A Math-Aware Search Engine for Math Question Answering System

Tam T. Nguyen, Kuiyu Chang, and Siu Cheung Hui
Nanyang Technological University
50 Nanyang Avenue, Singapore 639798
nguy0080@e.ntu.edu.sg, {askychang, asschui}@ntu.edu.sg

ABSTRACT

We propose a math-aware search engine that is capable of handling both textual keywords as well as mathematical expressions. Our math feature extraction and representation framework captures the semantics of math expressions via a Finite State Machine model. We adapt the passive aggressive online learning binary classifier as the ranking model. We benchmarked our approach against three classical information retrieval (IR) strategies on math documents crawled from *Math Overflow*, a well-known online math question answering system. Experimental results show that our proposed approach can perform better than other methods by more than 9%.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Performance

Keywords

math document retrieval, math-aware search engine, learning to rank

1. INTRODUCTION

Math question answering (Q&A) systems deal with mathematical questions and answers embedded in math documents. A math document can contain both textual data and math expressions.Major online Math Q&A systems such as *Math Overflow* are constantly monitored and updated by their online user community. Therefore, an up-to-date and effective *math-aware search system* is essential for users to find the latest math documents. This system must support searching based on both keywords and math expressions. Although there are many conventional keyword-based search engines available for searching textual documents, to the best of our knowledge, there exist no publicly-available math-aware search engine for math Q&A systems.

Wolfram research hosts a public computation knowledge search engine called Wolfram *Alpha* [1], which can parse simple math queries and expressions. It is not a search engine per se, but more of a computational engine *Mathematica*[2]. For example, *Alpha* will compute and return the numerical result of a math expression like $e^{\sin 20}$, in addition to providing its alternate form and continuous fraction representation. For math related search queries like "Riemann", *Alpha* returns a brief biographical summary of Bernhard Riemann the Mathematician, including his full name, date of birth/death, etc. What it will not do is return a list of relevant documents like conventional Web search engines. Although this system does not require huge amount of storage for operation, it is not really an all-purpose search engine since it returns largely the computational results and/or knowledge from a few selected common topics instead of Web pages crawled from the entire Internet. For instance, if we search for "iphone5", *Alpha* will respond with the message, "development of this topic is under investigation" even though "iphone5" is a very popular term [3]. On the other hand, *Alpha* understands search queries of stock symbols and also geographical entities such as "Los Angeles".

Another approach to tackle the problem is to extend text search systems for math search. In the preprocessing phase, math expressions are simply converted to textual data, after which classical information retrieval (IR) techniques are applied for retrieval, as was previously reported in *Mathdex* [18] and *MathWebSearch* [12, 14]. Developing math search systems in this way is fast and simple, but their performance depends heavily on the fidelity of the conversion from symbolic math expressions to textual representations.

To overcome the above limitations, we propose a novel approach for math-aware search, which comprises two steps: (1) extract meaningful math index terms based on the semantically rich *MathML* format; (2) apply the learning-to-rank approach to improve the accuracy of the ranking process.

1.1 A Motivating Example

A question posted by user *"Alejandro Erickson"* from the Math Overflow system is shown below. The question belongs to the probability category.

[1]http://www.wolframalpha.com
[2]http://www.wolfram.com/mathematica
[3]as of August 2012, prior to the actual product launch date.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

Title: Formulate edge length problem as convex optimization problem

Content: I want to use convex optimization to describe a problem in computational geometry. Let $E = (E_1, E_2, \ldots, E_m)$ be a sequence of line segments in the plane, where E_1 and E_m may be points and the rest are non-degenerate segments. A *critical path* on E is a selection of points $(p_1, p_2, \cdots, p_m)$ with $p_i \in E_i$ with edges $e_i = (p_i, p_{i+1})$ such that (1) all e_i has the same length l and (2) no other selection of points results in a path with edges that are not longer than l and some that are shorter.

Find the critical path using convex optimization (if it exists). Can you use the convex optimization problem to show that the solution is unique? ...

To provide a search facility for questions like this, one can ignore math expressions and simply deploy a conventional keyword-based search engine. However, users could also be interested in finding math documents containing a particular math expression. In this case, conventional search engines fail miserably. For one, it cannot handle mathematical expressions in the query. Second, it does not know that a+b is semantically equivalent to x+y. These are just two of the many limitations of the simple approach. We shall discuss more in the following section.

1.2 Our Key Contributions

The key contributions of this paper are summarized below.

- We propose a two-view representation format for math documents.
- We propose a new math *feature extraction* algorithm for math expressions.
- We propose a novel *matching model* for math document retrieval, which utilizes both math and text index terms in a balanced manner.
- We propose a new learning-to-rank algorithm in a *ranking model* for improving the ranking of the search results.

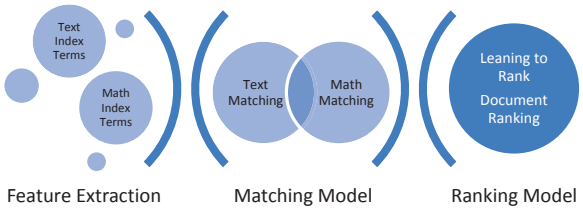

Figure 1: Math-Aware Search Engine Framework.

Figure 1 shows the proposed math-aware search engine framework. The idea is to extract and index text and math terms independently in two separate inverted indexes. A user query is first matched against both the math and text inverted indexes to retrieve an initial set of unranked candidates based on the intersection of matching documents in both the math and text query terms. A continuously learning *ranking model* assign scores to each document in this candidate set. Figure 2 shows the graphical user interface of the search engine, in which the user-friendly graphical equation editor CODECOGS allows math expressions to be entered effortlessly. The CODECOGS[4] editor converts user entered math expressions automatically into our preferred LaTeX storage format.

2. RELATED WORK

There exists several math-aware retrieval systems such as MathFind [19], ActiveMath [15], Wolfram Formula Search[5], Wikipedia Formula Search[6], Whelp [1], Mathdex [18], and MathWebSearch [12, 14]The vast majority of these systems are developed based mainly on conventional text retrieval techniques, which treats math expressions as textual data.

