KPatch: Knowledge Patch to Pre-trained Language Model for Zero-Shot Stance Detection on Social Media

Shuohao Lin[1,3], **Wei Chen**[1,3], **Yunpeng Gao**[2], **Zhishu Jiang**[1,3], **Mengqi Liao**[1,3], **Zhiyu Zhang**[1,3], **Shuyuan Zhao**[1,3], **Huaiyu Wan**[1,3,†]

[1] School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
[2] Zhipu AI, Beijing, China
[3] Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing, China
{shhlin, w_chen, jiangzhishu, mqliao, zyuzhang, sy_zhao, hywan}@bjtu.edu.cn
atomgyp@gmail.com

Abstract

Zero-shot stance detection on social media (ZSSD-SM) aims to distinguish the attitude in tweets towards an unseen target. Previous work capture latent variables between source and target domains to perform this task, but the lack of context knowledge hinders the detection performance. Recent studies have been devoted to obtaining the accurate representation of tweets by bringing additional facts from Knowledge Graph (KG), showing promising performance. However, these knowledge injection methods still suffer from two challenges: (i) The pipeline of knowledge injection causes error accumulation and (ii) irrelevant knowledge makes them fail to understand the semantics. In this paper, we propose a novel knowledge injection method for ZSSD-SM, which adopts two training stages, namely knowledge compression and task guidance, to flexibly inject knowledge into the pre-trained language model (PLM) and adaptively expand tweets context. Specifically, in the knowledge compression stage, the latent representation of KG is reconstructed by the triplet denoising task and compressed into external matrices; while in the task guidance stage, the frozen matrices are employed to guide the PLM to adaptively extract its own context-related knowledge, and then complete the fine-tuning of the ZSSD-SM task. Extensive experiments on multiple datasets show the effectiveness of our proposed method. The code is available at: https://github.com/ShuohaoLin/KPatch.

Keywords: Zero-Shot Stance Detection, Knowledge Graph, Knowledge Injection

1. Introduction

Zero-shot stance detection on social media (ZSSD-SM) has become hot research, which assists managers in quickly detecting fake news and identifying communities on social platforms. The aim of ZSSD-SM is to determine the attitude or standpoint (e.g. Favor, Against, or Neutral) expressed in tweets towards an unseen target (Liang et al., 2022a).

Previous work (Allaway et al., 2021; Allaway and McKeown, 2020; Liang et al., 2021, 2022a) focuses on domain transfer, hoping to transfer the latent variables for each stance learned from the source domain to the target domain. However, traditional ZSSD-SM methods fail to understand the target domain's tweet knowledge that does not appear in the source domain, since the short length of tweets contains less context information.

Recently, some studies (Zhang et al., 2020; Liu et al., 2021) attempt to inject external knowledge into the pre-trained language model (PLM) (Devlin et al., 2019) in the form of knowledge graph (KG) (Chen et al., 2023) for ZSSD-SM task, and achieve impressive results. Similar to common knowledge injection methods (Zhang et al., 2019; Peters et al., 2019; Liu et al., 2020; Sun et al., 2020; Wang et al., 2021; Yasunaga et al., 2022), these methods employ graph neural networks or other multi-modal alignment techniques to map external knowledge into the hidden space of the PLM.

Yet, traditional knowledge injection methods involve complex pipelines including (NER) identifying the location of the entity, (EL) linking entity to the knowledge graph entity, (KS) sampling knowledge related to the entity, and (KI) injecting knowledge into the PLM. NER, EL, and KS are collectively known as the preparatory work for knowledge injection. The process of multi-step knowledge injection poses two challenges:

A) **Error accumulation during the preparatory work**: Due to the independent steps in the preparatory work, error information will be transferred and amplified by entity missing, recognition error and confusion. Figure 1 illustrates an example where the "electricity" is not recognized by NER, leading to the disregard of "electricity" entity information, which serves as a critical node for navigating to the topic entity of "climate". Additionally, NER may wrongly take "Be" as an entity, and EL may link it to "beryllium", amplifying the error over time. Therefore, how to effectively mitigate the accumulation of errors is of great significance for the task of stance detection.

† Corresponding author.

LREC-COLING 2024, pages 9961–9973
20-25 May, 2024.

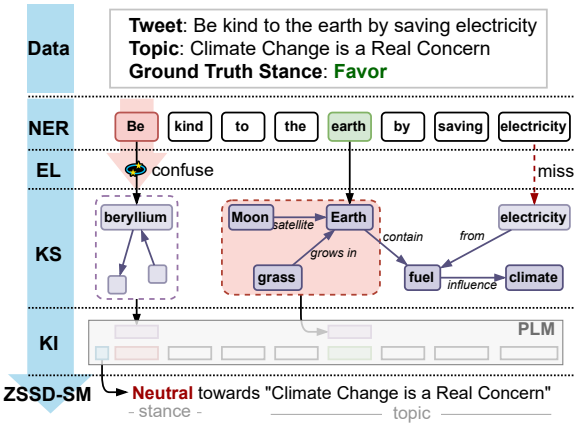


Figure 1: A data flow example of knowledge injection (KI) methods on ZSSD-SM ("Topic" is omitted) and the explanation of two challenges faced by KI. **NER**: A rectangle indicates a token, which colored in green and red represents whether the recognition is correct or not. **EL**: "confuse" means that the entity link is wrong, and the red dotted line indicates that misses some entities because the NER did not recognize them. **KS**: The dotted box indicates the sampling scope. **KI**: The blue square represents the tweet embedding. Due to error accumulation and the influence of irrelevant information, the input tweet is incorrectly classified as "Neutral", whose ground truth is "Favor".

B) **Irrelevant knowledge injection**: Limited by the narrow sampling scope and the invisibility of KG to the backbone PLM, it may not be possible to introduce information conducive to judging the stance of a tweet. Figure 1 demonstrates that when the sampling scope of KS is limited, and the KG is invisible to the PLM, the knowledge that is relevant to the context (Earth, contain, fuel) may not be taken into account. Thus, wider sampling scope and more accurate injection of knowledge related to the tweet is critical for understanding the semantics of the tweet.

