What Is Needed for Intra-document Disambiguation of Math Identifiers?

Takuto Asakura **Yusuke Miyao**
Dept. of Computer Science, The University of Tokyo
{takuto,yusuke}@is.s.u-tokyo.ac.jp

Abstract

In automated scientific document analysis, accurately interpreting math formulae is imperative alongside comprehending natural language. Ambiguity in math identifiers within a single document poses significant challenges to understanding math formulae. While disambiguating math identifiers across documents has seen some progress, resolving ambiguity within a document remains inadequately researched due to complexity and insufficient datasets. The level of difficulty and information required to accomplish this task was uncertain. This study aims to determine which information is necessary for the intra-document disambiguation of math identifiers. Our findings indicate that the position data and local formula structure surrounding the identifiers, including modifiers, are particularly critical. For our study, we expanded a dataset for formula grounding and doubled its size to include annotations for 27,655 math identifier occurrences. We have created a multi-layer perceptron model that performs similarly to humans, with an 85% accuracy and a kappa value of 0.73, outperforming rule-based baselines. We trained and evaluated the model with papers in natural language processing (NLP). Our findings were also confirmed valid in fields other than NLP by applying the trained models to papers from various fields. These results will aid in improving mathematical language processing, such as mathematical information retrieval.

Keywords: Math Linguistics, Math Information Retrieval (MathIR), Feature Engineering, Coreference Relations

1. Introduction

In the automatic comprehension of scientific documents, a significant challenge arises from the inherent need to process mathematical expressions, which differ significantly in nature from natural language. A crucial difference between natural language and mathematical expressions lies in the nature of words versus math identifiers. These math identifiers are often concise, occasionally reduced to a single alphabetical character, and lack descriptive depth. Even in a single document, the meanings of math identifiers can be unclear, requiring intra-document disambiguation. This ambiguity has consistently been recognized as a significant hindrance in tasks such as Presentation-to-Computable (P2C) conversion, Math Reasoning, and Math Information Retrieval (MathIR) (Meadows and Freitas, 2022; Shan and Youssef, 2021; Youssef, 2017). Furthermore, there has been a lack of clarity regarding what information would help resolve these ambiguities. Our research aims to evaluate the important information needed for this disambiguation quantitatively and presents an automated technique.

Figure 1 depicts an instance of intra-document ambiguity in math identifiers (from Bishop (2006)). The bolded **y** is used in two distinct meanings: as a corresponding function to the behavior of a machine learning algorithm (Concept 1) and as the output vector of the function (Concept 2). We aim to assign Concept 1 to the first and third occurrences of **y** and Concept 2 to the second. Since occurrences assigned to the same concept have a coreference relation, this task can also be seen as coreference analysis for math identifiers.

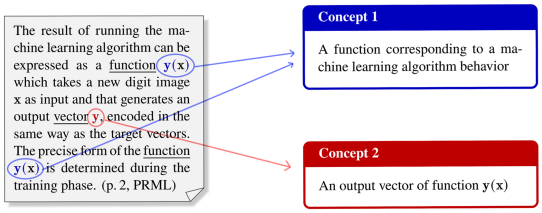

Figure 1: An example of intra-document ambiguity

In this study, we identified which information types are crucial for assigning math concepts by training a multi-layer perceptron model using various rule-based extracted features. Our experiments discerned three effective features for the task. The most critical of these is the position data within the document. The second is affix type, or local formula structure surrounding the math identifier occurrence, includes information such as the presence of superscripts or subscripts and whether parentheses follow. Although the natural language context surrounding the occurrence is the least important among the three features, it still provides benefits.

To address the complexity of the problem, in this study, we provided math concept candidates for each math identifier type (character and font) employed in the target papers. We included the position data of the initial occurrence associated with each math concept candidate. Although we provided this information, identifying the corresponding math concept for each math identifier occurrence remains challenging due to their intricate scopes. Assuming no scope switches occur ex-

LREC-COLING 2024, pages 17500–17512

cept the provided initial occurrence for each math concept (a cascade baseline), an 83% accuracy and a kappa value of 0.64 are achieved. In contrast, human inter-annotaor agreement (IAA) range from 75–90% accuracy, with kappa values ranging from 0.75 to 0.95 (Asakura et al., 2022). The top-performing model we developed, combining position data and affix type, attained 85% accuracy and a kappa value of 0.73, which lies between the cascade baseline and human performance.

Furthermore, we confirm that the information required for intra-document disambiguation of math identifiers is uniform across disciplines. The results mentioned above stem from developing and evaluating a model using 20 papers from the natural language processing (NLP) field. Upon evaluating the identical model with papers outside the NLP field, findings show that the three features are influential in the same order, consistent with evaluation for the NLP papers.

2. Related Work

Analyzing small structures within formulae, such as math identifiers, is recognized as extractive tasks within mathematical language processing (MLP) (Meadows and Freitas, 2022). Over time, multiple variations of the task have been proposed. The most popular task among these is *definition extraction* for math identifiers, also known as *description alignment*. This is the task to link each type of math identifiers within a document to its corresponding description in natural language.

Several automated techniques, including rule-based systems and machine learning-based methods, have been proposed for definition extraction. For instance, Pagel and Schubotz (2014) utilized the Gaussian heuristic ranking, while later on, Schubotz et al. (2016) and Schubotz et al. (2017) introduced hybrid techniques that combined Gaussian ranking with machine learning techniques such as *K*-means clustering and SVM. Recently, Alexeeva et al. (2020) put forth a rule-based method based on the Odin grammar.

Two alternative tasks focus on math identifiers by narrowing down the type of information attributed instead of natural language description. First, *variable typing* aims to deduce the mathematical type of a math identifier. Stathopoulos et al. (2018) formalized the task as a link prediction problem and utilized BiLSTM to solve the task. Secondly, *Part-of-Math (POM) tagging* targets all math tokens, including math identifiers, and assigns role labels within math formulae (Youssef, 2017). The primary labels given in this task are syntactic and consistent, such as operation, relation, and delimiter. A task similar to the POM tagging that centers on specific internal math structures, such as superscripts and primes, has also been proposed to classify their semantic roles (Shan and Youssef, 2021). The performance of various machine learning models, including random forests, SVM, and LSTM, for this task was evaluated. Although the target of disambiguation in this research differs from ours, it resembles our study.