Since text retrieval based math search systems follow more or less the same principles, we will only review a few representative math search systems in this section. In the MathFind system [19], Munavalli and Miner encoded math expressions into a sequence of textual data called math fragments. In this approach, math expressions are preprocessed and converted into the corresponding *MathML* format [2]. Similarly, Nguyen et al. [21] extracted math features from *MathML* and used Formal Concept Analysis as the formula search mechanism. To index math expressions and support sub-expression search, Miner and Munavalli [18] defined math n-grams, where each math token is considered a *1*-gram. For instance, "a+b" has three 1-grams, i.e., 'a', 'b', and '+'. To search for math expressions, a query is decomposed into n-grams. The indexes are then searched for each n-gram in the query. Similarly, Libbrecht and Melis [15] proposed the ActiveMath Search Tool, which was built on top of Apache Lucene[7]. In the ActiveMath Search Tool, math data are represented in the OMDoc [13] format. They are subsequently converted into tokens and indexed using the Lucene text search engine.

Wolfram Formula Search[8] is another math search system that was developed by Wolfram Research. Wolfram Formula Search allows users to search for formulas from its formula compendium of math functions via the Web. Its database contains more than three hundred thousand formulas classified into 14 categories. The formula search engine provides a friendly interface for users to explore the vast collection of formulas. However, although Wolfram Formula Search engine provides semantic search, it only supports search queries based on predefined constants, operations, and function names. For example, it understands common constants like e, π, etc., and operations including sum, product, etc., appearing anywhere in a formula.

For the past few years, many learning-to-rank algorithms have been proposed to improve the ranking of search results. Some adapted binary classification algorithms such as Perceptron and its variants [4]. In [9], Gao et al. applied the Perceptron algorithm to learn the ranking function. Although the Perceptron algorithm is elegant and fast, it performs poorly on datasets that are not linearly separable. To overcome this limitation, Crammer et al. [7] proposed the Passive Aggressive (PA) algorithm, which uses large margin optimizations. Improved versions of PA include Confidence-Weighted (CW) Linear Classification [8] and Passive-Aggressive Mahalanobis (PAM) [20]. In the

[4]http://www.codecogs.com/latex/about.php
[5]http://functions.wolfram.com/formulasearch
[6]http://shinh.org/wfs/
[7]http://lucene.apache.org
[8]http://functions.wolfram.com/formulasearch

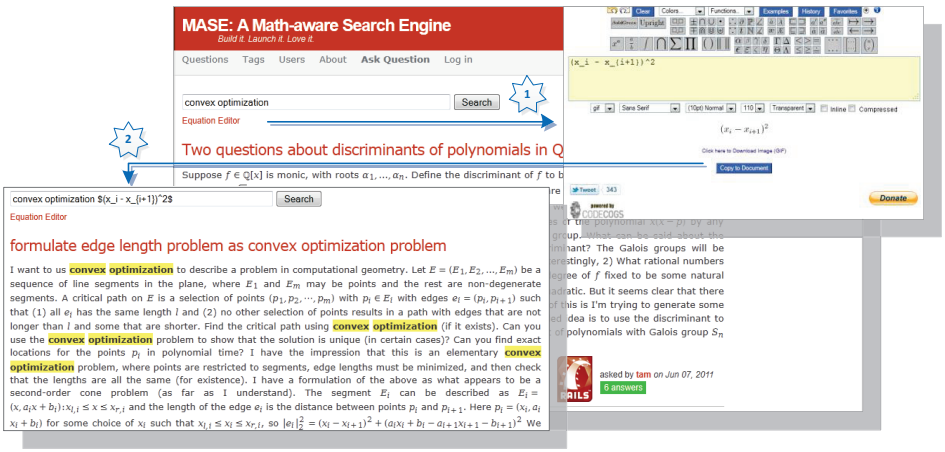

Figure 2: Math-Aware Search Engine.

realm of batch learning, the SVM algorithm [6, 26] has also been applied to the ranking problem. However, for dynamic systems like math Q&A systems, where math questions are posted every minute, an online learning algorithm is more desirable as it can incrementally update itself whenever new data is added. Therefore, in this research, we extend and apply the PA algorithm to learn the document ranking function.

3. MATH DOCUMENT PREPROCESSING

Math documents can be significantly more complex than textual documents; it can involve math expressions that are sequence sensitive in two dimensions, e.g., matrices, fractions, nested forms. Conventional text retrieval methods, which typically throw away the linear sequence information in text, are thus extremely poorly suited to process math documents. Therefore, new retrieval methods must be devised to effectively handle math documents. We introduce our math document format and feature extraction approach in this section.

3.1 Math Document Representation

Mathematical markup languages play an important role in the development of math search systems. It affects the storage and techniques for the processing of math expressions. There are several common mathematical markup languages: LaTeX, *ASCIIMath*[9], *OMDoc* [13], *OpenMath* [5] and *MathML* [2]. The content-oriented LaTeX markup language has been commonly used by many researchers, especially mathematicians, for document processing. ASCIIMath markup is a simpler variant of the LaTeX markup language [25], which is often used as a format for specifying math expression queries for math-aware search systems. Math expressions represented using ASCIIMath markup are easy to store as it mainly uses *ASCII* characters to represent the math expressions. However, the disadvantage of ASCIIMath is that its data representation is not well-structured, thereby requiring additional processing time to parse and extract math expressions. OMDoc and OpenMath are two other less popular standards for the semantic-oriented representation of math or expressions.

Among the various format, the MathML markup of W3C is the most popular, and is currently used by many math search engines for the presentation and storage of math expressions [15, 17, 19]. There are two variants of MathML: MathML content markup and MathML presentation markup. Among the two markup representations, MathML content markup is commonly used for the processing of math expressions as it contains richer semantic content than the MathML presentation markup.

Despite the advantages of MathML, our system uses the LaTeX markup as the storage and user interface language for entering queries. This is because LaTeX markup has a significantly smaller storage footprint than MathML. More importantly, LaTeX markup has a large user base and is more intuitive for users to enter arbitrary math expressions online. In fact, open source Javascript display engines like MathJax[10] and ASCIIMath can parse LaTeX markups and render the corresponding graphical math expressions on the web browser.

Moreover, during the feature extraction step, we transform our LaTeX queries and documents into MathML, which are easier for machines to analyze. Listing 1 shows an example question in LaTeX format, comprising the title and content of the example question from Section 1.