In this paper, we propose a novel method, named KPatch, a two-stage knowledge injection method for ZSSD-SM. In the first stage, KPatch compresses the external knowledge graph into a hidden space and stores it into the external matrix. In the second stage, KPatch adopts the knowledge from the latent KG space to guide the PLM in modeling ZSSD-SM. KPatch is capable of capturing latent variables between the source and target domains and injecting knowledge into the PLM without relying on the preparatory work.

The main contributions of our work are summarized as follows:

- We propose a knowledge injection method that skips the preparatory work of traditional knowledge injection pipeline, which guides the PLM to adaptively select the most suitable knowledge from external matrix through its latent modeling ability.
- We design the knowledge compression and the task guidance stage to flexibly introduce knowledge to the PLM. The former injects knowledge by mapping KG into a hidden space through triplet denoising task, while the latter guides the PLM to complete ZSSD-SM task with external parameter matrices learned in the former stage.
- The extensive experiments on two public datasets demonstrate that our KPatch achieves better performance against the baselines on the ZSSD-SM task.

2. Related Work

2.1. Zero-shot Stance Detection on Social Media

The application of social media has made zero-shot stance detection on social media (ZSSD-SM) crucial in various social media management scenarios, including argument mining, fake news detection, and fact checking, etc.

Previous studies aim to capture both the differences and similarities among topics. Allaway et al. (2021) uses a domain adaptive method to capture differences between topics. Allaway and McKeown (2020) constructs a cluster space and introduces an attention mechanism to explore similarities between topics. Liang et al. (2022b) learns both domain similarity and difference using a contrastive learning strategy based on prototype graphs.

However, previous methods only consider latent variables between domains, while ignoring the differences in knowledge involved across different domains. As a result, Zhang et al. (2020) and Liu et al. (2021) utilize the knowledge graph to inject relevant knowledge into the language model, which effectively overcome the performance bottleneck caused by the lack of knowledge.

2.2. Knowledge Injection to PLM

PLM uses knowledge acquired during the pretraining phase to model high-quality representations of text, which greatly facilitates the development of natural language tasks. However, retraining a PLM to update its knowledge can be costly.

To address this problem, some researchers have proposed injecting a knowledge graph into the PLM to update its knowledge. Zhang et al. (2019) and Peters et al. (2019) align pre-embedded vectors of entities in the KG with the PLM in hidden spaces.

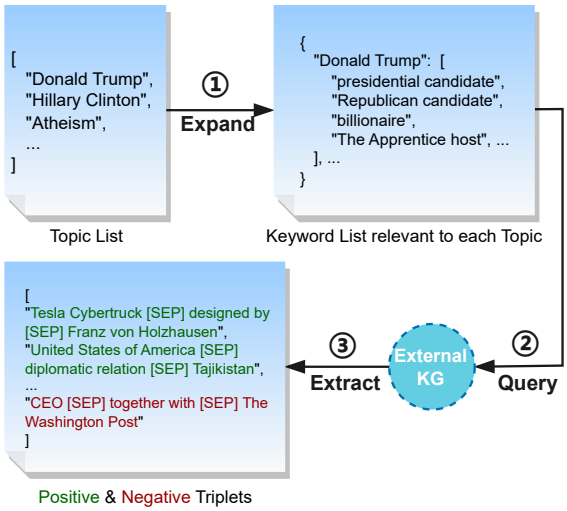


Figure 2: Knowledge Searching Stage: (1) **Expand** each topic to obtain a list of secondary keywords. (2) **Query** subgraphs related to all keywords from the external KG. (3) **Extract** triplets from subgraph as positive triplets, and randomly replace the head or tail entity to form negative triplets.

Liu et al. (2020) and Sun et al. (2020), on the other hand, extract relevant subgraphs from the KG and convert them into the input format of the PLM, which allows for injecting knowledge in the form of text.

However, as shown in Figure 1, these methods often require the preparatory work before knowledge injection, which accumulate errors through the pipeline. Additionally, due to the limited sampling scope and the invisibility of knowledge graph to the PLM, these methods may introduce irrelevant knowledge during the KS phase.

3. Methodology

3.1. Task Description

Let $D_s = \{(r_s^i, t_s^i, y_s^i)\}_{i=1}^{N_s}$ be an annotated training set, where r, t and y are the tweet, target topic and the stance label respectively. Then let $D_d = \{(r_d^i, t_d^i)\}_{i=1}^{N_d}$ be the testing set for the topics which are known but not present in D_s. N_s and N_d are the size of D_s and D_d respectively. The task of ZSSD-SM is to predict a stance label $\hat{y} \in \{Favor, Against, Neutral\}$ given the tweet r and the topic t from D_d.

3.2. Method Overview

Our method consists of two parts: Knowledge Searching (as shown in Figure 2) and KPatch, where KPatch is a two-stage framework that includes knowledge compression and task guidance stages (as shown in Figure 3):

1) Knowledge Searching: We enrich the semantics of each topic on topic list by large language model (LLM), then query the corresponding triplets from the external KG and construct negative examples by randomly replacing the head or tail entities.

2) Knowledge Compression (KPatch Stage 1): To map external knowledge triplets into the same high-dimensional space as the backbone PLM and avoid knowledge forgetting while fine-tuning towards ZSSD-SM, we freeze the backbone PLM and compress the knowledge into the external matrix M_K under the triplet denoising task (TDT);

3) Task Guidance (KPatch Stage 2): To bypass the lengthy preparatory work and complete the ZSSD-SM task by allowing the backbone PLM to learn to extract the most suitable knowledge for the input tweet and topic from the external matrix M_K, we fine-tune the trainable backbone PLM towards ZSSD-SM under the guidance of external knowledge that stores in the frozen M_K.

