SemEval-2021 Task 4: Reading Comprehension of Abstract Meaning

Boyuan Zheng[2][*], **Xiaoyu Yang[1]**, **Yu-Ping Ruan[3]**, **Zhenhua Ling[3]**,
Quan Liu[3], **Si Wei[4]**, **Xiaodan Zhu[1]**

[1]Queen's University, Kingston, Canada; [2]Northeastern University, Shenyang, China
[3]University of Science and Technology of China; [4]iFlytek Research, Hefei, China
steven.zheng010@gmail.com; xiaoyu.yang@queensu.ca;
{quanliu,zhling}@ustc.edu.cn; siwei@iFlytek.com.cn;
xiaodan.zhu@queensu.ca

Abstract

This paper introduces the SemEval-2021 shared task 4: Reading Comprehension of Abstract Meaning (ReCAM). This shared task is designed to help evaluate the ability of machines in representing and understanding abstract concepts. Given a passage and the corresponding question, a participating system is expected to choose the correct answer from five candidates of abstract concepts in a cloze-style machine reading comprehension setup. Based on two typical definitions of *abstractness*, i.e., the *imperceptibility* and *nonspecificity*, our task provides three subtasks to evaluate the participating models. Specifically, Subtask 1 aims to evaluate how well a system can model concepts that cannot be directly perceived in the physical world. Subtask 2 focuses on models' ability in comprehending *nonspecific* concepts located high in a hypernym hierarchy given the context of a passage. Subtask 3 aims to provide some insights into models' generalizability over the two types of abstractness. During the SemEval-2021 official evaluation period, we received 23 submissions to Subtask 1 and 28 to Subtask 2. The participating teams additionally made 29 submissions to Subtask 3. The leaderboard and competition website can be found at https://competitions.codalab.org/competitions/26153. The data and baseline code are available at https://github.com/boyuanzheng010/SemEval2021-Reading-Comprehension-of-Abstract-Meaning.

1 Introduction

Humans use words with abstract meaning in their daily life. In the past, research efforts have been exerted to better understand and model abstract meaning (Turney et al., 2011; Theijssen et al.,

2011; Changizi, 2008; Spreen and Schulz, 1966). Modelling abstract meaning is closely related to many other NLP tasks such as reading comprehension, metaphor modelling, sentiment analysis, summarization, and word sense disambiguation.

In the past decade, significant advancement has been seen in developing computational models for semantics, based on deep neural networks. In this shared task, we aim to help assess the capability of the state-of-the-art deep learning models on representing and modelling abstract concepts in a specific reading comprehension setup.

We introduce SemEval-2021 Task 4, Reading Comprehension of Abstract Meaning (**ReCAM**). Specifically, we design this shared task by following the machine reading comprehension framework (Hermann et al., 2015; Onishi et al., 2016; Hill et al., 2016), in which computers are given a passage D_i as well as a human summary S_i to comprehend. If a model can digest the passage as humans do, we expect it to predict the abstract word used in the summary, if the abstract word is masked. Unlike the previous work that requires computers to predict concrete concepts, e.g., named entities, in our task we ask models to fill in abstract words removed from human summaries. During the SemEval-2021 official evaluation period, we received 23 submissions to Subtask 1 and 28 submissions to Subtask 2. The participating teams additionally made 29 submissions to Subtask 3. In this paper, we induce the shared task and provide a summary for the evaluation.

2 Task Description

We organize our shared task based on two typical definitions of *abstractness*, named as *imperceptibility* and *nonspecificity* in this paper, implemented in Subtask 1 and Subtask 2, respectively. Subtask 3 further evaluates models' generalizability over the two definitions of abstractness.

---

[*] This work was performed when Boyuan Zheng visited Queen's University.

Passage	... Observers have even named it after him, "Abenomics". It is based on three key pillars of monetary policy to ensure long-term sustainable growth in the world's third-largest economy, with fiscal stimulus and structural reforms. In this weekend's upper house elections, ...
Question	Abenomics: The @*placeholder* and the risk.
Answer	(A) chance (B) prospective (C) government **(D) objective** (E) threat

Table 1: An example for Subtask 1. The correct answer to the question is ***objective***.

2.1 Subtask 1: ReCAM-Imperceptibility

In one definition (Turney et al., 2011; Theijssen et al., 2011; Spreen and Schulz, 1966), concrete words refer to things, events, and properties that humans can directly perceive with their senses, e.g., *trees* and *flowers*. In contrast, abstract words refer to "ideas and concepts that are distant from immediate perception", e.g., *objective*, *culture*, and *economy*. In Subtask 1, we perform reading comprehension on *imperceptible* abstract concepts, named as **ReCAM-ImPerceptibility**. Table 1 shows an example.

2.2 Subtask 2: ReCAM-NonSpecificity

The second typical definition of abstractness is based on *nonspecific* concepts (Theijssen et al., 2011; Spreen and Schulz, 1966). Compared to specific concepts such as *groundhog* and *whale*, words such as *vertebrate* are regarded as more *abstract*. Our Subtask 2, named as **ReCAM-NonSpecificity**, is designed based on this viewpoint. We will discuss how the datasets are constructed in Section 3.

2.3 Subtask 3: ReCAM-Cross

In this subtask, participants are asked to submit their predictions on the test data of Subtask 2, using models trained on the training data of Subtask 1, and vice versa. This subtask aims to demonstrate models' generalizability between modelling the two typical definitions of abstractness.

3 Data Construction

We develop our multi-choice machine reading comprehension datasets based on the XSum summarization dataset (Narayan et al., 2018). We first locate words with abstract meaning using our abstractness scorers. Then we perform data filtering to select our target words to construct our datasets.

3.1 The XSum Data

By collecting online articles from the British Broadcasting Corporation (BBC), Narayan et al. (2018) developed a large-scale text summarization dataset, XSum, in which each article has a single sentence summary. We developed our ReCAM dataset based on XSum.

