UIUC_BioNLP at SemEval-2021 Task 11: A Cascade of Neural Models for Structuring Scholarly NLP Contributions

Haoyang Liu, Janina Sarol, and **Halil Kilicoglu**
School of Information Sciences, University of Illinois at Urbana-Champaign
{hl57, mjsarol, halil}@illinois.edu

Abstract

We propose a cascade of neural models that performs sentence classification, phrase recognition, and triple extraction to automatically structure the scholarly contributions of NLP publications in English. To identify the most important contribution sentences in a paper, we used a BERT-based classifier with positional features (Subtask 1). A BERT-CRF model was used to recognize and characterize relevant phrases in contribution sentences (Subtask 2). We categorized the triples into several types based on whether and how their elements were expressed in text, and addressed each type using separate BERT-based classifiers as well as rules (Subtask 3). Our system was officially ranked second in Phase 1 evaluation and first in both parts of Phase 2 evaluation. After fixing a submission error in Phase 1, our approach yielded the best results overall. In this paper, in addition to a system description, we also provide further analysis of our results, highlighting its strengths and limitations. We make our code publicly available at https://github.com/Liu-Hy/nlp-contrib-graph.

1 Introduction

With the deluge of scientific publications in recent years, keeping pace with the literature and managing information overload have become increasingly challenging for researchers. There is a growing need for tools that can automatically extract and structure semantic information from scientific publications to facilitate advanced approaches to information access and knowledge curation (Shen et al., 2018).

The field of natural language processing (NLP) has witnessed an enormous growth in recent years with advances in deep learning, and there are increasing efforts in developing methods to extract scholarly knowledge from NLP publications (QasemiZadeh and Schumann, 2016; D'Souza and Auer, 2020b). One such effort is NLPCONTRIBUTIONS, an annotation scheme for describing the scholarly contributions in NLP publications and a corpus annotated using this annotation scheme (D'Souza and Auer, 2020b). This corpus has been proposed for training and testing of machine reading models, whose output can be integrated with the Open Research Knowledge Graph framework (ORKG) (Jaradeh et al., 2019). ORKG formalizes the research contributions of a scholarly publication as a knowledge graph, which can further be linked to other publications via the graph. The goal of the NLPContributionGraph (NCG) shared task (D'Souza et al., 2021) is to facilitate the development of machine reading models that can extract ORKG-compatible scholarly contribution information from NLP publications. The shared task consists of three subtasks (see D'Souza et al. (2021) for a more detailed description):

- Subtask 1: Identification of contribution sentences from NLP publications
- Subtask 2: Recognition of scientific terms and relations in contribution sentences
- Subtask 3: Extraction and classification of triples that pair scientific terms with relations

In this paper, we describe our contribution to NCG shared task. We built a cascade of neural classification and sequence labeling models based on BERT (Devlin et al., 2019). For subtask 3, we characterized triples based on whether and how their elements are expressed in text, and employed different models for each category. We also explored rule-based heuristics to improve model performance. Our models had the best overall performance in the shared task (57.27%, 46.41%, and 22.28% F_1 score in subtasks 1, 2, and 3, respectively). The results are encouraging for extracting

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 377–386
Bangkok, Thailand (online), August 5–6, 2021. ©2021 Association for Computational Linguistics

