Getting the Most out of Simile Recognition

Xiaoyue Wang[1, 2*] **Linfeng Song**[3*] **Xin Liu**[1] **Chulun Zhou**[1] **Jinsong Su**[1, 2†]
[1]School of Informatics, Xiamen University, China
[2]Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan (Xiamen University), Ministry of Culture and Tourism, China
[3]Tencent AI Lab, Bellevue, WA, USA
xiaoyuewang@stu.xmu.edu.cn lfsong@tencent.com
{liuxin, clzhou}@stu.xmu.edu.cn jssu@xmu.edu.cn

Abstract

Simile recognition involves two subtasks: *simile sentence classification* that discriminates whether a sentence contains simile, and *simile component extraction* that locates the corresponding objects (i.e., tenors and vehicles). Recent work ignores features other than surface strings. In this paper, we explore expressive features for this task to achieve more effective data utilization. Particularly, we study two types of features: 1) input-side features that include POS tags, dependency trees and word definitions, and 2) decoding features that capture the interdependence among various decoding decisions. We further construct a model named *HGSR*, which merges the input-side features as a heterogeneous graph and leverages decoding features via distillation. Experiments show that *HGSR* significantly outperforms the current state-of-the-art systems and carefully designed baselines, verifying the effectiveness of introduced features. Our code is available at https://github.com/DeepLearnXMU/HGSR.

1 Introduction

Simile is a type of figurative that compares two objects (named *tenor* and *vehicle*) of different categories using comparator words such as "*like*", "*as*" or "*than*". Table 1 shows a simile sentence, where the tenor "*sheep*" and the vehicle "*clouds*" are compared using comparator "*like*". Generally, simile recognition involves two subtasks (Liu et al., 2018): *simile sentence classification* that discriminates whether a sentence contains simile expressions, and *simile component extraction*, which aims to find simile components (i.e., tenor and vehicle). Because simile usually involves implicit sentiment, it can provide essential information for sentiment analysis (Li et al., 2012; Qadir et al., 2015) (e.g. hate speech detection) and dialogue understanding (Vanzo et al., 2019). Besides, simile recognition can help language learners to better understand the implicit meanings expressed by the simile expressions in novels and fairy tale stories. Therefore, simile recognition has become an important task in natural language processing.

羊群看上去像白云。 (*The sheep look like white clouds.*)
她看上去像我姐姐。 (*She looks like my sister.*)

Table 1: Two examples: the first is a simile sentence that uses "*like*" to compare tenor "*sheep*" and vehicle "*clouds*"; the second is a literal sentence.

Previous studies on simile recognition (Niculae, 2013; Niculae and Yaneva, 2013; Niculae and Danescu-Niculescu-Mizil, 2014) demonstrate that exploiting syntactic features is beneficial to simile recognition. However, they resort to handcrafting feature templates, which usually requires extensive efforts from linguistic experts and is hard to be adapted to new domains and languages. With the recent release of annotated data in a descent scale and the success of deep learning on a wide range of NLP tasks, Liu et al. (2018) first propose a neural model for simile recognition. Specifically, they adopt multi-task learning and let the two subtasks share an LSTM (Hochreiter and Schmidhuber, 1997) encoder that only consumes input sentences. Along this line, Zeng et al. (2020) propose a cyclic multitask learning model with a pretrained BERT (Devlin et al., 2019) encoder, where both subtasks and an extra language modeling subtask are stacked into a loop. This cyclic model yields the current state-of-the-art (SOTA) performance. In spite of these successful attempts, they suffer from the data hunger issue, and ignore other features except surface strings.

In this paper, we explore more expressive fea-

*Equal contribution.
†Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 3243–3252
December 7-11, 2022

tures to achieve more effective data utilization for neural simile recognition. The studied features can be categorized into two major types: one type (**input-side features**) covers the task input, and the other type (**decoding features**) captures the interdependence among various decoding decisions. Particularly, our input-side features include POS tags, dependency trees and word definitions, and we propose a novel heterogeneous graph that used to effectively merge the input-side features. In the heterogeneous graph, some nodes represent input words, and we use POS tags to distinguish *noun* nodes (in blue) from *non-noun* nodes (in green) as shown in 1(a). The noun words are highlighted because simile components are usually nouns (Hanks, 2012), and their dictionary definitions (if any) are added to help learn their representations. We also introduce two *subsentence* nodes divided by the given comparator (e.g., "*like*") to help contrast the both sides of the comparator. Meanwhile, each edge may correspond to a dependency arc (e.g., "nsubj") or point from a noun node to a subsentence node. We use multiple GAT (Veličković et al., 2017) layers to represent each graph.

We introduce the decoding features for simile component extraction. As the tenor and the vehicle are different entities with the same properties (Niculae, 2013), intuitively, the tenor information can help to locate vehicle, and vice versa. To model such intuition, we sequentially extract tenor and vehicle, where the encoder states of the first extracted component (e.g., the tenor) are consumed as extra decoding features for recognizing the second component (e.g., the vehicle). To leverage all possible decoding features, we take the ensemble of the models for all three decoding orders (tenor $\rightarrow$ vehicle; vehicle $\rightarrow$ tenor; in parallel) as the teacher model. The teacher model then simultaneously guide each individual model via distillation during training. During inference time, we only use one model to avoid the computational consumption caused by their ensemble.