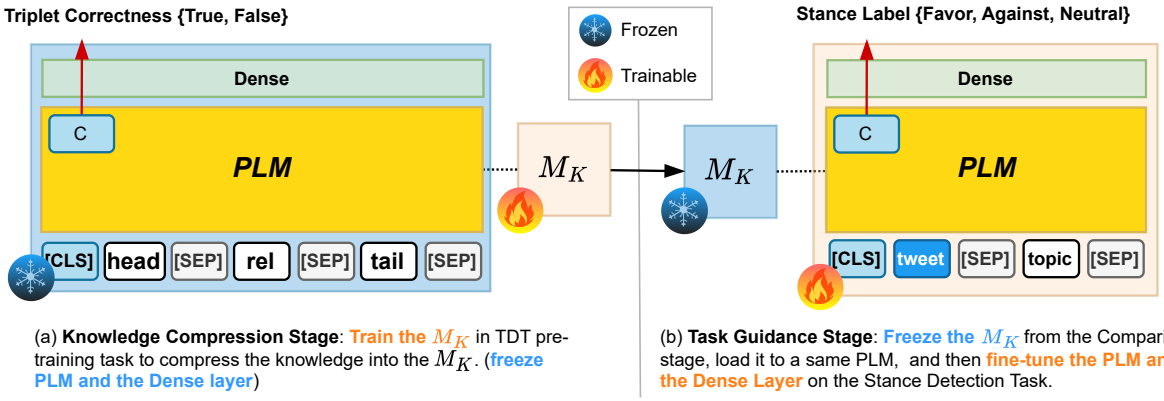
3.3. Knowledge Searching

The conversation of a topic discussed on social media tends to focus on the particular content related to the topic in a certain time period. Therefore, the ZSSD-SM only needs to consider knowledge that is relevant to the topic. Based on this observation, as shown in Figure 2, we first extend all known topics (from training and testing sets) to obtain some secondary keywords explaining each topic. Then we query external knowledge graphs related to these keywords, thereby constructing a topic-related subgraph. Finally, we extract the data that is needed for KPatch from the topic-related subgraph.

Expand Phase: In social media, a topic of discussion is often highly condensed (such as "Hillary Clinton", "Feminist Movement", etc.), and its condensed semantics helps less in selecting knowledge relevant to the topic. Therefore, expanding the known topic list L into the expanded keyword list L_e is necessary. Given a large amount of knowledge that is stored in the parameters of LLM (Petroni et al., 2019), we propose using LLM to expand each topic in L by constructing a prompt.

Query Phase: Since the knowledge graph is a structured semantic graph extracted from a large amount of text, each entity or triplet has an abstract or source text. So we perform full-text searching through these introductory texts to extract the subgraph KG_e related to each phrase in L_e.

Extract Phase: To map the queried KG subgraph into the hidden space, we follow KG-BERT (Yao et al., 2019) to extract self-supervised data from KG_e via triplet denoising task (TDT). Specifically,



(a) **Knowledge Compression Stage**: Train the M_K in TDT pre-training task to compress the knowledge into the M_K. (freeze PLM and the Dense layer)

(b) **Task Guidance Stage**: Freeze the M_K from the Comparison stage, load it to a same PLM, and then fine-tune the PLM and the Dense Layer on the Stance Detection Task.

Figure 3: The architecture of KPatch. The two sub-figures (a) and (b) represent the two stages of KPatch respectively.

all triplets in KG are regarded as positive examples $\mathbb{D}^+$, and the head entity h or tail entity o of triplet $(h,l,o) \in \mathbb{D}^+$ is randomly replaced with an entity h' or o', i.e.,

$$\mathbb{D}^- = \{(h',l,o) \mid h' \in \mathbb{E} \wedge h' \neq h \wedge (h',l,o) \notin \mathbb{D}^+\} \cup \{(h,l,o') \mid o' \in \mathbb{E} \wedge o' \neq o \wedge (h,l,o') \notin \mathbb{D}^+\}, \quad (1)$$

where $\mathbb{E}$ is the set of entities from KG_e. We denote $\mathbb{D} = \mathbb{D}^+ \cup \mathbb{D}^-$ as the training set of TDT, N^+ and N^- represent the size of $\mathbb{D}^+$ and $\mathbb{D}^-$ respectively. In particular, when $N^- > N^+$, we randomly select N^+ triplets from $\mathbb{D}^-$ as a set of negative triplets to ensure the balance of TDT training.

3.4. KPatch

3.4.1. Knowledge Compression Stage

The goal of knowledge compression is to compress knowledge into the matrix M_K independent of the backbone PLM. To this purpose, we regard Parameter-Efficient Fine-Tuning (PEFT) (Mangrulkar et al., 2022), which can flexibly transform explicit knowledge into implicit parameter, as the implementation of M_K. And then we employ the TDT to complete knowledge compression.

Triplet Denoising Task (TDT): Given a triplet $\tau = (h,l,o) \in \mathbb{D}$, the object of TDT is to determine whether the triplet τ is a positive or negative case.

As shown in Figure 3(a), we freeze the PLM and the classification layer, and only update M_K. We input triplet $\tau = (h,l,o) \in \mathbb{D}$ into the PLM with M_K:

$$\vec{h}_\tau = \underline{\text{PLM}}_{M_K}(\text{Format}_T(\tau))_{\text{[CLS]}}, \quad (2)$$

where $\text{Format}_T(\cdot)$ is a function to format τ in the form of "[CLS]h[SEP]l[SEP]o[SEP]", and underline denotes to freeze the parameters of the module while training.

We judge whether the triplet τ is positive or negative and employ the cross-entropy loss as an objective function:

$$\vec{y}_\tau = \text{sigmoid}(W\vec{h}_\tau + b), \quad (3)$$

$$\mathcal{L}_\tau = -\sum_{\tau \in \mathbb{D}} y_\tau log(\vec{y}_\tau^0) + (1-y_\tau)\, log(\vec{y}_\tau^1), \quad (4)$$

where $y_\tau \in \{0,1\}$ is the label (negative or positive) of the triplet.

Since no parameter is updated in the backbone PLM and classification layer, we believe that the knowledge graph is stored in M_K.

Through the above two strategies of PEFT and TDT, we can effectively compress the structured knowledge into the PLM and retain the semantic information of knowledge. It is worth noting that PEFT is a general implicit knowledge compression strategy and is not limited to a single type of manner. We conduct experiments in Section 4.2 to verify the generality of our proposed knowledge compression.

3.4.2. Task Guidance Stage

Traditional knowledge injection methods require the preparatory work to determine what knowledge should be introduced into the sentence. However, these methods do not accurately learn the knowledge semantics associated with tweets. van Aken et al. (2019) has demonstrated that PLMs have the capability to perform unsupervised operations such as NER required by KI. Inspired by it, we present the task guidance stage, where the PLM acquires the ability to automatically select the relevant knowledge from M_K according to the context, instead of relying on the preparatory work.