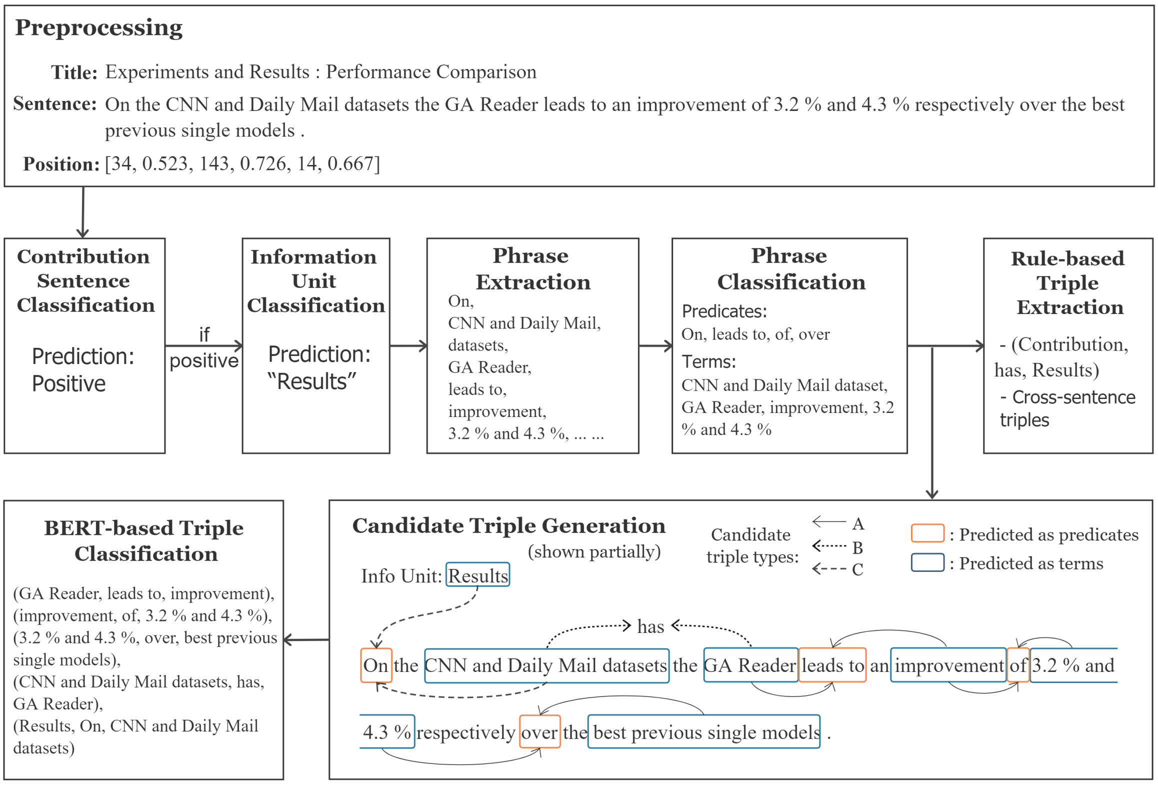
Preprocessing
Title: Experiments and Results : Performance Comparison
Sentence: On the CNN and Daily Mail datasets the GA Reader leads to an improvement of 3.2 % and 4.3 % respectively over the best previous single models .
Position: [34, 0.523, 143, 0.726, 14, 0.667]
Contribution Sentence Classification
Prediction: Positive
if positive
Information Unit Classification
Prediction: "Results"
Phrase Extraction
On, CNN and Daily Mail, datasets, GA Reader, leads to, improvement, 3.2 % and 4.3 %, ...
Phrase Classification
Predicates: On, leads to, of, over
Terms: CNN and Daily Mail dataset, GA Reader, improvement, 3.2 % and 4.3 %
Rule-based Triple Extraction
- (Contribution, has, Results)
- Cross-sentence triples
BERT-based Triple Classification
(GA Reader, leads to, improvement), (improvement, of, 3.2 % and 4.3 %), (3.2 % and 4.3 %, over, best previous single models), (CNN and Daily Mail datasets, has, GA Reader), (Results, On, CNN and Daily Mail datasets)
Candidate Triple Generation (shown partially)
Candidate triple types:
A
B
C
: Predicted as predicates
: Predicted as terms
Info Unit: Results
has
On the CNN and Daily Mail datasets the GA Reader leads to an improvement of 3.2 % and
4.3 % respectively over the best previous single models .

Figure 1: End-to-end system diagram.

scholarly contributions from scientific publications, although there is much room for improvement.

2 System Overview

In this section, we first describe our data preprocessing steps. Next, we discuss our models for each subtask, and the experimental setup for our end-to-end system (Phase 1). We provide an overview of the system in Figure 1 and provide examples for illustration, when necessary.

2.1 Data preprocessing

The participants of the shared task were provided three kinds of input: a) plain text files of the publications converted from PDF using Grobid[1], b) sentences and tokens identified using Stanza (Qi et al., 2020), and c) triples and source texts organized by their information units (e.g., APPROACH) in JSON format.

2.1.1 Identifying headers and positional information

One major preprocessing step was to identify section headers in the publications and associate them with individual sentences. For sentence classification (subtask 1), we incorporated the topmost and innermost section headers associated with a sentence into its representation. The topmost header indicates the general role that a sentence plays in the article, while the innermost header provides more specific context for the sentence. For example, one topmost/innermost header pair is EXPERIMENT/DATA SET AND EXPERIMENT SETTINGS.

