EnFVe: An Ensemble Fact Verification Pipeline

John Joy Kurian*, Deborah Zenobia Rachael Menezes*, Avinash Ronanki*,
Gaurang Sharma*, Sandeep Krishna Prasad*, Ashish Chouhan§, Ajinkya Prabhune§
SRH University
Heidelberg, Germany
Email *{john.joykurian, DeborahZenobiaRachael.Menezes, avinash.ronanki, gaurang.sharma, sandeepkrishna.prasad}
@stud.hochschule-heidelberg.de,
§{ashish.chouhan, ajinkya.prabhune}@srh.de

Abstract—**Recent years have seen exponential growth in the volume of data generated. An undesirable consequence of this rapid expansion is the alarming rise of misinformation that is spreading throughout social networks and publishing platforms. Fact verification is the act of verifying the correctness of a particular statement. Manual fact-checkers find it increasingly difficult to keep up with the rapid proliferation of fake news in the information ecosystem. Fact verification pipelines aim at assisting the fact-checkers. These pipelines involve retrieving potentially relevant documents for a given claim, checking the reliability of the document media sources, evaluating the significance of each document, and verifying the correctness of given claims. However, there are a few limitations to these pipelines. The first limitation is that they treat fact verification as a stance detection or textual entailment task and places less emphasis on pure reasoning as the basis of fact verification. The second limitation is the robustness of the models. The fact verification models do not perform well on real-world data due to the text constructed using a myriad of complex ways. The reason for low accuracy is due to the evaluation and training of models on synthetic data that does not resemble the real-world data. In this paper, we present EnFVe, an end-to-end pipeline that integrates various components of fact verification. EnFVe pipeline improves upon the existing pipelines by integrating: (1) a Data Acquisition module that routinely scrapes data for constructing and updating the ground truth, (2) Preprocessing modules such as Coreference Resolution and Sentence Simplification that improves the robustness of the verification, (3) a Continuous Retrieval Engine module that captures the semantic context of the text, and (4) an Ensemble module that consists of various state-of-the-art fact verification models equipped with hyperparameter optimization. EnFVe achieves an accuracy of (dev/test) 79.26%/73.28%, which is better as compared to the state-of-the-art fact verification pipelines.**

Index Terms—**FEVER, fact verification pipeline, fact-checking and reasoning, fake news detection.**

I. Introduction

The emergence of fake news illustrates the weakening in long-standing structural bulwarks toward disinformation in the digital era. Concern over this problem is global, and addressing fake news requires a multidisciplinary effort [1]. This digital era that we live in allows us the opportunity to hear something mentioned from anybody, anytime, at any moment. Ironically, this flood of information makes it challenging to determine what is true. Vosoughi et al. [2] show that false news reaches more people than the truth. Currently, qualified humans mostly perform fact verification. Manual fact-checkers have difficulty keeping up with the wide dissemination of misinformation. Manual verification of facts involves evaluating speeches, debates, legislation, and published documents, thus combining them with the reasoning for reaching a verdict. Based on the complexity of the claim, the above-referred approach may take from an hour to a few days. Research in automating this process has been carried out in a variety of fields, including natural language processing, machine learning, knowledge representation, databases, and journalism. Thus, it has become an absolute necessity to have an automated approach for fact verification.

Over the last several years, the increasing research literature has established Automated Fact Verification (AFV) as an intriguing subject in the field of artificial intelligence. Initial works involved Natural Language Inference (NLI) models [3], [4] to label the relationship between a pair of premise and hypothesis as “entailment”, “contradiction”, or “neutral”. Although this is achieved typically by concatenating all evidence sentences into a single string, the approach fails to reason between isolated evidences. Research has been performed to capture linguistic features to differentiate between the journalistic and sensational way of reporting a claim to suggest fake news [5], [6]. Another similar approach by Zellers et al. [7] leverages Generative Adversarial Network (GAN) to generate neural fake news for a given headline. This generated fake news pretends to be trustworthy for humans because it mimics the journalistic style. Several other research work focuses on stance detection that aims in identifying whether a particular news headline “agrees”, “disagrees”, “discusses”, or “unrelated” to a particular news article in order to allow journalists and others to easily find and investigate possible instances of “fake news” [8], [9], [10], [11], [12]. One of the recent open competitions, Fake News Challenge (FNC-I)[1] aims to combat fake news using stance detection as their initial approach towards fact verification.

In order to mitigate the shortage of labelled datasets for fact verification, Wang et al. [13] provide Politifact[2], a large dataset of claims. This dataset has associated speaker-related metadata with each claim and the claim’s credibility rating provided by the editors on a scale of six verdicts (*true,*

[1]http://www.fakenewschallenge.org/
[2]https://www.politifact.com/

mostly true, half true, mostly false, false and pants-on-fire). However, one drawback is that it does not use the evidence and the justification provided by humans to predict the label. Thorne et al. [14] provide FEVER, a dataset facilitated for fact verification with emphasis on reasoning. This dataset facilitates the training of models for better explainability than the PolitiFact dataset. Usually, where false claims are propagated, the claims are lengthy and complicated statements with the context not always embedded [15], [16]. However, the claims are short sentences with the context completely embedded in the statement for the FEVER dataset. Therefore, the limitation of FEVER dataset is that the model trained with this dataset would only work for short sentences with context.

Although, considerable progress has been made towards AFV, research on an end-to-end pipeline for fact verification integrating various components is limited. Some of the current end-to-end pipelines [11], [8], [17] have certain limitations:

- Existing end-to-end pipelines treat fact verification as a stance detection task. The drawback of these approaches is that they place less emphasis on pure reasoning as the basis of fact verification.
- The fact verification models are not scalable for real-world data, as evaluation and training are performed using synthetic data.
- Current end-to-end pipelines do not continuously update the ground truth sources.