The workflow of the task guidance stage is shown in Figure 3(b). We first initialize the backbone PLM and classification layer identical to that of the knowledge compression stage. Then we load

and freeze M_K onto the backbone PLM. For each $(r, t, y) \in D_s$, we get the representation of $v = (r, t)$ by:

$$\vec{h}_v = \text{PLM}_{\underline{M_K}}(\text{Format}_V(v))_{\text{[CLS]}}. \quad (5)$$

Different from the Eq. 2, we input tweet and target topic in the form of "[CLS]r[SEP]t[SEP]" (Liang et al., 2022b) by function $\text{Format}_V(\cdot)$. And fine-tune the backbone PLM with calculating the cross-entropy loss:

$$\vec{y}_v = \text{softmax}(W\vec{h}_v + b), \quad (6)$$

$$\mathcal{L}_v = -\sum_{i=1}^{N_s}\sum_{j=1}^{d_y} y_i^j \log((\vec{y}_v)_i^j), \quad (7)$$

where $\mathcal{L}_v$ is the final object function for ZSSD-SM task.

4. Experiments

4.1. Datasets and Evaluation Metrics

Datasets: We perform the ZSSD-SM task on two public datasets: Sem16 dataset(Mohammad et al., 2016) and MiDe22-EN dataset(Toraman et al., 2022).

Sem16 dataset comprises tweets related to 6 predefined topics, collected up until January 2016, including Atheism (A), Donald Trump (DT), Hillary Clinton (HC), Climate Change is a Real Concern (CC), Feminist Movement (FM) and Legalization of Abortion (LA). Each tweet is categorized as either "Favor", "Against", or "Neutral" towards a specific topic.

MiDe22-EN dataset collected tweets from 2020 to 2022 on 4 predefined topics for fake news detection, including Russia-Ukraine war (RU), Refugees (R), COVID-19 (C) and Miscellaneous (M). Formally, each piece of data in the MiDe22-EN dataset can be expressed as (r, t, e, y), where $e \in E_{MiDe}$ is the event to be detected, and the size of E_{MiDe} is 40. The purpose of MiDe22-EN dataset is to evaluate whether the content expressed in r provides evidence to support e. We feed the model as "[CLS]r[SEP]e[SEP]" and aggregate all results according to t.

Following Allaway et al. (2021), for each topic $t \in T$ of Sem16 or MiDe22-EN dataset (T is the target topic set of the dataset), we utilize all examples from topics in $\{T - t\}$ and split them 85/15 for the training and validation. Finally, we test the model on all examples of t. The statistics of datasets are shown in Table 1. More details about dataset can be found in Appendix A.

Evaluation Metrics: We follow Mohammad et al. (2016) and report the average F1 score on "Favor" and "Against" labels in the Sem16 dataset. Additionally, we report the F_{macro} score in the MiDe22-EN dataset.

Dataset	Target	Train	Valid	Test
Sem16	A	3519	618	733
	CC	3663	643	564
	FM	3336	585	949
	HC	3306	580	984
	LA	3349	588	933
	DT	3542	621	707
MiDe22-EN	C	2712	479	1048
	M	2632	465	1142
	R	2732	482	1025
	RU	2733	482	1024

Table 1: Statistics of Sem16 and MiDe22-EN datasets.

Dataset	$\lvert L\rvert$	$\lvert L_e\rvert$	N_L^+	$N_{L_e}^+$
Sem16	6	971	6k	390k
MiDe22-EN	100	680	131k	468k

Table 2: The data statistics of Sem16 and MiDe22-EN dataset during the knowledge searching and knowledge compression stage. $|L|$, $|L_e|$, N_L^+ and $N_{L_e}^+$ respectively represents the size of the topic list L, the expanded keyword list L_e, positive example set $\mathbb{D}^+$ constructed based on L and L_e.

4.2. Baselines and Modules

Baselines: We compare with ZSSD-SM methods: **BiCond**(Augenstein et al., 2016)–conditional encoder method based on bidirectional LSTM (Jia et al., 2023), **TOAD**(Allaway et al., 2021)–domain adaptation method based on adversarial learning, **BERT**(Allaway et al., 2021)–Bidirectional Encoder Representation from Transformers (Devlin et al., 2019) with learning rate warm-up and decay after 10% training steps, **TGA Net**(Allaway and McKeown, 2020)–topic clustering method based on BERT, **JointCL**(Liang et al., 2022b)–target-aware prototypical graph contrastive learning method and **WS-BERT**(dual)(He et al., 2022)–text splicing knowledge injection method on ZSSD-SM.

We also compare with traditional knowledge injection model: **K-BERT**(Liu et al., 2020)–constructs a sentence tree with limited-scale subgraph of entities, and converts the sentence tree as sequence via visible matrix to input into the PLM.

Modules: We use pre-trained uncased BERT-base with an embedding dimension of 768 as the backbone PLM. During the "Expand Phase", we use artificially constructed prompts (Appendix B) to guide ChatGPT (OpenAI, 2023) to expand the topic list. We use WikiData[1] as the external knowledge graph. To verify the generality of KPatch, we adopt two widely used PEFT methods: LoRA and P-Tuning V2, which serves as the implementation of external matrix M_K to conduct experiments:

[1] https://wikidata.org/

Model	Sem16 (F_{avg}%)					
	A	LA	HC	CC	FM	DT
BiCond	31.98(±4.35)	32.10(±5.12)	35.88(±4.06)	11.54(±1.59)	38.55(±2.21)	28.14(±0.69)
TOAD	31.07(±7.46)	28.65(±5.60)	32.45(±7.06)	8.31(±3.04)	38.44(±9.78)	36.16(±7.14)
BERT	30.74(±3.31)	37.43(±5.24)	42.56(±3.78)	14.44(±7.71)	43.05(±4.37)	20.30(±1.30)
TGA Net	34.71(±0.43)	39.97(±2.41)	40.15(±5.72)	11.27(±1.85)	**45.18(±1.49)**	25.15(±1.25)
JointCL	33.43(±5.31)	34.51(±1.18)	39.15(±3.55)	4.76(±0.34)	38.58(±4.25)	31.61(±10.37)
WS-BERT	30.13(±4.00)	41.29(±3.13)	34.30(±15.58)	8.87(±4.16)	35.16(±2.95)	21.78(±7.97)
K-BERT	39.39(±1.09)	40.52(±3.33)	39.75(±1.36)	12.34(±9.89)	42.76(±6.75)	32.43(±2.49)
KP(LoRA)	**39.86(±2.84)**	43.80(±5.01)	**49.75(±4.97)**	**31.92(±9.54)**	43.92(±3.46)	41.14(±3.74)
KP(PTV2)	38.61(±2.13)	**44.94(±0.64)**	45.85(±7.60)	27.99(±7.28)	44.16(±2.49)	**41.81(±2.73)**

