빅데이터를 활용한
공정거래 감시 역량 강화방안 연구
(A Tutoring System for Strengthening Capacity Using Big Data Analysis)

2020년 9월

공정거래위원회
나상태

차 례

국외훈련 개요

1. 훈련국 : 영국

2. 훈련기관명 : 더럼 대학 (Durham University)

3. 훈련분야 : 경쟁정책

4. 훈련기간 : 2019.9.8.~2020.9.7. (12개월)

훈련기관 개요

명칭	Durham University
홈페이지	https://www.dur.ac.uk/
주소	Durham University, The Palatine Centre, Stockton Rd, Durham DH1 3LE, UK
전화번호	+44 191 334 2000
설립목적	○ 잉글랜드 북부에 위치한 더럼주에 있는 대학교로써 잉글랜드에서 옥스퍼드와 캠브리지 다음으로 3번째로 오래된 대학이며, 1832년에 설립 ○ 2012년 영국연구대학의 Russell 그룹에 가입함. ○ 초기 교회대학을 시작으로 현재에 이르기까지 학생들에게 세계적 수준의 research와 education을 제공하는 것을 목표로 함.
조직	○ 더럼대학은 크게 4개의 Faculty(Faculty of Social Science & Health, Faculty of Arts and Humanities, Faculty of Science and Faculty of Business)로 구성되어 있으며, Department of Computer Science는 Faculty of Science에 소속되어 있음 ○ 더럼대학은 칼리지 시스템을 운영하고 있다. 16개의 칼리지로 이루어져 있으며 더럼대학교를 입학하는 경우 16개의 칼리지 중 한 개를 선택하여야 하며, 각각의 칼리지는 Collingwood, Grey, Hatfield, Josephine Butler, St Aidan's, St Chad's, St Cuthbert's Society, St Hild & St Bede, St John's, St Mary's, Trevelyan, University, Ustinov, Van Mildert, Stephenson, John Snow 임.

주요 연구분야	○ 사회 전번에 걸쳐 다양한 연구를 진행 중이며, 특히 Process Industries and Surface Science, Energy and Clean Growth, Cosmology and Astronomy, Heritage and Culture, and Hazard and Risk 분야에 강점이 있음. ○ 최근에는 Covid-19 pandemic에 대한 연구들이 진행되고 있음.
주요인사 인적사항	- Justin Welby: Archbishop of Canterbury - Lord Hughes: Supreme Court Justices - Richard Adams: Founder of fair trade organisation Traidcraft - John D. Barrow: Winner of the Templeton Prize

Abstract

How do we learn? What makes education effective? What are the characteristics of the best model for an intelligent tutoring system? Over the past several decades, many researchers have tried to answer these questions, but they remain unresolved. Enormous volume of data is generated every day, and Korea Fair Trade Commission (KFTC) is no exception. Hence, it is difficult for KFTC employees to analyze and understand all the data by themselves. A system which can help them to analyze and understand the data can be of great help to strengthen their capabilities. However, each individual has a different background, and so starts with a different learning style and knowledge base. Thus, it is extremely difficult to develop a tutoring system that can act as a human teacher. Moreover, the development of such a system is time- and effort-intensive. The aim of this project is to develop an enhanced tutoring system which can deliver meaningful content for a class using key concepts. This system starts by asking simple questions to collect user data and uses a k-means clustering method to group user and key concept data. Key concept data is automatically extracted from specific given data. After grouping the data, our system finds the best user group and key concept group for a user. Then, this tutoring system generates a lesson plan using the information from the user and key concept groups. Consequently, the system not only provides the lesson plan to a user but uses it to deliver a lesson by interacting with the user, providing constructive and friendly feedback. While

participating in a lesson, a user can build their own key concept list. As a result, our system serves as a proof of concept for a tutoring system that can deliver a lesson through simple data analysis to improve the competence of KFTC' s employees. Our key concept-based tutoring system is relatively simple, but can nonetheless provide useful information

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

There is a steady increase in data every year. The amount of data is increasing in Korea Fair Trade Commission (KFTC) as well. KFTC is considering systems to strengthen its fair trade surveillance capabilities by making good use of its growing data. Although there are many ways to enhance the KFTC's surveillance capabilities using big data, the method proposed in this project is a tutoring system. It is important to develop a tutoring system to help individuals' learning because understanding all the contents within the limited lecture period and studying a lot of data on their own take a lot of time and effort. Such learning assistant services using a tutoring system can be used in various educational fields as well as in KFTC. In this project, therefore, I introduce a general method of proposed tutoring system and present the results of applying the method to the KFTC.

For the past several years, the development of Information and Communication Technology (ICT) has grown alongside increasing demand for e-learning (Jong-Pil *et al.*, 2002). E-learning, recognized as a new education system centered on learners, provides more learning opportunities with less time and

space constraints. It has been facilitated by the development of Internet technology and changes in the learning environment, with varying content is available on the web. Early e-learning systems were self-directed learning systems, where learners led the overall learning process. However, such self-directed systems do not offer adaptive learning models that provide differentiated learning environments according to the individual learner's characteristics.

The purpose of web-based adaptive tutoring is to create learning environments that adapt and accommodate learners' needs and characteristics (Baldoni *et al.*, 2004). Web-based adaptive tutoring systems make up an area of research and development that integrates the concepts of an Intelligent Tutoring System (ITS) and Adaptive Hypermedia System (AHS) (Astleitner and Hufnag, 2003; Bra *et al.*, 1999). ITS is a tutor-centered system based on traditional artificial intelligence technology, and AHS is a learner-centered system based on more flexible search techniques (Astleitner and Hufnag, 2003; Bra *et al.*, 1999).

ITS is not simply a computer-based teaching and learning tool that provides predetermined content, but a system that maximizes learning effects, by providing the type of support teachers do, such as encouraging, giving feedback and assignments, and understanding learners' learning styles by observing them. However, the development of such a system is complex and time- and effort-intensive. Thus, there has been a greater focus on helping teachers rather than playing all of their roles. Several ongoing projects, such as PELARS and

CENTURY tech, focus on such an approach (Luckin and Cukurova, 2019).