Our EnFVe pipeline improves on the existing end-to-end pipelines by introducing the below key differentiating features:

- We present an extensible ensemble model that consists of various state-of-the-art fact verification models that focus on pure reasoning, and a component for hyperparameter optimization of these models.
- We integrate two preprocessing steps, Coreference Resolution [18] and Sentence Simplification [19] that aims to transform large complex text bodies into syntactically simpler sentences.
- We introduce a Data Acquisition module that routinely scrapes data for constructing and updating the ground truth.

The rest of this paper is structured as follows: In Section II, we address the related contributions made towards fact verification. In Section III, we discuss the architecture of our pipeline EnFVe. In Section IV, we evaluate and benchmark various sections of the pipeline. In Section V, we conclude and recommend potential enhancements for EnFVe.

II. Related Work

In this section, we discuss the relevant contributions made towards fact verification.

Linguistic analysis and metadata-based methods: The existing contributions focus on detecting fake news in news articles. These approaches include identifying linguistic features such as capturing specific writing styles [5], exploiting publishers' emotions while posting information on social media, and user emotions while consuming the information for fake news detection [6]. Few approaches aim towards feature engineering using news source features such as identification of bias, credibility, and trustworthiness. Some other approaches focus on identifying the domain location and credibility of the news source. Reis et al. [20] present a supervised learning approach for fake news detection, where the focus is to analyze environment features such as statistics of user's engagement and identifying temporal patterns from social media about explainable fake news detection. The meta-attributes used are the news context and speaker.

PolitiFact Based Models: A novel approach based on the PolitiFact dataset is DeClarE [21]. DeClarE provides linguistic metrics, the trustworthiness of the sources, and external evidence from the web. This method does not require any feature engineering, lexicons, or other manual intervention. It considers the context of the claim through word embeddings and the language of web documents captured via a bidirectional LSTM (biLSTM). Additionally, it emphasizes parts of the claim using the attention mechanism. DeClarE then aggregates all the claim source information, web document context, attention weights, and trustworthiness of the underlying sources to assess the claim. It also derives informative features like source embeddings that capture trustworthiness and salient words through attention.

FEVER Based Models: The FEVER shared task [14] aims to develop AFV for veracity checking of human-generated claims by extracting evidence from Wikipedia. The FEVER dataset consists of 185,445 annotated Wikipedia instances. It is the largest benchmark dataset in this domain to the best of our knowledge.

The majority of participating teams in FEVER used a three-stage pipeline approach consisting of Document Retrieval, Evidence Sentence Selection, and Claim Verification [22], [23], [24]. In Document Retrieval, the named entities are extracted from a claim and used as a query to search Wikipedia through the Wikipedia search API. In the Evidence Sentence Selection stage, the similarity is measured between the claim and individual sentences from retrieved documents. The similarity is measured by converting the sentences to vectors and computing cosine similarity. Based on the similarity, the top k evidences are selected and fed to Claim Verification stage. In the Claim Verification stage, the veracity of the claim is checked, and labels such as "SUPPORTS", "REFUTES", or "NOT ENOUGH INFO" is assigned.

Enhanced Sequential Inference Model (ESIM) [25] is one of the most widely used NLI models in FEVER. UKP-Athene [24] uses an extended ESIM model to predict the entailment relation between the claim and input sentences. The claim-evidence pair is encoded separately using ESIM after which a pooling function is used to aggregate features for prediction. Yoneda et al. [26] use ESIM to infer the veracity of each claim-evidence pair. However, they incorporate an aggregation stage to make a final prediction by aggregating multiple predicted labels. Nie et al. [27] employs a semantic matching neural network for both evidence selection and claim verification. The GEAR model presented by Zhou et al. [22] is employed in EnFVe and utilizes Bidirectional Encoder Representations from Transformers (BERT) [28]. An attention mechanism learns contextual relations between words in a text, to achieve claim-specific representation for each evidence sentence. It also constructs an evidence graph network by identifying each evidence sentence as a node in the graph. A similar approach by Liu et al. [23] uses node kernels to measure the importance of evidence node and edge kernels to conduct fine-grained evidence propagation in the evidence graph network.

Fact Verification Pipelines: Perspectroscope [17] outputs the perspectives, their stances, and evidences on an input discussion-worthy natural language claim. They first construct the ground-truth of perspective sentences and evidence paragraphs using Wikipedia and news articles. This created ground truth then use information retrieval engines like Google Search API[3] and Elastic Search[4] for searching resources similar to the claim. Perspectroscope uses BERT for perspective extraction, stance classification, perspective equivalence, and extracting supporting evidence.

CredEye [8] is an automated credibility assessment system that takes a natural language claim as input from the user, and automatically checks its credibility by analyzing the related articles on the internet. The assessment of credibility is carried out in a five-step approach:

- *Retrieval* - Querying the web using Bing.
- *Stance Detection* - By checking the unigram and bigram correlation between the input claim and the obtained article snippet, followed by a logistic regression classifier trained on claims and evidence articles to determine the stance.
- *Content analysis* - For improving the credibility, CredEye captures the article linguistic style by eliminating the sensational style of reporting the claim using a logistic regression classifier.
- *Credibility aggregation* - Measure the trustworthiness of the source, evaluate, and rate the credibility of the source.
- *Evidence Extraction* - CredEye highlights the bigrams from the article snippet most relevant to the claim using a gradient of green (credible) and red (non-credible).