Listing 1: An Example Question in LaTeXformat.

```
Formulate edge length problem as convex
    optimization problem

I want to use convex optimization to describe a
    problem in computational geometry. Let $E = (E
    _1, E_2,\ldots , E_m) $ be a sequence of line
    segments in the plane, where $E_1$ and $E_m$
    may be points and the rest are non-degenerate
    segments. A \emph{critical path} on $E$ is a
    selection of points $(p_1,p_2,\cdots ,p_m)$
    with $p_i\in E_i$ with edges $e_i=(p_i ,p_{i
    +1})$ such that (1) all $e_i$ has the same
    length $l$ and (2) no other selection of
    points results in a path with edges that are
    not longer than $l$ and some that are shorter
    ...
```

[9]http://www1.chapman.edu/~jipsen/mathml/asciimath.html

[10]http://www.mathjax.org

3.2 Math Feature Extraction

Due to the existence of both semantic and structural information, the preprocessing step for math expressions is more complex than that of text documents. For the purpose of math search, math features should be representative enough to reflect the underlying characteristics of each math expression. We thus extract math features as follows:

- Content MathML conversion. Convert math expressions into Content MathML format.
- Math feature extraction. Extract math features by traversing the Content MathML tree.

To convert math expressions from LaTeX, we use the *SnuggleTeX* library[11]. We first convert math expressions from LaTeX to the representation MathML format, then we use cascading stylesheets to map the representation MathML to content MathML. Listing 2 shows an example of the content MathML for the math expression $(x+y)^2$.

Once the expression is converted into content MathML, we use XML tree traversal to extract the math features. In this paper, we only use two kinds of features, namely single and combination features. Single features are used to express constant numbers, variable names, function names, etc. On the other hand, combination features refer to combinations of math operators and operands.

Listing 2: MathML Content Markup of $(x+y)^2$.

```
<apply>                 % apply operator
  <power/>              % power operator
  <mfence>
    <apply>             % apply operator
      <plus/>           % plus operator
      <ci>x</ci>        % variable (ci) x
      <ci>y</ci>        % variable (ci) y
    </apply>
  </mfence>
  <cn>2</cn>            % constant (cn) 2
</apply>
```

Take the sub-expression $x+y$ in Listing 2 as an example, based on the content MathML data, we have four single features, namely *ci*, *plus*, *cix*, and *ciy*, where *ci* stands for a variable and *cix* stands for a variable named x. We also have one combination feature *pluscixciy*, where *plus* stands for the operator + and *pluscixciy* stands for operator + applied to two operands x and y.

4. DATA MODEL

In this section, we define two types of index terms, namely text and math. These two types of index terms collectively form our math-aware search engine.

Definition 1 (Text Index Term). *Let n be the number of index terms in the system. Denote each text index term by t_i, and the set $T = \{t_1, \ldots, t_n\}$ contains all text index terms.*

Definition 2 (Math Index Term). *Let m be the number of math index terms in the system. Each math index term m_j is an element of the math feature set. Let $M = \{m_1, \ldots, m_m\}$ be the set of all math index terms.*

[11]http://www2.ph.ed.ac.uk/snuggletex

Let $\mathbb{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_k\}$ and $\mathbb{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_l\}$ be a collection of k math documents and a set of l user queries, respectively. The math-aware IR problem aims to determine a *ranking function* score : $\mathbb{D} \times \mathbb{Q} \mapsto \mathbb{R}$, which defines an ordering among documents based on the queries via a *framework* F. This framework considers all math documents and queries. For example, in text IR, the framework F is typically the vector space of document and query points, with the ranking function determined by cosine similarity.

In this section, we formally define the query and ranking models for math Q&A search based on the vector space model. The proposed approach is an extension of the vector space model for conventional text IR.

4.1 Matching Model

In this section, we formally define math documents and queries in the math-ware search engine, where the data consists of both text and math expressions. The search engine should index both types of data simultaneously.

Definition 3 (Math Predicate). *A math predicate is defined as p_i if the math index term m_i appears in a document $\mathbf{d}_j$. In other words, $\mathbf{d}_j$ satisfies p_i if $\mathbf{d}_j$ contains the math index term m_i.*

Definition 4 (Math Specification). *A math specification $P = p_1 \wedge p_2 \wedge \ldots \wedge p_s$ specifies a subset of documents $\mathbb{D}_P \subset \mathbb{D}$, where $\mathbf{d}_i \in \mathbb{D}_P$, if $\mathbf{d}_i$ satisfies P.*

Definition 5 (Math Document). *A math document $\mathbf{d}_j \in \mathbb{D}$ is determined by a vector $\mathbf{d}_j = (d_{1j}, \ldots d_{nj})$ and a math specification P_j, where $d_{ij} \geq 0$. If the text index term t_i appears in document $\mathbf{d}_j$, then $d_{ij} > 0$; otherwise, $d_{ij} = 0$. Thus P_j is the corresponding math specification satisfied by document $\mathbf{d}_j$.*

Definition 6 (Math Query). *A query in a math question answering system, $\mathbf{q} = \mathbf{q}_t|\mathbf{q}_m$, is comprised of two parts: a keyword query $\mathbf{q}_t \subset T$ and a math expression query $\mathbf{q}_m \subset M$. The keyword query $\mathbf{q}_t$ is a list of keywords and the math expression query $\mathbf{q}_m$ is a list of math expressions.*

For the math query $\mathbf{q} = \mathbf{q}_t|\mathbf{q}_m$, suppose that $\mathbf{q}_t = \{t_1, \ldots, t_i\}$ and $\mathbf{q}_m = \{m_1, \ldots, m_j\}$ corresponds to math specification P, the unranked result $\mathbf{q}(\mathbb{D})$ denotes the subset of documents that satisfies P, and those that contain all the terms in $\mathbf{q}_t$ as follows:

$$\mathbf{q}(\mathbb{D}) = \sigma_P(\mathbb{D}) \cap \sigma_{t_1}(\mathbb{D}) \cap \ldots \cap \sigma_{t_i}(\mathbb{D})$$

where the selection function $\sigma_{t_i}(\mathbb{D}) = \{\mathbf{d}_j | t_i \in \mathbf{d}_j \wedge \mathbf{d}_j \in \mathbb{D}\}$.

4.2 Ranking Model

4.2.1 A Straightforward Approach

Now, we define the ranking of the result of $\mathbf{q}$. We start by presenting a generic representation of the conventional ranking function for keyword queries. We then extend it into the math document ranking function.