Extensive experiments on a simile recognition benchmark (Liu et al., 2018) show that our proposed model largely outperforms previous SOTA system and several competitive baselines by 1.7 and 9.3 points for simile sentence classification and simile component extraction, respectively. Besides, our model trained with 40% data reaches comparable performances than the baseline using full training data, indicating that our model is less data hungry.

2 Problem Formulation

Formally, given an input sentence $S = w_1, \ldots, w_i, \ldots, w_N$ containing a comparator w_c, the goal is to detect whether the comparator w_c is a simile and what spans in S correspond to the simile components (tensors and vehicles). Note that a comparator may correspond to a literal comparison (rather than a simile), such as "*the sheep looks like an Australian sheep breed*." Following previous work (Niculae and Danescu-Niculescu-Mizil, 2014; Liu et al., 2018; Zhang et al., 2019b; Zeng et al., 2020), we formulate the two subtasks as binary classification and sequence labeling, respectively.

3 Baseline: BERT for Simile Recognition

In this section, we introduce a baseline for simile recognition based on BERT (Devlin et al., 2019), temred as **BSR**.

3.1 Encoding

Given an input sentence $S = w_1, \ldots, w_i, \ldots, w_N$, we first place special tokens `[CLS]` and `[SEP]` at its beginning and the ending, before feeding the sequence into a BERT encoder with extra self-attention (Vaswani et al., 2017) layers. Let $\mathbf{H} = \{h_i\}_{i=0}^{N+1}$ be the hidden states of tokens at the top layer, the hidden state (h_0) of `[CLS]` is used as the sentence representation.

3.2 Simile Sentence Classification

For simile classification, we feed the sentence representation h_0 into a linear layer:

$$p(c|S) = \text{softmax}(W_c h_0), \tag{1}$$

where $c \in \{\texttt{true}, \texttt{false}\}$ is the label indicating whether the input sentence S contains simile and W_c is a model parameter. The corresponding loss is defined as:

$$\mathcal{J}_{sc}(c|S;\theta) = -\log p(c|S) \tag{2}$$

3.3 Simile Component Extraction

Afterwards, we feed the final node states $\mathbf{H}$ into a fully-connected layer with softmax to conduct component extraction:

$$\begin{aligned} p(T|S) &= \prod_{i=1}^{N} p(t_i|S) \\ &= \prod_{i=1}^{N} \text{softmax}(W_e \cdot h_i + b_e), \end{aligned} \tag{3}$$

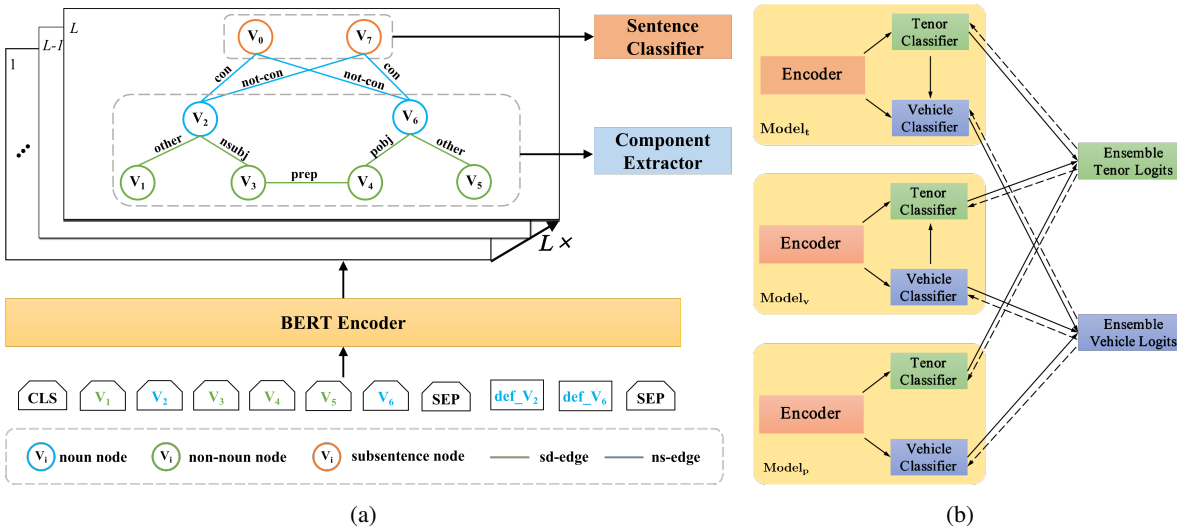

Figure 1: (a) The architecture of our model taking a heterogeneous graph constructed from input-side features, and (b) The process of distillation using decoding features. The decoding orders of the three models are "tenor → vehicle", "vehicle → tenor" and "in parallel", respectively.