AHS is intended to provide better and wider knowledge to learners. Before Internet access became widespread, most information was provided by a teacher. Hence, learners were limited by the teacher’s own knowledge. Given the availability of ICT, AHS removes this artificial boundary. AHS is a technique that can be used to provide the information presented to the learner's current level of knowledge and to guide the learner in the learning process. However, AHS is not used much, even though it has been the subject of many studies. Many obstacles to implementation of AHS have not yet been overcome; Somyurek (2015) presents seven trends in AHS and four challenges. According to Somyurek, these challenges are inter-operability, open corpus knowledge, usage across a variety of delivery devices, and the design of meta-adaptive systems.

1.1 Motivations

Every person learns with a different style and at a different rate. Thus, it is difficult for lecturers to teach so that all people can understand within the limited lecture period. With the development of ICT, many studies are under way to address these difficulties. Online classes using ICT are relatively easy to access for people under time and space constraints. When there is a pandemic, such as COVID-19, lectures can be conducted online. However, there are challenges to overcome in

ICT-based teaching, such as issues with implementation time and cost. The motivation of this project is to develop an online system that can provide a lesson on a given topic through simple data analysis and use of existing online content.

In this project, we propose an enhanced tutoring system which can deliver a lesson using key concepts extracted from given data. Our system has two main data types: user information and key concept data. By analyzing this data, our tutoring machine can group these two data types using the k-means clustering method, and then match a current user with the best user group and key concept group. Then, our system provides online resources using the key concepts in the group. Our system plays the role of AHS by linking each key concept to Open Education Resource (OER). A user can easily access predefined OERs by clicking key concepts. In addition, the system maintains two lists: OER and key concept lists. An OER list contains predefined OERs, and a key concept list contains key concepts automatically extracted from lecture notes. A user can personalize both of these lists.

1.2 Contributions

Although various studies are being conducted in this area, the key concept-based method is a fairly new approach. I started this project with the belief that I could develop a simple and useful tutoring system using given data.

To implement such a system, we first proposed a key concept

extraction method based on the Term Frequency Inverse Document Frequency (TF-IDF). In many cases, bigrams and trigrams are more meaningful than a single word. Using traditional TF-IDF, however, several unigrams ranked higher than n-grams due to the characteristics of this method. Thus, we developed a modified method that allows n-grams to be extracted as much as unigram.

Our system allows a user to freely modify the key concept and OER lists, accounting for the fact that each user has different prior knowledge and preference. A user can personalize the key concept list by adding, deleting and highlighting key concepts, and modify an OER list by selecting preferred OERs from a predefined OER list. After selection, a user can find open resources on the Internet using the selected OERs.

Our system also generates a lesson plan for a class. It automatically extracts key concepts to make a lesson plan; based on that plan, the system can provide a preview, assessment, review and summary, and related information, such as the lecture title and objectives. Our system also analyzes duration information in user data and predicts a duration for the lecture and forwards the information to the lesson plan, so a user can estimate how long it will take to study the topic.

Lastly, we intended to test the proposed system using the KFTC's annual training materials used for strengthening the capabilities of the employees, but most of the materials used in the training are written in Korean, making it complex to conduct the test smoothly. This is because, developing a system which

can preprocess and keyword extract in both English and Korean languages in a limited period of time is difficult and we decide to implement English version first. Further study should be considered for Korean letters. However, a test is conducted using the documents written in English provided by KFTC to check the possibility of the proposed system.

In conclusion, in this work we developed an enhanced tutoring system that can help learners' studies using key concepts from given data, and the system can interact with learners by providing constructive, friendly and sometimes critical feedback as well.

Chapter 2

Related work

Tutoring systems have evolved considerably over the past several decades. In this section, I present related work on tutoring systems. First, however, I explain key concept extraction methods.

2.1 Key Concept Extraction

Key Concept Extraction (KCE) is a type of text analysis technique that can automatically extract keywords or key phrases from text data. KCE methods allow the extraction of important concepts from big data sets. Preprocessing is required before the application of KCE methods; after preprocessing, various methods can be applied to extract key concepts.

- *Word frequency and co-occurrence*

Keyword extraction based on word frequency (Luhn, 1957) is the simplest statistical approach. This method can be useful for document overview; however, this method only counts the

words that appear most often. It does not consider the critical aspects of the words' meaning and order. Synonyms, for example, are ignored by this approach. Matsuo and Ishizuka (2004) proposed another method for keyword extraction using word co-occurrence information; co-occurrences are words that are frequently found in the same sentences.

- *TF-IDF*

The TF-IDF method is designed to evaluate the relative importance of words within a document for information searching and text mining. This method not only allows determination of the ranking of the most similar documents, but also allows grouping of documents by similarity. A word with a large TF-IDF value is highly likely to determine the meaning of the subject of the document and can therefore be used as a measure to extract the main keyword (Rajaraman and Ullman, 2011).

The TF-IDF weighted value is multiplied by the Term Frequency (TF) and the Inverse Document Frequency (IDF) (Aizawa, 2003). The TF value refers to the frequency of a specific term appearing in a single document. To include this value in the weighted model, the more often the given term appears in the document, the more importance it is assigned. The actual TF value is normalized by dividing the frequency of each term appearing inside the document by the total number of times all words appear. This normalization is intended to prevent bias of the TF value depending on the size of the document. However, it is insufficient to describe the document only with this TF

value, because a term with excessively large TF values may be meaningless common terms. To address this issue, the IDF factor is introduced. This value is the number of documents in a set of documents divided by the number of documents in which a particular term appears. This means that the IDF value of a term appearing in relatively many documents will be small, whereas the value of a term that appears in only a few documents will be large. Thus, a term with a small IDF value is more likely to be a common term and one with a large IDF value is likely to have a major meaning within the documents. Hu and Wu (2006) offer a theoretical justification of the IDF figures. The TF-IDF value is the product of the TF factor and IDF factor, and based on this TF-IDF weighted value, a term that appears frequently in a specific document and has a low document frequency in the whole collection of documents can be considered as an important term.