FAKTA [11] predicts the authenticity of given claims and provides evidence at the document and sentence level to explain its predictions. FAKTA uses Apache Lucene to search and access related Wikipedia articles and utilizes the Google API to filter through predefined collections of media sources dependent on their authenticity and reliability. Finally, the re-ranking model [29] is used to select the top-K relevant documents. The FAKTA approach integrates Bag Of Words (BOW) and Convolutional Neural Networks (CNNs) in a two-step hierarchical system. The first step determines whether the document is related or unrelated. The related documents are then transferred to the second step to define their stances: “agree”, “disagree”, or “discuss”.

The EnFVe pipeline discussed in this paper enhances the existing research by implementing the following key distinguishing features in the fact verification pipeline:

- We introduce an extensible ensemble modelling approach consisting of state-of-the-art fact verification models focused on reasoning with hyperparameter optimization.
- We combine two preprocessing steps: Coreference Resolution [18] and Sentence Simplification [19] to transform broad, complicated text bodies into syntactically simpler sentences.
- We incorporate a Data Acquisition module that routinely scraps data to construct and update the ground truth.

III. EnFVe Architecture

In this section, the architecture of EnFVe is described. Fig. 1 outlines the structure of the EnFVe pipeline that comprises of two primary workflows: Ground Truth Creation and Inference. The source code of the EnFVe pipeline is available online[5].

Ground Truth Creation: This workflow concerns with creating a regularly updated evidence retrieval engine. The Data Acquisition module deals with creating and updating the ground truth data from two sources: Wikipedia and news articles. An essential step is to pass the ground truth text data through the Preprocessing module as the FEVER dataset is a collection of short and straightforward sentences without frequent occurrences of pronouns. Therefore, the models would fail when it encounters syntactically-complex claims with pronouns referring to multiple entities. The Encoder module converts the preprocessed text data into a vector format that is subsequently inserted into a Vector Database, forming the base of the evidence retrieval engine.

Inference: The Inference workflow deals with the fact verification of a claim that user inputs. The user inputs the claim through the EnFVe User Interface (UI). The claim can be a single sentence or a long-form multi-paragraph text document. The claim is provided to the Preprocessing module that performs Coreference Resolution and Sentence Simplification. The Encoder then converts the preprocessed claim into a vector representation. Considering the vector representation and an Approximate Nearest Neighbor algorithm [see section III-C], the relevant evidences are fetched from the ground truth Vector Database. The claim as well as the evidences are fed into the Ensemble module to perform fact verification.

[3]https://cse.google.com/cse/
[4]https://www.elastic.co/
[5]https://github.com/JohnKurian/EnFVe