where $T = t_1, \ldots, t_N$ is the gold label sequence of simile component extraction. The possible values for each t_i are $\{\mathtt{T}, \mathtt{V}, \mathtt{O}\}$, indicating w_i being part of a tenor, part of a vehicle and none, respectively. The loss is defined as:

$$\mathcal{J}_{ce}(T|S;\theta) = -\sum_{i=1}^{N} \log p(t_i|S), \qquad (4)$$

3.4 Training

Given training data $\mathcal{D}$, we train the model by a linear interpolation between the two subtasks:

$$\mathcal{J}(\mathcal{D};\theta) = \sum_{(S,c,T)\in\mathcal{D}} \Big(\alpha \cdot \mathcal{J}_{sc}(c|S;\theta) + (1-\alpha) \cdot \mathcal{J}_{ce}(T|S;\theta)\Big), \qquad (5)$$

where θ denotes all model parameters, and α is the interpolation coefficient between the two subtasks.

4 Model

In this section, we give more details of our model that takes input-side features (§4.1) and decoding feature (§4.2) for simile recognition. For fair comparison, our model is mostly consistent with the baseline (§3) with slight changes (shown below) for incorporating our introduced features.

4.1 Including Input-side Features

We explore the following three types of input-side features to enhance each input sentence:

- **POS Tags**: We mainly use the part-of-speech (POS) information to distinguish nouns and other words in each given sentence. This is because simile components are usually nouns.
- **Dependency Tree**: Dependency trees have been shown to capture long-range word-to-word dependencies and some shallow semantic information. We adopt such knowledge to help our model learn better sentence representations.
- **Word Definitions**: We adopt a word sense analyzer (Yao et al., 2021) to find definitions for the nouns. The definitions are then appended as input features to our model. Intuitively, using definitions can improve model robustness and achieve more effective data utilization. It also shares a similar spirit with recent prompt-based research (Schick and Schütze, 2021).

Combination with Heterogeneous Graph We merge the input-side features for each instance by constructing a heterogeneous graph $G = (V, E)$, which includes the set of nodes V and edges E:

Node. In the node set V, each node may represent a noun, a non-noun word or a subsentence. This is based on the POS tagging results by a pre-trained LTP parser*. As simile usually involves the comparison between a tenor and a vehicle located on both sides of a comparator, we introduce two subsentence nodes that correspond to the left

*https://github.com/HIT-SCIR/ltp

and right parts split by the comparator, respectively. Taking Figure 1(a) as the example, the graph contains two noun nodes (v_2 and v_6), four non-noun nodes (v_1, v_3 v_4, and v_5) and two subsentence nodes (v_0 and v_7).

The reason for the above design is to better determine whether the subsentences divided by the comparator contain different types of objects with similar attributes, which is crucial for simile sentence classification. We expect that the subsentence nodes can help highlight the difference between the two sides of a comparator during information aggregation within the graph neural network. Moreover, since the considered objects are usually nouns, we believe that nouns are more important than other words in this task. Thus, we deliberately differentiate noun and non-noun nodes to emphasize the positive impact of nouns.

Edges. We consider two main types of edges in the edge set E. The edges of the first type (named *sd-edge*) are essentially dependency arcs. To avoid excessive trainable parameters, we restrict the edges to only cover the top 8 most frequent dependency relations in the training data. Meanwhile, we convert all rest dependency relations as "*other*". By doing so, we expect that the *sd-edges* are able to capture the long-distance dependency between each tenor and its vehicle. An edge of the second type (named *ns-edge* that is short for *noun-subsentence edge*) connects a noun node with a subsentence node. In this way, the impacts of nouns are highlighted when aggregating information for subsentence nodes. In order to distinguish the two subsentence nodes, we assign each *ns-edge* with a label that can either be "*con*" or "*not-con*", indicating whether the subsentence contains the noun. Using Figure 1(a) as the example, the labels of many *sd-edges* are set to *"other"* except for "*nsubj*", "*prep*" and "*pobj*". For the *ns-edges*, as "v_2" belongs to the left subsentence, the edge labels for "$v_2 \rightarrow v_0$" and "$v_2 \rightarrow v_7$" are *"con"* and *"not-con"*, respectively.

Heterogeneous Graph Encoding As shown in Figure 1(a), we modify the baseline encoding phase (§3.1) by replacing the extra self-attention layers with GAT (Veličković et al., 2017) layers in order to consume the proposed heterogeneous graphs. The initial state (e.g., $g_i^{(0)}$) for a word node (in blue and green) is initialized from the corresponding BERT output (h_i). For a subsentence node (in orange), we initial its state using average pooling over the hidden states of the words within the subsentence. The embeddings (e.g, e_{ij}) for the edge labels are randomly initialized.

At each GAT layer, we sequentially conduct graph attention and gating mechanisms to update all node states. Taking the l-th layer for example, we first update each $g_i^{(l)}$ with the hidden states (e.g., $g_j^{(l)}$) of its directly connected neighbors as follows:

$$
\begin{aligned}
z_{ij} &= \text{LeakyReLU}(W_a[W_q g_i^{(l)}; W_k g_j^{(l)}; e_{ij}]), \\
\alpha_{ij} &= \text{softmax}(z_{ij}) = \frac{\exp(z_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(z_{ik})}, \\
g_i^{(l+1)} &= \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W_v g_j^{(l)}\right),
\end{aligned} \tag{6}
$$

where α_{ij} is the attention score indicating the importance of node j to node i, $\mathcal{N}_i$ is the set of neighborhood nodes to the node i in the graph, W_* are model parameters[†], and $\sigma(*)$ is a sigmoid function.