- *RAKE*

The Rapid Automatic Keyword Extraction (RAKE) is another well-known keyword extraction method. This method considers that keywords frequently contain multiple words but rarely contain standard punctuation and stopwords. The RAKE method uses these punctuation and stopwords to partition a document into candidate keywords. In addition to the stopwords list, this algorithm uses two more parameters; a set of phrase delimiters and a set of word delimiters. A document text is split into an array of words by the word delimiters, and then this array is split into sequences of contiguous words at the phrase delimiters and stopword positions. Words within a sequence are

assigned the same position in the text and together are considered a candidate keyword. Among candidate words, co-occurrences of words are considered as words with significant meaning. Thus, the RAKE method frequently finds keywords which contain phrases or multiple words rather than few words. Then, a score is calculated for each candidate keyword based on the degree and frequency of words (Rose *et al.*, 2010).

- *Machine Learning*

Machine learning methods can be seen for many text analysis tasks, including KCE. Various machine learning techniques are used for extracting keywords, including Support Vector Machines (SVM) and deep learning such as recurrent neural network (Zhang et al., 2016; Kaur and Gupta, 2010).

2.2 Adaptive Hypermedia System

AHS can offer a personalized learning environment. In contrast to the traditional educational systems, which generally focus on a closed corpus of information set by a teacher, AHS can provide personalized information based on a model of the user's goals, preferences and existing knowledge. AHS consists of a user model, a domain model and an interaction model (Benyon and Murray, 1993). The domain model has the application used for providing the adaptive capabilities, and interaction model represents the interaction between user model and domain model (Figure 1).

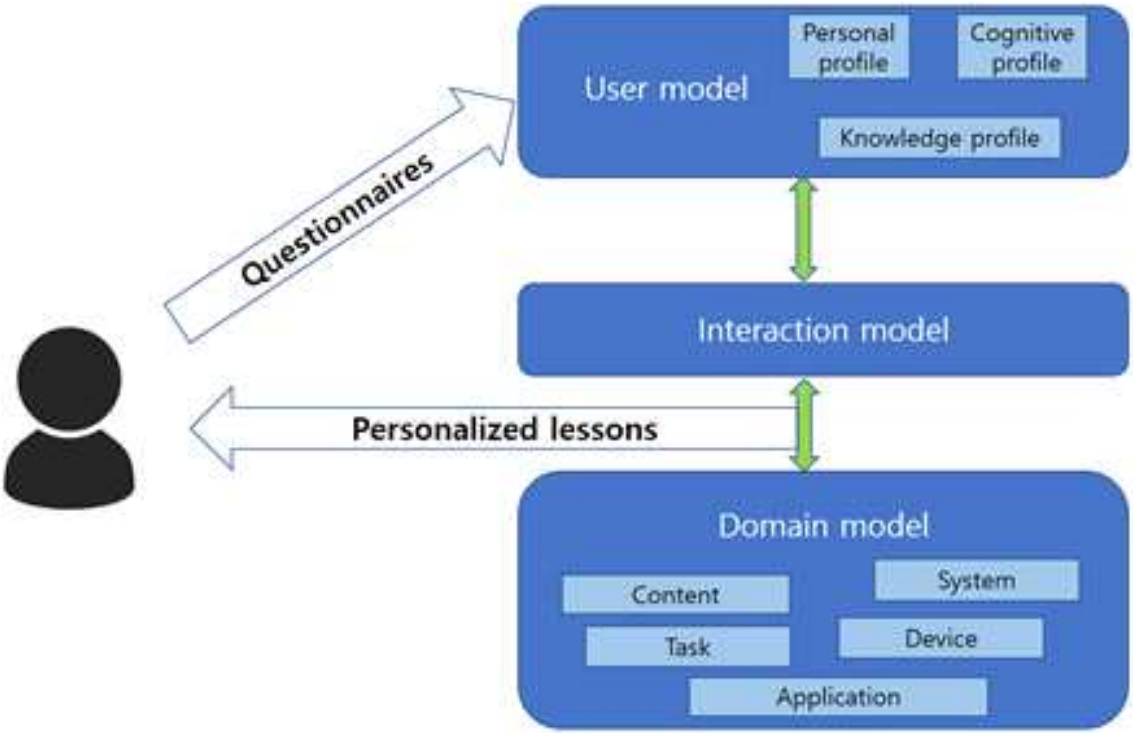


Figure 1. Adaptive hypermedia system

- *User model*

AHS requires a user model that can recognize and manage changes in the knowledge status of each user and apply the user's characteristics to the system; these characteristics can vary over time, even for the same user. The user model represents several profiles of each user. The profiles can be acquired implicitly, by making inferences about users from their interactions with the system, and explicitly, by asking related questions (Virvou *et al.*, 2012).

User models can be classified into two main classes: stereotype and overlay models (Nguyen *et al.*, 2008). A

stereotype user model consists of one or more stereotypes or cluster of characteristics and is useful to quickly build user models. Although stereotype models are useful and effective, each represents the characteristics of the group. Thus, the representation is not absolutely true for all users in the group. An overlay user model consists of a subset of elements from domain model. Each element represents the knowledge, concept, subject or topic in the area. Each user can be represented by a different aspect of the user's relationship with elements and stored in personal, cognitive and knowledge profiles.

- *Domain model*

AHS also needs a domain model contains several features such as content. Domain model has the application which is used for providing the adaptive capabilities.

- *Interaction model*

The other model for AHS is an interaction model which represents the interaction between user model and domain model.

2.3 Intelligent Tutoring System

ITS is a computer system designed to realize personalized lessons on the same level as those conducted by human teachers using computing technologies (Al-Hanjori *et al.*,

2017). The main purpose of ITS is the development of education and authoring systems, study of learner models by cognitive analysis, and research on natural language understanding to support user access. These applications are used in various fields of society where education and training are required, including medical education and various types of vocational training. ITS consists of four components: expert knowledge, student model, and tutoring and a user interface module (Nwana, 1990), (Figure 2).