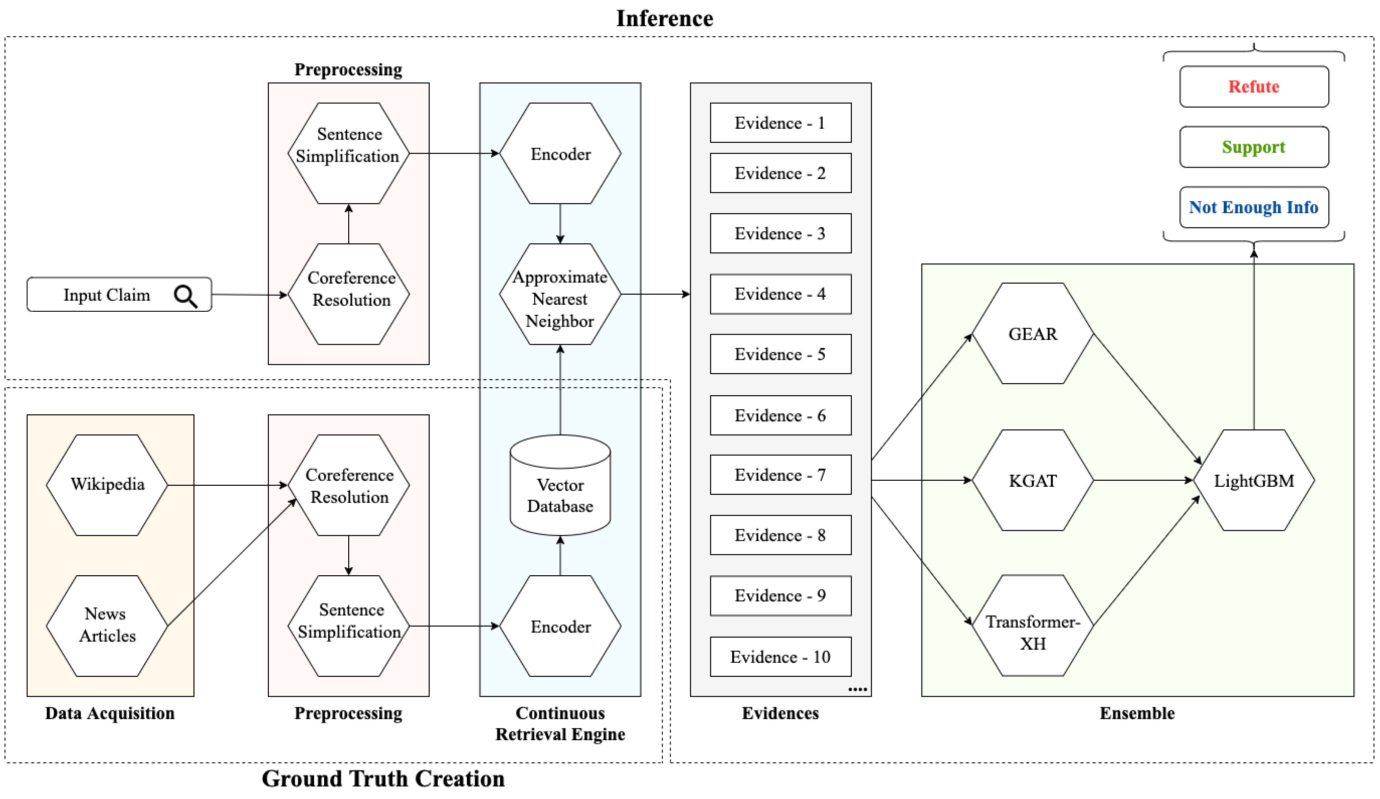

Fig. 1. EnFVe: Ensemble Fact Verification Pipeline

A. *Data Acquisition*

The Data Acquisition module deals with periodically scraping Wikipedia and news articles data from various sources and transforming it into a format that is suitable for the Preprocessing module.

- **Wikipedia:** Wikipedia provides an entire data XML format that is available online[6]. The file *enwiki-latest-pages-articles.xml* contains the latest updated Wikipedia articles in a compressed format. *wikiextractor*[7] is used to extract the text data in JSON format. We then keep the URL, text, and title fields. We use the URL field to provide the user with the link pointing to the source of the evidence through the UI.

- **News Articles:** We design a near real-time news article crawler for adding reputed news articles to the ground truth alongside Wikipedia. News API[8] provides an API where we can fetch the news articles published by eighty-two trusted sources[9]. The API, however, does not supply the full-text content of the news article. Hence, we use *newspaper3k*[10] library for fetching the text content from the article link retrieved from the News API. The full-text content contains boilerplate such as sitemap, the related articles section, and many more. These irrelevant entities were removed using the *justText*[11] python library.

B. *Preprocessing*

The Preprocessing module handles the essential transformations required for the inference. It primarily consists of Coreference Resolution and Sentence Simplification.

- **Coreference Resolution:** Coreference Resolution is the task of finding all expressions that refer to the same entity in a text. End-to-end Neural Coreference Resolution [18] is a neural network architecture that considers all possible spans as potential mentions in the document. The neural network also learns distributions over possible antecedents for each span, using aggressive pruning strategies to retain computational efficiency. We employ the huggingface implementation[12] of this model to perform the task of Coreference Resolution. The text content of Wikipedia articles and news articles are passed through this module to resolve the entity references.

- **Sentence Simplification:** Sentence Simplification (SS) is the process of reducing the linguistic complexity of

[6]https://dumps.wikimedia.org/enwiki/latest/
[7]https://github.com/attardi/wikiextractor
[8]newsapi.org
[9]https://newsapi.org/docs/endpoints/sources
[10]https://pypi.org/project/newspaper3k/
[11]https://pg/project/jusText/
[12]https://github.com/huggingface/neuralcoref

natural language text by utilizing a more readily accessible vocabulary and sentence structure. SS is separated into two categories: lexical simplification and syntactic simplification. Syntactic simplification is beneficial to break down the grammatical complexities of a sentence by dividing the sentence into shorter segments. The resulting shorter segments are simplified in a standardized format that is easier to understand for downstream applications. As FEVER-based models are trained on short claims, syntactically complex sentences cannot be used for inference as the models will fail to capture the complexity. Therefore, we use DISSIM (Discourse Simplification) [19], a recursive sentence splitting technique that provides a semantic hierarchy of simplified sentences. This hierarchy helps to break the sentences syntactically after incorporating Coreference Resolution.

C. *Continuous Retrieval Engine*

After performing Coreference Resolution and Sentence Simplification on the incoming text, constructing a Continuous Retrieval Engine [30] becomes feasible. Continuous retrieval is the process of building complex representations and retrieving neighbors. This method of retrieval is in contrast to discrete retrieval, where the inverted index maximizes the sparse representations. In principle, continuous retrieval has significant benefits: better recall (unrestricted by particular word-choice), more granular similarity ranking, established interaction between query and candidates, and the likelihood of retrieval across-modalities. The primary components of the Continuous Retrieval Engine are the dual encoder and the Approximate Nearest Neighbor search engine. Dual encoders are a series of models where pairs of objects are encoded into a shared space. The following elements constitute the Continuous Retrieval Engine:

- **Approximate Nearest Neighbor:** A similarity function *sim(E1, E2)* takes two encodings of the same dimension, and outputs a score between [0, 1] with 0 denoting the least similarity and 1 denoting the highest. Similarity functions can be arbitrarily complex, including neural networks that learn interactions between encodings [30]. For the current version of EnFVe, we use cosine similarity to enable the nearest neighbor search. Locality-Sensitive Hashing (LSH) is an algorithmic strategy that provides identical inputs in the same "buckets" with high likelihood. Since a related element is placed in the same bucket, this strategy contributes to data clustering and the nearest neighbor search. The random projection method of LSH called SimHash[13] intends to estimate the cosine distance between vectors. The Annoy[14] library uses SimHash to compute the approximate nearest neighbors.

[13]https://www.cs.princeton.edu/courses/archive/spr04/cos598B/bib/CharikarEstim.pdf
[14]https://github.com/spotify/annoy

- **Encoder:** An Encoder is any learnable function *f(X)* that takes an item *X* as input and returns a *d*-dimensional real-valued encoding vector [30]. We use the multilingual Universal Sentence Encoder (USE) [31] trained on Semantic Textual Similarity (STS) datasets [32]. A dual encoder has the form *g(X1, X2) = sim(f1(X1), f2(X2))* where *f1, f2* are two possibly identical encoders [30]. In our case, both *f1* and *f2* are identical USE modules. During the Ground Truth Creation workflow, the ground truth sentences are encoded by the Encoder module and indexed off-line. At inference time, the input claim is encoded by the same Encoder module and neighbors are retrieved from the Vector Database according to cosine distance.