In the absence of explicit section information in the input, we used rule-based heuristics to extract these headers. With the first heuristic (Heuristic1), we simply identified the sentences following blank lines in plain text files as section headers. In Heuristic2, we first identified candidate headers as sentences that contain fewer than 10 words, have the first letter capitalized, do not end with several stopwords (*by*, *as*, *in*, *that*, or *and*), do not contain question marks in the middle or end with some punctuation (comma, colon or full stop). Next, we determined the case format used for headers in the publication by counting the occurrences of each case format type (e.g., all uppercase: EXPERIMENTAL SETUP). Among the headers that conform to the determined case format, we dis-

[1] https://grobid.readthedocs.io/

tinguished topmost headers as those that contain several lexical cues (e.g., *background*, *method*) and are shorter than 5 words. Finally, we associated each sentence with the nearest preceding topmost and innermost header.

To incorporate headers into the sentence representation, we join the topmost and innermost header together with a colon between them and refer to it as the "title" of the sentence. In the case where a sentence is directly governed by a top-level header or it is a header itself, the title consists of the topmost header only.

We characterize the position of each sentence in the document with a combination of numeric features:

- The offset of the sentence in the entire paper.
- The offset of the sentence with respect to its topmost header.
- The offset of the sentence with respect to the header extracted using Heuristic1.

Each of these offset features are divided by the number of sentences in the corresponding discourse (entire paper or the section) to extract a proportional sentence position feature. Thus, for every sentence, a total of six positional features (three offsets, three proportional sentence positions) are computed.

2.1.2 JSON Parsing

We created two additional models to assist with triple extraction: a) a multi-class sentence classifier that labels each sentence with a single information unit and b) a binary phrase classifier that labels phrases as scientific terms vs. predicates (described below). To train these models, we extracted additional information from JSON files. First, we matched the contribution sentences with the source text in the JSON files to get the information unit labels of the sentences. Second, we aligned the phrases with the triples in the same information unit, and determined whether each phrase is a predicate or term based on its location in the triple.

2.2 Subtask 1: Contribution Sentence Classification

We built a binary classifier to determine whether each sentence describes a contribution of the publication. Our analysis revealed that this decision was not simply based on the semantics of the sentence, but also its position in the document. On one hand, the section header associated with the sentence provides important clues about the role of the sentence in the larger context. For example, the header "Related Work" indicates that sentences in this section are likely to discuss the contributions of prior research. On the other hand, some parts of the documents are naturally more salient than others (e.g. title, abstract, the first few lines of each section), where authors tend to summarize the most important information. To operationalize these insights, we designed a model that captures the information about the sentence, its topmost and innermost headers as well as its position in the document, as discussed above.

We used a BERT model to encode the sentence and its title (i.e., concatenated headers) separately and concatenated their textual representation together with the positional features to obtain a sentence representation. We then fed this representation into two dense layers, and used a final softmax layer for classification (Figure 2).

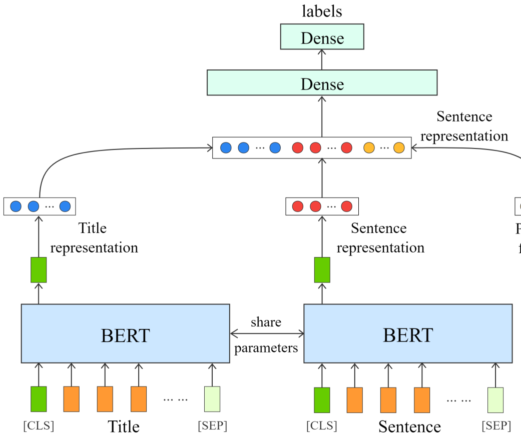

Figure 2: Sentence classification model architecture

2.3 Subtask 2: Phrase Recognition

Subtask 2 is similar to a named entity recognition (NER) task, although the participating systems were only required to extract relevant text spans and not to categorize them. One major difficulty with this subtask is that phrases do not align neatly with sentence constituents (e.g., noun phrases) and they vary greatly in length and in what counts as their boundaries (e.g. *best results* and *our best results* are both valid phrases).

For this subtask, we used a BERT-CRF model for phrase extraction and type classification (Souza et al., 2019). The raw text of the sentence is taken as the model input. A BIO scheme that incorporates phrase types (scientific term vs. predicate) is used (e.g., B-Predicate, I-Term, O). The probabilities produced by the BERT model are fed into a Conditional Random Field (CRF) layer (Lafferty et al., 2001) for end-to-end training. We note that while phrase type classification is not necessary for subtask 2, we perform it since it is useful for our subtask 3 model, described next.