Note that different from the BERT-based simile recognition model, which has only one sentence representation (§3.2), there are two subsentence representations ($g_0^{(L)}$ and $g_{N+1}^{(L)}$). Hence, we adjust Eq. 1 for simile sentence classification as follows:

$$
\begin{aligned}
p(c|S) = \text{softmax}(W_c[&g_0^{(L)}; g_{N+1}^{(L)}; \\
&|g_0^{(L)} - g_{N+1}^{(L)}|]E_c^\top),
\end{aligned} \tag{7}
$$

4.2 Leveraging Decoding Features

As shown in Figure 1(b), we adopt two extra models to extract simile components (tenor and vehicle) sequentially: one extracts the *tenor* before the *vehicle* (model$_t$) while the other functions in the opposite direction (model$_v$). Different from the baseline, which extracts simile components in parallel (model$_p$), both models use the encoder state of the first component as extra features to the second component extractor. In particular, the extractor for the second component is defined as

$$
\begin{aligned}
p(T_{c_2}|S) &= \prod_{i=1}^{N} p(t_{c_2,i}|S) \\
&= \prod_{i=1}^{N} \text{softmax}(W_e^{c_2} \cdot [g_i^{(L)}; g_{c_1}^{(L)}] + b_e^{c_2}),
\end{aligned} \tag{8}
$$

[†]For the remaining of this paper, we use W_* and b_* to denote model parameters.

where T_{c_2} is the gold label sequence for extracting the second simile component, and $g_{c_1}^{(L)}$ denotes the hidden state for the first simile component. Both $W_e^{c_2}$ and $b_e^{c_2}$ represent the parameters for the extra simile component extractor.

To leverage all possible features, inspired by (Zhang et al., 2019a; Wu et al., 2022), we apply distillation to encourage each model to mimic the behaviors of their ensemble:

$$\mathcal{J}_{kl}^{*}(\mathcal{D};\theta^{*}) = \mathrm{KL}(p_{*}(T|S)||p_{ensemble}(T|S)) \tag{9}$$

where $p_{*}(T|S)$ and θ^{*} are the output probabilities and parameters for one individual model, $\mathrm{KL}(*)$ denotes the Kullback-Leibler divergence, and $\mathcal{J}_{kl}^{*}(\mathcal{D};\theta^{*})$ is the distillation objective. We sum up the logits of all models as their ensemble, and $p_{ensemble}(T|S)$ denotes the probability for the ensemble. The final objective of each model can be denoted as:

$$\begin{aligned}\mathcal{J}_{final}^{*}(\mathcal{D};\theta^{*}) &= \lambda\mathcal{J}^{*}(\mathcal{D};\theta^{*}) + (1-\lambda)\mathcal{J}_{kl}^{*}(\mathcal{D};\theta^{*}) \\ &= \lambda \sum_{(S,c,T)\in\mathcal{D}} \Big(\alpha\cdot\mathcal{J}_{sc}(c|S;\theta) + \\ &\quad (1-\alpha)\cdot\mathcal{J}_{ce}(T|S;\theta)\Big) + (1-\lambda) \\ &\quad \mathrm{KL}(p_{*}(T|S)||p_{ensemble}(T|S))\end{aligned} \tag{10}$$

where $\mathcal{J}^{*}(\mathcal{D};\theta^{*})$ is the training objective defined in Eq.5, and λ is the hyper-parameter to control the impacts of two objectives. Here we linearly increase λ from 0 to 1 throughout training. It is worth noting that for less consumption, only one model is used during inference time.

5 Experiments

We conduct detailed experiments and analysis to investigate the effectiveness of our model.

5.1 Setup

Datasets. We choose the Chinese Simile Recognition benchmark (Liu et al., 2018), which consists of 11,377 sentences (roughly half of them contain simile). Since no data split on training, developing and testing is provided, we follow previous work to conduct 5-fold cross validation[‡]. We also follow previous work to evaluate our models using the official scorer that measures Precision, Recall and F1 score.

[‡]In our experiments, the standard deviations of the 5-fold cross validation for simile sentence classification and simile component extraction are 0.29 and 0.32, respectively.

Comparisons. To comprehensively evaluate the **BSR** baseline and our **HGSR** model, we compare them with the following systems:

- **MTL** (Liu et al., 2018). A multi-task learning model, where simile sentence classification, simile component extraction and sentence reconstruction are jointly modeled.
- **Self_Attn+POS** (Zhang et al., 2019b). It extends (Liu et al., 2018) with POS information and uses several self-attention layers to enhance the original LSTM encoder.
- **Cyc-MTL** (Zeng et al., 2020). It extends (Liu et al., 2018) by stacking the three subtasks into a cycle to let them better benefit from each other.