Each of the existing tasks described above aims to address the problem of math identifiers having varied meanings across documents but does not consider the ambiguity of these identifiers within a single document. Math identifiers can carry multiple meanings and demonstrate complex scoping in a document. The Symlink shared task of SemEval 2022 (Lai et al., 2022) partially addresses the intra-document ambiguity but is limited to paragraph-level analysis. Specifically, it involves extracting coreference relationships among math tokens within a single paragraph. The state-of-the-art (SOTA) for this relation extraction task is a method called JBNU-CCLab (Lee and Na, 2022), utilizing SciBERT, though it achieves an F_1 score of only 37.19, indicating significant room for improvement. The context feature in our work essentially covers this (see Section 5.1).

While datasets are scarce that address intra-document polysemy of math identifiers, the Formula Grounding Dataset (Asakura et al., 2022) offers annotations for the issue. Our research aims to enhance the dataset and spearhead an initiative to disambiguate math identifiers within documents automatically.

3. Dataset and Task

3.1. Dataset Overview

We use a dataset that is an extension of the Formula Grounding Dataset (Asakura et al., 2022). Originally consisting of 15 papers, our extended dataset now contains a total of 40 papers[1]. The extension followed the same methodology as the original study, which involved manual annotation using the MioGatto annotation tool (Asakura et al., 2021). Each occurrence of a math identifier within the 40 target papers in this dataset is labeled with its corresponding math concept. Those math identifiers that are coreferenced are associated with the same math concept, while those not coreferenced are associated with different math concepts. Thus, the annotations in this dataset inherently contain coreference information.

To ensure the quality and reliability of the annotations in the extended dataset, we calculated the inter-annotator agreement rate on a random subset of the data. Specifically, for seven out of the 40

[1]The dataset is available on `https://sigmathling.kwarc.info/resources/grounding-dataset/`.

Dataset	#papers	#words	#types	#occr	#con
Original (Asakura+, 2022)	15	86098	680	12352	1418
Extended (Ours)	**40**	237062	1742	**27657**	**3603**

Table 1: The extended formula grounding dataset

papers included in our dataset, the average agreement rate was 95.7% (kappa value of 0.81). This is consistent with the results of the previous work (agreement rates of 84.2–96.5%, kappa values of 0.75–0.94), which we deemed sufficiently reliable.

Table 1 gives an overview of our extended dataset. In the table, "#papers" denotes the number of annotated papers,"#words" denotes the total number of words in those papers, "#types" denotes the total number of different types of math identifiers (character or symbol type along with its font), "#occr" denotes the total number of occurrences of math identifiers, and "#con" denotes the number of math concepts associated with the math identifiers in the papers. Due to a 2.6-fold increase in the number of target papers from the original dataset, the number of occurrences of annotated math identifiers increased by a factor of 2.23.

In the original dataset, there were only a total of 15 papers, with just a few papers for each specific domain. However, our extended dataset includes 20 papers from the NLP domain, along with eight papers from astronomy, five from other Computer Science (CS) domains besides NLP, three from economics, two from mathematics, and one each from physics and biology[2]. This expansion permits a somewhat quantitative analysis of variations across diverse domains. Primarily, our research employs 20 NLP papers for model design and evaluation. Additionally, we aim to evaluate the model's generalizability by applying it to papers from other fields.

3.2. Task Description

We propose a task of disambiguating math identifiers within a single paper. The task aims to assign math concepts to each math identifier occurrence.

Input:

- *Structured document representation* of the target paper (in XHTML format)
- *The initial occurrence position* associated with each math concept of the math identifiers

Output:

- *Math concepts* assigned to every occurrence of the math identifier within the target paper

The input paper's structured data is an XHTML document converted from a LaTeX document source using LaTeXML (B. Miller, 2018). Math formulae are encoded in Presentation MathML format (Ausbrooks et al., 2014). Each occurrence of a math identifier is represented by the `<mi>` tag. We can determine the precise character or symbol type and its typeface by examining the `<mi>` tag content and attributes.

Information regarding candidate math concepts to assign to each occurrence of a math identifier is provided as position information for the initial occurrence of the identifier associated with each concept. It indicates where a math concept is first used in the paper, often where it is defined or declared. This information only partially provides correct labels and is limited in number. For instance, our dataset (Table 1) contains 27,657 occurrences. The number of initial positions is equivalent to the number of math concepts, which is 3,603. If we focus only on annotating these initial positions, the manual annotation effort is reduced to about one-tenth. Additionally, we plan to automate identifying these initial positions in the future.

We split our dataset into development and evaluation data on a per-paper basis. During evaluation, the model will assign math concepts to each occurrence of a math identifier based on input from papers not in the training data. While a single paper may contain numerous math identifier occurrences, these occurrences will not be divided between the training and evaluation datasets. This data requirement is formulated assuming a realistic scenario in which, when provided with the structured data of an unfamiliar paper and its initial position annotations, the objective is to complete the remaining annotations.

3.3. Dataset Usage

Considering the varying writing styles and conventions across different research fields, we primarily utilized 20 research papers in NLP for both training and evaluation purposes. The NLP subset contains 9,314 instances of math identifiers (Table 2). Each of these math identifiers has been annotated with a math concept from a set of 1,518.

The dataset contains a limited number of papers due to the costly nature of annotation, and it is imperative to separate the evaluation data from the development data reasonably. However, the length of the paper and the complexity of the formulae used in the paper cause the difficulty of solving the task to vary significantly from paper to paper. We calculate the average candidate number to obtain a relatively accurate representation of this difficulty (the "#cand" column in Table 2). This metric is calculated as an average of the number of math concepts for each math identifier type

[2]We decided on the domain of the paper with respect to the arXiv categories. For example, we treat papers labeled with the category `cs.CL` on arXiv as NLP papers.

arXiv ID	#words	#type	#occr	#con	#cand	Test?
2004.08500	8165	102	2528	308	5.6	
1905.11006	4820	58	404	108	5.1	✓
1711.02281	4108	49	402	85	4.7	
2104.05336	9839	46	453	101	3.8	
2002.08046	5415	73	1175	167	3.3	
1802.08545	5900	44	1147	99	3.0	
2103.04350	4451	33	237	61	2.9	✓
1906.05149	2881	25	162	30	2.7	
2002.06823	6255	34	510	74	2.7	
2106.02134	5082	46	333	85	2.7	
1508.01745	4267	42	266	73	2.6	
2001.05139	5543	31	203	47	2.6	✓
1609.06038	3563	38	433	79	2.5	
2010.00710	4362	26	189	38	1.8	
2012.14116	4261	31	186	39	1.7	
1606.02821	2091	11	86	16	1.3	
2011.09553	4444	37	165	35	1.2	✓
2011.04946	4728	17	94	19	1.2	
2105.12523	2257	23	124	27	1.2	
2005.00175	4613	23	217	27	1.1	
Total	97045	789	**9314**	**1518**		

Table 2: Details for the NLP subset of our dataset

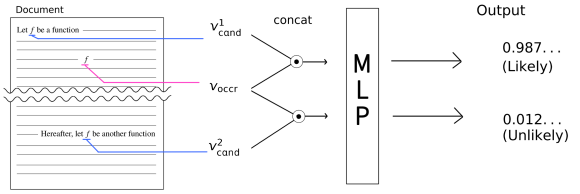

Figure 2: Architecture overview

weighted by the occurrence frequency. As a result, the expected probability of correctly guessing at random would be in accordance with the inverse of this average number of candidates. Test data was chosen to ensure the average number of candidates in the development and evaluation data was not heavily skewed (see "Test?" column in Table 2).