2.4 Subtask 3: Triple Extraction

Subtask 3 involves organizing phrases into triples. In information extraction, semantic triples are typically composed of subject, predicate, and object terms each corresponding to specific textual spans. This is not always the case in this subtask. While in most cases all three terms are extracted from a single sentence, a non-negligible number of triples consist of at least one phrase that does not come from the sentence (e.g. (TASKS, has, *Coreference resolution*), where the subject is an information unit and the predicate is not a sentence element).

To better understand triple characteristics, we categorized them into several types based on their composition, and created separate relation classification models for each type. The triple categorization is presented in Table 1. For each type, we list their functions in information organization, their proportion to all triples, along with some examples. We note that input to the training process for triple extraction varies by the type of the triple (described for each type in Section 2.4.2).

2.4.1 Information Unit Classification

To aid triple extraction, we modified the binary classification model that we trained for subtask 1 to further classsify contribution sentences by their information units (multi-class classification). The process of labeling contribution sentences with information units was briefly described in Subsection 2.1.2.

In analyzing the information units, we identified two special pairs (MODEL vs. APPROACH and EXPERIMENTAL-SETUP vs. HYPERPARAMETERS). In the dataset, no document contains both units of a pair. The decision of which unit to choose is made at the document level. Therefore, we merged the labels of similar units before feeding the examples into the multi-class classification model.

After classification, we used lexical rules to split these units. Our rules were based on the following observations. First, the MODEL vs. APPROACH distinction seems related to how the authors mention their work in the abstract and section headers of the paper. Second, EXPERIMENTAL-SETUP is often used instead of HYPERPARAMETERS when the hardware or the framework used in the study is specified (e.g. *V100 GPU, Pytorch*).

We did not recognize CODE information units using this model, since we found that such sentences can be identified with a very high accuracy using a simple rule based on presence of a URL in the sentence.

2.4.2 Neural models for triple extraction

We extract triples of type A, B, C and D (Table 1) by formulating them as neural relation classification tasks. All the classifiers are vanilla BERT classifiers (one linear layer followed by softmax). For each type, we observed the patterns in the training data, and addressed the most common ones. Ignoring the less frequent patterns inevitably led to a lower recall ceiling in our models.

Type A This type, in which all triple elements are mentions in the sentence, represents the majority of the triples. The corresponding model classifies the triples as a whole ("triple classification"). To the best of our knowledge, little research has been done on relation classification among three phrases; however, the Transformer model at the core of BERT is versatile enough to succeed in a wide range of tasks. As our training examples, we take every combination of a predicate and two terms in a sentence as a candidate triple, and train a model that predicts whether the three phrases constitute a triple or not. We encode the relation between three phrases by marking their boundaries in the sentence, as shown in Example 1. We use angle brackets to enclose predicates, and square brackets to enclose terms.

(1) *In this paper , we explore an alternate [[semisupervised approach]] which does << not require >> [[additional labeled data]] .*

Type B To identify triples of type B (two terms from the sentence and the relation type one of has, name, or None), we classify the relation between each pair of terms in a sentence that are not related by a type A triple. We found that 96% of these triples preserve the order of the two terms in the sentence, so we also preserve the order for extraction.

	Composition	Examples	Role	Pct.
Type A	Three phrases in a sentence	(*Deep - ED*, `obtain`, *BLEU score*)	Organize the semantics of a sentence.	57%
Type B	Two terms in a sentence with an added predicate `has` or `name`	(*ByteNet Decoder*, `has`, *30 residual blocks*)	Organize the semantics of a sentence.	7%
Type C	Information unit (subject), and two phrases in a sentence (predicate and object)	(HYPERPARAMETERS, `use`, *cross - entropy loss*)	Link a sentence to its information unit.	9%
Type D	Information unit (subject), `has` (predicate), and a term in the sentence (object)	(HYPERPARAMETERS, `has`, *starting learning rate*)	Link a sentence to its information unit.	9%
Type E	CONTRIBUTION (subject), `has` (predicate), information unit (object) OR CONTRIBUTION (subject), fixed (predicate), and a phrase (object) for the information units RESEARCH PROBLEM and CODE	(CONTRIBUTION, `has`, RESULTS), (CONTRIBUTION, `has research problem`, *neural machine translation*)	Link the "Contribution" node of each paper to an information unit.	9%
Type F	Cross-sentence triples	(*Positional Encoding*, `inject`, *some information*)	Structure the information across sentences	3%

Table 1: Triple types, their roles, and frequency. Types A-D are addressed using neural models and Types E-F with rules. 6% of triples do not fit in these categories and are not shown.