To verify that our heterogeneous graph can effectively incorporate POS features, we also build a variant of our model (HGSR-ConcatPOS), which removes noun nodes from the graph and concatenates each word embedding with its POS embedding.

Implementation Details. We determine hyperparameters α as 0.1 according to the model performance on the validation set. Following Zeng et al. (2020), we employ a pre-trained Chinese BERT[§] model to learn contextual word embeddings and then finetune this model using our training data. Besides, we randomly initialize the representation vectors for edges and label embeddings with 50 and 100 dimension vectors, respectively. The max length and word number of sentence are set to be 128 and 100 by padding shorter sentences and cutting longer ones. The hidden state size of the GAT layer is set to 300. Parameter optimization is performed using Adam with learning rate 2e-5 and batch size 8. And we stack two layers of GAT to gather global information after conducting simile sentence classification. For fair comparison, we also stack two self-attention layers on the *BSR* baseline. After training, we evaluate model$_t$, model$_v$ and model$_p$ on the development set and pick up the best one for inference.

5.2 Effect of GAT Layer Number L

We first investigate the effect of the GAT layer number L on the development set. When L varies from 1 to 3, the F1 scores of our model on simile classification are 91.32, 92.44, and 92.32, those on component extraction are 86.74, 87.53, and 87.49,

[§]https://github.com/ymcui/Chinese-BERT-wwm

Model	Sentence Classification			Component Extraction		
	Precision	Recall	F1	Precision	Recall	F1
MTL (Liu et al., 2018)	80.84	92.20	86.15	61.60	73.61	67.07
Self_Attn+POS (Zhang et al., 2019b)	80.44	91.69	85.70	58.91	74.65	65.85
Cyc-MTL (Zeng et al., 2020)	85.81	**94.43**	89.92	73.97	77.61	75.74
HGSR-ConcatPOS	88.73	94.30	91.43	81.23	87.76	84.37
HGSR	**89.04**	94.39	**91.64**	**81.86**	**88.37**	**84.99**

Table 2: Main test results. Please note that we outperform all baselines, including the SOTA Cyc-MTL.

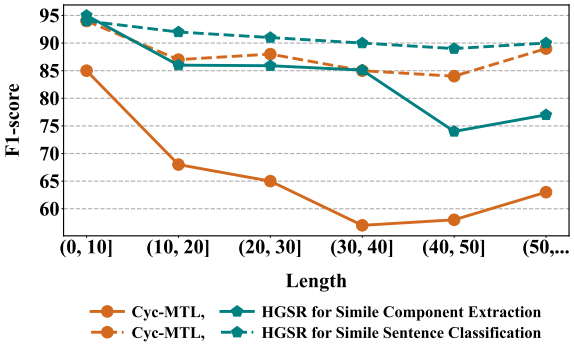

Figure 2: F1 scores on different groups of test instances according to sentence lengths. Dashed lines and solid lines represent simile sentence classification and simile component extraction, respectively.

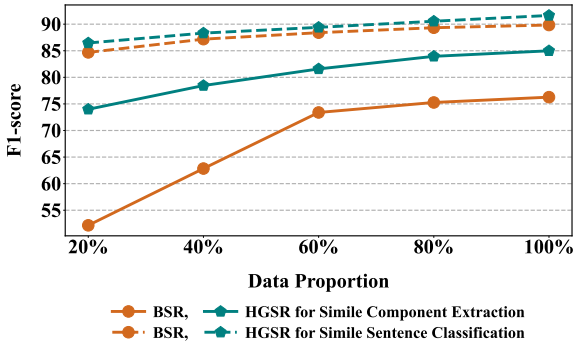

Figure 3: F1 scores on low-resource settings where only a certain percent of data is available for training. Different line colors and styles represent different tasks and systems.

respectively. Therefore, we set L as 2 in subsequent experiments.

5.3 Main Results

The main test results are shown in Table 2. We can observe *HGSR* outperforms all the baselines and achieves SOTA F_1 scores on both subtasks, which demonstrates the effectiveness of our methods. In order to further understand advantages of *HGSR*, we follow Zeng et al. (2020) to conduct more evaluations.

F1 Score against Sentence Length. As shown in Figure 2, we compare our model with *Cyc-MTL* (Zeng et al., 2020) regarding different ranges of sentence length, where we use their provided results that correspond to the reported performance. Results show that our model is consistency better than *Cyc-MTL* in all groups, and our model always performs better with the increase of sentence length. This verifies the effectiveness of our dependency-based features for helping handle the long-range dependency problem.