3.2 Finding *Imperceptible* Concepts

Abstractness Scorer for Imperceptibility   Following Turney et al. (2011), we use the MRC Psycholinguistic Database (Coltheart, 1981), which includes 4,295 words rated with a degree of abstractness by human subjects, to train our abstractness scorer for *imperceptibility*. The rating of the words in the MRC Psycholinguistic Database ranges from 158 (highly abstract) to 670 (highly concrete). We linearly scale the rating to the range of 0 (highly abstract) to 1 (highly concrete). The neural regression model accepts fixed Glove embedding (Pennington et al., 2014) as input and predicts the abstractness rating score between 0 and 1. Our regression model is a three-layer network that consists of two non-linear hidden layers with the ReLU activation and a sigmoid output layer. The mean square error (MSE) is used as the training loss.

To test the regression model's performance, we randomly split the MRC Psycholinguistic Database into train and test set with the size of 2,148 and 1,877, respectively. Table 2 shows the final performance of the neural regression model on the MRC database. We use the Pearson correlation between ratings predicted by models and original ratings from MRC as the evaluation metric. We can see that the regression model achieves high correlation coefficients (the higher, the better), i.e., 0.934 and 0.835, on the training and test set. The correlations are significant (p-values are smaller than 10^{-5}), reflecting the quality of our models in finding abstract words. Note that Turney et al. (2011) report a correlation score of 0.81 on their MRC test set. Their training-test split is unavailable, so we run cross-validation here in our experiment. The scorer can then be used to assign an *imperceptibility* score to a word that is not in the MRC Psycholinguistic Database.

Using the abstractness scorer described above, we assign an abstractness value to each word in summaries and select words with a value lower than 0.35 as the candidates for our *target words* (words that will be removed from the summaries

	#samples	Pearson r	p-value
train	2,148	0.934	$p < 10^{-5}$
test	1,877	0.854	$p < 10^{-5}$

Table 2: Fitting performance of neural regression model on the MRC database.

to construct questions). We only consider content words as potential target words, i.e., nouns, verbs, adjectives, and adverbs. For this purpose, we use part-of-speech tagging model (**?**) implemented in Stanza (Qi et al., 2020).

3.3 Finding *Nonspecific* Concepts

Nonspecificity Scorer Following the work of Changizi (2008), we assign a *nonspecificity* score to a word token based on the hypernym hierarchy of WordNet (Miller, 1998). Specifically, the root of the hierarchy is at level 0 and regarded as the most abstract. The abstractness of a node in the hierarchy is measured by the maximal length of its path to the root. The hypernym level in WordNet is between 0 and 17. For each word token in summaries, we use Adapted Lesk Algorithm (Banerjee and Pedersen, 2002) to label the sense since the WordNet hypernym hierarchy works at the sense level. Since a summary sentence may be short, we concatenate each summary sentence with the corresponding passage for word sense disambiguation. Built on this, each token, which is labelled with a sense, receives an abstractness score based on the WordNet hierarchy.

Using the *nonspecificity* scorer, we assign an *nonspecificity* value to each word in summaries and select words with a value smaller than six as the candidate target words. The targets words will be nouns and verbs since the hypernym hierarchy in WordNet (**?**) consists of these two POS types.

3.4 Filtering

We aim to avoid developing simple questions. For example, if a target word also appears in the passage, it is likely that a model can easily find the answer without the need to understand the passage in depth.

Filtering by Lemmas We lemmatized passages and summaries. If a lemma appears both in a summary and the corresponding passage, the lexemes of the lemma will not be considered as target words. Note that a strict filter may exclude some good candidates for target words but helps avoid introducing many simple questions.

Filtering by Synonyms and Antonyms For a word in a summary, if a synonym or antonym of the word appears in the corresponding passage, we will not consider this word to be our target word. We use WordNet (**?**) to derive synonyms and antonyms. Instead of using word sense disambiguation (WSD), for a word w_i in a summary, we use all senses of this word and add all synonyms and antonyms into a pool. Only if none of the words in the pool appear in the passage, we consider w_i as a candidate target word. Otherwise, we will not use w_i to construct a question for this passage-summary pair.

Filtering by Similarity We further filter words by similarity. For each candidate target word in a summary and each word in the passage, we calculate similarity and use that to perform further filtering.

We use 300-dimension GloVe word embedding trained on 840 billion tokens (Pennington et al., 2014). We calculate the cosine similarity between a candidate target word and a passage word. For contextual embedding, we embed each sentence in a passage as well as the summary into a context-aware representation matrix using the BERT-large uncased language model. Then, we calculate the similarity between each passage token and question token with the cosine similarity. If the similarity is higher than 0.85, we will not consider the involved summary words as candidate target words.

3.5 Constructing Multiple Choices

We train machine reading comprehension models using the data built so far to generate four choices for each question. Together with the ground-truth (the target word identified above and removed from the human summary), we have five choices/options for each question. In our work, we propose to use three models, Gated-Attention Reader (Hermann et al., 2015), Attentive Model and Attention Model with Word Gloss to generate the candidate options. Please find details of the models in Appendix B and Appendix C as well as the training details in Appendix D.

We adopt the idea of k-fold cross validation to train the above mentioned three models to generate candidate answer words. Specifically, we split the data into 4 folds. Each time, we train the baseline models on 3 folds of data and use the trained

	MRR	R@1	R@5	R@10
GAReader	0.245	0.175	0.314	0.378
AttReader	0.235	0.167	0.300	0.363
+gloss	0.179	0.123	0.227	0.276

Table 3: Three baseline models are used to generate candidate multiple choices for Subtask 1. The table shows their performance on the XSum dataset, evaluated with MRR(Craswell, 2009), Recall@1, Recall@5, and Recall@10.

	MRR	R@1	R@5	R@10
GAReader	0.343	0.268	0.422	0.484
AttReader	0.348	0.273	0.424	0.490
+gloss	0.228	0.166	0.286	0.345

Table 4: Three baseline models are used to generate candidate multiple choices for Subtask 2. The table shows their performance on the XSum dataset, evaluated with MRR, Recall@1, Recall@5, and Recall@10.

models to predict candidate words on the remaining 1-fold data. With 4-fold iteration, we obtain predication of each model on the entire data. The performance of the three baseline models are listed in Table 3 for Subtask 1 and Table 4 for Subtask 2, using several typical retrieval-based evaluation metrics.