Type C Type C triples involve an information unit name as the subject along with a predicate and object from the sentence. We found that 89% of these triples take the first predicate and the first term in a sentence as their predicate and object respectively. Furthermore, in 98% of these sentences, the first predicate precedes the first term. Therefore, we classify each sentence whose first predicate precedes the first term, to predict whether a triple of this type can be extracted from the sentence. To train this classifier, we prepend the information unit name to the sentence text with a colon in between, as in Example 2 (*Model* is the information unit).

(2) *[[Model]] : In this work , we << introduce >> [[a new type of linear connections]] for multi - layer recurrent networks .*

Type D Type D triples are similar to Type C, but instead of a predicate phrase from the sentence, they involve the non-sentence predicate `has`. We found that 95% of these triples in the training set take the first term in the sentence as their object, and the first predicate in the sentence, if one exists, almost always follows the first term. Therefore, we classify each sentence that conforms to this pattern, to predict whether the information unit name and the first term constitute a `has` relation. We prepend the info unit name to the sentence in the same way as in Type C.

2.4.3 Rule-based triple extraction

Triples of type E and F are extracted using heuristic rules. For type E, the subject is always CONTRIBUTION. The predicate can be `has`, in which case the object is the name of an information unit. If the related information unit is CODE or RESEARCH PROBLEM, the predicate is a fixed predicate (`Code` or `has research problem`, respectively) and the object is a phrase from the sentence. These rules use phrase and information units identified in earlier steps (Sections 2.3 and 2.4.1, respectively).

We developed the following rules to extract cross-sentence triples (type F):

1. If the first sentence has a single entity, and the second sentence has at least 2 entities, we assign the entity in sentence 1 as the subject and the first and second entities in sentence 2 as the predicate and object, respectively. We add this triple to the list only if both subject and predicate are noun phrases, which prevents many false positives. We also add the corresponding triple in the form of INFO-UNIT-`has`-*subject* (e.g. MODEL-`has`-*Encoder*). In many sentences that follow this rule, the first sentence is a section header.

	Avg F_1	Information Units			Sentences			Phrases			Triples		
		F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R
Our system	49.72	72.93	66.67	80.49	57.27	53.61	61.46	**46.41**	42.69	50.83	**22.28**	22.30	22.26
IAA	52.82	**79.73**	78.83	80.65	**67.44**	67.25	67.63	41.84	45.36	38.83	**22.28**	23.76	20.97

Table 2: End-to-end performance (Evaluation Phase 1). IAA: intra-annotator agreement.

2. If the two sentences each contain a single term and sentence 1 term is a substring of sentence 2 term or if sentence 1 term is an acronym of sentence 2 term, we create the following triple: *term 1*-`name`-*term 2*. We extract a term's acronym by combining the initials of each token in the entity. An example of a term pair that follows this rule is (*GLUE*, *General Language Understanding Evaluation*).

These rules are applied to consecutive sentences only. In the training set, we found 812 triples that follow these rules, 649 (80%) of which could be identified correctly using these rules.

2.5 Experimental Setup

We implemented our models using Simple Transformers[2]. We used SciBERT (Beltagy et al., 2019) as the pre-trained language model. To train our models, we used a batch size of 16, and empirically found the best learning rate for each model between 10^{-5} and 10^{-4}. One exception was that in our sentence classification model (subtask 1), we used a fixed learning rate of 10^{-5} to fine-tune the BERT, and a larger learning rate between 5×10^{-5} and 10^{-3} for the dense layers. We used the AdamW optimizer (Loshchilov and Hutter, 2017) and the polynomial decay scheduler with the power of 0.5. We ran the experiments on a Google Cloud VM instance, using a Tesla V100 GPU.

3 Results

All the subtasks were evaluated on F_1 scores, and among them, triple extraction is evaluated by the micro-average of F_1 scores on each information unit. In the end-to-end evaluation (Phase 1), the participants were provided with the raw input to perform all three subtasks sequentially. We were officially ranked second in Phase 1, due to a submission error that resulted in phrase extraction F_1 of zero. Our correct submission achieved an average F_1 of 49.7%, the best score among all participating teams. Table 2 shows our performance in Phase 1, and the intra-annotator agreement (IAA) on each subtask (D'Souza and Auer, 2020a).

We observe that, although the performance of our system on sentence classification is lower than human performance (57.27% vs. 67.44% F_1), using its own sentence predictions, our system outperforms human annotators on phrase recognition (46.41% vs. 41.84% F_1), and reaches comparable performance to human annotators on triple extraction. We also note that our system generally performs better in terms of recall than precision.