This extracted test dataset was crucial and exclusively reserved for the model's final evaluation. For all other processes, including model development, hyperparameter tuning, and granular analysis, we relied on the residual 16 papers classified under development data. To obtain maximum usefulness from this 16-paper dataset, we primarily employed the Leave-One-Out Cross Validation (LOOCV) approach for most experiments.

4. Machine Learning Model

In our task, the possible math concepts assigned to a math identifier occurrence vary by document and identifier type. Consequently, this task cannot be considered a straightforward multi-class classification problem. Instead, feature vectors for pairs of the occurrence and math concepts are produced for each candidate math concept. A three-layer perceptron is then trained to generate the probability for each pair.

Figure 2 presents a summary of the inputs and outputs of the perceptron. The input consists of feature vectors corresponding to the target occurrence and candidate math concept pairs. A vector of these is formed by concatenating the feature vector v_{occr} extracted from the target occurrence with the feature vector v^n_{cand} extracted from the first occurrence associated with a candidate math concept. The perceptron's output indicates the likelihood that the input pair is correct. For the final concept assignment, we select the math concept with the highest perceptron output, i.e., probability, from each pair by taking the argmax.

The model's performance was evaluated using the method employed by Asakura et al. (2022) to calculate IAA. The accuracy was determined by considering the human annotations as ground truth and calculating the simple agreement rate. Additionally, Cohen's kappa (Cohen, 1960) was calculated. This was calculated with the weighted average being determined by the number of occurrences for each identifier type.

Shifting our focus to the intricacies of our model's training, the Mean Squared Error (MSE) function was used as the preferred loss function. We methodically optimized hyperparameter using Optuna (Akiba et al., 2019) although detailed information has been included in Appendix A for brevity. At a high level, our exploration included a range of optimizers such as SGD, Adam, and RMSprop and a suite of activation functions including ReLU, ELU, Sigmoid, and tanh. Among the tested combinations, the combination SGD with ReLU consistently produced superior results, cementing its selection throughout our experimentation. We optimized other key hyperparameters, such as the dimension of the hidden layers, learning rates, and batch sizes specific to each model. It is important to note that, during the hyperparameter refinement phase, we exclusively used LOOCV on our development dataset.

5. Feature Engineering

In constructing the input for the multi-layer perceptron, we extract three types of information from each math identifier occurrence (Table 3). The first feature is *context*, i.e., the natural language text surrounding the occurrence. To convert the context into a vector representation, we utilize embeddings generated by the Sentence Transformer (Reimers and Gurevych, 2019). The second feature is *affix type*, which captures the local formula structure around the targeted math identifier. This encompasses whether the occurrence has been modified by elements such as a prime or is followed by parentheses. The analysis of the affix type is rule-based, and the resulting data is transformed into a vector representation. Lastly, the third feature is *position data*, which includes two types of information. The first is the binary

Feature	Description	Example
Context	Natural language text surrounding the occurrence	`the feature vector $v'_{x}}$ extracted from the`
Affix type	Local structure in the math formula	prime and subscript
Position data	With or without cascade effect and distance from the first instance	In cascade effect?: yes, Distance: 20 words

Table 3: Three types of the feature extracted from a math identifier occurrence

value of whether or not the target occurrence is within the cascade effect. The second specifies how far the target occurrence is from the initial instance of the paired math concept.

Regarding feature engineering, all design decisions were based on the development dataset, which included 16 papers; the test dataset remained untouched. To measure the effectiveness of various features, we utilized LOOCV as our evaluation tool. Anticipating potential variations in model performance resulting from initial settings, such as initial parameter values, our approach involved training iterations using three different seeds. Subsequently, we compared the performances based on their averaged outcomes. An early stopping was implemented to prevent overfitting, with a patience of 3 and a cap of 20 epochs.

5.1. Context Embeddings

Following recent advances in NLP techniques based on the distributional hypothesis, we utilize the context information of math identifiers. To make context compatible with our perceptron, we incorporated the functionality of the Sentence Transformer. The feature vectors are constructed for the occurrence of each math identifier and its corresponding math concept, with both being concatenated. The context of the identifier occurrence is extracted from the text surrounding it. On the other hand, regarding the math concept's context, we base it on the position of its initial introduction. We use the surrounding text from this debut position as its context.

MiniLM (Wang et al., 2020) served as our pretrained model for Sentence Transformer. We used LOOCV to compare the performance of various pretrained models and concluded that MiniLM is the fastest and the most effective for our task (Table 4). The widely recognized high-performing generic model, MPNet (Song et al., 2020), performs slightly worse than MiniLM. We also used the STS-fine-tuned versions of MPNet and SciBERT (Beltagy et al., 2019; Deka and Jurek-Loughrey, 2021) models, but their performance is inferior to MiniLM for our task.

Variations in the context window size and representation methods for formulae have minimal influence on task performance. We confirmed the negligible variance among these variations by LOOCV (see Appendix B). In our evaluation experiments, we used a window size of 20 and LaTeX representation for the math formulae in the contexts.