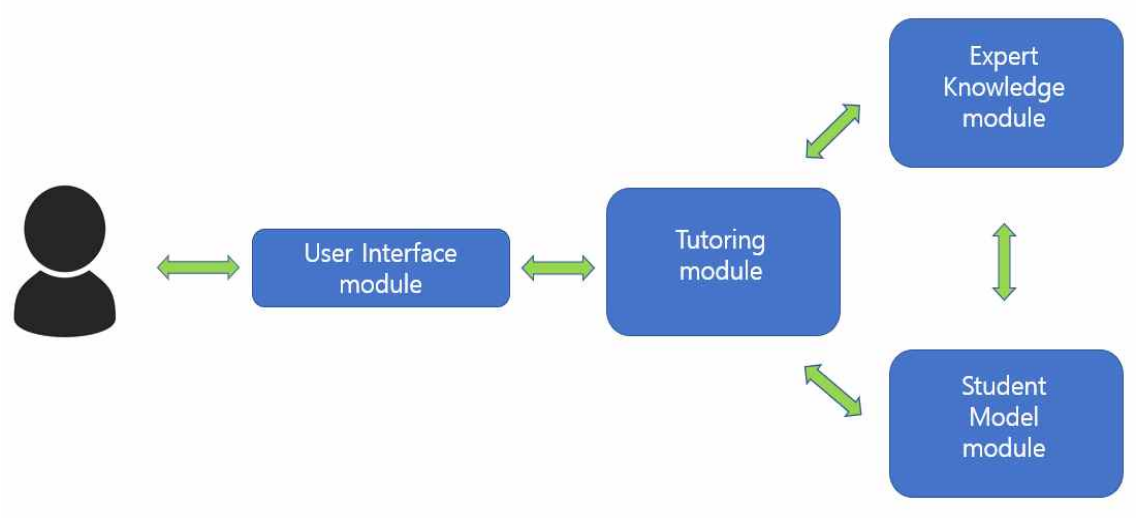


Figure 2. Intelligent tutoring system

- *Expert Knowledge module*

Expert knowledge module consists of a knowledge of the concepts and explanations of topics intended to be taught to learners. This module should also include the ability to learn new knowledge by analogy from existing knowledge. These

expert modules relate to the subject areas that you want to teach learners such as interrelationships between the concepts to be learned, procedural knowledge or rules, and knowledge gained or explored through expert experience Thus, When building expert modules, it is important to be able to accurately determine the difficulties of learners in solving problems and to present a solution to them. It should be also considered to ensure that rudimentary and general knowledge is turned into more specific and professional knowledge without losing its basic usability as learners become closer to experts.

- *Student model module*

A learner module is a module that assesses learners' propensity and learners' abilities regarding a specific learning process To this, this module has information about individual learners to understand their current status of knowledge. These Modules support specific tutoring decisions used to guide learners in solving problems and to construct learners' learning experiences. These Learner Modules adapt properly to the teaching field and teaching history to adapt specific learners to the selection of hints and corrections, and selection of examples or assignments. Learner modules are also used for various purposes to evaluate learners' propensity for a particular learning process and their ability in teaching areas, especially important factors in determining progress, providing advice, creating problems, and adopting explanations.

- *Tutoring module*

The Tutoring model is a module that makes decisions such as selecting problems to be solved, evaluating learners' performance, and providing assistance to learners' requests in the tutoring process. In other words, the Tutoring model is a collection of teaching experts' knowledge, such as teaching strategies. Rather than simply conveying knowledge, this module should be able to provide the most optimal teaching method to a particular learner. It should know what to say to each learner and when to say it. In addition, it should know how to get learners from one level to the next and how to help learners from the current level of knowledge. Also, Tutoring models include teaching techniques. This uses diagnostic and coaching rules to be performed based on data collected in the learner mode. These rules can also provide feedback to learners. The teaching model improves the learner to a better level of understanding by providing relevant hints or messages with the most timely. Therefore, the tutoring model has information on how to guide according to the information obtained from the student module. In other words, all modules are important in the ITS, but the crucial part is the tutoring model

- *User Interface module*

Interface modules are necessary modules for communication between learners and computers. When an interface module is designed, it is important to find ways to allow learners to communicate at different levels and adapt to individual learners' preferences. Interface modules play an important role for bi-directional communication between learners and computer teachers. For such communication, natural language processing

is needed. This module is designed to communicate between ITS and learners. In addition to simply entering the correct answer to a problem presented by a computer during the communication, the learner should be able to get answers from a professional teacher about any unknown problem

2.4 Lesson Plan

A lesson plan is a guide teachers prepare in advance of a class. In general, since the lesson plan is written and used by teachers, they must possess expertise in the principles, characteristics, and content of the lesson plan (Nagro *et al.*, 2019). The lesson plan is aimed at how to construct the plan to effectively achieve the learning objectives of the day for learners. This lesson plan should be designed with a number of considerations in order to achieve the optimal educational goals of learners. A typical lesson plan has several sub-parts: title, objectives, related requirements, progress plan, evaluation plan, assessment method, reflection and duration. Lecturers have traditionally spent substantial time writing lesson plans manually, but the development of ICT has led to the development of automated lesson planner systems (Jong-Pil *et al.*, 2002; Kouno *et al.*, 2002). In this project, we used key concepts to automatically generate a lesson plan.

- *Lesson title*

Lesson title is a title of a lesson which is usually located on the top of the lesson plan.

- *Lesson objectives*

Lesson objectives are a list of things that students need to learn as they progress through class. In other words, lesson objectives are goals for the class. When determining the lesson objectives, people should consider whether the goals are specific or achievable for students. It is also important to set the objectives not too much or too little, depending on the level of knowledge of students.

- *Prerequisites*

Prerequisites indicate the subjects students should have already taken to take the lesson. Some of them are optional and some are mandatory.

- *Lesson progress plan*

Lesson progress plan is planning on how to proceed with the class and composed of four stages. First of all, it introduces the objectives and discuss core concepts. Then, each student works independently to get into the details of your lesson. Next, students will look back what they have learned in the lesson. Lastly, students organize what they have learned once more and then reinforce it.

- *Assessments*

Assessments allow students to measure how much they have achieved the class objectives. There are two types of assessments, formative and summative. Common assessments are quizzes, writing assignments, and hands-on activities.

- *Lesson reflection*

Lesson reflection provides lessons learned in a class in the form of review so that students can remain what they learned in class. It also gives students a chance to remind on what they learned in the whole class.

- *Duration*

Duration represents the time it takes to proceed a class.