- **Vector Database:** We use the Annoy library to store the text vectors. Since Annoy uses static files to save and load the indices, it is possible to share index across processes. Due to the decoupled nature of index files, Annoy makes it easy to achieve horizontal scalability. The library focuses on minimizing memory footprint, so the indexes are quite small. The static file indexes and the minimal footprint together makes scaling the Vector Database across multiple machines a seamless procedure.

D. *Ensemble*

1) **Fact Verification Models**
   - **Graph-based Evidence, Aggregating and Reasoning for Fact Verification (GEAR):** GEAR [22] framework focuses on reasoning over multiple evidences using a fully-connected Graph Attention Network. The architecture enables various evidences to be interlinked together in order to perform fact verification. The GEAR framework has three components - Sentence encoder, Evidence Reasoning Network (ERNet), and Evidence aggregator.

     Sentence encoder - For each claim and the corresponding retrieved evidence set, the sentence encoder converts each evidence-claim pair using BERT to get evidence representation, and the claim is encoded alone to obtain the claim representation.

     Evidence Reasoning Network (ERNet) - ERNet aims to build a graph network to facilitate reasoning between the pieces of evidence. Initially, a fully connected evidence graph is constructed with each node representing a piece of evidence. Then the model uses a Multi-Layered Perceptron to compute the attention coefficients between a node i and its neighbor j, followed by normalization using the softmax function. The normalized attention coefficients α_{ij} are used to obtain the features inside hidden state h_i^t for node i at layer t as shown in Eq. 1 [22].

$$\mathbf{h}_i^t = \sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{h}_j^{t-1} \quad (1)$$

where N are the set of neighbors of i.
The final hidden states of evidence nodes $h_1^T, h_2^T, ..., h_N^T$ are obtained by stacking T layers of ERNet. They are capable of grasping information by communicating with neighbouring evidences. These final hidden states obtained are then fed into the evidence aggregator.

Evidence aggregator - The evidence aggregator obtains the final hidden state o by gathering information from different evidence nodes. The attention aggregator employed in the GEAR framework uses the claim representation c obtained from BERT to attend the final hidden states of evidence and get the final aggregated state o.

- **Kernel Graph Attention Network (KGAT):** KGAT [23] performs claim verification by first constructing an evidence graph using retrieved evidence sentences for a claim. Then it conducts fine-grained joint reasoning on the evidence graph by introducing the concept of edge kernels and node kernels. Evidence Graph G is constructed by KGAT considering claim-evidence pair as a node and making it a fully-connected evidence graph with l nodes by connecting all node pairs with an edge. Its predecessor, GEAR mostly inspire the architecture of KGAT model. Each node of the fully-connected evidence graph represents a claim-evidence pair. BERT model is used to formulate the initial node representations. Then, information propagation is performed along with node-level prediction with edge-kernels. The novelty in KGAT is the edge-kernels and node-kernels. Edge-kernels introduced, enhances the flexibility in modelling the interaction between claim-evidence pair, whereas the node-kernels filter out the most significant nodes in the graph using weights.

- **Transformer-XH:** Transformer-XH [33] is a transformer-based model that achieves state-of-the-art results on the Hotpot Question Answering dataset [34]. Transformer-XH follows the guiding principles behind GEAR and KGAT, but it emphasizes more on the interconnectedness of evidences. In order to capture the essence of multi-hop reasoning, the model employs a concept called "extra-hop attention" that lets information to propagate efficiently over the text sequence network. While constructing the evidence graph, each set of evidence can be envisioned as a cluster of nodes having a set of edges that signifies the linkage between the evidences. The fundamental idea behind Transformer-XH is to formalize the information contained in the text sequences and also the global contexts such as the entire set of evidences. This is achieved by two attention mechanisms which consist of "in-sequence attention", that attends over the tokens of an individual evidence set, and the extra-hop attention that attends over the entire evidence set.

2) **Ensembler**
An effective way to deal with the variance of neural network models is to train multiple models instead of a single model and combine predictions using ensemble learning. Ensembling not only reduces the variance of predictions, yet also can bring predictions that are better to any single model. Techniques for ensemble learning are grouped by the varied element, such as training data, the model, and how predictions are combined.

- **LightGBM:** Gradient boosting decision tree (GBDT) [38] is a widely-used machine learning algorithm due to its efficiency, accuracy, interpretability, and implementations such as XGBoost and pGBRT. LightGBM speeds up the training process of conventional GBDT by up to over 20 times [39] while achieving almost the same accuracy by proposing two novel techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). LightGBM is ideal for ensembling individual models, but the problem is choosing the right hyper-parameters. For this, we use a Tree-structured Parzen Estimators (TPE) [40], that is a Sequential Model-Based optimization (SMBO) approach. TPE takes the LightGBM model as the objective function and tries to minimize the objective function loss by finding the optimal set of hyperparameters in the search space. The hyperparameters found by the TPE algorithm are shown in Table II.