Model	Accuracy	Kappa
`all-MiniLM-L6-v2`	**0.7682**	**0.3441**
`all-mpnet-base-v2`	0.7583	0.3151
`stsb-mpnet-base-v2`	0.7260	0.2261
`pritamdeka/S-Scibert-snli-multinli-stsb`	0.6911	0.0000

Table 4: Sentence Transformer models

Affix type (Example)	Occurred	Recall	Precision	F1
subscript (B_a)	1914	0.971	0.898	0.933
superscript (B^a)	548	0.987	0.646	0.781
comma ($B(a_0, a_1)$)	150	0.940	0.613	0.742
semicolon ($B(a_0; a_1)$)	26	0.961	0.641	0.769
colon ($B(a_0 : a_1)$)	0	—	—	—
prime (B')	117	0.931	1.000	0.964
asterisk (B^*)	92	1.000	0.978	0.989
circle (B°)	0	—	—	—
hat ($\hat{B}$)	93	0.924	0.934	0.929
tilde ($\tilde{B}$)	44	1.000	1.000	1.000
bar ($\bar{B}$)	146	0.863	0.984	0.919
over ($\overset{a}{B}$)	0	—	—	—
over right arrow ($\vec{B}$)	4	1.000	1.000	1.000
over left arrow ($\overleftarrow{B}$)	4	1.000	1.000	1.000
dot ($\dot{B}$)	0	—	—	—
double dot ($\ddot{B}$)	0	—	—	—
open parenthesis ($B(a)$)	611	0.901	0.880	0.890
close parenthesis ($B(a)$)	611	0.901	0.880	0.890
open bracket ($B[a]$)	4	1.000	1.000	1.000
close bracket ($B[a]$)	4	1.000	1.000	1.000
open brace ($B\{a\}$)	5	1.000	0.454	0.625
close brace ($B\{a\}$)	5	1.000	0.454	0.625
vertical bar ($B(a_0 \mid a_1)$)	82	0.878	0.900	0.888
leftside argument (aB)	2	—	—	—
rightside argument (Ba)	4	—	—	—
leftside base (a^B)	13	1.000	0.812	0.896

Table 5: Performance of rule-based affix detection

5.2. Affix Types

The term *affix type* refers to the type of affix that accompanies math identifiers, such as subscripts, accent marks, and others. Our dataset contains a total of 26 annotated affix types, as shown in the first column of Table 5. These affix types are utilized to extract the inherent structural information within math formulae that may not be fully captured by context embeddings.

We developed a rule-based algorithm to distinguish the different types of affixes. After evaluating the algorithm over the full development dataset, we found a 90.56% accuracy rate. Table 3 provides a comprehensive performance breakdown for each affix type. It is worth noting that certain affix types, e.g., colon and circle, have not been used in the NLP domain's development data despite appearing in the evaluation data and data from other domains. Such affix types are not supported in our current algorithm.

Most errors for the detection fall into the following:

- *Indistinguishable by pattern*: for these cases, correct identification is challenging solely based on patterns. For example, the precision for superscript is comparatively lower because discerning whether a superscript functions as an exponent operator or forms a part of the identifier is unfeasible through pattern recognition alone. Similarly, discerning between omitted multiplications and left/right-side arguments is also challenging, and thus, these affix types are not supported.
- *Annotation errors*: Affix types such as the comma, which are easy to overlook, primarily contribute to errors resulting from annotation issues rather than the algorithm's design.

We assessed our perceptron using output from a preliminary algorithm with 88.77% accuracy, as well as our final rule-based algorithm. The preliminary algorithm's average LOOCV accuracy was 82.88%, whereas the final algorithm achieved 83.19%. Despite improving the affix type identification accuracy by 1.79 percentage points, the overall performance only increased by 0.31 percentage points. This indicates that addressing additional corner cases have limited significance.

Some efforts have been made to acquire embeddings for the mathematical expressions (Greiner-Petter et al., 2020; Youssef and B. R. Miller, 2018), which might serve as alternatives for the affix type feature, and their performance comparison could be significant in efforts towards SOTA. However, our objective is not to aim for SOTA but to identify the crucial information for intra-document disambiguation of math identifiers. While exploring various methods for formula embeddings is important, it has little impact on our key finding according to the error analysis of our models (see Section 6.2).

5.3. Position Data

Based on the annotation data analysis, position data is integrated as a model feature. Observations regarding the scope of math identifiers in documents uncovered a cascade effect (Asakura et al., 2022). In most cases, a math identifier occurrence aligns with the preceding occurrence of the same identifier type. The position data of the target occurrence within a document is a powerful clue for identifying the associated math concept.

As for the model input, position data comprises two distinct types. The first is a binary value that indicates whether the target occurrence falls within the cascade effect's scope. The value is 1 (true) if there is no initial position of another candidate concept between the position of the target occurrence and the initial instance of the paired candidate concept and 0 (false) otherwise. We call a method of concept assignment that relies solely on this binary value a *cascade baseline*.

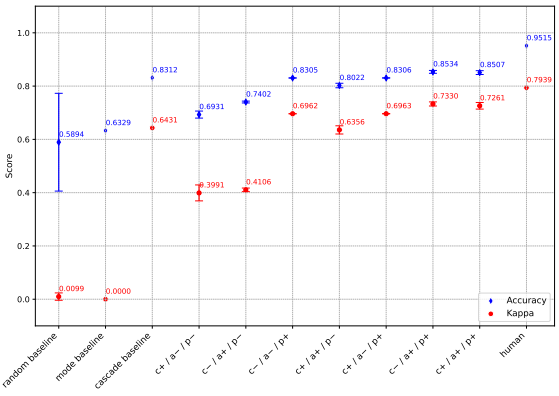

Figure 3: Results of test evaluation

The second is the distance between the target occurrence and the initial instance of the paired candidate concept. This value is standardized to $[-1.0, 1.0]$ when fed into the model. Although the distance information has no impact in case the position data is the only feature given to the perceptron, it slightly boosts the performance (on average, 0.61 points in accuracy and 0.0084 in kappa value) when combined with other features.

6. Experiment I: Model Comparison

To determine which information is most helpful for intra-document disambiguation of math identifiers, we inputted the three types of features in various combinations into a three-layer perceptron and compared their performance. The model was trained on all development data, which included 16 papers in the field of NLP, and then evaluated on test data consisting of four other NLP papers.

For comparative analysis, we created three different baselines. The *random baseline* conducts all assignments at random. The *mode baseline* assigns to all occurrences the math concept first introduced for each identifier type. Finally, the *cascade baseline* takes advantage of the cascade effect (see Section 5.3).

The evaluation results are shown in Figure 3. Except for the baseline labels on the *x*-axis, each indicates the features used: 'c' for context, 'a' for affix type, and 'p' for position data. The use of a feature in a given model is indicated by '+', while its absence is indicated by '-'. Error bars attached to each data point are derived from the standard deviations. We also computed the agreement rate between two human annotators on the same test data for reference. The first annotator, a graduate student majoring in NLP, performed both the listing of candidate math concepts and the assignment of them. The second annotator, a postdoc-

toral researcher majoring in fields other than NLP, has prior experience with this type of annotation.