Given a query $\mathbf{q}$ and a document $\mathbf{d} \in \mathbb{D}$, a conventional ranking function $f(\cdot)$ takes as argument statistics from $\mathbf{q}$, $\mathbf{d}$, and computes a score of $\mathbf{d}$ with respect to $\mathbf{q}$ as follows:

$$\text{score}(\mathbf{q}, \mathbf{d}) = f(\mathbf{q}_t, \mathbf{q}_m, \mathbf{d})$$

For example, $tf \times idf$ weighting [16, 23] is a well-known ranking model. Among its variants, the pivoted normalization formula is one of the best performing vector space models, and is widely used in many text search systems. Specifically, it is defined as

$$\text{score}(\mathbf{q}, \mathbf{d}) = \sum_{t \in \mathbf{q}} \frac{1 + \ln(1 + tf(t, \mathbf{d}))}{(1 - s) + s \cdot \frac{len(\mathbf{d})}{|\mathbb{D}|}} \cdot tf(t, \mathbf{q}) \cdot ln \frac{|\mathbb{D}| + 1}{df(t, \mathbb{D})}$$

where $len(\mathbf{d})$ is the length of document $\mathbf{d}$, $|\cdot|$ denotes the set cardinality function, $tf(t, \mathbf{d})$ is the term frequency of term t in document $\mathbf{d}$, $df(t, \mathbb{D})$ is the document frequency of term t in the document collection $\mathbb{D}$, and s is a parameter.

We note that the scoring function depends on the number of documents. Therefore, we revise this scoring function for ranking math document search results as follows:

$$\text{score}(\mathbf{q}, \mathbf{d}) = \sum_{t \in \mathbf{q}} \frac{1 + \ln(1 + tf(t, \mathbf{d}))}{(1 - s) + s \cdot \frac{len(\mathbf{d})}{|\mathbb{D}_P|}} \cdot tf(t, \mathbf{q}) \cdot ln \frac{|\mathbb{D}_P| + 1}{df(t, \mathbb{D}_P)}$$

4.2.2 Ranking Passive Aggressive Algorithm

In this section, we describe the Ranking Passive Aggressive algorithm in details based on the Passive Aggressive algorithm [7]. Let $(\mathbf{x}_t, y_t)_{t=1,T}$ be a sequence of examples, Crammer et al. proposed the PA algorithm, which learns the weight $\mathbf{w}$ of a linear prediction function $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ based on the following optimization problem:

$$\begin{aligned} \mathbf{w}_{t+1} = \quad & \underset{\mathbf{w} \in \mathbb{R}^n}{\text{argmin}} \quad \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 \\ & \text{s.t.} \qquad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0 \end{aligned} \tag{1}$$

where ℓ is the Hinge loss function.

We adapt the Passive Aggressive algorithm to learn-to-rank by training on pairs of instances. A correct classification corresponds to ranking a pair so that the more relevant instance is scored higher and vice versa.

We know that math and text features are extracted from different data types and therefore should be treated differently. We can apply the cascade ranking model [24], where one type of data is used to rank documents at each stage of a multi-stage ranking procedure. However, in practice, this approach may not be practical because math questions do not always contain math expressions. Therefore, we propose another approach that combines math and text data as follows. Let $\mathbf{a}_t \in \mathbb{R}^k$ and $\mathbf{b}_t \in \mathbb{R}^l$ be the math and text vectors representing the document $\mathbf{d}_t \in \mathbb{R}^n$, respectively, where $k + l = n$. The simplest way is to concatenate the math and text data as follows:

$$\mathbf{d}_t = (\eta \mathbf{a}_t, (1 - \eta)\mathbf{b}_t)$$

where $\eta \in [0, 1]$ weighs the relative importance of the text and math data.

Remark $\eta = 0$ means that we ignore math expressions in the math document. If $\eta = 1$, we only rank the search result based on the math expressions and ignore the text data. Normally, η will be assigned values between 0 and 1. If we treat math and text data as equals, then $\eta = 0.5$. If $\eta < 0.5$, we consider text data to be more important than math data, and vice versa.

Let the feature score vector of a document $\mathbf{d}_t \in \mathbb{R}^n$ and query $\mathbf{q}$ be defined as follows:

$$\phi_t = (f_1(\mathbf{d}_t, \mathbf{q}), \ldots, f_n(\mathbf{d}_t, \mathbf{q}))$$

The goal is to learn the weight vector $\mathbf{w}$ of the following scoring function.

$$\text{score}(\phi_t, \mathbf{w}) = \mathbf{w} \cdot \phi_t \tag{2}$$

Suppose that R and N denote relevant and non-relevant feature score vectors. Then the feature score vectors of relevant and non-relevant documents to the query $\mathbf{q}$ are defined as ϕ_t^R and ϕ_t^N, respectively. Let $(\phi_t^R, \phi_t^N)_{t=1,T}$ be a sequence of training examples. We should have $\mathbf{w} \cdot \phi_t^R \geq \mathbf{w} \cdot \phi_t^N$ because the score of the relevant document to the query should be greater than that of the non-relevant document. For the Ranking Perceptron [9], the weight vector $\mathbf{w}_{t+1}$ is learnt on each round t as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau(\phi_t^R - \phi_t^N)$$

where τ is a constant learning rate. Similar to the Perceptron algorithm, the Ranking Perceptron algorithm is very sensitive to the learning rate, and also works poorly on non-linearly separable data.

To overcome this problem, we introduce a new Hinge loss function,

$$\ell((\phi_t^R, \phi_t^N); \mathbf{w}) = \max\left\{0, 1 - (\mathbf{w} \cdot \phi_t^R - \mathbf{w} \cdot \phi_t^N)\right\},$$

which penalizes the difference of the two scores if the score of the non-relevant document is greater than that of the relevant document, with respect to the query. We then derive a soft margin optimization problem by introducing a slack variable ξ as follows:

$$\begin{aligned} \mathbf{w}_{t+1} = \quad & \underset{\mathbf{w} \in \mathbb{R}^n \wedge \mathbf{w} \geq 0}{argmin} \quad \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 + C\xi \\ \text{s.t.} \quad & 1 - (\mathbf{w} \cdot \phi_t^R - \mathbf{w} \cdot \phi_t^N) \leq \xi; \quad \xi \geq 0 \end{aligned} \tag{3}$$

where C is an aggressiveness parameter.

PROPOSITION 1. *The optimization problem (3) has the closed-form solution as follows:*

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t(\phi_t^R - \phi_t^N)$$

where the learning rate τ_t has the form

$$\tau_t = \min\left\{C, \frac{1 - (\mathbf{w}_t \cdot \phi_t^R - \mathbf{w}_t \cdot \phi_t^N)}{(\phi_t^R - \phi_t^N)^2}\right\}.$$

Algorithm 1 summarizes the *Ranking Passive Aggressive* algorithm.