For each target word that has been removed from the corresponding summary sentence (again, a question is a summary sentence containing a removed target word), we collect top-10 words predicted by each of the three models. In this way, we can collect a candidate word pool of 30 predicted word tokens for each removed target word. To avoid including multiple correct choices for each question, we adopt synonym and context similarity filtering methods described in Section 3.4. Specifically we first calculate similarity between the ground-truth target word and each word type in the pool. We exclude a word type from the multiple choices if its similarity to the ground-truth is higher than 0.85. In addition, we also exclude synonyms of the ground-truth target word. For the remaining word tokens in the pool, we select four most frequent word types (a word type may have multiple tokens in the pool). Together with the ground-truth word, we obtain five choices for each question.

3.6 Further Quality Control

We further make the following efforts to remove noise in the dataset and improve the datasets' qual-

ity. We observe that up to now, there are mainly two kinds of noise in our dataset: 1) some target words cannot be inferred solely based on the corresponding passage; 2) more than one of the multiple choices are correct answers.

The first issue is mainly related to the property of the XSum dataset, in which the first sentence of a passage is used as the summary. The second type of problems are often caused by our automatic generation method. Although we have applied strict rules in Section 3.4 to handle this, among a small portion of the resulting data, multiple potentially correct answers still exist in candidate answers.

To further ensure the quality of our dataset, we invite workers in Amazon Mechanical Turk to perform further data selection. Each annotator needs to follow the procedure of Appendix A to answer the question and annotate relevant information, with which further data selection is applied. To ensure quality, we only include workers from English-speaking countries and only if their previous HITs' approval rates are above 90%. To see more details about this process, please refer to Appendix E.

3.7 ReCAM Data Statistics

Table 5 lists the size of our ReCAM datasets, i.e., numbers of questions. For example, in total Subtask 2 has 6,186 questions, which are split into training/development/test subsets.

Dataset	Subtask 1	Subtask 2	Total
Train	3,227	3,318	6,545
Dev	837	851	1,688
Test	2,025	2,017	4,042
Total	6,089	6,186	12,275

Table 5: Size of the ReCAM Dataset.

4   Systems and Results

Our shared task received 23 submissions to Subtask 1, 28 submissions to Subtask 2, and 29 submissions to Subtask 3. We use *accuracy* as the evaluation metric for the three subtasks.

In general, most participating teams use pre-trained language models in their systems such as BERT (Devlin et al., 2019), ALBERT (Lan et al., 2020), DistilBERT (Sanh et al., 2019), RoBERTa (Liu et al., 2019), ELECTRA (Clark et al., 2020), DeBERTa (He et al., 2020), XL-Net (Yang et al., 2019), T5 (Raffel et al., 2020). Data augmentation, external knowledge resources,

and/or transfer learning are additionally used by many teams to further enhance their model performance.

4.1 Subtask 1: ReCAM-Imperceptibility

Table 6 shows all the official submissions and most of them outperform the baseline model. The baseline used for Subtask 1 is the Gated-Attention (GA) Reader (Dhingra et al., 2017). The GA Reader uses a multi-layer iterated architecture with a gated attention mechanism to derive better query-aware passage representation. The motivation behind using GA Reader is to have a simple comparison between our task and the CNN/Daily Mail reading comprehension dataset since GA Reader achieves reasonably good performance on the CNN/Daily Mail reading comprehension dataset.

Note that the last column of the table lists the accuracy (*Acc. Cross*) for models trained on the Subtask 2 training data and tested on the Subtask 1 testset. We will discuss those results later in Section 4.3.

The best result in Subtask 1 was achieved by team SRC-B-roc (Zhang et al., 2021) with an accuracy of 0.951. The system was built on a pre-trained ELECTRA discriminator and it further applied upper attention and auto-denoising mechanism to process long sequences. The second-placed system, PINGAN omini-Sinitic (Wang et al., 2021), adopted an ensemble of ELECTRA-based models with task-adaptive pre-training and a multi-head attention based multiple-choice classifier. ECNU-ICA-1 (Liu et al., 2021) ranked third in this subtask with a knowledge-enhanced Graph Attention Network and a semantic space transformation strategy.

Most teams in Subtask 1 utilize pre-trained language models (PLM), like BERT (Devlin et al., 2019), ALBERT (Lan et al., 2020), DistilBERT (Sanh et al., 2019), RoBERTa (Liu et al., 2019), ELECTRA (Clark et al., 2020), DeBERTa (He et al., 2020), XLNet (Yang et al., 2019), T5 (Raffel et al., 2020). SRC-B-roc (Zhang et al., 2021) conducted an ablation study regarding the performance discrepancy of different transformers-based pre-training models. They tested BERT, ALBERT, and ELECTRA by directly fine-tuning the pre-trained LMs on the ReCAM data. ELECTRA outperforms BERT and ALBERT by large margins, which may be due to the different learning objectives of these pre-trained models.

Rank	Team	Acc	Acc. Cross
-	GA Reader	25.1	-
1	SRC-B-roc	95.1	91.8 (↓ 3.3)
2	PINGAN-Omini-Sinitic	93.0	91.7 (↓ 1.3)
3	ECNU-ICA-1	90.5	88.6(↓ 1.9)
4	tt123	90.0	86.2(↓ 3.8)
5	cxn	88.7	-
6	nxc	88.6	74.2(↓ 14.4)
7	ZJUKLAB	87.9	-
8	IIE-NLP-Eyas	87.5	82.1(↓ 5.4)
9	hzxx1997	86.7	-
10	XRJL	86.7	81.8(↓ 4.9)
11	noobs	86.2	78.6(↓ 7.6)
12	godrevl	83.1	-
13	ReCAM@IITK	82.1	80.7(↓ 1.4)
14	DeepBlueAI	81.8	76.3(↓ 5.5)
15	LRG	75.3	61.8(↓ 13.5)
16	xuliang	74.7	-
17	Llf1206571288	72.8	-
18	Qing	71.4	-
19	NEUer	56.6	51.8(↓ 4.8)
20	CCLAB	46.3	35.2(↓ 11.1)
21	UoR	42.0	39.4(↓ 2.6)
22	munia	19.3	-
23	BaoShanCollege	19.0	-

Table 6: Official results of Subtask 1 and Subtask 3. *Acc* is the accuracy of the models trained on the Subtask 1 training data and tested on the Subtask 1 testset. *Acc. cross* is the accuracy of models trained on the Subtask 2 training data and tested on the Subtask 1 testset.