6.1. Feature Significance

We found that the three features we identified, context, affix type, and position data each significantly contribute to the effectiveness of our models. Furthermore, all of these features, even when used alone, outperform both the random and mode baselines. The evaluation data contain identifier types used unambiguously within a document, and even the baselines demonstrate a certain degree of accuracy. However, it is essential to note that the kappa value, intended to eliminate random predictions, is meager for these baselines, as expected. Including any of the features leads to a substantial increase in kappa values.

Of the three features, position data stands out as the most critical due to its inherent ability to encapsulate both the cascade effect and relative distance information. This results in the position-only model (c-/a-/p+) has nearly identical, if not significantly better, performance than the cascade baseline. In contrast, while the other two features are valuable, they do not match the level of performance achieved by the cascade baseline and appear less significant compared to position data.

Shifting our attention to the outcomes of models that blend two features, we observe that incorporating affix type with position data surpasses the cascade baseline, resulting in the most successful model detected in this evaluation. In contrast, the addition of context to position data barely enhances performance. This implies that much of the knowledge obtained from context is already contained within position data. While the experiments in Section 5.1 suggested limited integration of intra-formula information in the embedding representation by the Sentence Transformer, our present findings agree with this. In that, information from context is majorly covered in position data, while affix type provides exceptional insights that are not accessible in position data.

When all features are combined, the model's performance slightly falls behind the position data and affix type combination in our evaluation. This slight difference could be due to the context acting as unintentional noise. However, this deviation as possibly falling within the margin of error since it contrasts with the performance order in the LOOCV.

6.2. Error Analysis

Differences by paper To estimate the difficulty for various papers, we analyzed the performance of our model on each of the 16 development data papers. We depicted the correlation between the average number of candidates per paper and the

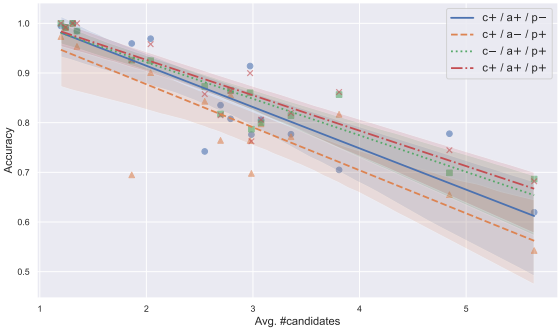

Figure 4: Performance by paper in LOOCV

Position	0–20%	20–40%	40–60%	60–80%	80–100%
Accuracy	0.9336	0.9137	0.8391	0.8395	0.8779
Kappa	0.7658	0.6909	0.6358	0.5080	0.4142

Table 6: Performance by position

models' performances in Figure 4. Our focus is on four models that employed two or more features.

Across all models, a strong negative correlation was found between the two variables, with correlation coefficients ranging from -0.944 to -0.849. This means that the model's performance tends to deteriorate as the average number of candidates increases, which indicates a more challenging task. This trend aligns with what would be expected when selecting candidates randomly.

Additionally, the performance was observed to vary significantly among papers, even for the same model. To thoroughly compare task difficulty across diverse fields, a significant amount of data is necessary to minimize the variations observed among individual papers. As our existing dataset consists of 40 papers, making task-difficulty comparisons across domains remains challenging.

Error trends Using the model c+/a+/p+ trained in the LOOCV, we conducted an in-depth analysis of specific error cases. Figure 5 presents a graphic display of the contrast for paper 1711.02281 between human-annotated correct labels and the output generated by the model. The horizontal axis represents the positions in the paper, and the vertical axis represents the math concepts assigned to math identifiers that occurred at each position. In this figure, when the correct label matches the model output, it is represented by a single gray line. Discrepancies between the correct label and the model's outputs are highlighted with a thick blue line for the correct label and a dotted red line for the model output. In this figure, only the math identifier types with discrepancies between correct labels and model outputs are extracted.

Observing Figure 5, two error trends can be identified: (1) errors occur more frequently towards

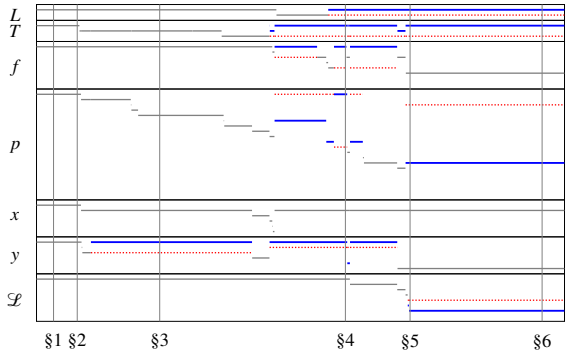

Figure 5: Contrast of labels and model outputs

the end of the documents, and (2) the thick blue line (correct labels) often appears above the dotted red line (model outputs). Computing the performance against the 16 papers in the LOOCV by partitioning according to each paper's position yields results that agree with trend (1) (Table 6). Quantitatively, the model performance, especially the kappa value, decreases as the document progresses toward the latter sections. This can be attributed to the fact that the initial sections of the papers introduce fewer math concepts, thus simplifying the assignment. In contrast, the later sections present many choices due to introducing numerous concepts. Trend (2) suggests that our model has difficulty with patterns where math concepts, previously introduced in the document, are used again at distant positions. In 90.97% of errors observed during LOOCV, the correct math concept is introduced at an earlier position than the model output.

Frequent error patterns The most common error pattern identified was the *scope-switch oversight*, in which the model output failed to correspond with the updated scope indicated by the human annotation. This oversight accounted for 62.79% of all errors produced during the LOOCV. In Figure 5, an illustration of this error type is given by the discrepancy related to the math identifier L in the latter part of §3. The thick blue line indicates the correct scope, which diverges from the preceding grey line. Nonetheless, the model's output, shown by the dotted red line, stays connected to that previous grey line. This oversight happens frequently for math identifiers that are used in multiple meanings even though they appear closely together. For instance, in the paper 1711.02281, the variables T and T' occurred nearby within a math formula. Additionally, the adjacent context contains the matrix transpose symbol denoted as T. While our rule-based affix type detector can distinguish between T and T' accurately, the perceptron model heavily relied on the position feature. Consequently, this preference led to frequent errors in the output.

While rare, the model's output sometimes suggests a change in the meaning of a math identifier despite the correct annotation not indicating a scope switch. This phenomenon is referred to as a *false scope switch*. These discrepancies constitute 13.7% of the total errors. Although difficult to visually discern from Figure 5, the error regarding the identifier f can be observed in §4. Before the mismatch, there is a match aligned with the y-coordinate of the thick blue line, from which only the model's output, represented by the dotted red line, deviates. The cause for false scope switches often remains unknown upon analyzing individual instances. Despite the proximity of positions and matching affix types, the model sometimes mistakenly introduces false scope switches. These errors may stem from the model's prioritization dilemma between position and affix type features. False scope switches are more likely to occur in situations with regular scope changes or math identifier types prone to affix type misclassification. For example, superscript judgments have low precision (see Section 5.2), and identifiers involving this are susceptible to false scope switching.

