU, 16 GB of RAM and NVIDIA GeForce RTX 4090 GPU running Windows 11.

Table I depicts the classification report of our model. We can observe from the table that precision is higher for "True" class whereas recall is higher for "False" class. A higher precision value for the "True" class means that the model has a lower tendency to make false positive predictions for true claims. It indicates that when the model predicts the veracity of a claim as true (positive prediction for the "True" class), it is more likely to be correct. In other words, the model is better at avoiding false claims or misinformation while identifying true claim instances. At the same time a higher recall for the "False" class indicates that the model is better at capturing most of the actual false claims present in the dataset, reducing the risk of false negatives for false information.

Table I
Classification performance of the proposed model

	Precision	**Recall**	**F1 score**
Class True	**0.88**	0.86	0.86
Class False	0.84	**0.92**	**0.87**
Macro	0.86	0.89	**0.87**

A. Ablation Study

To study the contribution of the individual knowledge graphs we carry out ablation experiments. The experimental results are shown in Table II.

The ablation experiments include the following three variants of the proposed model:

- With only true claims knowledge graph (TCKG)
- With only false claims knowledge graph (FCKG)
- With both knowledge graphs

For the first two experiment the respective scores from the FFCM and TFCM are set to zero for each triplet.

Table II
Ablation study results

Our model	**Precision**	**Recall**	**Accuracy**	**F1 Score**
With only False KG	0.78	0.83	0.76	0.80
With only True KG	0.84	0.81	0.81	0.82
With both KGs	**0.86**	**0.89**	**0.89**	**0.87**

From Table II, we can observe that with the accuracy of with only false claims knowledge graph is the least. This can be attributed to the fact that the knowledge graph constructed from false claims lacks the necessary factual relationships and context, leading to limitations in identifying reliable patterns and distinguishing false claims accurately. Without a comprehensive representation of true claims and their associations, the model's ability to differentiate between reliable and misleading information is compromised.

On the other hand, the accuracy with only the true knowledge graph is higher, showcasing the importance of a knowledge graph built from verified and reliable claims. The true knowledge graph contains valuable structured information, capturing genuine relationships between entities and their attributes, which aids in the identification of trustworthy patterns in factual claims. Consequently, the model trained on the true knowledge graph demonstrates improved performance in detecting true claims with higher precision and accuracy.

However, the highest accuracy is achieved when the model uses both the true claims knowledge graph and the false claimns knowledge graph, with an increase of almost 8% to that of only using true knowledge graph. Integrating knowledge from both sides provides a more comprehensive and balanced representation of information. By combining the knowledge of true and false claims, the model gains a broader understanding of the entire information landscape. This enables the combined model CCM to effectively leverage contrasting patterns and identify distinctive features between reliable and misleading claims.

B. Comparison with Baseline Models

We conducted a comparative analysis of our proposed model against six state-of-the-art MDM classification baseline models, all of which were evaluated using the same LIAR dataset by their authors as documented in their respective publications.

- Jayaraman [17]: XLNet base fine-tuning model for fake news detection. This model uses permutation language modeling to capture the right and left context information jointly and computes contextualized input representation.
- Khan [18]: Explored several advanced pre-trained language models for misinformation detection along with the traditional and deep learning ones. They found that that BERT and similar pre-trained models perform the best for misinformation detection, especially RoBERTa for the LIAR dataset.
- Mehta [19]: In this work BERT is fine-tuned for domain-specific datasets, incorporating human justification and metadata to enhance model performance. The approach involves employing multiple BERT models with shared weights to manage various inputs and use extra justification and metadata.
- Yang [20]: An unsupervised learning framework, UFD, which utilizes a probabilistic graphical model to model the truths of news and the users' credibility. An efficient collapsed Gibbs sampling approach is proposed to solve the inference problem.
- Bhatt [21]: The authors used an enhance sequence model with four branches having inputs as statement (S branch), metadata (M branch), justification (J branch) and credit score (C branch) for classification. The authors use credibility information and metadata associated with the article.
- Nida [22]: The authors used a multi-modal deep learning model for misinformation detection. For the textual attributes Bi-LSTM-GRU-dense deep learning model was used, while for the remaining attributes, dense deep learning model was used.

Table III
PERFORMANCE COMPARISON BETWEEN THE PROPOSED MODEL AND STATE-OF-THE-ART MODELS

Model	Accuracy	Precision	Recall	F1 Score
Khan [18]	0.62	0.63	0.62	0.63
Jayaraman [17]	0.72	0.52	0.39	0.45
Mehta [19]	0.74	0.70	0.85	0.77
Yang [20]	0.76	0.75	0.75	0.75
Nida [22]	**0.89**	**0.87**	0.88	**0.87**
Bhatt [21]	0.82	0.73	0.71	0.72
Our model	**0.89**	0.86	**0.89**	**0.87**

Table III presents a comparison between our approach and baseline models. The results suggest that the proposed model demonstrates superior performance compared to the majority of state-of-the-art approaches. Notably, our model achieves an accuracy of 89%, which is on par with the best performing model presented by Nida [22].

Importantly, it should be highlighted that Nida [22] used supplementary features, including author information and content metadata, which might not always be available in real world scenarios. In contrast, our model exclusively relies on textual features of the content.

VII. LIMITATIONS OF THE MODEL

Our model has the following limitations:

- The accuracy of our model heavily depends on the quality and completeness of the underlying knowledge graphs. Inaccurate or biased information within these graphs could lead to erroneous predictions.
- The interpretability provided by explainable artificial intelligence techniques might not always cover all aspects of the model's decision making process. For instance, it does not attribute the sources of the information.
- Although our model benefits from periodic updates to the knowledge graphs, it remains constrained by the intervals between updates.