Most participating systems performed intermediate task pre-training (Pruksachatkun et al., 2020) for their language models. For example, CNN/Daily Mail dataset was selected by ZJUKLAB (Xie et al., 2021a) to further pre-train their language models. The CNN/Daily Mail dataset and Newsroom dataset boost model performance on both Subtask 1 and Subtask 2. Data augmentation methods are also popular among participants. ZJUKLAB (Xie et al., 2021a) performed negative data augmentation with a

language model to leverage misleading words. `IIE-NLP-Eyas` (Xie et al., 2021b) adopted template-based input reconstruction methods to augment their dataset and further fine-tuned their language models based on the dataset.

Most teams also used an ensemble of multiple pre-trained language models to further enhance model performance. `SRC-B-roc` (Zhang et al., 2021) applied Wrong Answer Ensemble (Kim and Fung, 2020) by training the model to learn the correct and wrong answer separately and ensembled them to obtain the final predictions. Stochastic Weight Averaging (Izmailov et al., 2018) was also performed across multiple checkpoints in the same run to achieve better generalization.

In addition, some interesting approaches were additionally used to tackle the task from different perspectives. `PINGAN omini-Sinitic` (Wang et al., 2021) turned the original multi-choice task into a masked-sentence classification task by adding each option to the placeholder. Noise detection methods and auto de-noising methods were further proposed by adding a noise-tolerant loss. `ZJUKLAB` (Xie et al., 2021a) used label smoothing to encourage the activations of the penultimate layer. `ECNU-ICA-1` (Liu et al., 2021) utilized a semantic space transformation strategy to convert ordinary semantic representations into abstract representations for classification.

Many teams used external knowledge resources to further improve model performance. WordNet (Fellbaum, 1998) was widely used to provide candidate word definitions. `ECNU-ICA-1` (Liu et al., 2021) also used ConceptNet5 (Speer et al., 2016) and Graph Neural Network in their systems. To alleviate the noise induced by incorporating structured knowledge through unimportant edges, they propose a noise reduction strategy. `owlmx` used the MRC Psycholinguistic Database to obtain a measurement of *imperceptibility* abstractness.

Different pre-processing techniques were proposed in multiple systems. `ZJUKLAB` (Xie et al., 2021a) used a sliding window to limit input length in training. `PINGAN Omini-Sinitic` (Wang et al., 2021) used the cycle noisy label detection algorithm to make models more robust.

Much interesting analysis regarding the failure cases and data distribution was discussed in several system description papers. `XRJL` (Jiang et al., 2021) found that for a few questions, common

Rank	Team	Acc.	Acc. Cross
-	GA Reader	24.3	-
1	PINGAN-Omini-Sinitic	95.3	94.2 (↓ 1.1)
2	SRC-B-roc	94.9	93.9(↓ 1.0)
3	tt123	93.4	85.8(↓ 7.6)
4	ECNU-ICA-1	93.0	92.8(↓ 0.2)
5	cxn	92.9	-
6	ZJUKLAB	92.8	-
7	nxc	92.7	-
8	hzxx1997	90.2	-
9	XRJL	90.0	87.6(↓ 2.4)
10	IIE-NLP-Eyas	89.6	84.1(↓ 5.5)
11	ReCAM@IITK	87.6	85.2(↓ 2.4)
12	noobs	87.1	82.4(↓ 4.7)
13	DeepBlueAI	86.2	80.7(↓ 5.5)
14	xuliang	81.0	-
15	LRG	77.8	65.6(↓ 12.2)
16	Yotta	71.6	-
17	sayazzad	68.3	-
18	itanhisada	67.7	-
19	NEUer	66.9	45.0(↓ 21.9)
20	YaA@JUST	66.1	-
21	NLP-IIS@UT	64.4	-
22	CCLAB	48.1	31.8(↓ 16.3)
23	K-FUT	47.6	-
24	owlmx	44.8	31.0(↓13.8)
25	UIT-ISE-NLP	42.0	27.3(↓ 14.7)
26	UoR	39.1	34.2(↓ 4.9)
27	Noor	19.9	-
28	BaoShanCollege	17.6	-

Table 7: Official results of Subtask 2 and Subtask 3. *Acc* is the accuracy (%) of the models trained on the Subtask 2 training data and tested on the Subtask 2 test-set. *Acc. Cross* is the accuracy(%) of models trained on the Subtask 1 training data and tested on the Subtask 2 testset.

sense knowledge was further needed to help find the answer. They also pointed out that there were still a few questions in which multiple candidate choices may serve as appropriate answers.

4.2 Subtask 2: ReCAM-Nonspecificity

In Subtask 2, we received 28 submissions. Table 7 shows the official leaderboard. The best result in Subtask 2 was achieved by team `PINGAN omini-Sinitic` (Wang et al., 2021) with an accuracy of 0.953, using a model similar to the team's model in Subtask 1. The second-placed team `SRC-B-roc` (Zhang et al., 2021) also adopted the same model it used in Subtask 1 with a data augmentation method based on the hypernym hierarchy in WordNet.

In general, the participating teams in Subtask 2 used pre-trained language models and neural networks similar to those they used in Subtask 1. The main differences lie in how the participants performed data augmentation and leveraged external knowledge. For example, in addition to `SRC-B-roc` (Zhang et al., 2021), the `IRG` team (Sharma et al., 2021) also performed data augmentation using hypernyms from WordNet.

4.3 Subtask 3: Cross-task Performance

In this section, we explore models' performance across the two types of definitions of abstractness. Specifically, in this subtask, participants train their models on the training set of one subtask and test on the testset of the other subtask. We received 29 submissions in total from the participants.

Cross-task performance: Subtask 2-to-1 testing. We asked participants to test their models trained on the Subtask 2 training data on the Subtask 1 test data. The results are shown in the last column of Table 6.