IV. Evaluation and Discussion

In this section, we evaluate the Data Acquisition, Preprocessing, Continuous Retrieval Engine, and Ensemble modules of the EnFVe pipeline. First, we start with a comparison of XGBoost, CatBoost, and LightGBM algorithms used for ensembling FEVER-based models. Second, we evaluate the Data Acquisition module by contrasting it with the limitation of existing pipelines. Third, we discuss the utility of the Preprocessing module and how it allows FEVER based models to perform inference on multiple line sentences of fairly high complexity. Finally, we evaluate the performance of the Encoder module used in the Continuous Retrieval Engine.

We benchmarked various ensembling models in combination with GEAR, KGAT, and Transformer-XH. Tree-based Pipeline Optimization Tool (TPOT) is a python library[15] used for automatically optimizing machine learning pipelines using genetic programming. TPOT selected XGBoost classifier as the optimal classifier with an accuracy of 78.8% on FEVER

[15]https://github.com/EpistasisLab/tpot

TABLE I
ENCODER PERFORMANCE ON VARIOUS SIMILARITY TASKS [35]

MODEL	DIM	STS12	STS13	STS14	STS15	STS16	STSB	SICK-R	AVG.
Avg. GloVe embeddings [36]	300	52.3/53.3	50.5/50.7	55.2/55.6	56.7/59.2	54.9/57.7	65.8/62.8	80.0/71.8	59.3/58.7
InferSent-GloVe [37]	4096	59.3/60.3	58.8/58.7	69.6/66.7	71.3/72.2	71.5/72.6	75.7/75.3	88.4/82.5	70.7/69.8
USE [31]	**512**	**61.4/62.0**	**63.5/64.2**	**70.6/67.0**	**74.3/75.9**	**73.9/77.3**	**78.2/77.1**	**85.9/79.8**	**72.5/71.9**
BERT [CLS] [28]	768	27.5/32.5	22.5/24.0	25.6/28.5	32.1/35.5	42.7/51.1	52.1/51.8	70.0/64.8	38.9/41.2
AVG.BERT embeddings [28]	768	46.9/50.1	52.8/52.9	57.2/54.9	63.5/63.4	64.5/64.9	65.2/64.2	80.5/73.5	61.5/60.6

TABLE II
HYPERPARAMETERS IDENTIFIED BY TPE

Parameter name	Value
boosting type	dart
multi logloss	metric
learning rate	0.05
num iterations	550
num leaves	31
tree learner	tree learner
max depth	-1
min data in leaf	20
min sum hessian in leaf	$1e^{-3}$

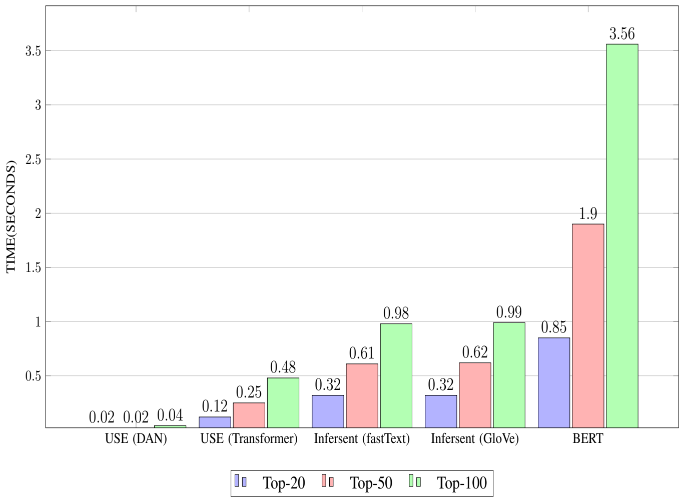

Fig. 2. Top-n evidences retrieval time [43]

dev dataset. CatBoost [41], an implementation of ordered boosting is an alternative to gradient boosting. An advantage of the CatBoost library is the GPU implementation that significantly speeds up the training process. CatBoost model achieved an accuracy of 78.91% on FEVER dev dataset. Arik et al. [42] proposed Tabnet, a high-performance, and interpretable canonical in-depth tabular data learning architecture. Tabnet uses sequential attention to choose the decision step, enabling interpretability, and more efficient learning at each interval. Tabnet achieves better performance than other neural network and decision tree variants on a tabular dataset. However, for the ensembling task, the Tabnet model achieved an accuracy of 79.01% on dev dataset.

EnFVe chooses the LightGBM ensemble of GEAR, KGAT, and Transformer-XH. An accuracy of 79.26% and fast convergence is achieved with the help of LightGBM ensembler. Table III shows the results of the individual models as well as the ensemble models. Among the individual models, KGAT and Transformer-XH show superior results when compared to GEAR and Athene. The ensemble model achieves a marginal increase in accuracy than the best individual model. A feature of the ensemble model is its extensibility. We can provide several FEVER-based models as well as textual entailment models as the input in order to achieve better accuracy. Also, feeding in other text indicators like the measure of toxicity, racism, sexism, and political views may support the ensemble model in producing better results.

The existing pipelines use a combination of search APIs to gather news information. The drawback of using search APIs like Google, Bing, and other search engine APIs is that it restricts scalability. A real-time fact verification pipeline serving fact-checkers will require many claim verification requests to be processed. Currently, the search APIs have very limited query allowance[16] and increasing the query limit would be prohibitively expensive. Also, a clear procedure to routinely scrape and update news articles is missing in the existing work. The essential building blocks of a fact verification pipeline that are missing among existing pipelines are Coreference Resolution and Sentence Simplification. We observe that current pipelines cannot handle multiple line claims or claims containing pronouns referring to an entity. In addition to improved inference, the EvFVe Preprocessing module comprising of the Coreference Resolution and Sentence Simplification helps in building a robust continuous evidence retrieval engine by decomposing the sentence complexity along with performing entity disambiguation.