Model	MiDe22-EN (F_{macro}%)			
	C	M	R	RU
BiCond	34.30(±0.21)	42.73(±0.18)	43.25(±4.07)	33.65(±1.69)
TOAD	32.92(±1.02)	38.80(±2.42)	38.43(±0.99)	32.27(±1.26)
BERT	57.41(±1.21)	52.43(±5.04)	61.28(±8.72)	47.22(±1.49)
TGA Net	48.90(±1.71)	42.26(±0.97)	55.56(±1.32)	42.37(±5.53)
JointCL	54.89(±3.28)	57.42(±3.15)	58.50(±11.38)	43.41(±8.32)
WS-BERT	55.54(±1.60)	57.23(±2.18)	59.09(±5.65)	40.33(±3.45)
K-BERT	40.63(±1.13)	36.97(±3.61)	42.65(±3.44)	37.50(±4.13)
KP(LoRA)	55.06(±2.90)	60.55(±5.25)	**69.50(±1.67)**	**54.92(±1.67)**
KP(PTV2)	**58.14(±4.05)**	**62.84(±1.40)**	69.02(±2.31)	52.24(±3.05)

Table 3: Performance comparison of KPatch and ZSSD-SM models on two public ZSSD-SM datasets. The best scores are in bold and the suboptimal scores are in underline (KPatch and P-Tuning V2 is represented by KP and PTV2 respectively and the same below).

1) **LoRA** (Hu et al., 2021): Based on the assumption of lower intrinsic dimensions of downstream tasks, two bypass matrices are added to the attention module, whose dimension r (rank) between them is the intrinsic dimensions of downstream tasks.

2) **P-Tuning V2** (Liu et al., 2022): Stores the information in the parameter matrix of the prefix encoder, and inserts virtual tokens (as a soft prompt) before the input sequence of each layer to adapt to the downstream task. The number of virtual tokens is recorded as l_{vt}.

We design two variants for ablation study:

1) "**w/o Expand**": Cancels the "Expand Phase" during "Knowledge Searching", and directly uses the topic list to query the external knowledge graph.

2) "**w/o Compress**": Cancels the "Knowledge Compression Stage", trains the backbone PLM (freeze the last classification layer) on $\mathbb{D}$ without M_K, and then fine-tunes it on the ZSSD-SM task.

4.3. Implementation Settings

All models are implemented with PyTorch (Paszke et al., 2019) framework and Huggingface transformers (Wolf et al., 2020) on one NVIDIA GeForce RTX 2080 (8GB) card.

In our KPatch, we use AdamW (Loshchilov and Hutter, 2019) as the optimizer without warm-up, and determine the timing of early stopping based on the evaluation metric of different datasets in the fine-tuning of ZSSD-SM.

Knowledge Searching Stage: We query 4 candidate entries for each keyword by WikiPedia[2] full-text search engine (Koren, 2012), and then query their one-hop neighbor subgraphs from WikiData. For triplets in subgraphs, we only keep those relations considered in WikiData5M (Wang et al., 2021). During the extract phase, we consider the WikiData aliases of each entity as its label as well, and replace them into the original triplets to augment the set of positive triplets.

Knowledge Compression Stage: The training epoch is set to 2 to save resources and reduce the risk of overfitting, since the value of loss will decrease slowly after 2 epochs according to our previous experiments. The learning rate of LoRA and P-Tuning V2 are set to 2e-5 and 1e-3 respectively. The batch size is set to 64. (It will be 80% of the value and rounded down when the external matrix M_K is implemented by P-Tuning V2. The batch size parameter mentioned below also follows this setting.) r and l_{vt} both take 64, except l_{vt} on MiDe22-EN takes 16.

Task Guidance Stage: The learning rate is set to 2e-5 on both datasets. The batch size is 64 and 10 on Sem16 and MiDe22-EN respectively. We train

[2] https://www.wikipedia.org/

Model	Sem16 (ΔF_{avg}%)						MiDe22-EN (ΔF_{macro}%)			
	A	LA	HC	CC	FM	DT	C	M	R	RU
K-BERT	0.22	-2.83	-1.32	-8.23	**1.56**	-1.38	**5.76**	1.09	0.12	3.35
KP(LoRA)	**9.12**	6.36	**7.19**	**17.48**	0.87	20.84	-2.35	8.12	**8.22**	**7.70**
KP(PTV2)	7.87	**7.50**	3.29	13.55	1.11	**21.51**	0.73	**10.41**	7.74	5.02

Table 4: Performance comparison between KPatch and K-BERT fine-tuned for ZSSD-SM on two public ZSSD-SM datasets. Scores represent the difference in performance between the model and its backbone PLM.

Model	Sem16 (F_{avg}%)						MiDe22-EN (F_{macro}%)			
	A	LA	HC	CC	FM	DT	C	M	R	RU
KP(LoRA)	**39.86**	43.80	**49.75**	**31.92**	43.92	41.14	55.06	60.55	**69.50**	**54.92**
w/o Expand	34.44	**46.26**	48.44	22.18	42.52	35.68	54.08	57.31	67.85	52.72
KP(PTV2)	38.61	44.94	45.85	27.99	**44.16**	**41.81**	**58.14**	**62.84**	69.02	52.24
w/o Expand	36.32	39.00	43.98	18.03	37.19	33.66	55.48	55.60	66.61	48.55
w/o Compress	39.83	45.36	48.69	29.84	39.32	36.04	48.74	53.70	59.60	48.42

Table 5: Experimental results of ablation study. For KPatch variants that implement M_K by different PEFT methods, we complete w/o Expand experiments using the same hyper-parameters.

KPatch for 20 epochs under the early stopping setting with the patience value of 3.