F1 Score against the Distance between a Tenor and a Vehicle. As shown in Figure 4, we also analyze the results regarding different groups

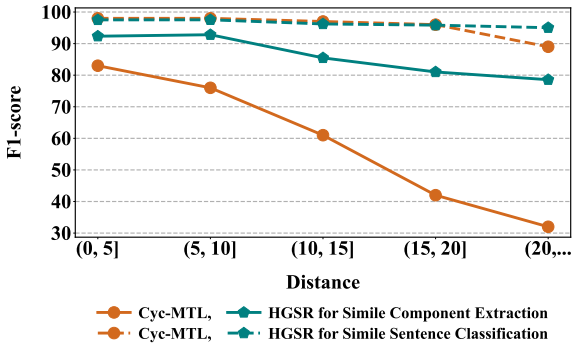

Figure 4: F1 scores on different groups of test instances according to the distance between a tenor and a vehicle.

of distances between a tensor and a vehicle. We can observe that both models yield descent performances on simile sentence classification, while the challenge is still large for simile component extraction. For *Cyc-MTL*, the performance in terms of F1 score drops to around 40% when there are more than 15 words in between, and the number further drops to 30% if there are more than 20 words. On the other hand, our model can yield F1 scores of more than 80% and more than 60% for these situations, respectively. This indicates the robustness of our model on the most challenging cases.

Besides, we analysis models **F1 Score on Low**

Resource Settings to measure their capabilities when data is insufficient. As shown in Figure 3, we train several *HGSR* and *BSR* models under the supervision of 20%, 40%, 60%, 80% and 100% training data and then evaluate them on the test set. We can observe that the *HGSR* consistently surpass *BSR* in all data settings. Besides, it is encouraging to see that *HGSR* only drops 12.9% on simile component extraction when data is insufficient (20%), compared with 31.6% of *BSR*. And only 40% training data is enough to train a satisfactory *HGSR*, which even surpasses the *BSR* trained by 100% training data. This indicates that the proposed *HGSR* is less data hungry with the help of input-side and decoding features.

Finally, we replace the pretrained model from Chinese BERT with a Chinese RoBERTa-wwm (Cui et al., 2020b), so as to further investigate the generality of our model. We also apply the Chinese RoBERTa-wwm to *Cyc-MTL*, which is our most competitive baseline. Results are shown in Table 3. We can observe that *HGSR* still surpasses both *Cyc-MTL* and *BSR* on for both sentence classification and component extraction tasks. This verifies that our model is effective with various pretrained models.

5.4 Ablation study

To investigate the influence of different features on the model effects, we conducted an ablation study regarding the two major features.

Input-side features. We explore the following variants to investigate the impacts of input-side features: 1) ***w/o dependency***. In this variant, we let each word node to connect all other word nodes rather than following the dependency arcs. This is for pinpointing the effect of using dependency information. 2) ***w/o POS***. This baseline does not utilize the POS information, where each subsentence node connects all word nodes rather than only noun nodes. 3) ***w/o definitions***. In this variant, only the input sentence will be fed into the model. 4) ***w/o subsentence nodes***. In this variant, we merge the two subsentence nodes into one global node and the initial state of the global node is the representation of "[CLS]".

As shown in group ① of Table 4, consistent performance decrease on both subtasks is witnessed after removing each input-side features. These results verify the effectiveness of using dependency tree, POS information, word definitions as well as the subsentence nodes. Among these factors, we observe that the subsentence nodes cause the greatest impact on sentence classification, which indicates the effectiveness of highlighting the difference between the two sides of the comparator. Besides, the word definitions give the greatest impact on component extraction. This quite fits our expectation and is consistent with previous observations (Niculae and Danescu-Niculescu-Mizil, 2014).

Decoding features. We further study the effectiveness of the decoding features by removing model_t/model_v/model_p, respectively.

As shown in group ② of Table 4, we find consistently decline of the model performance when one or more sub models are removed, suggesting that each model learns advantages from other models. Most importantly, we observe a large decrease (4.5/5.1 F1 points) on Vehicle F1/Tenor F1 when model_t/model_v is removed. These results strongly suggest the importance of utilizing the proposed decoding features. Besides, we also observe a performance decrease by removing model_p, suggesting that it also helps improve the whole system. By combining both comparison results, we can reach the conclusion: knowledge distillation with the ensemble of all three sub models (model_t, model_v and model_p) is important for the overall performance of our model.

5.5 Case Study

Based on the ground-truth results, we analyze the prediction results of *Cyc-MTL* and *HGSR* on the test set, then we group the errors into four major types and count their respective occurrences. The four types of errors are: *component dropping*, where an output misses important simile components; *locating error*, where a wrong span is extracted as a simile component; *simile classification error*, where a simile/literal sentence is erroneously considered as the other type; *redundant component extraction*, where extra spans (in addition to the gold spans) are extracted as simile components. For better illustration, we list several representative cases, as shown in Table 5.

In general, *HGSR* is much better than *Cyc-MTL* regarding the first 3 types of errors. It can particularly reduce the *locating error* issue, where the error reduction is more than 50%. Besides, it also largely alleviates the *component dropping* issue. Both situations are highly correlated with the dependency information, which can be well represented by *HGSR*. One typical example is the second

Model	Sentence Classification			Component Extraction		
	Precision	Recall	F1	Precision	Recall	F1
BSR	87.06	93.48	90.16	74.51	78.56	76.48
Cyc-MTL (Zeng et al., 2020)	86.46	**95.03**	90.54	75.07	79.91	77.41
HGSR	**89.04**	94.39	**91.93**	**81.86**	**88.37**	**85.42**

Table 3: The model performance with RoBERTa.