For quantitative analysis, we use constituency tree depth as a decision criteria for measuring the efficacy of Sentence Simplification module and number of pronouns for Coreference Resolution. Constituency tree is the syntactical structure of a sentence created based on the relationship between the words or sub-phrases. Formally, the complexity of a sentence

[16]https://developers.google.com/custom-search/v1/overview

can be defined by the depth of its constituency tree. We calculated the constituency tree depth and the number of pronouns occurring in the FEVER dataset as well as a sample of news articles obtained from News API. FEVER dataset has a mean tree depth and pronoun count of 9.08 and 1.12 respectively, whereas it was 12.46 and 1.64 respectively for the sample of news articles. We observe that for constituency tree depth > 15 and the number of pronouns > 3, the individual model performance drops by 2.4% on average if Coreference Resolution and Sentence Simplification is not included in the pipeline.

The retrieval engines, primarily Elasticsearch and Apache Lucene, used by the existing pipelines, are discrete (term-based). Discrete retrieval engines excel in search operations involving substrings and single keyword searches [44]. However, in the case of evidence retrieval, the input claim is a formed sentence, and the word order and structure cannot be ignored. Encoders are pretty effective at capturing convoluted synonyms, homonyms, and typos. Gillick et al. [30] have shown that by training simple models with appropriate architecture is sufficient to improve on a discrete baseline by a significant margin for similar-question retrieval tasks. This forms the baseline for building a fast and accurate semantic search engine on models trained on the downstream task of STS. Wang et al. [35] use the SICK-R (Sentences Involving Compositional Knowledge - Relatedness) [45] and the STS benchmark datasets to measure the semantic relatedness of a pair of sentences. Wang and Kuo [35] use the cosine similarity between sentence pairs as the similarity score and report both Pearson and Spearman's rank correlation coefficient.

Table I shows the evaluation result of various encoder models on SICK-R and STS datasets. DIM (Dimensions) is the output dimension of the vector representation of the sentence, and the final column AVG presents the average Pearson and Spearman's rank correlation coefficient across SICK-R and STS datasets. We extend Gillick's model by replacing it with the multilingual USE having a competent STS score that outperforms GloVe, Infersent, and BERT. USE further helps with the scalability of the system by being computationally efficient, see Fig. 2. Exploiting the dual encoder framework of the Continuous Retrieval Engine allows the pipeline to handle multiple data modalities that enable the retrieval engine to incorporate images, speech, and videos.

V. Conclusion

In this paper, we presented EnFVe, an end-to-end pipeline for real-time explainable fact verification. EnFVe uses a Data Acquisition module that continuously scrapes, encodes, and stores the vector format of ground-truth information from Wikipedia and news articles. We also employ two vital preprocessing steps, Coreference Resolution and Sentence Simplification to capture semantic and syntactic contexts of the claim. The existing end-to-end fact verification pipelines make use of search APIs for querying the documents related to the claim and is limited to stance detection or textual entailment for fact-checking the claim. When given a claim, EnFVe retrieves related evidences from the ready-to-infer Vector Database of Wikipedia and news articles. The computational efficiency of the Continuous Retrieval Engine makes the process of real-time evidence extraction possible. EnFVe places a strong emphasis on pure reasoning aspect unlike detecting stance on retrieved evidence sentences to fact check the veracity of the claim. We further employ a decision tree ensemble model (LightGBM) that consists of various FEVER-based models in combination with our evidence retrieval engine that provides near real-time evidence extraction. We are able to achieve an accuracy of (dev/test) 79.26%/73.28%, which is better as compared to the existing fact verification models.

TABLE III
FEVER Accuracy

MODEL	DEV	TEST
Athene [24]	68.49	65.46
GEAR [22]	74.84	71.60
KGAT (BERT Base) [23]	78.02	72.81
Transformer-XH [33]	78.05	72.39
G + K + T + XGBoost(TPOT)	78.80	72.51
G + K + T + Catboost	78.91	72.86
G + K + T + Tabnet	79.01	72.90
G + K + T + LightGBM (EnFVe)	**79.26**	**73.28**

G=GEAR K=KGAT T=Transformer-XH

In our ongoing work, we are aiming to enhance accuracy by including text indicators such as the measure of toxicity, racism, sexism, and political views. Further, the pipeline can be extended with a social media monitoring module that enables EnFVe to capture and fact-check the claims at scale. The EnFVe architecture enables a fully automated real-time fact-checking of news articles and social media by integrating a check-worthiness module that filters out and groups similar claims.

References

[1] David MJ Lazer, Matthew A Baum, Yochai Benkler, Adam J Berinsky, Kelly M Greenhill, Filippo Menczer, Miriam J Metzger, Brendan Nyhan, Gordon Pennycook, and David Rothschild. "The science of fake news". In: *Science* 359.6380 (2018), pp. 1094–1096.

[2] Soroush Vosoughi, Deb Roy, and Sinan Aral. "The spread of true and false news online". In: *Science* 359.6380 (2018), pp. 1146–1151.

[3] I. Dagan, D. Roth, F. Zanzotto, and M. Sammons. *Recognizing Textual Entailment: Models and Applications*. Morgan & Claypool, 2013.

[4] Gabor Angeli and Christopher D. Manning. "NaturalLI: Natural Logic Inference for Common Sense Reasoning". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, pp. 534–545.

[5] Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. “A Stylometric Inquiry into Hyperpartisan and Fake News”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018, pp. 231–240.