We were officially ranked first in both parts of Evaluation Phase 2. In Part 1, the participants were provided with the sentences labels to conduct phrase recognition and triple extraction sequentially; in Part 2, both the sentence labels and the phrase labels were provided to extract triples. We essentially followed our method in Phase 1 on phrase recognition and triple extraction, but made several attempts to improve the performance, which we discuss in Section 4. Our results in both parts of the Phase 2 evaluation are shown in Table 3. Compared to Phase 1 evaluation, we observe a significant improvement in phrase recognition (46.41% vs. 78.57% F_1) in Part 1 and in triple extraction (22.28% to 43.44% and 61.29% F_1) when ground truth contribution sentences and phrases are provided.

4 Performance Analysis

In this section, we analyze the performance of several components of our system and compare different schemes for entity representation and triple extraction. We also discuss some possible methods for improvement based on our shared task results and follow-up experiments.

4.1 Contribution Sentence Classification

We conducted ablation experiments to evaluate the effect of features for contribution sentence classification. Table 5 shows the model performance on the 10% validation set when using all features, using either the title or the position features together with the sentence, and using the sentence only.

[2] https://github.com/ThilinaRajapakse/simpletransformers

	Information Units			Phrases			Triples		
	F_1	P	R	F_1	P	R	F_1	P	R
Part 1	82.49	76.84	89.02	78.57	76.86	80.35	43.44	45.06	41.94
Part 2	82.49	76.84	89.02	-	-	-	61.29	65.19	57.82

Table 3: Performance in phrase and triple extraction (Evaluation Phase 2). Note that we focused only on triple extraction in Part 2, therefore the information unit extraction performance remains the same.

Unit name	Research problem	Approach	Model	Code	Dataset	Experimental Setup	Hyperparameters	Baselines	Results	Tasks	Experiments	Ablation analysis
F_1	94.64	24.14	86.22	87.50	80.00	58.29	72.61	91.45	94.65	90.48	83.16	90.68

Table 4: Information unit classification performance.

Settings	F_1	P	R
Sentence + title + position	65.11	63.96	66.30
Sentence + title	63.87	61.00	67.03
Sentence + position	52.28	46.38	59.89
Sentence only	51.39	49.00	54.03

Table 5: Results of ablation experiments on contribution sentence classification task.

We observe the title information significantly improves the performance, and the position features are also helpful, to a lesser extent. Combining the title and the position features gives the best performance on contribution sentence classification.

4.2 Information Unit Classification

In Evaluation Phase 2, the ground truth labels for contribution sentences increased the performance of our base model on information unit classification from 72.93% to 76.84% F_1. To further improve our method, we ensembled 45 multi-class sentence classifiers by averaging their output (using *bagging*), which increased the F_1 score to 78.65%. Next, we improved our rules for distinguishing the special pairs (MODEL vs. APPROACH and EXPERIMENTAL-SETUP vs. HYPERPARAMETERS) by adjusting the lexical cues with more careful observation of the data, which results in our final performance (82.49% F_1 in Table 3).

For further analysis, we evaluated the classification performance on each information unit, as shown in Table 4. The related confusion matrix is shown in Fig. 3. We observe that severe confusion mainly occurs between MODEL vs. APPROACH and EXPERIMENTAL-SETUP vs. HYPERPARAMETERS, pairs that we grouped together in neural classification. This shows that while our sentence classification model has good accuracy, there is still much room for improvement in the rule-based differentiation of similar units.

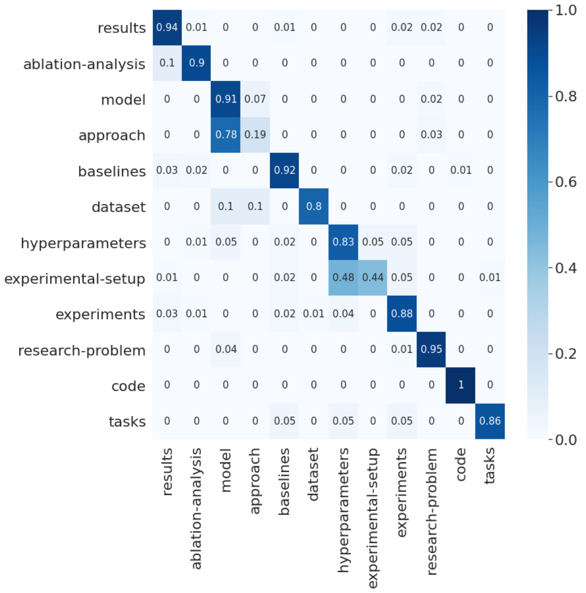

Figure 3: Confusion Matrix

The differentiation between MODEL and APPROACH is particularly challenging. While some papers aim at discussing an abstract idea and some focus on system implementation, most papers fall in the gray area between them. We also attempted neural classification on the abstracts to deal with this issue, but the result were not satisfactory.