Algorithm 1 Ranking Passive Aggressive Algorithm

Input:
C = positive aggressiveness parameter
Output:
None
Process:
Initialize $\mathbf{w}_1 \leftarrow \mathbf{0}$;
for $t = 1, 2, \ldots$ **do**
  Receive feature score vectors $\phi_t^R \in \mathbb{R}^n$ and $\phi_t^N \in \mathbb{R}^n$
  Suffer loss $\ell_t \leftarrow \max\left\{0, 1 - (\mathbf{w}_t \cdot \phi_t^R - \mathbf{w}_t \cdot \phi_t^N)\right\}$
  if $\ell_t > 0$ **then**
    Set $\tau_t \leftarrow \min\left\{C, \frac{1 - (\mathbf{w}_t \cdot \phi_t^R - \mathbf{w}_t \cdot \phi_t^N)}{(\phi_t^R - \phi_t^N)^2}\right\}$
    Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \tau_t(\phi_t^R - \phi_t^N)$
  end if
end for

Let ϕ_t^R and ϕ_t^N be the feature score vectors of the relevant and non-relevant documents for query $\mathbf{q}$, respectively. Let $\mathbf{w}_t$ be the weight vector of the Ranking Passive Aggressive algorithm on round t. We say that the proposed algorithm makes a mistake if $\mathbf{w}_t \cdot \phi_t^R < \mathbf{w}_t \cdot \phi_t^N$. The total number of mistakes made by the proposed algorithm is bounded as shown in Preposition 2. The proofs of Prepositions 1 and 2 are given in the Appendix.

PROPOSITION 2. *Let $(\phi_t^R, \phi_t^N)_{t=1,T}$ be a sequence of examples, where $\phi_t^R \in \mathbb{R}^n$ and $\phi_t^N \in \mathbb{R}^n$ for all t. Then for any weight vector $\mathbf{u} \in \mathbb{R}^n$ and its loss $\ell_t^* = \ell((\phi_t^R, \phi_t^N); \mathbf{u})$, the number of mistakes made by the Ranking Passive Aggressive algorithm is bounded by*

$$\max\Big\{\frac{1}{C}, \max_{t=1,T}(\phi_t^R - \phi_t^N)^2\Big\}\Big(\parallel \mathbf{u} \parallel^2 + 2C\sum_{t=1}^{T}\ell_t^*\Big)$$

4.2.3 Adding Metadata to Math Question Retrieval

In math question answering system, math questions typically contains additional metadata such as user rating, number of views, number of answers, etc. The metadata can be very useful for improving ranking results, faceted search, and personalization [3]. The question is how to combine the document content and its metadata to answer a user's information need. To address this, we have to first choose a suitable set of metadata. For instance, user may prefer to see the most popular and/or answered questions. Moreover, he could also be looking for questions that have already been answered by other users. In each of these two scenarios, the number of views and answers are useful information for re-ranking the search results.

To retrieve documents based on user ratings, Zhang et al. [27] proposed a simple IR technique, which considers thumbs-up and thumbs-down as a term in the document. They came up with a scoring function, which is the ratio of the number of thumbs-up (n^+) to the number of thumbs-down (n^-). However, in practice systems like Math Overflow only store the difference ($n^+ - n^-$) between these two user ratings. Therefore, the above technique cannot be directly applied. In order to overcome this limitation and consider other kinds of metadata, we propose another IR approach, which is based on the $tf \times idf$ ranking model.

Since users want to retrieve "good" math questions, e.g., questions with thumbs-up, questions with many views, questions with several answers, questions with highly-rated answers, and questions with high reputation, we can embed this need into a virtual query $\mathbf{v}$ comprising metadata. The scoring function of this query is defined as follows:

$$score(\mathbf{d}_t, \mathbf{v}) = \sum_{w \in \mathbf{v}} \frac{1 + \ln(1 + \mathbf{m}_t(w))}{(1-s) - s\frac{len(\mathbf{m}_t)}{avg(len(\mathbf{m}_t))}} \quad (4)$$

where $\mathbf{m}_t$ is the metadata of document $\mathbf{d}_t$.

Let the scoring function between document $\mathbf{d}_t$ and query $\mathbf{q}$ be defined as in Equation (2). The overall scoring function is then defined as the linear combination of the query and virtual-query scoring functions, $score(\mathbf{d}_t, \mathbf{q})$ and $score(\mathbf{d}_t, \mathbf{v})$, as

$$score(\mathbf{d}_t, \mathbf{q}, \mathbf{v}) = \alpha \cdot score(\mathbf{d}_t, \mathbf{q}) + (1-\alpha) \cdot score(\mathbf{d}_t, \mathbf{v}) \quad (5)$$

where $\alpha \in [0, 1]$ determines the relative importance of content and metadata for document $\mathbf{d}_t$. The advantage of this approach is that we are able to determine the metadata score beforehand and update it whenever users update the metadata.

5. EXPERIMENTAL RESULTS

5.1 Experimental Testbed

The major obstacle to evaluating math search is that we do not have a standard benchmark dataset like for other more common IR tasks. Also, it is hard to compare our proposed approach with other systems such as MathFind and Wolfram Function Search because they are either unavailable or inaccessible.

Therefore, we build our own math search dataset by crawling and downloading 31,288 math questions and answers (or math documents for short) from the Math Overflow online question answering system. Each math document can have one or more of the 20 categorical label. We parsed and converted the math documents from HTML into the *Text REtrieval Conference (TREC)*[12] XML format. We also manually generated 30 queries in the TREC format. Each query has a set of relevant and non-relevant documents deterimined manually.

5.2 Experimental Setup

Each document is identified by a numerical document ID. All math documents are indexed using standard inverted index techniques with stop-word removal based on the English stop-word list provided by *Terrier Information Retrieval Platform* [22]. The total number of index terms is 29,745.

To learn the ranking function, all math documents and queries are processed and converted into $tf \times idf$ [16] vectors. The feature score vectors are calculated based on these documents and queries. The feature score vectors (consisting of more than 1000 vectors) are used to train the Ranking Perceptron and Ranking PA algorithms.

After training with cross validation, we obtained an optimal weight vector $\mathbf{w}$, which is used to determine the scores of all retrieved math documents. Since other math-aware search systems only focus on math feature extraction and apply classical IR techniques, we only compared our approach against standard retrieval methods such as BM25 [11], InL2, and $tf \times idf$, where InL2 stands for "Inverse Document Frequency with Laplace after-effect and normalization 2" [10]. While BM25 and $tf \times idf$ are established IR techniques, the InL2 model was only proposed recently. For the $tf \times idf$ ranking model, we applied the straightforward approach describe in Section 4.2.1.