[6] Chuan Guo, Juan Cao, Xueyao Zhang, Kai Shu, and Miao Yu. “Exploiting emotions for fake news detection on social media”. In: *arXiv preprint arXiv:1903.01728* (2019).

[7] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. “Defending against neural fake news”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 9051–9062.

[8] Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. “CredEye: A credibility lens for analyzing and explaining misinformation”. In: *Companion Proceedings of the The Web Conference 2018*. 2018, pp. 155–158.

[9] Ramy Baly, Mitra Mohtarami, and James Glass. “Integrating Stance Detection and Fact Checking in a Unified Corpus”. In: *Proceedings of NAACL-HLT*. 2018, pp. 21–27.

[10] Valeriya Slovikovskaya and Giuseppe Attardi. “Transfer Learning from Transformers to Fake News Challenge Stance Detection (FNC-1) Task”. In: *Proceedings of The 12th Language Resources and Evaluation Conference*. European Language Resources Association, 2020, pp. 1211–1218.

[11] Moin Nadeem, Wei Fang, Brian Xu, Mitra Mohtarami, and James Glass. “FAKTA: An Automatic End-to-End Fact Checking System”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Association for Computational Linguistics, 2019, pp. 78–83.

[12] Ali K Chaudhry, Darren Baker, and Philipp Thun-Hohenstein. “Stance detection for the fake news challenge: identifying textual relationships with deep neural nets”. In: *CS224n: Natural Language Processing with Deep Learning* (2017).

[13] William Yang 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)*. Association for Computational Linguistics, 2017, pp. 422–426.

[14] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. “FEVER: a Large-scale Dataset for Fact Extraction and VERification”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2018, pp. 809–819.

[15] Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. “Where the truth lies: Explaining the credibility of emerging claims on the web and social media”. In: *Proceedings of the 26th International Conference on World Wide Web Companion*. 2017, pp. 1003–1012.

[16] Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. “How Noisy Social Media Text, How Diffrnt Social Media Sources?” In: *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, 2013, pp. 356–364.

[17] Sihao Chen, Daniel Khashabi, Chris Callison-Burch, and Dan Roth. “PerspectroScope: A Window to the World of Diverse Perspectives”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, 2019, pp. 129–134.

[18] Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. “End-to-end Neural Coreference Resolution”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017, pp. 188–197.

[19] Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. “Transforming Complex Sentences into a Semantic Hierarchy”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2019, pp. 3415–3427.

[20] J. C. S. Reis, A. Correia, F. Murai, A. Veloso, and F. Benevenuto. “Supervised Learning for Fake News Detection”. In: *IEEE Intelligent Systems* 34.2 (2019), pp. 76–81.

[21] Kashyap Popat, Subhabrata Mukherjee, Andrew Yates, and Gerhard Weikum. “DeClarE: Debunking Fake News and False Claims using Evidence-Aware Deep Learning”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018, pp. 22–32.

[22] Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. “GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2019, pp. 892–901.

[23] Zhenghao Liu, Chenyan Xiong, and Maosong Sun. “Kernel Graph Attention Network for Fact Verification”. In: *arXiv preprint arXiv:1910.09796* (2019).

[24] Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. “UKP-Athene: Multi-Sentence Textual Entailment for Claim Verification”. In: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. Association for Computational Linguistics, 2018, pp. 103–108.

[25] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. "Enhanced LSTM for Natural Language Inference". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017, pp. 1657–1668.

[26] Takuma Yoneda, Jeff Mitchell, Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. "UCL Machine Reading Group: Four Factor Framework For Fact Finding (HexaF)". In: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. Association for Computational Linguistics, 2018, pp. 97–102.

[27] Yixin Nie, Haonan Chen, and Mohit Bansal. "Combining fact extraction and verification with neural semantic matching networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 6859–6866.

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL-HLT (1)*. 2019.

[29] Nayeon Lee, Chien-Sheng Wu, and Pascale Fung. "Improving large-scale fact-checking using decomposable attention models and lexical tagging". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 1133–1138.

[30] Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. "End-to-end retrieval in continuous space". In: *arXiv preprint arXiv:1811.08008* (2018).

[31] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. "Universal Sentence Encoder for English". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2018, pp. 169–174.

[32] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. "SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, 2017, pp. 1–14.

[33] Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. "Transformer-XH: Multi-Evidence Reasoning with eXtra Hop Attention". In: *International Conference on Learning Representations*. 2019.

[34] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018, pp. 2369–2380.

[35] Bin Wang and C-C Jay Kuo. "SBERT-WK: A Sentence Embedding Method by Dissecting BERT-based Word Models". In: *arXiv* (2020), arXiv–2002.

[36] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, pp. 1532–1543.

[37] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017, pp. 670–680.

[38] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *Annals of Statistics* 29 (2000), pp. 1189–1232.

[39] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems*. 2017, pp. 3146–3154.

[40] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems*. 2011, pp. 2546–2554.

[41] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. "CatBoost: unbiased boosting with categorical features". In: *Advances in neural information processing systems*. 2018, pp. 6638–6648.

[42] Sercan O Arik and Tomas Pfister. "Tabnet: Attentive interpretable tabular learning". In: *arXiv preprint arXiv:1908.07442* (2019).

[43] Hebatallah A. Mohamed Hassan, Giuseppe Sansonetti, Fabio Gasparetti, Alessandro Micarelli, and Joeran Beel. "BERT, ELMo, USE and InferSent Sentence Encoders: The Panacea for Research-Paper Recommendation?" In: *RecSys*. 2019.

[44] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2013.

[45] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. "A SICK cure for the evaluation of compositional distributional semantic models". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), 2014, pp. 216–223.