The results we received show that the performance of all systems drops substantially. For some systems ranking among top 10, the accuracy can decrease by 5 points (`IIE-NLP-Eyas` (Xie et al., 2021b) and `XRJL` (Jiang et al., 2021)), or even more (14 points for `nxc`). Some systems show good generalization ability in this Subtask 2-to-1 scenario; the performance of `PINGAN-Omini-Sinitic` (Wang et al., 2021) is only 1.3 point less, which may be due to the the data augmentation and task adaptive training used in the model.

Cross-task Performance: Subtask 1-to-2 Testing. Participants are asked to test their Subtask 1 systems on the Subtask 2 testset. Details of the results can be seen in the last column of Table 7. All systems' performances drop. For example, among the top-10 systems, the accuracy decreases by 5 points (`IIE-NLP-Eyas` (Xie et al., 2021b)) or 7 points (`tt123`).

However, `ECNU-ICA-1` (Liu et al., 2021) shows a very good generalization ability in Subtask 1-to-2 testing. `PINGAN-Omini-Sinitic` (Wang et al., 2021), `SRC-B-roc` (Zhang et al., 2021) and `XRJL` (Jiang et al., 2021)'s systems are rather consistent in this Subtask 1-to-2 cross testing. Some algorithms they used may explain the models' good generalization ability. `ECNU-ICA-1`'s algorithm of using knowledge-enhanced Graph Attention Network can provide external knowledge to the model. The Wrong Answer Ensemble algorithm (Kim and Fung, 2020) used in `PINGAN-Omini-Sinitic` (Wang et al., 2021) is a relatively simple but an effective way of improving model performance and generalization ability. Also, the Stochastic Weight Averaging algorithm across multiple checkpoints is effective for better generalization. `XRJL` (Jiang et al., 2021) retrieves the definitions of candidate answers from WordNet and feeds them to the model as extra inputs. We also think data augmentation methods contribute to the generalization ability.

5 Related Work

There have been tasks being proposed to evaluate machines' ability on reading comprehension, which either require models to find an entity or text span from the source document as the answer (Hermann et al., 2015; Hill et al., 2016; Onishi et al., 2016; Rajpurkar et al., 2016; Trischler et al., 2017), or further generate an answer (Nguyen et al., 2016; He et al., 2018; Kočiský et al., 2018). The cloze-style MRC tasks (Hermann et al., 2015; Onishi et al., 2016; Hill et al., 2016) are most similar to ours, in which the missing words in the cloze questions are entities appearing in source documents. Unlike previous work, ReCAM questions specifically focus on abstract words unseen in the corresponding source documents.

In general, multi-choice questions have been widely used as a tool for language examination to test both humans and machines. In this paper, we follow the multiple-choice framework for our proposed ReCAM task to evaluate computers' ability in comprehending abstract concepts, in which computers are asked to predict the missing abstract words in human-written summaries.

6 Summary

This shared task aims to study the ability of machines in representing and understanding abstract concepts, based on two definitions of abstractness, the *imperceptibility* and *nonspecificity*, in a specific machine reading comprehension setup. We provide three subtasks to evaluate models' ability in comprehending the two types of abstract meaning as well as their generalizability. In Subtask 1, the top system achieves an accuracy of 0.951, and in Subtask 2, an accuracy of 0.953, suggesting the current systems perform well in the specific setup of our share task. In Subtask 3, we found that in general the models' performances dropped in both Subtask 2-to-1 and Subtask 1-to-2 testing. However, some models generalize well, benefiting from technologies such as data augmentation and task adaptive training. We hope the shared task can help shed some light on modelling abstract concepts and help design more challenging tasks in the future.

References

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Computational Linguistics and Intelligent Text Processing, Third International Conference, CICLing 2002, Mexico City, Mexico, February 17-23, 2002, Proceedings*, volume 2276 of *Lecture Notes in Computer Science*, pages 136–145. Springer.

Steven Bird and Edward Loper. 2004. NLTK: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Mark A. Changizi. 2008. Economically organized hierarchies in wordnet and the oxford english dictionary. *Cogn. Syst. Res.*, 9(3):214–228.

Kyunghyun Cho, Bart van Merriënboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pretraining text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Max Coltheart. 1981. The MRC psycholinguistic database. *The Quarterly Journal of Experimental Psychology*, 33(4):497–505.

Nick Craswell. 2009. Mean reciprocal rank. In *Encyclopedia of Database Systems*, pages 1703–1703. Springer.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1832–1846. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with disentangled attention. *CoRR*, abs/2006.03654.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. 2018. Dureader: a chinese machine reading comprehension dataset from real-world applications. In *Proceedings of the Workshop on Machine Reading for Question Answering@ACL 2018, Melbourne, Australia, July 19, 2018*, pages 37–46. Association for Computational Linguistics.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 876–885. AUAI Press.

Yuxin Jiang, Ziyi Shou, Qijun Wang, Hao Wu, and Fangzhen Lin. 2021. XRJL-HKUST at SemEval-2021 Task 4: Wordnet-enhanced dual multi-head co-attention for reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.

Hyeondey Kim and Pascale Fung. 2020. Learning to classify the wrong answers for multiple choice question answering (student abstract). In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 13843–13844. AAAI Press.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM.

Pingsheng Liu, Linlin Wang, Qian Zhao, Hao Chen, Yuxi Feng, Xin Lin, and Liang He. 2021. ECNU_ICA_1 SemEval-2021 Task 4: Leveraging knowledge-enhanced graph attention networks for reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

George Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1797–1807. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5231–5247. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Abheesht Sharma, Harshit Pandey, Gunjan Chhablani, Yash Bhartia, and Tirtharaj Dash. 2021. LRG at SemEval-2021 Task 4: Improving reading comprehension with abstract words using augmentation, linguistic features and voting. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2016. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence*.

Otfried Spreen and Rudolph W. Schulz. 1966. Parameters of abstraction, meaningfulness, and pronunciability for 329 nouns. *Journal of Verbal Learning and Verbal Behavior*, 5(5):459 – 468.

DL Theijssen, H van Halteren, LWJ Boves, and NHJ Oostdijk. 2011. On the difficulty of making concreteness concrete.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, pages 191–200. Association for Computational Linguistics.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 680–690. Association for Computational Linguistics.