Model		Sentence Classification (F1)	Component Extraction		
			Tenor (F1)	Vehicle (F1)	Overall (F1)
HGSR		**91.64**	**90.54**	**91.11**	**84.99**
①	w/o dependency	91.36	85.04	90.77	83.43
	w/o POS	91.20	87.41	89.13	83.69
	w/o definitions	90.83	85.75	89.54	82.07
	w/o subsentence nodes	90.64	87.62	90.91	83.83
②	w/o model$_t$	91.23	89.95	86.61	83.26
	w/o model$_v$	91.42	85.40	90.95	83.52
	w/o model$_p$	91.44	88.84	90.30	83.12
BSR		89.84	84.13	87.18	75.12

Table 4: Ablation study on the main test set, where "①" and "②" represent the input features and decoding features, respectively.

case in Table 5. *Cyc-MTL* extracts "tortoise", probably because it is the main entity in the sentence. On the other hand, *HGSR* correctly extracts "shell", which directly connects with the comparator "like" in the dependency tree. For the last case, *HGSR* predicts both "sky" and "it", where they form a coreference relation. We consider the prediction of *HGSR* as reasonable, though the reference does not contain "it".

6 Related work

Our related work mainly includes the studies of simile recognition and heterogeneous neural network for NLP.

Simile Recognition. Early studies mainly focus on classifiers based on manually created patterns and syntactic features. For example, Bin et al. (2008) adopts a maximum entropy model to recognize simile sentences. In addition, Niculae (2013) uses syntactic patterns to extract potential simile components. Niculae and Danescu-Niculescu-Mizil (2014) aims to distinguish between figurative and literal by using a series of linguistics cues as features. However, such pattern-based methods can not deal with the sentences with complex syntactic structures. Inspired by promising results of deep neural networks, Liu et al. (2018) and Zhang et al. (2019b) introduce multitask learning into simile recognition. Furthermore, Zeng et al. (2020) proposes Cyc-MTL that considers the inter-correlation between different subtasks of simile recognition.

HGNN for NLP. Recently, heterogeneous graph neural network (HGNN) has been shown effective in several NLP tasks, such as relation extraction(Zhang et al., 2018), sentence ordering(Yin et al., 2019; Lai et al., 2021; Yin et al., 2021), graph node classification (Wang et al., 2019), question answering (Tu et al., 2019), intent recommendation (Fan et al., 2019), text classification (Wang et al., 2021), event detection (Wang et al., 2018; Cui et al., 2020a), machine translation (Yin et al., 2020) and document summarization (Wang et al., 2020). To our knowledge, this is the first attempt to explore HGNN for simile recognition. Besides, different from previous work that mainly focuses on encoding one type of features (e.g. dependency tree), ours explores more relevant features to enhance graph representations.

7 Conclusion

In this paper, we propose *HGSR*, which gets the most out of task features to alleviate the data hunger issue for simile recognition. Concretely, we explore the input-side features and the decoding fea-

Type & Cnt	Example	Cyc-MTL	HGSR
Component dropping (137/77)	一篇篇作文、日记像泉水一样从笔下涌出。 *Pieces of essays and diaries gush out from the writing like spring.*	Type: simile Tenor: 日记*(diaries)* Vehicle: 泉水*(spring)*	Type: simile Tenor: 日记*(diaries)* Tenor: 作文*(essays)* Vehicle: 泉水*(spring)*
Locating error (86/57)	乌龟的壳像小山。 *The shell of the tortoise is like a hill.*	Type: simile Tenor: 乌龟*(tortoise)* Vehicle: 小山*(hill)*	Type: simile Tenor: 壳*(shell)* Vehicle: 小山*(hill)*
Simile classification error (197/156)	如果不保护动物，大熊猫迟早会像恐龙一样灭绝。 *If animals are not protected, pandas will become extinct like dinosaurs sooner or later.*	Type: simile Tenor: 熊猫*(pandas)* Vehicle: 恐龙*(dinosaurs)*	Type: literal
Redundant component extraction (15/18)	天空中没有星星，它像一个巨大的黑洞。 *There are no stars in the sky, it likes a huge black hole.*	Type: simile Tenor: 它*(it)* Vehicle: 星星*(stars)*	Type: simile Tenor: 天空*(sky)* Tenor: 它*(it)* Vehicle:黑洞*(black hole)*

Table 5: Case Study on four major types of errors. The gold answers are highlighted with blue and green colors in the "*Example*" column. The counts separated by "/" in the first column represent the number of mistakes made by *Cyc-MTL* and *HGSR* for each error type, respectively.

tures. The input-side features, which includes POS tags, dependency tree and word definitions are encoded via heterogeneous graph encoding. For the decoding features, we build two models sequentially extracted simile components in the opposite orders, then force these two models and the basic model which extracts components in parallel to mimic the behavior of their ensemble. During inference time, only one of these models will be used. Experimental results and in-depth analyses demonstrate the superiority of our model under both sufficient and insufficient data settings.