2.5 Clustering Method

Clustering is a type of unsupervised machine learning algorithm commonly used in data analysis. This method is used when there are no clear classification criteria for classifying objects in a given dataset. Two main principles of clustering are to maximize the degree of aggregation in each cluster and maximize the degree of separation between clusters. Euclidean distance is commonly used for measuring these degrees. In this project, we used the k-means clustering algorithm (MacQueen, 1967). This algorithm clusters objects into predefined number, k, of groups. It first selects k random objects as initial cluster centroids; each data object is grouped into the nearest cluster

centroids. Then, it calculates the new cluster center. Each object is readjusted based on the new cluster center. It repeats the steps until there is no change to the centers. The objective function for the process is defined as

$$V = \sum_{i=1}^{k} \sum_{x \in s_i} \left| x - \mu_i \right|^2 \quad (1)$$

where k is the number of clusters, x is a data point in cluster s_i, i=1,2,⋯,k, and μ_i is the centroids of k groups (Dogan, 2008). In this project, the k-means algorithm is used for user model clustering and key concept clustering.

- *User model clustering and keyword clustering*

Nguyen (2014) proposed user model clustering method based on the k-means algorithm. When user model U_i is represented as vector U_i={u_ij }, i is one of user's features and j is considered as an instance. For example, if there are three different features which are age, knowledge and gender, and 100 instances, u_age21 represents the 21st user's age vector. Then, the k-means algorithm is applied to cluster the user model.

Sang-Woon and Joon-Min (2019) proposed a system that can cluster papers using the k-means algorithm. In it, the TF-IDF method is applied to calculate the distance of keywords of each paper. Since it is difficult to apply text data itself directly to clustering, we must transform the data into something the system can understand. Computer systems are exceptionally

good at processing numbers; hence, the text data should be transformed into vector of numbers using the TF-IDF method. Then, keywords can be clustered using the k-means algorithm.

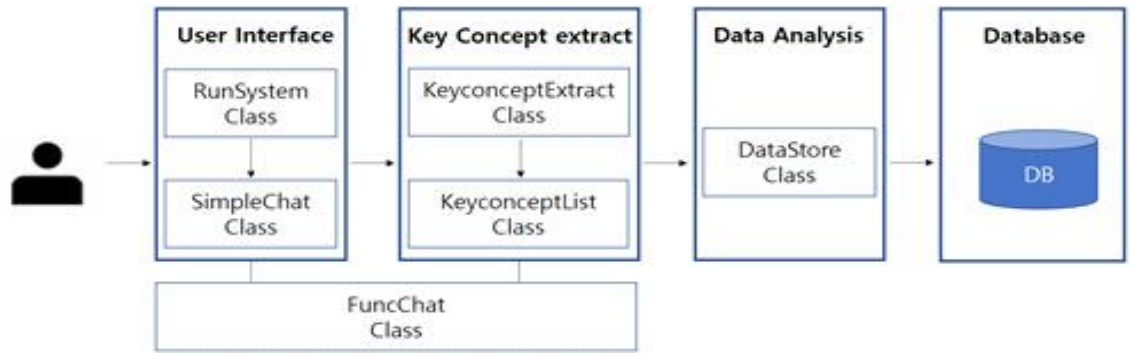
Chapter 3

Solution

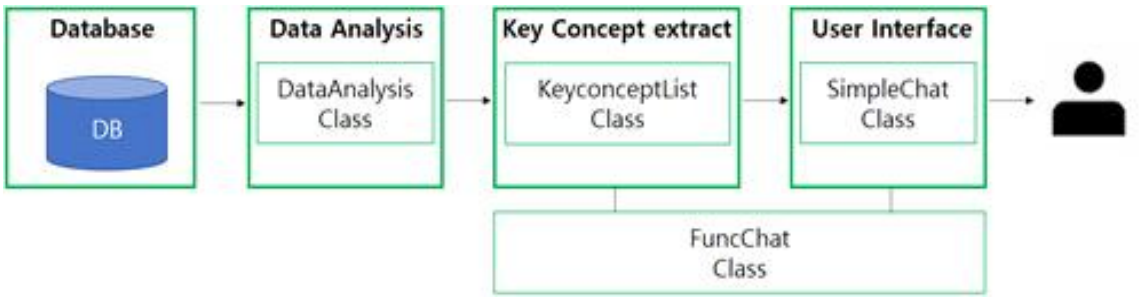
In this section, I introduce a new solution for a tutoring system that can deliver a lesson from given data. I present details of the system, a part of source codes and testing process in this section.

3.1 System Structure

The structure of our system consists of four components (Figure 3): a user interface component for interacting with users, a key concept (KC) extract component for extracting key concepts, a data analysis component for analyzing user and key concept data, and a database component for storing user and key concept data.