We repeat the experiment for ZSSD-SM on 3 different random seeds. In addition, to restore the real performances of baselines, all results are the average performance obtained after re-experimenting on three random seeds under the same dataset, rather than citing from other papers. For most baselines, we use settings from the original paper. On the Sem16 dataset, TOAD uses about two thousand unlabeled target domain tweets crawled at the same time as training data. However, on the MiDe22-EN dataset, there are not unlabeled target domain data for adversarial learning. Therefore, we cancel the adversarial learning setting of TOAD on MiDe22-EN.

4.4. Overall Performance

The performance comparison on two datasets is shown in Table 3. We can observe that the performance of KPatch is significantly better than traditional ZSSD-SM models on two datasets. This verifies the effectiveness of KPatch on the ZSSD-SM task.

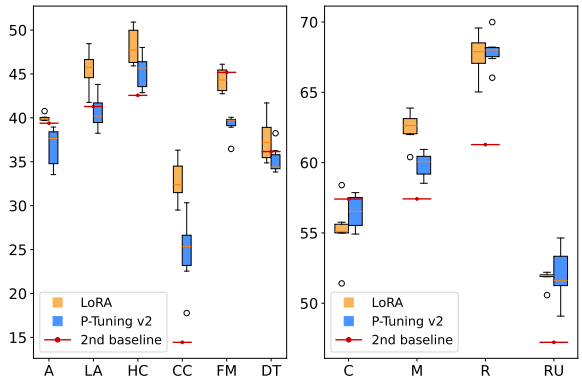
Another observation is that the vanilla BERT without introducing knowledge achieves similar performance to the suboptimal traditional ZSSD-SM model. This indicates that the vanilla BERT can also learn the capability of domain transfer. In addition, the performance of vanilla BERT is worse than the knowledge-introduced KPatch, which proves that the introduction of external knowledge is beneficial for the model to achieve better performance on the ZSSD-SM task.

Considering the different backbone PLMs used by each KI approach, to ensure a fair comparison, we conduct a comparative analysis of the model's performance fluctuations before and after the injection of knowledge into the PLM for the ZSSD-SM task. As shown in Table 4, the fluctuations of KPatch are generally positive and significant. On the contrary, K-BERT is limited by the preparatory work, and the knowledge injected leads to poorer performance on Sem16. Due to KPatch canceling the preparatory work of KI, there is no challenge of error accumulation, limited sampling scope and invisible KG to the PLM faced by traditional KI models.

4.5. Ablation Study

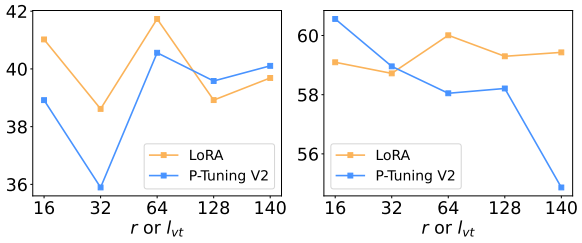
w/o Expand: We can intuitively see that a significant degradation in the performance of "w/o Expand" is compared to KPatch on two datasets in Table 5. Specifically, on the Sem16 dataset, the average F_{avg} of "w/o Expand" lagged behind KPatch by 3.48% and 5.86% (LoRA and P-Tuning V2 implementation respectively and the same below), while on the MiDe22-EN dataset, the gap narrowed to 2.02% and 4.0%.

The reason for this phenomenon is that there are different sizes of topic list lengths on different datasets. As shown in Table 2, on the Sem16 dataset for example, the size of external knowledge triplet N_L^+ (6k) is much smaller than $N_{L_e}^+$ (390k). It indicates that without sufficient knowledge, KPatch cannot effectively complete "knowledge compression" and construct effective M_K, and therefore cannot effectively "guide" the backbone PLM to complete the ZSSD-SM task. On the MiDe22-EN dataset, the number of knowledge used for "knowledge compression" by KPatch and "w/o Expand" is on the same order of magnitude, which allows the



(a) Sem16 (b) MiDe22-EN

Figure 4: Box plot of the impact of TDT data generated under the different random seed s_{ns} on KPatch (LoRA or P-Tuning V2 implementation) performance. ○ represents outliers, and "2nd baseline" is the suboptimal baseline performance reported in Table 3 except KPatch. (suboptimal baseline does not refer to a specific model)



(a) Sem16 (b) MiDe22-EN

Figure 5: On Sem16 and MiDe22-EN, the relationship between performance and the size of M_K (i.e., the rank r of LoRA or the number of virtual tokens l_{vt} of P-Tuning V2).

Dataset	The value of l_{vt}					
	0	16	32	64	128	140
Sem16	30	46	62	**94**	158	170
MiDe22-EN	65	**81**	97	129	193	205

Table 6: The average length of input tokens in Sem16 and MiDe22-EN (the first column) and the actual input length to the backbone PLM under different l_{vt} values (remaining columns on the right).

"w/o Expand" variant to close the gap with KPatch.

w/o Compress: Essentially, "w/o Compress" is equivalent to KPatch degenerating into KG-BERT, which learns external knowledge into the parameters of the backbone PLM through TDT and then fine-tunes the PLM on the downstream task.

However, due to the lack of the "knowledge compression" stage, the PLM may forget previously injected knowledge while fine-tuning downstream tasks. Therefore, although the average performance of "w/o Compress" is only reduced by 1.89% and 0.71% compared to KPatch on Sem16, it decrease sharply increases to 7.39% and 7.95% on the MiDe22-EN with larger injection knowledge. This further demonstrates the role of "knowledge compression" in isolating knowledge from interference from downstream tasks.

4.6. The Effect of Random Negative Triplets Generation in Extract Phase

When conducting "Knowledge Searching Stage", the sampled negative triplets will changed due to differences in random seeds. In order to study the impact of TDT data generated by different random seeds on KPatch, we set up six different random seeds to conduct repeated experiments. Specifically, during the negative triplets generation, we set a random seed s_{ns} to control the negative sampled triplets. After obtaining TDT data, we train KPatch on three different random seeds s_{train} mentioned in Section 4.3. Finally, we average the results corresponding to all s_{train} under the same s_{ns} and draw a box plot as shown in Figure 4.