4.3 Phrase extraction and classification

Specific BIO VS. simple BIO Alternative to our method of using specific BIO tags to indicate phrase types (Subsection 2.3), we also used another scheme ("simple BIO"), in which we only used (B, I, O) tags to mark phrase boundaries.

With this scheme, we first trained a BERT-CRF model to extract the phrases, and then trained a binary BERT classifier to predict phrase types. The sentence along with the phrase marked by special tokens is fed into the BERT model for binary clas-

Settings	Phrase extraction			Phrase classification		
	F_1	P	R	F_1	P	R
Specific BIO	76.09	75.57	76.62	98.13	98.00	98.25
Simple BIO	77.13	76.33	77.95	98.27	98.70	97.85
Simple BIO + ensemble	78.57	76.86	80.35			

Table 6: Phrase extraction and classification performance. We take *predicate* as the positive label to calculate the F_1 score for phrase classification.

sification. The performance comparison of these schemes is shown in Table 6. While both schemes are effective, simple BIO outperforms specific BIO in phrase extraction by a small margin, so we used this scheme in Evaluation Phase 2.

The difference may be due to the noise in phrase types. Specifically, there is a good number of gerund phrases, on which the predicate-term differentiation is challenging. Moreover, in some cases, a verb phrase is used as a term to form triples. Combined with the relatively low intra-annotator agreement, these observations suggest that uncertainty and noise in the data affects the performance of the models. Note that the specific BIO scheme eliminates the need for a separate phrase classification model, making it preferable when the training and inference speed is a concern.

Error analysis and improvement We investigated the wrong predictions of our phrase extraction model, and found that most errors are due to boundary detection issues. For example, in one sentence, the model predicts *all layers of representation* as a phrase, while *all layers*, *of*, *representations* are annotated as three separate phrases. The opposite situation also occurs, when the model predicts a single unit as separate phrases. Another type of boundary error occurs when the model cannot predict correctly whether to include a non-core phrase element, like an adverb, in the phrase or not (e.g., it predicts *see that* whereas the annotated phrase is *also see that*). We believe that a relaxed boundary match evaluation can be considered for this task.

We attribute these errors to the uncertainty in semantic granularity, and attempted to alleviate the problem by ensembling. We get 12 bootstrap samples from the training data, and on each sample, we train the model and save its snapshot after each epoch from the 3th epoch to the 10th epoch, to get a total of 96 submodels. To aggregate their predictions, we extract a phrase in a sentence only if it is predicted by more than N submodels, where N is a hyperparameter around 48. We present the result in Table 6 for comparison. We observe that ensembling noticeably improved phrase extraction (from 77.13% to 78.57% F_1).

4.4 Triple extraction

Triple vs. pairwise classification In addition to triple classification method (Subsection 2.4.2) to extract type A triples, we also used pairwise classification for this task. In this scheme, we considered every (subject, predicate, object) triple as a composition of two (predicate, term) pairs, or "candidate pairs", and used a neural model to predict whether the two phrases in the pair are associated. After prediction, we reconstructed triples from the predicted pairs using rules. If a predicate is predicted to be associated with two terms, we combine them into a triple while preserving the order of the two terms in the sentence (subject first). If one predicate is associated with more than two terms, we only extract the triples in which the predicate is located between the two terms in the sentence. With only a few exceptions, we confirmed the effectiveness of these reconstruction rules; in other words, the performance of the pairwise scheme depends mainly on the classification accuracy on candidate pairs.

We compared the performance of the two schemes for type A triple extraction on the 10% validation set. We also attempted to address the imbalance of class labels resulting from both schemes by downsampling and class weight adjustment.

Settings		F_1	P	R
No adjustment	Pair	91.33	91.23	91.43
	Triple	75.95	70.58	82.20
Downsampling	Pair	91.31	89.09	93.63
	Triple	80.04	79.43	80.66
Class weight	Pair	91.30	88.93	93.79
	Triple	**80.37**	81.35	79.42

Table 7: Performance of the pairwise classification scheme.