(a) User to database



(b) Database to user

Figure 3. Conceptual diagram of the enhanced tutoring system

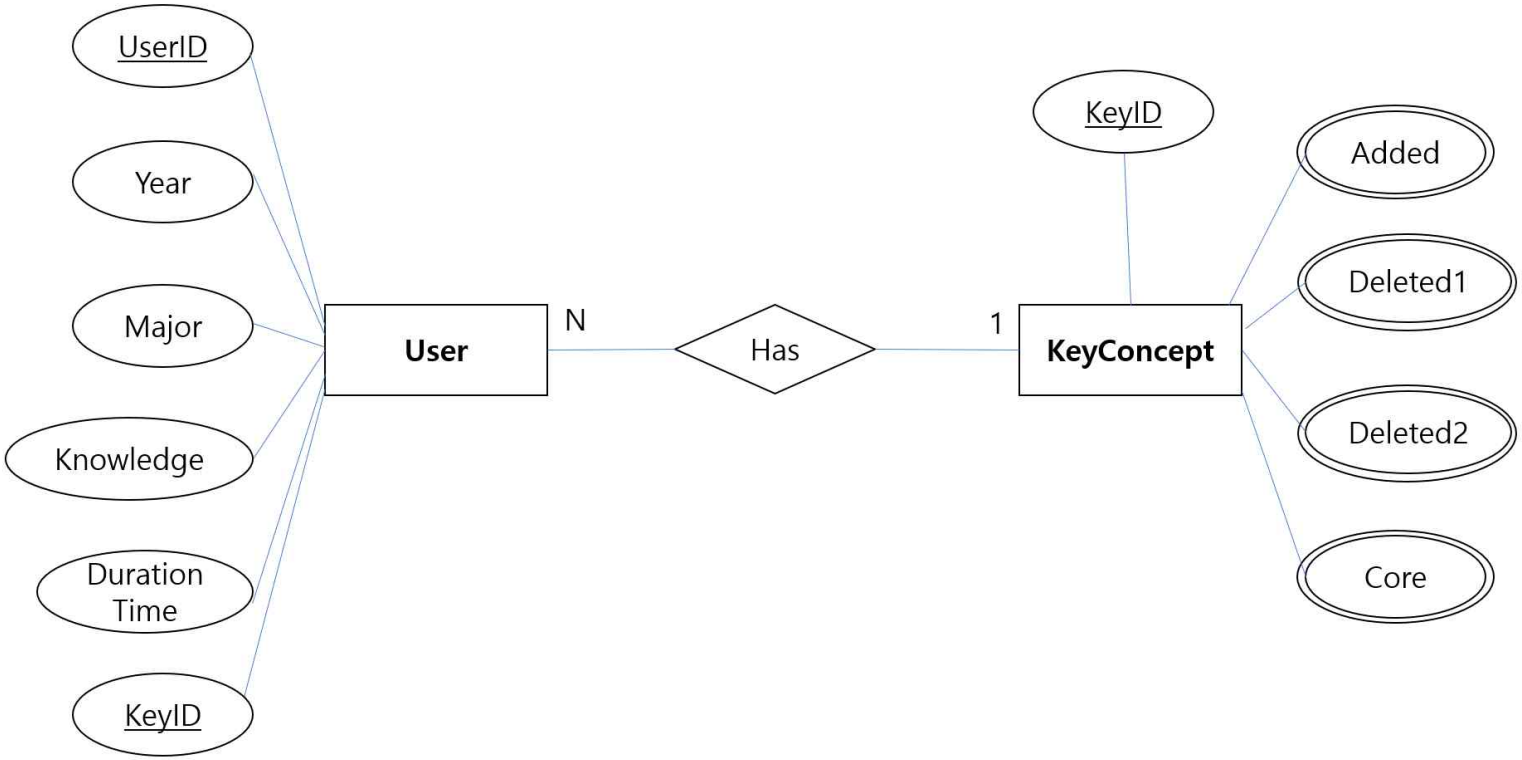
The user interface component acts as an interface when a user uses our system. A user has to go through this component to communicate with the tutoring system. Data collecting for analysis, delivering open educational content and a lesson plan are done in this component. It also acts as a chat bot that can answer simple questions.

The KC extract component is needed to extract key concepts from given lecture notes and make a key concept list according to the configuration value of the system. Depending on the value, the list can be short or long. This list is not only used for the data analysis component, it is also sent to the user interface component, so that the user can see the key concepts and link to their open educational content.

The data analysis component processes user data and key concept data. It stores these data temporarily for analysis. This component clusters user data using the k-means algorithm and

also clusters key concepts based on their similarity. After analysis, it passes these data to the database component.

Our system uses the database components for storing user data and key concepts. The Entity Relationship Diagram (ERD) for our database is shown in Figure 4. Currently, our system needs only two entities: User and KeyConcept, and there is a many-to-one relationship between the entities. A KeyConcept can have several Users, but a User can have only one KeyConcept. The User entity has six attributes, which are collected from the user's answers to simple questions. These attributes are generally filled when the system starts, except duration, which is asked at the end. The KeyConcept entity has five attributes, and they are gathered while the user is studying a lesson. In terms of a Database Management System (DBMS), each entity is a table in database, so the ERD presents the relationships between tables. Figure 4 also shows the table structure for converting the ERD to tables. According to the ERD, KeyID acts as a primary key in the KeyConcept table and a foreign key in the User table. UserID is a primary key in the User entity. Except for the primary key (KeyID), the other attributes in KeyConcept entity are multivalued because a user can add and delete multiple key concepts while studying.


UserID
Year
Major
Knowledge
Duration Time
KeyID
User
N
Has
1
KeyConcept
KeyID
Added
Deleted1
Deleted2
Core

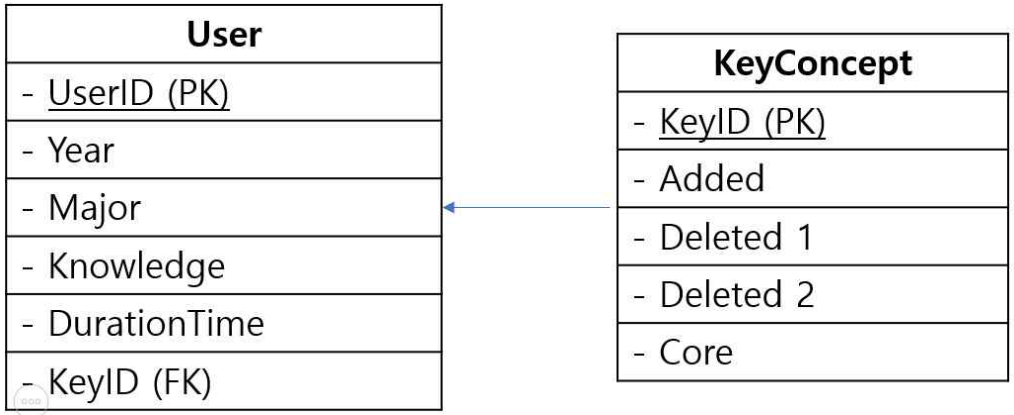


Figure 4. Entity relationship diagram

Python codes 3.1. DB table

```
def data_store_DB(self, all_data):
        key_table = """CREATE TABLE IF NOT EXISTS keyconcepts (
                    id integer PRIMARY KEY AUTOINCREMENT,
                    added text NOT NULL,
                    core text NOT NULL,
                    deleted1 text NOT NULL,
                    deleted2 text NOT NULL );"""

        users_table = """ CREATE TABLE IF NOT EXISTS users (
                    id integer PRIMARY KEY AUTOINCREMENT,
                    major integer NOT NULL,
                    year integer NOT NULL,
                     knowledge integer NOT NULL,
                    durationtime float NOT NULL,
                    keyconcept_id integer NOT NULL,
                    FOREIGN KEY (keyconcept_id) REFERENCES keyconcepts (id) ); """
```