Limitations

The limitations of this work are the following aspects: 1) In this work, experiments are conducted only on Chinese due to the availability of descent-scaled annotated data. We will evaluate the proposed model on other languages once the corresponding large-scale datasets are available. 2) Same as previous work (Liu et al., 2018; Zeng et al., 2020), we only focus on the result of simile recognition itself, ignoring further discussions on its contribution for other tasks. 3) The proposed HGSR uses model ensemble as the teacher model for knowledge distillation during training, making the training phase not eco-friendly. We plan to explore the lightweight models and investigate alternative eco-friendly plans in the future.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and suggestions. This research is supported by National Natural Science Foundation of China (No. 62276219), Natural Science Foundation of Fujian Province of China (No. 2020J06001), and Youth Innovation Fund of Xiamen (No. 3502Z20206059).

References

L. I. Bin, Y. U. Li-Li, Shi Min, and Q. U. Wei-Guang. 2008. Computation of chinese simile with "xiang". *Journal of Chinese Information Processing*.

Shiyao Cui, Bowen Yu, Tingwen Liu, Zhenyu Zhang, Xuebin Wang, and Jinqiao Shi. 2020a. Edge-enhanced graph convolution networks for event detection with syntactic relation. In *EMNLP findings 2020*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020b. Revisiting pre-trained models for chinese natural language processing. In *EMNLP findings 2020*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL 2019*.

Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-guided heterogeneous graph neural network for intent recommendation. In *KDD 2019*.

Patrick Hanks. 2012. The roles and structure of comparisons similes and metaphors in natural language (an analogical system). *Presented at the Stockholm Metaphor Festival*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Shaopeng Lai, Ante Wang, Fandong Meng, Jie Zhou, Yubin Ge, Jiali Zeng, Junfeng Yao, Degen Huang, and Jinsong Su. 2021. Improving graph-based sentence ordering with iteratively predicted pairwise orderings. In *EMNLP 2021*.

Bin Li, Haibo Kuang, Yingjie Zhang, Jiajun Chen, and Xuri Tang. 2012. Using similes to extract basic sentiments across languages. In *WISE 2012*.

Lizhen Liu, Xiao Hu, Wei Song, Ruiji Fu, Ting Liu, and Guoping Hu. 2018. Neural multitask learning for simile recognition. In *EMNLP 2018*.

Vlad Niculae. 2013. Comparison pattern matching and creative simile recognition. In *JSSP 2013*.

Vlad Niculae and Cristian Danescu-Niculescu-Mizil. 2014. Brighter than gold: Figurative language in user generated comparisons. In *EMNLP 2014*.

Vlad Niculae and Victoria Yaneva. 2013. Computational considerations of comparisons and similes. In *ACL workshop 2013*.

Ashequl Qadir, Ellen Riloff, and Marilyn Walker. 2015. Learning to recognize affective polarity in similes. In *EMNLP 2015*.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL 2021*.

Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. 2019. Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs. In *ACL 2019*.

Andrea Vanzo, Emanuele Bastianelli, and Oliver Lemon. 2019. Hierarchical multi-task natural language understanding for cross-domain conversational AI: HERMIT NLU. In *SIGDAL 2019*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS 2017*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *ACL 2020*.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW 2019*.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *EMNLP 2018*.

Ziyun Wang, Xuan Liu, Peiji Yang, Shixing Liu, and Zhisheng Wang. 2021. Cross-lingual text classification with heterogeneous graph neural network. *arXiv preprint arXiv:2105.11246*.

Changxing Wu, Liuwen Cao, Yubin Ge, Yang Liu, Min Zhang, and Jinsong Su. 2022. A label dependence-aware sequence generation model for multi-level implicit discourse relation recognition. In *AAAI 2022*.

Wenlin Yao, Xiaoman Pan, Lifeng Jin, Jianshu Chen, Dian Yu, and Dong Yu. 2021. Connect-the-dots: Bridging semantics between words and definitions via aligning word sense inventories. In *EMNLP 2021*.

Yongjing Yin, Shaopeng Lai, Linfeng Song, Chulun Zhou, Xianpei Han, Junfeng Yao, and Jinsong Su. 2021. An external knowledge enhanced graph-based neural network for sentence ordering. *JAIR*.

Yongjing Yin, Fandong Meng, Jinsong Su, Chulun Zhou, Zhengyuan Yang, Jie Zhou, and Jiebo Luo. 2020. A novel graph-based multi-modal fusion encoder for neural machine translation. In *ACL 2020*.

Yongjing Yin, Linfeng Song, Jinsong Su, Jiali Zeng, Chulun Zhou, and Jiebo Luo. 2019. Graph-based neural sentence ordering. In *IJCAI 2019*.

Jiali Zeng, Linfeng Song, Jinsong Su, Jun Xie, Wei Song, and Jiebo Luo. 2020. Neural simile recognition with cyclic multitask learning and local attention. In *AAAI 2020*.

Biao Zhang, Deyi Xiong, Jinsong Su, and Jiebo Luo. 2019a. Future-aware knowledge distillation for neural machine translation. *TASLP*.

Pengfei Zhang, Yi Cai, Junying Chen, Wenhao Chen, and Hengjie Song. 2019b. Combining part-of-speech tags and self-attention mechanism for simile recognition. *IEEE Access*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP 2018*.