Errors that do not fall into the above two patterns are more complicated, where the correct scope and the model output shift to different scopes. Identifying a consistent pattern in such errors proves challenging. An interesting case from the paper 1711.02281 in the second half of §3 illustrates this. An occurrence of the identifier f occurs within a sentence beginning with "As shown in Fig. 2." Interestingly, "Fig. 2" refers to a topic introduced much earlier, specifically in §2. Consequently, the introduction of this first sentence effectively reintroduces the topic from §2. As a result, the correct math concept associated with the identifier f in this case should be consistent with its occurrence in §2. However, the output of the model does not account for such cases. To enable the model to handle such complicated scenarios, capturing the topic to which "Fig. 2" refers is a formidable challenge. While such instances are rare in most papers, typically occurring once or twice, several papers in our constructed dataset contain sentences that revive previous topics.

7. Experiment II: Cross-domain

Through evaluation with a subset of our dataset containing papers from non-NLP fields, we examined the effectiveness of our proposed model and the findings presented in the previous section in a cross-domain setting. We trained our model using the same 16 NLP papers previously used in Experiment I and evaluated with 11 other papers selected from the non-NLP subset of our dataset. We limit the number of papers in a single field to a

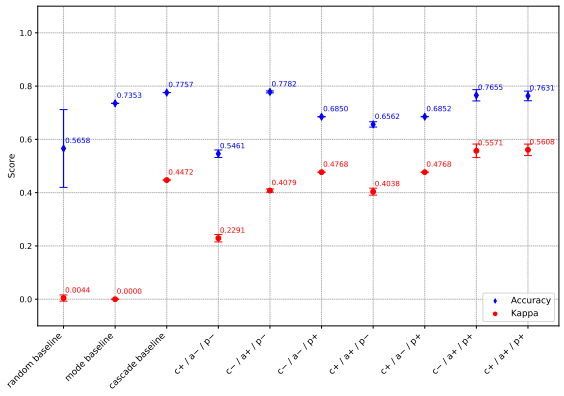

Figure 6: Results of cross-domain evaluation

maximum of three to prevent bias among the constituent areas.

The result of the cross-domain experiment is shown in Figure 6. Due to the uneven distribution of candidate numbers for each identifier, some models' accuracy appears outside the norm. However, the trend observed in the kappa values aligns closely with the results of Experiment I: in terms of feature importance, position data and affix type rank highest, followed by context. The model that combined position data and affix type exhibited notably high performance. Adding context to this model resulted in minimal impact on its effectiveness. Evaluating the performance of these two models based on kappa values demonstrates that they exceed all the baseline models.

8. Conclusion

In this study, we investigated the information types that aid in the intra-document disambiguation of math identifiers by training a multi-layer perceptron with various features derived from our expanded formula grounding dataset. Position data is found to be the most important feature, followed by the affix type and context. Although our experiments focused on NLP papers for training and evaluation, the model trained on such papers performs consistently when evaluated on papers from other disciplines. These domain-independent observations are expected to be advantageous for developing more efficient models for future intra-document disambiguation of math identifiers. Automating the intra-document disambiguation process will be crucial in various MLP tasks. For instance, this will allow text to be presented at a finer granularity than the document level in MathIR tasks and facilitate even more precise P2C conversions than previously achieved.

9. Acknowledgements

This work has been supported by JST, ACT-X Grant Number JPMJAX2002, Japan.

10. References

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2623–2631.

M. Alexeeva, R. Sharp, M. A. Valenzuela-Escárcega, J. Kadowaki, A. Pyarelal, and C. Morrison (2020). MathAlign: Linking Formula Identifiers to their Contextual Natural Language Descriptions. In *Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020)*, 2204–2212.

T. Asakura, Y. Miyao, and A. Aizawa (2022). Building Dataset for Grounding of Formulae — Annotating Coreference Relations Among Math Identifiers. In *Proceedings of 13th Conference on Language Resources and Evaluation (LREC 2022)*.

T. Asakura, Y. Miyao, A. Aizawa, and M. Kohlhase (2021). MioGatto: A Math Identifier-oriented Grounding Annotation Tool. In *13th MathUI Workshop at 14th Conference on Intelligent Computer Mathematics (MathUI 2021)*.

R. Ausbrooks et al. (2014). Mathematical Markup Language (MathML) 3.0 Specification. World Wide Web Consortium (W3C).

I. Beltagy, K. Lo, and A. Cohan (2019). SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.

C. M. Bishop (2006). Pattern Recognition and Machine Learning. Springer.

J. Cohen (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*.

P. Deka and A. Jurek-Loughrey (2021). Unsupervised Keyword Combination Query Generation from Online Health Related Content for Evidence-Based Fact Checking. In *The 23rd International Conference on Information Integration and Web Intelligence*, 267–277.

A. Greiner-Petter, A. Youssef, T. Ruas, B. R. Miller, M. Schubotz, A. Aizawa, and B. Gipp (2020). Math-word embedding in math search and semantic extraction. *Scientometrics* 125, 3017–3046.

V. D. Lai, A. P. B. Veyseh, F. Dernoncourt, and T. H. Nguyen (2022). SemEval 2022 Task 12: Symlink — Linking Mathematical Symbols to their Descriptions. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*.

S.-M. Lee and S.-H. Na (2022). JBNU-CCLab at SemEval-2022 Task 12: Machine Reading Comprehension and Span Pair Classification for Linking Mathematical Symbols to Their Descriptions. In *Proceedings of the 16th International*

Workshop on Semantic Evaluation (SemEval-2022), 1679–1686.

J. Meadows and A. Freitas (2022). A Survey in Mathematical Language Processing. arXiv:2205.15231. arXiv: 2205.15231 [cs].

B. Miller (2018). LaTeXML *The Manual — A LaTeX to XML/HTML/MathML Converter, Version 0.8.3*.

R. Pagel and M. Schubotz (2014). Mathematical Language Processing Project. In *Joint Proceedings of the MathUI, OpenMath and ThEdu Workshops and Work in Progress track at CICM*.