Ye Wang, Yanmeng Wang, Haijun Zhu, Bo Zeng, Zhenghong Hao, Shaojun Wang, and Jing Xiao. 2021. PINGAN Omini-Sinitic at SemEval-2021 Task 4: Reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.

Xin Xie, Xiangnan Chen, Xiang Chen, Yong Wang, Ningyu Zhang, Shumin Deng, and Huajun Chen. 2021a. ZJUKLAB at SemEval-2021 Task 4: Negative augmentation with language model for reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.

Yuqiang Xie, Luxi Xing, Wei Peng, and Yue Hu. 2021b. IIE-NLP-Eyas at SemEval-2021 Task 4: Enhancing plm for recam with special tokens, re-ranking, siamese encoders and back translation. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Jing Zhang, Yimeng Zhuang, and Yinpei Su. 2021. TA-MAMC at SemEval-2021 Task4: Task-adaptive pretraining and multi-head attention for abstract meaning reading comprehension. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.

A Annotation Script

Please read the instruction before starting answering the question (about 2 minutes reading). The instruction will help you better understand the task. Here is a question with one word missing. Please pick one word to complete it.

Question: Wales are waiting to discover the ©**placeholder** of the knee injury lock Jake Ball suffered in Scarlets ' 24 - 15 win over Treviso on Friday .

- Before reading the passage, please select one word to answer the question.
  ☐ challenge   ☐ duties   ☐ extent   ☐ causes   ☐ future

Passage: Ball, 24, limped off towards the end of the Pro12 game at Parc Scarlets, a week before Wales face England in the Six Nations at Twickenham. NJake took a knock to the knee at the stage when we scored the try," Scarlets head coach Wayne Pivac said. "It was sensible to get him off as well." Pivac added: "I think he'll be all right - he's thrown a bit of ice so we'll see what the medical boys say." Wales coach Warren Gatland released lock Ball and scrum-half Aled Davies to the region as the Welsh squad prepare for their Six Nations showdown with England on 12 March. Media playback is not supported on this device Pivac also revealed Davies, 23, had been struggling with a virus while Scotland forward John Barclay also suffered an injury during the victory over bottom side Treviso. "John just took a stinger and felt he was favouring one side so he was going to let the team down so it was the right decision and he came off but I'm pretty sure he'll be OK," Pivac added. "Aled Davies was crook for the last couple of days with a virus and had to get through the warm-up. He did that and started so he was always going to run out of petrol. "It was a matter of how long he'd last and he did well to get to where he did."

- After reading and understanding the passage, select one word to answer the question.
  ☐challenge ☐ duties ☐ extent ☐ causes ☐ future

- If you think there is another answer to this question, please pick your second choice. Else, please pick the *"No Other Answer"* option.
  ☐challenge ☐ duties ☐ extent ☐ causes ☐ future ☐ No Other Answer

- Which part of the passage supports your answer? Copy the text span to the line below:
  Paste corresponding text span here: _____

- Does the answer summarize or paraphrase the text span?
  ☐ Summarize ☐ Paraphrase ☐ Others

- Which part of the passage summarize the information included in the answer? Copy the text span to the line below:
  Paste corresponding text span here: _____

- How difficult is this problem?
  ☐ Easy ☐ Medium ☐ Difficult

B Gated-Attention Reader

The Gated-Attention (GA) Reader (Dhingra et al., 2017), the state-of-art model on CNN/Daily Mail reading comprehension dataset (Hermann et al., 2015), is adapted here in our experiments. The GA Reader uses a multi-layer iterated architecture with a gated attention mechanism, which is based on multiplicative interactions between the query embedding and the intermediate states of a recurrent neural network document reader, to derive better query-aware passage representation. To apply GA Reader to our ARC task, we input the news passage p as the document and the processed summary s as the query to GA Reader.

Specifically, for an input passage $p = [p_1, p_2, ..., p_{l_p}]$ with l_p words and its corresponding summary $s = [s_1, s_2, ..., s_{l_s}]$ with l_s words, we first derive their corresponding word embedding sequence $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_{l_p}]$ and $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_{l_s}]$ respectively. Then the GA Reader accepts the $\mathbf{P}$ and $\mathbf{S}$ as inputs and return the hidden states $\mathbf{H}_p = [\mathbf{h}_1^p, \mathbf{h}_2^p, ..., \mathbf{h}_{l_p}^p]$ and $\mathbf{H}_s = [\mathbf{h}_1^s, \mathbf{h}_2^s, ..., \mathbf{h}_{l_s}^s]$ as the sequential representation for passage p and summary s respectively. As for the final prediction process, we do not adopt the operations in Dhingra et al. (2017) because in ARC the answer words are unseen in the corresponding passage, however, GA Reader in Dhingra et al. (2017) tries to select a entity word in the passage as the final prediction since their target answer word appears in the passage. So we redesign the part of prediction.

First, the corresponding representation of "@*placeholder*" in $\mathbf{H}_s$, denoted as $\mathbf{h}_q^s$ (q is the position index of @*placeholder* in summary s), is used as the final vector representation for summary s. For the final vector representation $\mathbf{p}$ for passage p, a bilinear attention between $\mathbf{h}_q^s$ and $\mathbf{H}_p$ is used for its derivation:

$$e_i = \mathbf{h}_q^{sT} \mathbf{W}^{att} \mathbf{h}_i^p, \forall i \in [1, ..., l_p] \quad (1)$$

$$\mathbf{p} = \sum_{i=1}^{l_p} \frac{\exp e_i}{\sum_{j=1}^{l_p} \exp e_j} \mathbf{h}_i^p, \quad (2)$$

We set a token embedding $\mathbf{a}_t^e$ for each candidate abstractive word a_t ($t \in [1, ..., n_c]$, n_c is the size of candidate set). We first concatenate the $\mathbf{h}_p^s$ and $\mathbf{p}$, then use the bilinear product and softmax to predict the probability distribution over all n_c candidate

abstractive words.

$$r_t = [\mathbf{h}_q^s; \mathbf{p}]^T \mathbf{W}_p \mathbf{a}_t^e, \forall t \in [1, ..., n_c], \quad (3)$$

$$o_t = softmax_t(r_t), \forall t \in [1, ..., n_c] \quad (4)$$

in which o_t represents the probability of predicting the candidate abstractive word a_t as the final answer.