5.3 Math Feature Extraction Evaluation

In this section, we evaluate the proposed math feature extraction approach by comparing the retrieval precision on the math expressions with and without math feature extraction. The procedure is as follows:

- Extract all math expressions from the Math Overflow dataset to create a math-only dataset, called Math Expression dataset.
- Generate a textual math feature set, where the math expressions are treated as normal text.

[12]http://trec.nist.gov/

Table 1: Textual Math and Math Feature Properties.

	Textual Math	Math Feature	N-Gram
# Unique Terms	2959	6060	7145
Avg. DocLen	40.25	13.46	65.55
Avg. PLLen	25.53	8.63	28.01
Avg. QT (ms)	62.20	53.80	78.80

- Generate a math feature set, where math features are extracted as described in Section 3.2.
- Generate an n-gram feature set as described in [18].
- Conduct the experiments on these three feature sets and compare their performances.

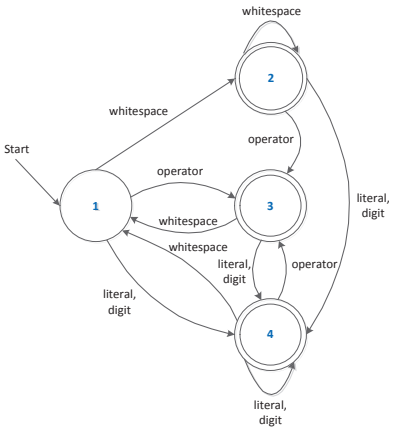

Figure 3: FSM for extracting Textual Math Features.

A finite state machine (FSM) as shown in Figure 3 is used to extract textual math features, where whitespace stands for invisible characters, end-of-stream characters, and delimiters such as space, tab, comma, etc. Operator refers to common math operators such as plus, minus, power, etc. Literal includes characters from the set 'a' to 'z' and 'A' to 'Z'. Digit refers to numbers '0' to '9'. The FSM accepts a math expression as a sequence of characters, on which it extracts math operators, math variables, function names, and constant numbers. Specifically, at state 2, it skips a whitespace. At state 3, it extracts the operator. Upon termination at state 4, it returns a variable, a function name, or a constant number.

The properties of all three feature sets are summarize in Table 1. We counted the number of unique terms and calculated the average *document length* (DocLen) and average *posting list length* (PLLen), where the document length is determined based on the number of index terms in the document (i.e., math expressions). While the number of unique terms represents the size of the *lexicon*, the average posting list length indicates the number of documents associated with a term in the inverted index. These two major factors impact the performance of a search system in terms of *query time* (QT) and accuracy.

Note that the total number of unique terms in the math feature set is greater than that of the textual math feature set. However, the average document length and posting list length of the math feature set are smaller than those of the textual math feature set. This implies that the math feature set is less uniform and therefore more discriminative than the textual math feature set.

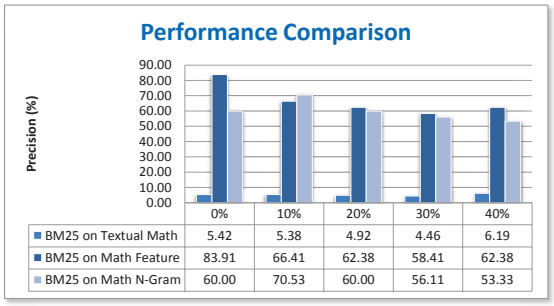

Figure 4: Performance Comparison on the Math Expression Dataset.

To verify the above claim, we carried out a retrieval experiment and compared the IR performance on both feature sets. Figure 4 shows the precision at different recall values on the three feature sets. In this experiment, the popular ranking method BM25 was applied. We see that BM25 does not work well on the textual math dataset. Its max precision on this set is 6% at 40% recall. On the other hand, its precision on the math feature set and n-gram set is significantly larger. Even in the worst case, the math feature set's precision is improved by more than 53%. This result confirms that math feature extraction is essential for an effective math-aware search engine, compared to the simplistic textual math set. Besides, the math feature set gave better precision than n-gram for all levels of recall greater than 10%, which makes it the practical feature choice.

5.4 Ranking Algorithm Evaluation

We trained both the Ranking Perceptron and Ranking PA algorithms on the feature score vectors. We then compared their performance by using the standard cumulative error rate, which is the ratio of mistakes made by the online learning algorithm over the total number of examples observed to-date. In this experiment, the performance of the Ranking PA algorithm and the Ranking Perceptron algorithm [9] is evaluated on the Math Overflow dataset.

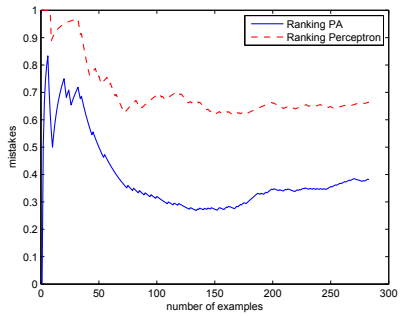

Figure 5: Cumulative Error Rate for Math Overflow Datatset.

Figure 5 shows the cumulative error rates of both the

Table 2: Metadata Properties (* using Equation (4)).

	Rate	#Ans	#Views	Ans Rate	Score*
Min	-3	0	14	-4	2.9215
Max	93	121	21606	86	27.2277
Avg	5.9049	1.9146	498.2342	5.8395	14.8662

Ranking Perceptron and Ranking PA algorithms on the Math Overflow dataset. Compared to the Ranking Perceptron algorithm, the proposed Ranking PA algorithm consistently achieved a lower cumulative error rate than the Ranking Perceptron algorithm. This shows that the proposed algorithm is effective in improving the online ranking function learning. Hence, in the ranking process, if the weight of the Ranking PA is utilized, the retrieval precision will be improved compared to that of the Ranking Perceptron algorithm. More experimental results shall confirm this claim in the next section.

5.5 Retrieval Performance Evaluation

For the Ranking PA algorithm, we used cross validation to choose the optimal parameter C. Then, we used the optimal solution/weight to rank retrieved documents based on the scoring function (5) of Section 4.2.3. We also evaluated the Ranking Perceptron algorithm. In this experiment, we used both text and math feature sets, where the text features were extracted using an English tokenizer with stop-word removal and the math features were extracted as described in Section 3.2. Moreover, the metadata of each question is extracted to calculate the score. We determined the optimal trade-off parameter α in Equation (5) to be around 0.958. This means that content largely rules over metadata. The properties of the metadata of all questions are shown in Table 2.