In the pairwise classification scheme (Table 7),

	F1	P	R
No adjustment	**87.54**	85.93	89.22
Downsampling	75.59	62.32	96.04
Class weight	83.35	74.94	93.89

Table 8: Performance of the triple classification scheme.

there is a 11% drop in the F1 score from the candidate pair classification to triple prediction, which is not unexpected as the model needs to correctly classify *both* of the candidate pairs to correctly predict a triple.

Table 8 shows the performance of the triple classification scheme, which achieves better performance compared to the pairwise classification scheme (87.54% vs. 80.37% F_1). We also observed that the best performance was obtained without dealing with the imbalanced data. It seems that despite constituting a small portion of the dataset (9.7%) , the number of the positive samples is large enough for the model to learn useful patterns.

Type-specific performance We also evaluated our deep learning methods for the extraction of the four types of triples, as shown in Table 9.

Type	F1	P	R
A	87.54	85.93	89.22
B	55.56	88.24	40.54
C	83.33	77.96	89.51
D	75.86	78.11	73.74

Table 9: Performance of triple extraction on each type.

Whereas our models for Type A, C, and D perform generally well, our model for Type B is far less accurate. Type B is a little special among the four types in that it requires the prediction of relation types. The type `has` is more difficult to predict than `name`, because the sentence often lacks semantic clues about the belonging or inclusion relationship between the two terms. A plausible idea is to incorporate *has* into the input, but it is difficult to do so without breaking the grammatical integrity of the sentence. We leave this improvement for future work.

Coordination in triple extraction A common problem we observed in our triple extraction models is the failure to account for coordination between terms. Example 3 shows a sentence with the terms in bold, and the two type C triples associating them. Our model only extracts the first triple, and misses the second.

(3) *The MoE consists of a* ***number of experts****, each a simple feed - forward neural network, and a* ***trainable gating network*** *which selects a sparse combination of the experts to process each input.*
(APPROACH, `consists of`, *number of experts*)
(APPROACH, `consists of`, *trainable gating network*)

We attempted to address this issue in post-processing, and used Stanza dependency parser (Qi et al., 2020) to detect coordination of words in phrases. If one phrase is used in a triple, we generated a parallel triple by replacing the term with the other. While this method improves recall (from 57.57% to 58.41%), it also led to precision errors (from 65.15% to 61.77%), its overall effect being negative (from 61.13% to 60.04% F_1). We plan to refine this approach in future work.

5 Conclusion

We developed a system to generate structured representations of research contributions described in NLP publications in a manner compatible with the ORKG framework, achieving the top performance in the NCG shared task. We combined a cascade of state-of-the-art BERT-based classification and sequence labeling models with rule-based methods. In particular, we proposed a novel approach for triple extraction, where we tackled triples with different characteristics using different relation classification methods. We also explored various alternatives to the components in our end-to-end system to analyze the contribution of individual components.

In future work, we plan to improve the differentiation of similar units (e.g., MODEL vs. APPROACH), improve the extraction of type B triples, and address coordinated triples more thoroughly. We did not attempt to extract approximately 6% of the triples that did not fit in our classification (Table 1). These often involve nested information units, and we also hope to explore them in more depth in future work.

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text.

In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

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

Jennifer D'Souza, Sören Auer, and Ted Pedersen. 2021. SemEval-2021 task 11: NLPContributionGraph - structuring scholarly NLP contributions for a research knowledge graph. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). Association for Computational Linguistics.

Jennifer D'Souza and Sören Auer. 2020a. Graphing Contributions in Natural Language Processing Research: Intra-Annotator Agreement on a Trial Dataset.

Jennifer D'Souza and Sören Auer. 2020b. NLPContributions: An Annotation Scheme for Machine Reading of Scholarly Contributions in Natural Language Processing Literature.

Mohamad Yaser Jaradeh, Allard Oelen, Kheir Eddine Farfar, Manuel Prinz, Jennifer D'Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. 2019. *Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge*, page 243–246. Association for Computing Machinery, New York, NY, USA.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Behrang QasemiZadeh and Anne-Kathrin Schumann. 2016. The ACL RD-TEC 2.0: A language resource for evaluating term extraction and entity recognition methods. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1862–1868, Portorož, Slovenia. European Language Resources Association (ELRA).

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Zhihong Shen, Hao Ma, and Kuansan Wang. 2018. A web-scale system for scientific knowledge exploration. In *Proceedings of ACL 2018, System Demonstrations*, pages 87–92, Melbourne, Australia. Association for Computational Linguistics.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2019. Portuguese named entity recognition using BERT-CRF. *arXiv preprint arXiv:1909.10649*.