3.2 Implementation Tools

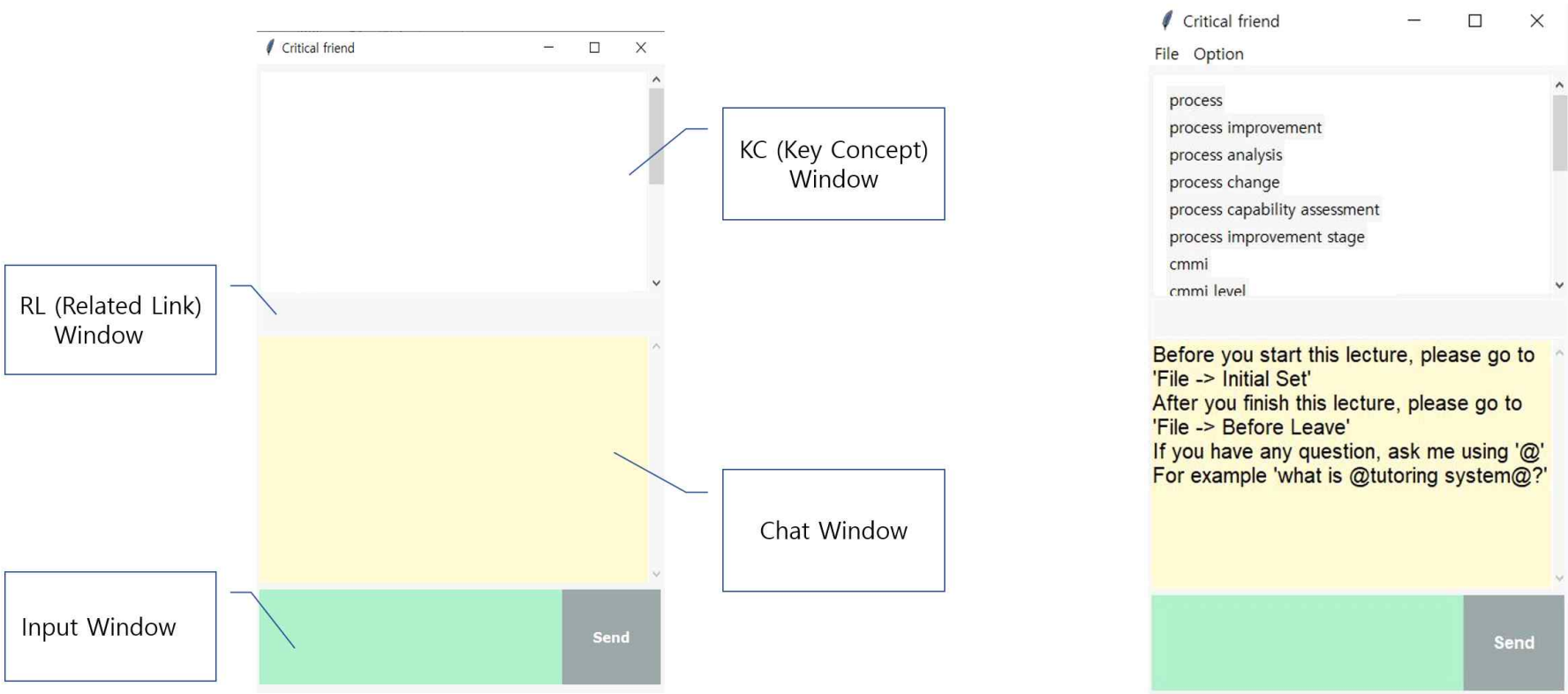
Python is one of the most popular programming languages for data analysis. It offers several benefits for data science, such as being an easy language to learn, but most importantly, it has a great number of data-oriented libraries that can simplify data processing and save time. While implementing our system, we applied many well-known libraries for data analysis: pandas is a fast, powerful, and easy to use python library for data analysis and manipulation (McKinney *et al.*, 2010); Natural Language Toolkit (NLTK) is used to preprocess natural language data (Bird *et al.*, 2009); and we employed the sklearn library for key concept extraction and k-means clustering (Pedregosa *et al.*, 2011). Tkinter is the standard GUI library for python, which we used to make dialog window (Lundh, 1999). Lastly, we developed the database based on the sqlite3 database engine (Hipp, 2020).

3.3 Enhanced Tutoring System

The goal is for this system to act as an enhanced tutoring system that can help a user study, one that can act as a critical friend who offers critiques of a user's work, as well as encouraging users in their studies and recommending meaningful content. Our enhanced tutoring system can deliver personalized experience based on a stereotype user model. By using the extracted key concepts and the user's selections,

our system can provide a personalized lesson plan and deliver a lesson based on the plan.

First, a user interface is needed to enable a user to communicate with our system in a simple conversation dialogue window. Figure 5(a) illustrates the dialogue window, which consists of four sub-windows. They are a key concept window for presenting extracted key concepts, a related link window for linking concepts to open education content, a chat window for the user to see communication with the system, and an input window for a user's input to the system. Each window can be operated separately or deleted. Figure 5(b) shows the initial state of our system. This initial window shows a list of key concepts extracted from a specific example lecture note. KCE methods are used for extracting key concepts