Figure 6 shows the performance comparison between our proposed approach and other methods. The retrieval precision of each method is compared at recall levels of 0%, 10%, and 20%. The experimental results show that the Ranking PA algorithm consistently outperformed other methods. Its precision is at least 9% better than other methods at 0% and 10% recall levels. In the worst case, the improvement is more than 7% at 20% recall. We did not further evaluate higher recall levels because the precision have fallen to way less than 50% to be of practical significance.

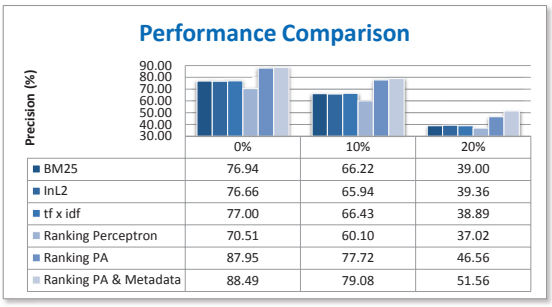

Figure 6: Performance Comparison on the Math Overflow Dataset.

Compared with other methods, the straightforward approach $tf \times idf$ is slightly better. The improvement is nearly 1%. Hence, term weighting method alone is not good enough to rank highly structured data such as math expressions. In this case, the learning-to-rank algorithm should be applied to improve the performance of the search system. In this experiment, the Ranking PA has been shown to be an elegant solution for this problem. If we take into account the metadata, the performance is just marginally improved. This improvement is nearly 5% at 20% recall but declines gradually with decreasing recall levels.

6. CONCLUSION

In this paper, we proposed a new approach to math-aware search engine, which consists of two major steps: feature extraction and learning-to-rank. In feature extraction, we proposed a new method for math expression feature extraction based on the content MathML format. To learn the score function, we formulated and derived an online optimization problem. We then derive a closed-form solution, leading to our Ranking PA algorithm. In addition, we derived the mistake bound of the Ranking PA algorithm.

Moreover, we crawled the Math Overflow question answering system and generated a new dataset for math document retrieval. Our Math Overflow dataset is a big dataset, which has more than 30,000 math documents. It is very useful for math document retrieval evaluation in general and can be downloaded from project.mosuma.net. We evaluated our proposed ranking approach and benchmarked it against other baseline techniques on the Math Overflow dataset. The experimental results show that the proposed approach performs better than the runner-up by more than 9%. Although we evaluated our proposed approach on the math question answering dataset, it is possible to apply it to other kinds of math document retrieval.

Acknowledgements

This research was supported in part by Singapore Ministry of Education's Academic Research Fund Tier 2 grant ARC 9/12 (MOE2011-T2-2-056).

7. REFERENCES

[1] A. Andrea, G. Ferruccio, C. C. Sacerdoti, T. Enrico, and Z. Stefano. A content based mathematical search engine: Whelp. In *TYPES*, pages 17–32, 2004.

[2] R. Ausbrooks, S. Buswell, S. Dalmas, S. Devitt, A. Diaz, R. Hunter, B. Smith, N. Soiffer, R. Sutor, and S. Watt. Mathematical markup language (mathml) version 2.0, 2000.

[3] P. N. Bennett, K. El-Arini, T. Joachims, and K. M. Svore. Sigir '11 workshop report: Enriching information retrieval, 2011.

[4] H. Block. The perceptron: A model for brain functioning. *Rev. Modern Phys.*, 34:123–135, 1962.

[5] S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaetano, and M. Kohlhase. *The Open Math standard version 2.0*. 2004.

[6] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 186–193, New York, NY, USA, 2006. ACM.

[7] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, pages 551–585, 2006.

[8] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 264–271, New York, NY, USA, 2008. ACM.

[9] J. Gao, H. Qi, X. Xia, and J.-Y. Nie. Linear discriminant model for information retrieval. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 290–297, New York, NY, USA, 2005. ACM.

[10] R. Guillén. Gir with language modeling and dfr using terrier. In *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access*, CLEF'08, pages 822–829, Berlin, Heidelberg, 2009. Springer-Verlag.

[11] K. S. Jones. Index term weighting. *Information Storage and Retrieval*, 9(11):619 – 633, 1973.

[12] A. Kohlhase and M. Kohlhase. Reexamining the mkm value proposition: From math web search to math web research. In *Calculemus '07 / MKM '07: Proceedings of the 14th symposium on Towards Mechanized Mathematical Assistants*, pages 313–326, Berlin, Heidelberg, 2007. Springer-Verlag.

[13] M. Kohlhase and I. Sucan. A search engine for mathematical formulae. In J. Calmet, T. Ida, and D. Wang, editors, *AISC '06: Proceedings of 8th International Conference on Artificial Intelligence and Symbolic Computation*, pages 241–253. Springer-Verlag, 2006.

[14] M. Kohlhase and I. A. Sÿucan. A search engine for mathematical formulae. In *Proc. of Artificial Intelligence and Symbolic Computation, number 4120 in LNAI*, pages 241–253. Springer, 2006.

[15] P. Libbrecht and E. Melis. Methods to access and retrieve mathematical content and activemath. In *ICMS '06: In Proceeding of the 2nd International Congress on Mathematical Software*, 2006.

[16] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[17] B. R. Miller and A. Youssef. Augmenting presentation mathml for search. *MKM '08: Proceedings of the 7th International Conference on Mathematical Knowledge Management*, pages 536–542, 2008.

[18] R. Miner and R. Munavalli. An approach to mathematical search through query formulation and data normalization. In *Calculemus '07 / MKM '07: Proceedings of the 14th symposium on Towards Mechanized Mathematical Assistants*, pages 342–355, Berlin, Heidelberg, 2007. Springer-Verlag.

[19] R. Munavalli and R. Miner. Mathfind: a math-aware search engine. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 735–735, New York, NY, USA, 2006. ACM.

[20] T. T. Nguyen, K. Chang, and S. C. Hui. Distribution-aware online classifiers. In T. Walsh, editor, *IJCAI*, pages 1427–1432. IJCAI/AAAI, 2011.

[21] T. T. Nguyen, S. C. Hui, and K. Chang. A lattice-based approach for mathematical search using formal concept analysis. *Expert Systems with Applications*, 2011.