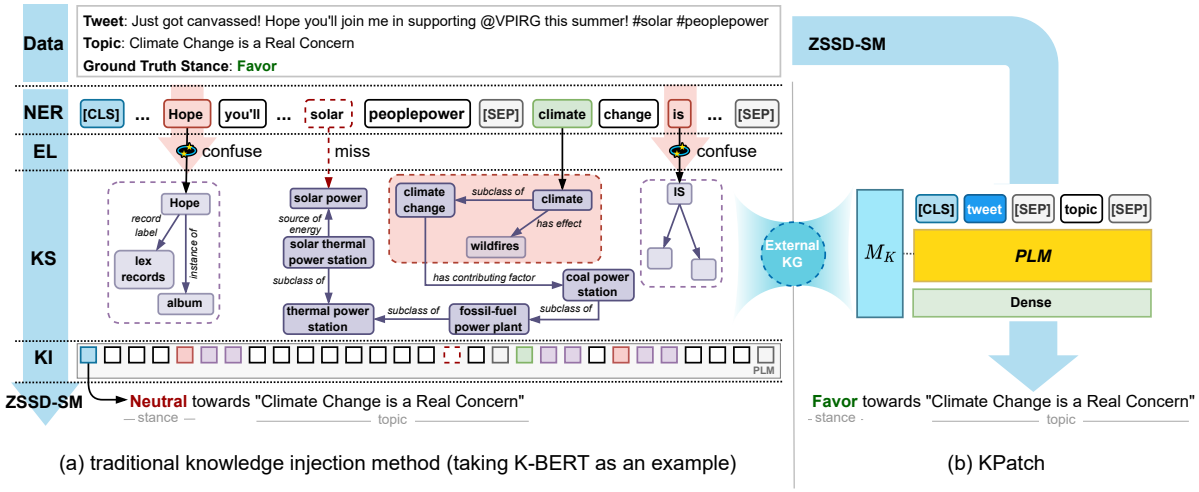
We observed that different negative sampling data will cause some fluctuations in model performance. However, KPatch performs better than the suboptimal baseline models on most subsets. In addition, due to the difference in assumptions between LoRA and P-Tuning V2 (see Section 4.2 for details), KPatch(LoRA) can usually achieve better and more stable results than KPatch(P-Tuning V2).

4.7. The Effect of Different Size of M_K

Referring to the practical conclusions of the PEFT method (Hu et al., 2021; Liu et al., 2022), different external matrix M_K sizes should be selected to adapt to different datasets and tasks.

As shown in Figure 5 (a), we observe that the performance of the two types of PEFT methods on the Sem16 dataset is relatively similar, and both achieve the best results at $r = 64$ and $l_{vt} = 64$.

On the MiDe22-EN dataset, as shown in Figure 5 (b), the LoRA implementation still shows stability, while the P-Tuning V2 implementation shows a negative correlation between performance and the value of l_{vt}. Empirically, the reason for this phenomenon is that the longer input is not conducive to the backbone PLM obtaining effective external knowledge from the virtual tokens introduced by P-Tuning V2: As shown in Table 6, on Sem16 and MiDe22-EN, KPatch(P-Tuning V2) achieves the best results with similar average input lengths, 94 and 81 respectively. In addition, longer input length may also cause the denoising ability of KPatch(P-Tuning V2) to decrease as l_{vt} increases.



(a) traditional knowledge injection method (taking K-BERT as an example)

(b) KPatch

Figure 6: Comparison of the prediction process and differences between traditional knowledge injection method and KPatch.

4.8. Case Study

We provide a case study of traditional knowledge injection method (taking K-BERT as an example) and KPatch on the Sem16 dataset. Figure 6 shows the details and comparison of the prediction process of two methods.

It can be observed that K-BERT requires the lengthy preparatory work before performing the knowledge injection (KI) stage, which may lead to serious error accumulation. Specifically, K-BERT (Figure 6 (a)) has errors in the NER, EL and KS stages:

1) **NER**: K-BERT identifies "Hope" and "is" as entities respectively, and misses the key entity "solar".

2) **EL**: On top of the errors caused by the NER stage, K-BERT links "Hope" and "is" to "Hope (Non-Prophets album)" and "Islamic State (IS)" respectively. In addition, if "solar" can be identified in the NER stage, it is difficult for traditional models to correctly understand the connection between "solar" and "solar power".

3) **KS**: Due to the errors in the first two stages, K-BERT introduces a lot of context-independent information through "Hope" and "is". In addition, if K-BERT can correctly recognize "solar" as an entity, the model will also be unable to sample the effective inference path between "solar power" and "climate change" entities due to sampling scope limitations.

In contrast, KPatch (Figure6 (b)) hands over these manually designed steps to the PLM for processing and implements knowledge injection through the hidden space. In other words, KPatch can directly perform stance detection without the need for the lengthy preparatory work.

5. Conclusion

We propose a simple but effective knowledge injection method (KPatch) for zero-shot stance detection on social media (ZSSD-SM), which skips the preparatory work required in the traditional knowledge injection (KI) method and avoids the problems of error accumulation and irrelevant knowledge injection faced by traditional KI. Based on the two stages of knowledge compression and task guidance, KPatch guides the backbone pre-trained language model (PLM) to adaptively introduce the most suitable knowledge by its latent modeling ability. During this process, we leverage the triplet denoising task to compress the external knowledge into external parameter matrices, acting as a bridge to communicate two stages. Tests on two public ZSSD-SM datasets show that our KPatch outperforms traditional ZSSD-SM and KI methods. In future work, we plan to investigate how to explain the knowledge considered during the inference process.

6. Acknowledgements

This work is supported by the National Key R&D Program of China (No. 2021QY1502).

7. Limitations

We have proven the effectiveness of KPatch on two public datasets, however there are still limitations:

1) In order to skip the traditional KI model's dependence on the preparatory work, we map KG to the hidden space. However, due to the limitations of deep learning, this makes KG invisible and unreadable to humans, exacerbating the inexplicability of the model's inference process.

2) Compared to full tuning, the efficient parameter fine-tuning method represented by LoRA and P-Tuning V2 has advantages in parameter quantity and can also bring the characteristic of "knowledge compression" to KPatch. But it leads the model performs a slower speed during training.

8. Ethics Statement

1) Our method may be used to identify the overall stance of the public towards a particular event. If the model provides incorrect predictions, it may mislead managers into making incorrect decisions. Therefore, there are still certain risks in using our method for stance detection before performance improvement.