VIII. CONCLUSION

The paper presents a method for fact checking using CNN-based knowledge graph embeddings. This method involves the creation of two KGs constructed from verified true and false claims, respectively, which are later used for training two separate KGE models. By employing the ConvE technique, which is based on convolutional neural networks, in conjunction with feature vector generation and classification methods, the model predicts the veracity of claims. Experimental results show that the proposed model outperforms the baseline models.

The output of the model is further enhanced by XAI techniques. True and false triplets are highlighted by different colors, green and red, respectively. In addition, the model generates confidence scores for triplets. XAI output allows end users to understand the reasons behind the model's predictions.

This work can be extended in the following directions:

- To evaluate the model's performance across a broader range of datasets, especially those from different domains.
- To investigate further into XAI techniques for the model's predictions.

References

[1] H. Allcott and M. Gentzkow, "Social media and fake news in the 2016 election," *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 211–36, May 2017. [Online]. Available: https://www.aeaweb.org/articles?id=10.1257/jep.31.2.211

[2] S. Lewandowsky, U. K. H. Ecker, C. M. Seifert, N. Schwarz, and J. Cook, "Misinformation and its correction: Continued influence and successful debiasing," *Psychological Science in the Public Interest*, vol. 13, no. 3, pp. 106–131, 2012. [Online]. Available: http://www.jstor.org/stable/23484653

[3] A. Coleman. (2020) 'hundreds dead' because of covid-19 misinformation. Accessed: 2021-07-15. [Online]. Available: https://www.bbc.com/news/world-53755067

[4] G. Pennycook and D. G. Rand, "Fighting misinformation on social media using crowdsourced judgments of news source quality," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 116, no. 7, pp. 2521–2526, 2019.

[5] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *SIGKDD Explor. Newsl.*, vol. 19, no. 1, p. 22–36, sep 2017. [Online]. Available: https://doi.org/10.1145/3137597.3137600

[6] E. Lazarski, M. Al-Khassaweneh, and C. Howard, "Using nlp for fact checking: A survey," *Designs*, vol. 5, no. 3, 2021. [Online]. Available: https://www.mdpi.com/2411-9660/5/3/42

[7] G. L. Ciampaglia, P. Shiralkar, L. M. Rocha, J. Bollen, F. Menczer, and A. Flammini, "Computational fact checking from knowledge networks," *PLOS ONE*, vol. 10, no. 6, pp. 1–13, 06 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0128193

[8] B. Shi and T. Weninger, "Discriminative predicate path mining for fact checking in knowledge graphs," *Knowledge-Based Systems*, vol. 104, pp. 123–133, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705116300570

[9] P. Lin, S. Qi, Y. Wu, and J. Pi, "Discovering patterns for fact checking in knowledge graphs," *Journal of Data and Information Quality*, vol. 11, pp. 1–27, 05 2019.

[10] P. Shiralkar, A. Flammini, F. Menczer, and G. L. Ciampaglia, "Finding streams in knowledge graphs to support fact checking," 2017.

[11] J. Z. Pan, S. Pavlova, C. Li, N. Li, Y. Li, and J. Liu, "Content based fake news detection using knowledge graphs," in *International Workshop on the Semantic Web*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:52900831

[12] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13. Red Hook, NY, USA: Curran Associates Inc., 2013, p. 2787–2795.

[13] Y. Dai, S. Wang, N. N. Xiong, and W. Guo, "A survey on knowledge graph embedding: Approaches, applications and benchmarks," *Electronics*, vol. 9, no. 5, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/5/750

[14] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[15] P.-L. Huguet Cabot and R. Navigli, "REBEL: Relation extraction by end-to-end language generation," in *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 2370–2381. [Online]. Available: https://aclanthology.org/2021.findings-emnlp.204

[16] W. Y. Wang, ""liar, liar pants on fire": A new benchmark dataset for fake news detection," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 422–426. [Online]. Available: https://aclanthology.org/P17-2067

[17] A. K. Jayaraman, T. Trueman, and E. Cambria, "Fake news detection using xlnet fine-tuning model," 11 2021, pp. 1–4.

[18] J. Y. Khan, M. T. I. Khondaker, S. Afroz, G. Uddin, and A. Iqbal, "A benchmark study of machine learning models for online fake news detection," *Machine Learning with Applications*, vol. 4, p. 100032, jun 2021. [Online]. Available: https://doi.org/10.1016%2Fj.mlwa.2021.100032

[19] D. Mehta, A. Dwivedi, A. Patra, and M. Kumar, "A transformer-based architecture for fake news classification," *Social Network Analysis and Mining*, vol. 11, 12 2021.

[20] S. Yang, K. Shu, S. Wang, R. Gu, F. Wu, and H. Liu, "Unsupervised fake news detection on social media: A generative approach," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 5644–5651, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4508

[21] S. Bhatt, N. Goenka, S. Kalra, and Y. Sharma, "Fake news detection: Experiments and approaches beyond linguistic features," in *Data Management, Analytics and Innovation*. Springer Singapore, sep 2021, pp. 113–128.

[22] N. Aslam, I. Khan, F. Alotaibi, L. Aldaej, and A. Aldubaikil, "Fake detect: A deep learning ensemble model for fake news detection," *Complexity*, vol. 2021, pp. 1–8, 04 2021.