[22] I. Ounis, G. Amati, V. Plachouras, B. He, C. MacDonald, and D. Johnson. Terrier information retrieval platform. *Advances in Information Retrieval*, 3408:517–519, 2005.

[23] A. Singhal. Modern information retrieval: a brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24:2001, 2001.

[24] L. Wang, J. Lin, and D. Metzler. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 105–114, New York, NY, USA, 2011. ACM.

[25] A. S. Youssef. Roles of math search in mathematics. In J. M. Borwein and W. M. Farmer, editors, *MKM '06: Proceedings of the 5th International Conference on Mathematical Knowledge Management*, pages 2–16, Berlin Heidelberg, 2006. Springer-Verlag.

[26] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 271–278, New York, NY, USA, 2007. ACM.

[27] D. Zhang, R. Mao, H. Li, and J. Mao. How to count thumb-ups and thumb-downs?: an information retrieval approach to user-rating based ranking of items. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 1223–1224, New York, NY, USA, 2011. ACM.

APPENDIX

Here, we prove Proposition 1 as follows:

PROOF. We define the Lagrangian of the optimization problem as follows:

$$\begin{aligned}\mathcal{L} = & \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 + C\xi - \lambda\xi \\ & + \tau(1 - \xi - (\mathbf{w} \cdot \phi_t^R - \mathbf{w} \cdot \phi_t^N)) \\ = & \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 + (C - \lambda - \tau)\xi \\ & + \tau(1 - (\mathbf{w} \cdot \phi_t^R - \mathbf{w} \cdot \phi_t^N)\end{aligned} \tag{6}$$

where $\lambda \geq 0$ and $\tau \geq 0$ are the Lagrangian multipliers.

Setting the partial derivatives of $\mathcal{L}$ with respect to the weight $\mathbf{w}$ to zero, we have,

$$\begin{aligned}0 = \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \mathbf{w}_t - \tau(\phi_t^R - \phi_t^N) \\ \Rightarrow \mathbf{w} = \mathbf{w}_t + \tau(\phi_t^R - \phi_t^N)\end{aligned} \tag{7}$$

Setting the partial derivatives of $\mathcal{L}$ with respect to weight ξ to zero, we have,

$$0 = \frac{\partial \mathcal{L}}{\partial \xi} = (C - \lambda - \tau) \Rightarrow \lambda + \tau = C \tag{8}$$

Substituting (7) and (8) into (6), we have,

$$\mathcal{L} = \quad -\frac{1}{2}\tau^2(\phi_t^R - \phi_t^N)^2 + \tau(1 - (\mathbf{w}_t \cdot \phi_t^R - \mathbf{w}_t \cdot \phi_t^N)) \tag{9}$$

Setting the partial derivatives of $\mathcal{L}$ with respect to weight τ to zero, we have,

$$0 = \frac{\partial \mathcal{L}}{\partial \tau} = -\tau(\phi_t^R - \phi_t^N)^2 + 1 - (\mathbf{w}_t \cdot \phi_t^R - \mathbf{w}_t \cdot \phi_t^N)$$

$$\Rightarrow \tau = \frac{1 - (\mathbf{w}_t \cdot \phi_t^R - \mathbf{w}_t \cdot \phi_t^N)}{(\phi_t^R - \phi_t^N)^2}$$

We have $\lambda + \tau = C$ and $\lambda \geq 0$. Therefore, we can conclude that $0 \leq \tau \leq C$, which concludes the proof. □

Next, we prove Proposition 2 as follows:

Proof. We have

$$\begin{aligned} &\sum_{t=1}^{T} \| \mathbf{w}_t - \mathbf{u} \|^2 - \| \mathbf{w}_{t+1} - \mathbf{u} \|^2 \\ =&\| \mathbf{w}_1 - \mathbf{u} \|^2 - \| \mathbf{w}_{T+1} - \mathbf{u} \|^2 \\ =&\| \mathbf{u} \|^2 - \| \mathbf{w}_{t+1} - \mathbf{u} \|^2 \leq \| \mathbf{u} \|^2 \end{aligned} \tag{10}$$

We also have

$$\begin{aligned} &\| \mathbf{w}_t - \mathbf{u} \|^2 - \| \mathbf{w}_{t+1} - \mathbf{u} \|^2 \\ =&\| \mathbf{w}_t - \mathbf{u} \|^2 - \| \mathbf{w}_t + \tau_t(\phi_t^R - \phi_t^N) - \mathbf{u} \|^2 \\ =& -\tau_t^2(\phi_t^R - \phi_t^N)^2 + 2\tau_t(\mathbf{w}_t - \mathbf{u}) \cdot (\phi_t^R - \phi_t^N) \\ =& \tau_t\Big(2\ell_t - \tau_t(\phi_t^R - \phi_t^N)^2 - 2\ell_t^*\Big) \end{aligned} \tag{11}$$

where $\ell_t = \ell((\phi_t^R, \phi_t^N); \mathbf{w}_t)$.

Based on (10) and (11), we can conclude that

$$\sum_{t=1}^{T} \tau_t\Big(2\ell_t - \tau_t(\phi_t^R - \phi_t^N)^2 - 2\ell_t^*\Big) \leq \| \mathbf{u} \|^2 \tag{12}$$

Recall that $\tau_t(\phi_t^R - \phi_t^N)^2 \leq \ell_t$, substituting it into (12), we have,

$$\sum_{t=1}^{T} \tau_t \ell_t \leq \| \mathbf{u} \|^2 + 2C \sum_{t=1}^{T} \ell_t^* \tag{13}$$

If the Ranking PA algorithm makes a mistake on round t, we have $\ell_t \geq 1$. We also have $\tau_t = \min\Big\{C, \frac{\ell_t}{(\phi_t^R - \phi_t^N)^2}\Big\}$. Therefore, $\tau_t \ell_t \geq \min\Big\{C, \frac{1}{\max_{t=1,T}(\phi_t^R - \phi_t^N)^2}\Big\}$

Let M be the number of mistakes made by the Ranking PA algorithm, we have,

$$\sum_{t=1}^{T} \tau_t \ell_t \geq \min\Big\{C, \frac{1}{\max_{t=1,T}(\phi_t^R - \phi_t^N)^2}\Big\} M \tag{14}$$

Substituting (14) into (13), we have,

$$\min\Big\{C, \frac{1}{\max_{t=1,T}(\phi_t^R - \phi_t^N)^2}\Big\} M \leq \| \mathbf{u} \|^2 + 2C \sum_{t=1}^{T} \ell_t^*$$

which concludes the proof. □