C Attentive Model

The word gloss, which defines a word sense meaning, has been mainly used in word sense disambiguation (WSD) task and its variants (Lesk, 1986; Moro et al., 2014). Since the goal of ARC is to predict a word that can summarize corresponding information from the source passage, which is an abstracting process, it may be helpful when the gloss, i.e., interpretation of candidate abstractive words, are provided.

We design an attentive model with word gloss (AMWG) as Figure 1 shows. Specifically, all the *encoders* are 1-layer bi-directional recurrent neural networks (RNNs) with Gated Recurrent Units (GRU) (Cho et al.). For an input news passage $p = [p_1, p_2, ..., p_{l_p}]$ with l_p words, we can derive its hidden states $\mathbf{H}_p = [\mathbf{h}_1^p, \mathbf{h}_2^p, ..., \mathbf{h}_{l_p}^p]$ by sending its word embedding sequence $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_{l_p}]$ to the *Passage Encoder*. Similarly, we can derive hidden states $\mathbf{H}_s = [\mathbf{h}_1^s, \mathbf{h}_2^s, ..., \mathbf{h}_{l_s}^s]$ for summary s by inputting its word embedding sequence $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_{l_s}]$ into the *Summary Encoder* and hidden states $\mathbf{H}_{g_t} = [\mathbf{h}_1^{g_t}, \mathbf{h}_2^{g_t}, ..., \mathbf{h}_{l_{g_t}}^{g_t}]$ for gloss g_t of the candidate word a_t by sending its word embedding sequence $\mathbf{G}_t = [\mathbf{g}_1^t, \mathbf{g}_2^t, ..., \mathbf{g}_{l_{g_t}}^t]$ to the *WordGloss Encoder*.

Similar to Section B, the corresponding representation of "@*placeholder*", i.e., $\mathbf{h}_q^s$, is used as the final vector representation for summary s. And an bilinear attention $f_{att}^p(\bullet)$ is applied to $\mathbf{h}_q^s$ and $\mathbf{H}_p$ as follows:

$$e_i = \mathbf{h}_q^{sT} \mathbf{W}_{att}^p \mathbf{h}_i^p, \forall i \in [1, ..., l_p] \quad (5)$$

$$\alpha_i = \frac{\exp e_i}{\sum_{j=1}^{l_p} \exp e_j}, \forall i \in [1, ..., l_p] \quad (6)$$

Then $\mathbf{p}$ is derived as the vector representation for passage p by the weighed sum of $\mathbf{H}_p$, which is further concatenated with the $\mathbf{h}_q^s$ to form the final summarization vector $\mathbf{v}$:

$$\mathbf{p} = \sum_{i=1}^{l_p} \alpha_i \mathbf{h}_i^p, \quad (7)$$

$$\mathbf{v} = concat(\mathbf{p}, \mathbf{h}_q^s), \quad (8)$$

48

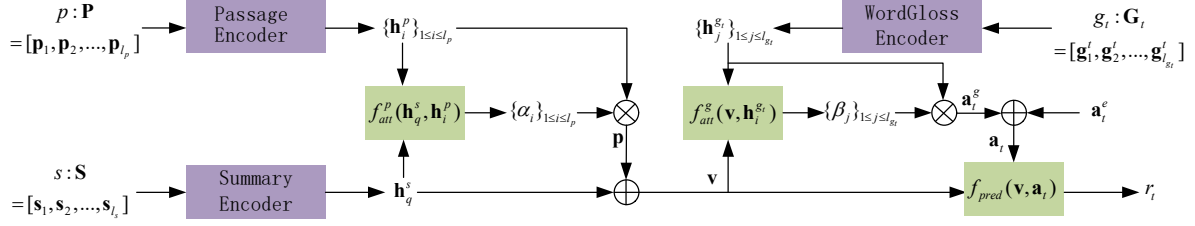Figure 1: The model architecture of the attentive model with word gloss (AMWG) implemented in this paper. $\bigoplus$ denotes the concatenation of input vectors. All the encoders are 1-layer bi-directional GRU-RNNs, $\bigotimes$ denotes the weighted sum of vectors.

Another attention $f_{att}^g(\bullet)$ is applied to $\mathbf{v}$ and $\mathbf{H}_{g_t}$,

$$e_j = \tanh(\mathbf{W}_{att}^g \mathbf{v} + \mathbf{b})^T \mathbf{h}_j^{g_t}, \forall j \in [1, ..., l_{g_t}] \tag{9}$$

$$\beta_j = \frac{\exp e_j}{\sum_{i=1}^{l_{g_t}} \exp e_i}, \forall j \in [1, ..., l_{g_t}], \tag{10}$$

The following weighted sum of $\mathbf{H}_{g_t}$, i.e., $\mathbf{a}_t^g$, is derive as the final vector representation for the gloss of candidate word a_t:

$$\mathbf{a}_t^g = \sum_{j=1}^{l_{g_t}} \beta_j \mathbf{h}_j^{g_t} \tag{11}$$

We also set a token embedding $\mathbf{a}_t^e$ for each candidate word a_t ($t \in [1, ..., n_c]$, n_c is the size of candidate set), which is further concatenated with $\mathbf{a}_t^g$ to build the final representation $\mathbf{a}_t$ for candidate word a_t. For the final prediction, we input the summarization vector $\mathbf{v}$ and candidate representation vector $\mathbf{a}_t$ to $f_{pred}(\bullet)$ and apply the softmax to derive the probability distribution over all n_c candidate abstractive words,

$$\mathbf{a}_t = concat(\mathbf{a}_t^g, \mathbf{a}_t^e), \tag{12}$$

$$r_t = \mathbf{v}^T \mathbf{W}_{pred} \mathbf{a}_t, \forall t \in [1, ..., n_c], \tag{13}$$

$$o_t = softmax_t(r_t), \forall t \in [1, ..., n_c] \tag{14}$$

in which o_t gives the probability of predicting the candidate word a_t as the final answer. The word gloss, which defines a word sense meaning, has been mainly used in word sense disambiguation (WSD) task and its variants (Lesk, 1986; Moro et al., 2014). Since the goal of ARC is to predict a word that can summarize corresponding information from the source passage, which is an abstracting process, it may be helpful when the gloss, i.e., interpretation of candidate abstractive words, are provided.