2) The external knowledge graph we use mainly comes from WikiData, and the training data we use comes from public datasets that contain controversial topics such as politics, war, and refugees. As a result, there may be toxic, biased, or offensive information in the data, which is used for model training. Therefore, during testing, the model may have biases against some special data, which requires further security review.

9. Bibliographical References

Abeer ALDayel and Walid Magdy. 2021. Stance detection on social media: State of the art and trends. *Information Processing & Management*, 58(4):102597.

Emily Allaway and Kathleen McKeown. 2020. Zero-Shot Stance Detection: A Dataset and Model using Generalized Topic Representations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8913–8931, Online. Association for Computational Linguistics.

Emily Allaway, Malavika Srikanth, and Kathleen McKeown. 2021. Adversarial learning for zero-shot stance detection on social media. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4756–4767, Online. Association for Computational Linguistics.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Wei Chen, Huaiyu Wan, Yuting Wu, Shuyuan Zhao, Jiayaqi Cheng, Yuxin Li, and Youfang Lin. 2023. Local-global history-aware contrastive learning for temporal knowledge graph reasoning. *arXiv preprint arXiv:2312.01601*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*.

Zihao He, Negar Mokhberian, and Kristina Lerman. 2022. Infusing knowledge from Wikipedia to enhance stance detection. In *Proceedings of the 12th Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis*, pages 71–77, Dublin, Ireland. Association for Computational Linguistics.

Edward Hu, Yelong Shen, Phil Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Yuxin Jia, Youfang Lin, Xinyan Hao, Yan Lin, Shengnan Guo, and Huaiyu Wan. 2023. Witran: Water-wave information transmission and recurrent acceleration network for long-range time series forecasting. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yaron Koren. 2012. *Working with MediaWiki*. WikiWorks Press San Bernardino, CA, USA.

Kalev Leetaru and Philip A Schrodt. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, volume 2, pages 1–49. Citeseer.

Bin Liang, Zixiao Chen, Lin Gui, Yulan He, Min Yang, and Ruifeng Xu. 2022a. Zero-shot stance detection via contrastive learning. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 2738–2747, New York, NY, USA. Association for Computing Machinery.

Bin Liang, Yonghao Fu, Lin Gui, Min Yang, Jiachen Du, Yulan He, and Ruifeng Xu. 2021. Target-adaptive graph for cross-target stance detection. In *Proceedings of the Web Conference 2021*, WWW '21, page 3453–3464, New York, NY, USA. Association for Computing Machinery.

Bin Liang, Qinglin Zhu, Xiang Li, Min Yang, Lin Gui, Yulan He, and Ruifeng Xu. 2022b. JointCL: A joint contrastive learning framework for zero-shot stance detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 81–91, Dublin, Ireland. Association for Computational Linguistics.

Rui Liu, Zheng Lin, Yutong Tan, and Weiping Wang. 2021. Enhancing zero-shot and few-shot stance detection with commonsense knowledge graph. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3152–3157, Online. Association for Computational Linguistics.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: Enabling language representation with knowledge graph. In *Proceedings of AAAI 2020*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Laura Eline Ruis, Akbir Khan, Stella Biderman, Sara Hooker, Tim Rocktäschel, and Edward Grefenstette. 2023. Large language models are not zero-shot communicators.

Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. 2020. CoLAKE: Contextualized language and knowledge embedding. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3660–3670, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Cagri Toraman, Oguzhan Ozcelik, Furkan Şahinuç, and Fazli Can. 2022. Not good times for lies: Misinformation detection on the russia-ukraine war, covid-19, and refugees.

Betty van Aken, Benjamin Winter, Alexander Löser, and Felix A. Gers. 2019. How does bert answer questions? a layer-wise analysis of transformer representations. In *Proceedings of the 28th ACM International Conference on Information*

and Knowledge Management, CIKM '19, page 1823–1832, New York, NY, USA. Association for Computing Machinery.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for knowledge graph completion. *CoRR*, abs/1909.03193.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D. Manning, Percy Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. In *Neural Information Processing Systems (NeurIPS)*.

Donghan Yu, Chenguang Zhu, Yiming Yang, and Michael Zeng. 2022. Jaket: Joint pre-training of knowledge graph and language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11630–11638.

Bowen Zhang, Xianghua Fu, Daijun Ding, Hu Huang, Yangyang Li, and Liwen Jing. 2023. Investigating chain-of-thought with chatgpt for stance detection on social media.

Bowen Zhang, Min Yang, Xutao Li, Yunming Ye, Xiaofei Xu, and Kuai Dai. 2020. Enhancing cross-target stance detection with transferable semantic-emotion knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3188–3197, Online. Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of ACL 2019*.

A. Dataset Details

Following TOAD (Allaway et al., 2021), our preprocessing methods to datasets include: (1) Remove URLs, non ASCII characters, and words with special meanings for tweet. (2) Split the hashtag. If the processed text is an empty string, the data will be discarded.

For MiDe22-EN, as it only provides a tweet ID, it is unable to download some expired or deleted data. Therefore, the dataset used in this paper is slightly smaller than the dataset proposed in the MiDe22 paper (Toraman et al., 2022), but it does not affect the overall distribution of the data.

B. The Design Details of Prompt for Topic List Expansion

We restricts the following conditions in the prompt:

- Using Chain of Thought: By adding "think step by step" (Ruis et al., 2023) in the prompt.
- The output content should be named entities as much as possible.
- Specify a time **time**, so that the output content is as active as possible within that time range.
- Each entity in the output list should be related to the target topic **T** or event keyword **kw**.
- The output list should not be longer than **amount**.
- Output as the format of json list.

The prompts we designed for Sem16 and MiDe22-EN datasets are as follows:

Sem16: *Imagine you are a person living in **time**! What can you think of about '**T**' in **time**? Please think step by step! Please generate **amount** named entities related to '**T**', which can be relevant to event, and output as json format. The example is as follows: ["phrase A", "phrase B", ...]. Consider some events and hotspots at that time. The json output is:*

MiDe22-EN: *What can you think of about '**kw**' and '**T**'? Please think step by step! Please generate **amount** named entities related to '**kw**' and*

*'**T**', which can be relevant to event, and output as json format. The example is as follows: ["phrase A", "phrase B", ...]. Considering some events and hotspots at that time. The json output is:*

We used ChatGPT based on gpt-3.5-turbo to expand the topic list on the Sem16 and MiDe22-EN datasets.