N. Reimers and I. Gurevych (2019). Sentence-BERT: Sentence embeddings using siamese bert-networks. arXiv:1908.10084.

M. Schubotz, A. Grigorev, M. Leich, H. S. Cohl, N. Meuschke, B. Gipp, A. S. Youssef, and V. Markl (2016). Semantification of Identifiers in Mathematics for Better Math Information Retrieval. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '16*, 135–144.

M. Schubotz, L. Krämer, N. Meuschke, F. Hamborg, and B. Gipp (2017). Evaluating and Improving the Extraction of Mathematical Identifier Definitions. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction — 8th International Conference of the CLEF Association, CLEF 2017, Dublin, Ireland*. Vol. 10456. Lecture Notes in Computer Science. Springer, 82–94.

R. Shan and A. Youssef (2021). Towards Math Terms Disambiguation Using Machine Learning. *Intelligent Computer Mathematics*. Ed. by F. Kamareddine and C. Sacerdoti Coen. Vol. 12833. Springer International Publishing, 90–106.

K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu (2020). MPNet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems* 33, 16857–16867.

Y. Stathopoulos, S. Baker, M. Rei, and S. Teufel (2018). Variable typing: Assigning meaning to variables in mathematical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 303–312.

W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou (2020). MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems* 33, 5776–5788.

A. Youssef (2017). Part-of-Math Tagging and Applications. In *Intelligent Computer Mathematics — 10th International Conference, CICM 2017, Edinburgh, UK, July 17–21, 2017, Proceedings*. Springer International Publishing, 356–374.

Parameter	Phase 1	Phase 2
optimizer	SGD, Adam, RMSprop	SGD
activation	Sigmoid, tanh, ReLU, ELU	ReLU
hidden size	2–input size	2–input size[4]
lr	10^{-5}–1.0	10^{-3}–1.0
batch size	10–512	20–512

Table 7: Settings of hyperparameter tuning

A. Youssef and B. R. Miller (2018). Deep learning for math knowledge processing. In *Intelligent Computer Mathematics — 11th International Conference, CICM 2018, Hagenberg, Austria, August 13–17, 2018, Proceedings*. Springer, 271–286.

A. Hyperparameters

In this study, we employed a three-layer perceptron to explore how various features, such as contexts surrounding math formulae and position data within the document, can assist in disambiguating math identifiers. Due to the varying dimensions of the input vector based on the combination of features used, multiple machine learning models with distinct hyperparameters were necessary. Neglecting to optimize hyperparameters may impede the perceptron's capacity to effectively utilize the built-in features, thus generating inaccurate evaluations of their importance.

We employed Optuna[3], a hyperparameter optimization framework utilizing Bayesian optimization, to conduct an adequate search for the optimal hyperparameter. The optimization process involved performing LOOCV on the development data for each trial, with the primary objective of maximizing accuracy.

The hyperparameter search was performed in two phases. In the first phase, a broad hyperparameter search was executed for a model that utilized all three features identified in the study. Table 7 presents a comprehensive names of the parameters and their search ranges for this phase. This initial search comprised 150 trials and lasted about seven hours on a 15-inch M2 MacBook Air (2023, 8-core CPU, 24GB RAM, 1GPU (mps)).

Figures 7 and 8 illustrate the progression of the initial search phase and the significance of each hyperparameter, respectively. It is worth noting that selecting the optimizer and activation function is paramount for our task. As a result, the optimizer was fixed to SGD, and the activation function to

[3] https://optuna.org

[4] Exceptions were made when the input size was smaller than five, in which case a range of one to five was searched.

ReLU for all further experiments. Although the impact of other parameters was relatively limited, we conducted the second phase search to fine-tune the parameters for each model, given the significant variation in input vector dimensions between models.

In the second phase, we narrowed our search based on findings from the first phase. The rightmost column of Table 8 enumerates the refined search ranges. We observed that the speed of LOOCV was greatly influenced by batch size. Although training time increased with smaller batches, our investigation in the first phase revealed that relatively larger batches did not negatively affect performance. Thus, in the second phase, we set a higher lower bound for the batch size to reduce optimization time. Each model underwent 50 trials during this phase, with each search taking approximately two hours on the M2 MacBook Air.

After the second phase of the search, we finalized hyperparameters for each model (Table 8). These hyperparameters were used for LOOCV during feature engineering and evaluation experiments on the test data. Throughout feature engineering, multiple features with slightly varying input dimensions were tested, and the values from the table were usually reused unless there was a considerable alteration in the input size. Nonetheless, there were two exceptions as follows.

- Given the significant difference in dimensions, 378 and 756, depending on the Sentence Transformer type utilized, we conducted separate optimizations for each case.
- Individual adjustments were conducted for models with a small input dimension, where altering the hyperparameters impeded adequate training.

B. Formatting inputs for Sentence Transformer

Representation of math formulae Options include keeping the LaTeX format, replacing all formulae with special `[MATH]` tokens, or using a `[TARGET]` token exclusively for the formula that has the target occurrence while replacing other formulae with `[MATH]`. No significant performance differences were noted among these variations, with accuracies slightly fluctuating around 76.82% for the original LaTeX format, 75.72% when using the `[MATH]` tag consistently, and 75.11% when employing both `[TARGET]` and `[MATH]` (Figure 9). This suggests that the embedding vectors generated by the Sentence Transformer may not effectively encode the innate structural data within the equations for this particular task.

Size of context window This means selecting the number of words encompassing the math formula to be considered context, where formulae are deemed a single word. We tested 5, 10, 15, 20, 25, and 30 window sizes, with the results illustrated in Figure 10. The impact of window size on model performance was insignificant, with no noticeable correlation exceeding the margin of error thresholds. All subsequent experiments utilized a consistent window size of 20.

Based on these results, for the final evaluation experiments, we use a window size of 20, keep the math formulae in their LaTeX format for context extraction, and use MiniLM-generated embeddings as the contextual feature vectors.

C. The non-NLP Subset

Details of the subset of papers used from fields other than NLP are presented in Table 9. We selected two papers each from fields such as astronomy and CS, as they had a relatively higher number of papers, to serve as test data. As with the NLP subset, we ensured that the chosen test data had an unbiased average number of candidates. Papers from other domains were exclusively used as test data and were not included in the training set.