We design an attentive model with word gloss (AMWG) as Figure 1 shows. Specifically, all the *encoders* are 1-layer bi-directional recurrent neural networks (RNNs) with Gated Recurrent Units (GRU) (Cho et al.). For an input news passage $p = [p_1, p_2, ..., p_{l_p}]$ with l_p words, we can derive its hidden states $\mathbf{H}_p = [\mathbf{h}_1^p, \mathbf{h}_2^p, ..., \mathbf{h}_{l_p}^p]$ by sending its word embedding sequence $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_{l_p}]$ to the *Passage Encoder*. Similarly, we can derive hidden states $\mathbf{H}_s = [\mathbf{h}_1^s, \mathbf{h}_2^s, ..., \mathbf{h}_{l_s}^s]$ for summary s by inputting its word embedding sequence $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_{l_s}]$ into the *Summary Encoder* and hidden states $\mathbf{H}_{g_t} = [\mathbf{h}_1^{g_t}, \mathbf{h}_2^{g_t}, ..., \mathbf{h}_{l_{g_t}}^{g_t}]$ for gloss g_t of the candidate word a_t by sending its word embedding sequence $\mathbf{G}_t = [\mathbf{g}_1^t, \mathbf{g}_2^t, ..., \mathbf{g}_{l_{g_t}}^t]$ to the *WordGloss Encoder*.

Similar to Section B, the corresponding representation of "@*placeholder*", i.e., $\mathbf{h}_q^s$, is used as the final vector representation for summary s. And an bilinear attention $f_{att}^p(\bullet)$ is applied to $\mathbf{h}_q^s$ and $\mathbf{H}_p$ as follows:

$$e_i = \mathbf{h}_q^{sT} \mathbf{W}_{att}^p \mathbf{h}_i^p, \forall i \in [1, ..., l_p] \tag{15}$$

$$\alpha_i = \frac{\exp e_i}{\sum_{j=1}^{l_p} \exp e_j}, \forall i \in [1, ..., l_p] \tag{16}$$

Then $\mathbf{p}$ is derived as the vector representation for passage p by the weighed sum of $\mathbf{H}_p$, which is further concatenated with the $\mathbf{h}_q^s$ to form the final summarization vector $\mathbf{v}$:

$$\mathbf{p} = \sum_{i=1}^{l_p} \alpha_i \mathbf{h}_i^p, \tag{17}$$

$$\mathbf{v} = concat(\mathbf{p}, \mathbf{h}_q^s), \tag{18}$$

Another attention $f_{att}^g(\bullet)$ is applied to $\mathbf{v}$ and $\mathbf{H}_{g_t}$,

$$e_j = \tanh(\mathbf{W}_{att}^g \mathbf{v} + \mathbf{b})^T \mathbf{h}_j^{g_t}, \forall j \in [1, ..., l_{g_t}]$$

(19)

$$\beta_j = \frac{\exp e_j}{\sum_{i=1}^{l_{g_t}} \exp e_i}, \forall j \in [1, ..., l_{g_t}],$$

(20)

The following weighted sum of $\mathbf{H}_{g_t}$, i.e, $\mathbf{a}_t^g$, is derive as the final vector representation for the gloss of candidate word a_t:

$$\mathbf{a}_t^g = \sum_{j=1}^{l_{g_t}} \beta_j \mathbf{h}_j^{g_t}$$

(21)

We also set a token embedding $\mathbf{a}_t^e$ for each candidate word a_t ($t \in [1, ..., n_c]$, n_c is the size of candidate set), which is further concatenated with $\mathbf{a}_t^g$ to build the final representation $\mathbf{a}_t$ for candidate word a_t. For the final prediction, we input the summarization vector $\mathbf{v}$ and candidate representation vector $\mathbf{a}_t$ to $f_{pred}(\bullet)$ and apply the softmax to derive the probability distribution over all n_c candidate abstractive words,

$$\mathbf{a}_t = concat(\mathbf{a}_t^g, \mathbf{a}_t^e),$$

(22)

$$r_t = \mathbf{v}^T \mathbf{W}_{pred} \mathbf{a}_t, \forall t \in [1, ..., n_c],$$

(23)

$$o_t = softmax_t(r_t), \forall t \in [1, ..., n_c]$$

(24)

in which o_t gives the probability of predicting the candidate word a_t as the final answer.

D Training Details

We train all models using the non-negative log-likelihood as the objective function. The gloss of candidate words are derived from *WordNet* using the NLTK tools (Bird and Loper, 2004). Specifically, we first lemmatize the candidate word and use the lemmatized word as the query word for the searching in *WordNet*. To cope with the semantic ambiguity of words, we just concatenate the gloss of the first sense in each retrieved POS for the query word with corresponding POS tag as the deliminator.

Models in our experiments are trained with the following hyperparameter settings: All word embeddings and token embeddings $\mathbf{a}_t^e$ have 300 dimensions and are initialized with Glove (Pennington et al., 2014). The passage p and summary s share one set of word embeddings, which are fixed during training. The glosses $\{g_t\}$ for candidate words $\{a_t\}$ keep its own word embeddings.

The hidden state vectors of all bi-directional GRU-RNNs in all models have 150 dimensions. The number of attention hops in GA Reader is set to 3. The batch size is set to 32. The method of Adam (Kingma and Ba, 2015) is adopted for optimization with initial learning rate $1e - 03$. A dropout with rate 0.3 is applied to the input layers for all GRU-RNN encoders and the final summarization vector $\mathbf{v}$.

E Annotation Selection

To ensure most of our annotation is valid, we select annotations satisfying the following criteria: a) the average accuracy is higher than 40%; b) both text spans should not be empty; c) if the difficulty level is rated as easy, then this data sample should be answered correctly.