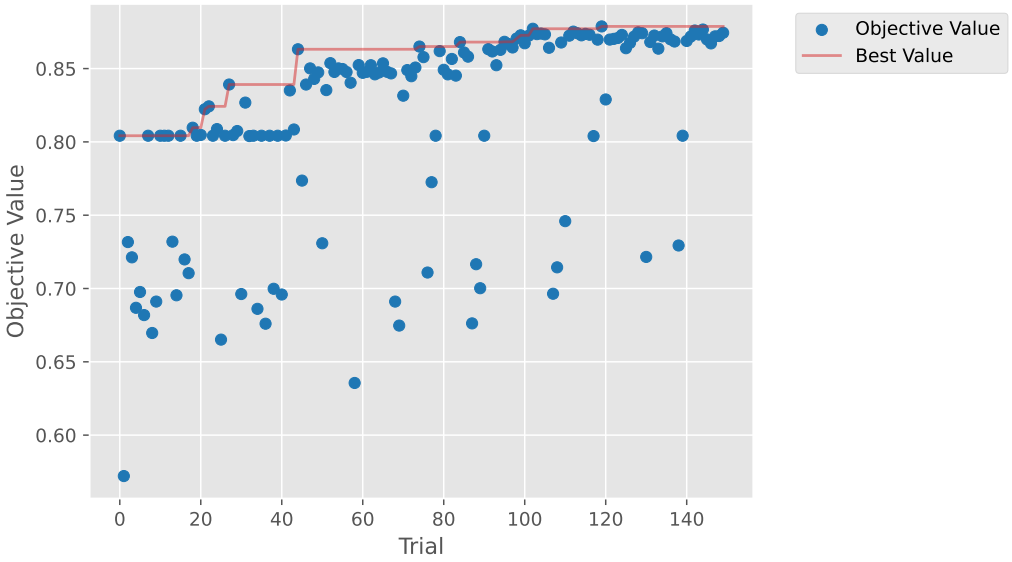

Figure 7: Optimization history

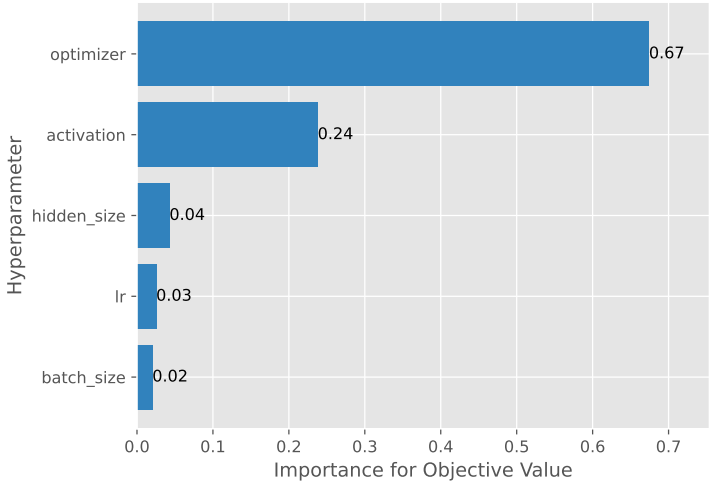

Figure 8: Importances of hyperparameters

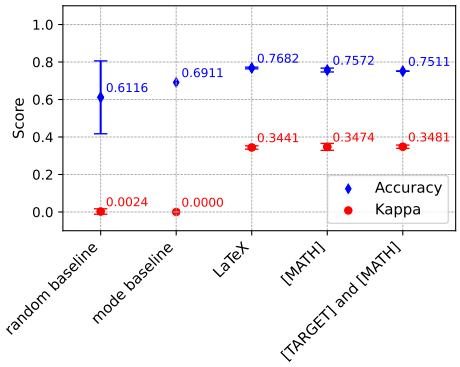

Figure 9: Representation of math formulae

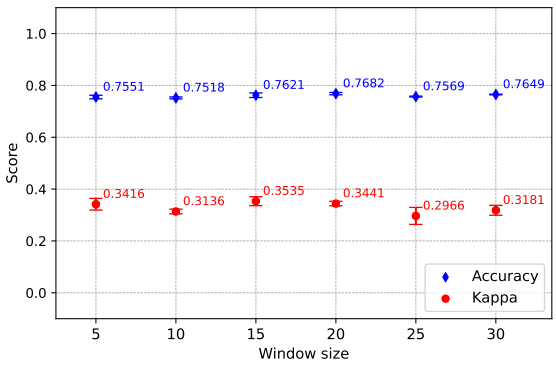

Figure 10: Window size and performance

model	input size	hidden size	lr	batch size
c+ (MiniLM) / a− / p−	768	694	0.639	23
c+ (Others) / a− / p−	1536	1136	0.722	67
c− / a+ / p−	52	47	0.328	70
c− / a− / p+ (w/dist)	2	3	0.244	85
c− / a− / p+ (wo/dist)	1	3	0.150	175
c+ / a+ / p−	820	701	0.310	21
c+ / a− / p+	770	335	0.483	57
c− / a+ / p+	54	34	0.698	35
c+ / a+ / p+	822	108	0.414	36

Table 8: Optimized hyperparameters for each model

arXiv ID	arXiv category	Our category	#words	#type	#occr	#con	#cand	Test?
2106.09995	astro-ph.SR	astronomy	7793	62	1349	191	5.7	
2012.07856	astro-ph.SR	astronomy	10032	59	1064	129	4.2	
2201.05402	astro-ph.SR	astronomy	5477	64	456	152	3.6	
2003.10641	astro-ph.SR	astronomy	6821	51	787	107	3.3	
1903.02241	astro-ph.SR	astronomy	6016	33	682	77	3.0	✓
1911.11277	astro-ph.SR	astronomy	9519	34	699	77	3.0	
1805.00495	astro-ph.IM	astronomy	7347	36	614	67	2.3	✓
1005.1008	astro-ph.CO	astronomy	5176	32	332	50	1.7	
1808.02342	cs.IT	cs	10976	40	935	104	6.4	
1805.08522	stat.ML	cs	10919	56	581	139	5.2	
2203.00854	cs.LG	cs	6835	28	120	55	3.8	
2107.10832	cs.LO	cs	3567	46	1648	64	1.9	✓
2205.03055	cs.CV	cs	4816	46	480	52	1.7	✓
2110.13716	q-fin.ST	economics	5963	46	676	110	3.6	✓
1903.06478	q-fin.CP	economics	4471	48	362	59	2.7	✓
1807.00939	q-fin.ST	economics	6443	10	35	8	1.2	✓
math0303074	math.AG	mathematics	13154	141	4628	424	5.2	✓
2008.08349	math.CO	mathematics	6091	49	1898	86	2.8	✓
0912.3513	q-bio.NC	biology	3738	31	470	61	2.7	✓
1708.02771	cond-mat.mtrl-sci	physics	4863	41	561	73	2.3	✓
Total			140017	953	**18377**	**2085**		

Table 9: Details for the other-domain subset of our dataset