(a) Dialogue window

(b) Initial state

Figure 5. Dialogue window and initial state window

Python codes 3.2. Dialogue window

```
import tkinter as tk
from tkhtmlview import HTMLLabel
from classes.KeyConceptList import KeyConceptList
from classes.SimpleChat import SimpleChat
from classes.FuncChat import FuncChat
from classes.KeyConceptExtract import KeyConceptExtract
from classes.KeyConceptExtract_pdf import KeyConceptExtract_pdf

MSG_SIZE = 0.35
HYPER_SIZE = 0.06

#initialize
base = tk.Tk()
base.title("Critical friend")
base.geometry("400x600")

canvas = tk.Canvas(base, borderwidth=0, background="#ffffff")
```

```
msg_frame = tk.Frame(canvas, bg='#ffffff', bd=10)
scrollbar_frame = tk.Scrollbar(base, command=canvas.yview)
canvas.configure(yscrollcommand=scrollbar_frame.set)

#Making menu buttons
func_chat=FuncChat(msg_frame)
main_menu = func_chat.make_menu(base)
base.config(menu=main_menu)
msg_frame.bind("<Configure>", lambda event, canvas=canvas: func_chat.frame_configure(canvas))

canvas.place(relx=0.01, rely=0.01, relwidth=0.95, relheight=MSG_SIZE)
canvas.create_window((4,4), window=msg_frame, anchor="nw")

#Initialzing chat window
simple_chat = SimpleChat()
#Create Chat window
START_M = "Before you start \n"
END_M = "After you finish \n"
```

```
Q_M = "If you have any question, ask me \n"
chat_win = tk.Text(base, bd=0, bg="#FCF3CF", font="Arial")
chat_win.insert(tk.INSERT, START_M+END_M+Q_M)
chat_win.config(state=tk.DISABLED)

#Bind scrollbar to Chat window
scrollbar = tk.Scrollbar(base, command=chat_win.yview)
chat_win['yscrollcommand'] = scrollbar.set

#Initializing further window
hyper_win = tk.Text(base, bd=0, bg="#FCF3FF", font="Arial",)
hyper_win.config(state=tk.DISABLED)
html_label=HTMLLabel(hyper_win)
html_label.pack()

#Create the box to enter message
input_win = tk.Text(base, bd=0, bg="#ABEBC6", font="Arial")
```

```
def send():
    simple_chat.send(input_win, chat_win, html_label)

#Create send button message
send_button = tk.Button(base, font=("Arial",10,'bold'), text="Send",
                        bd=0, bg="#99A3A4", activebackground="#515A5A", fg='#ffffff', command=send)

#Displaying
chat_win.place(relx=0.01, rely=MSG_SIZE+0.08, relwidth=0.95, relheight=0.8-MSG_SIZE-HYPER_SIZE)
hyper_win.place(relx=0.01, rely=MSG_SIZE+0.015, relwidth=0.99, relheight=HYPER_SIZE)
input_win.place(relx=0.01, rely=0.83, relwidth=0.74, relheight=0.15)
send_button.place(relx=0.75, rely=0.83, relwidth=0.24, relheight=0.15)
scrollbar.place(relx=0.96, rely=MSG_SIZE+HYPER_SIZE+0.02, relwidth=0.04,
relheight=0.8-MSG_SIZE-HYPER_SIZE)
scrollbar_frame.place(relx=0.96, rely=0.01, relwidth=0.04, relheight=MSG_SIZE)
```

Before we apply KCE methods, preprocessing is needed because the input natural language data is not structured. As a first step, it is necessary to remove all stopwords, generally the most common words, such as ‘the’, ‘he’ and ‘she’. Depending on the purpose, we can make a stopwords list that contains all words to be deleted through the process.

Tokenization is the next process, in which a series of words are chopped up into units called tokens. Tokenization is important because it is not easy for a computer to understand whole sentences at once. Breaking down the sentences allows the machine to understand the meanings more easily. We can break sentences down into words or n-grams.

After tokenization, the last step for preprocessing is stemming or lemmatization. The extraction of stems (stemming) can be viewed as a simplified version of morphological analysis, or as a type of guesswork that cuts the endings of words by looking at a set of rules. In other words, since this is not a delicate task, after stemming, the resulting word may be a non-dictionary word. Stemming is a rule-based approach, so it is easy to understand and implement. Unlike stemming, lemmatization is based on the dictionary; because words are processed based on dictionary, the probability of error is lower than with stemming, but new words which are not in the dictionary cannot be processed. Lemmatization also requires labor to build a foundation dictionary, which is not needed for stemming. After pre-processing, various KCE methods can be applied to extract key concepts, as illustrated in Figure 6.

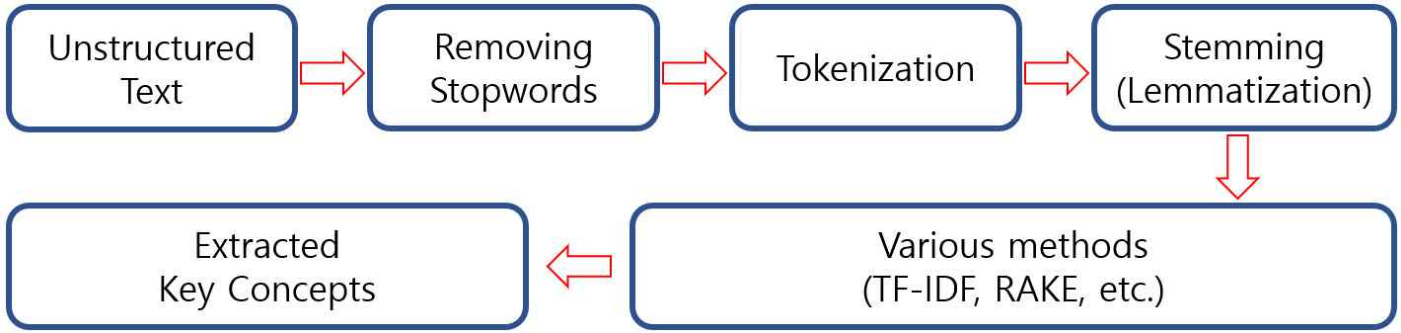


Figure 6. Natural language processing for key concept extraction

Python codes 3.3. Preprocessing

```
def creat_stopword(self):
    file1 = open("files\stopwordslist.txt")
    line=file1.read()
    file1.close()
    new_words=line.split()
    self.stop_words = self.stop_words.union(new_words)

def pre_process(self):
    tot_corpus = ""
    d_len = len(self.tt)
    for i in range(0, d_len):
        #Remove punctuations
        text = re.sub('[^a-zA-Z]', ' ', self.tt[i])
        #Convert to lowercase
        text = text.lower()
        #remove tags
        text=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",text)
```

```
# remove special characters and digits
text=re.sub("(\\d|\\W)+"," ",text)
##Convert to list from string
text = text.split()
#Stemming
#ps=PorterStemmer()
#Lemmatisation
lem = WordNetLemmatizer()
text = [lem.lemmatize(word) for word in text if not word in self.stop_words]
text = " ".join(text)
self.corpus.append(text)
tot_corpus += " "
tot_corpus += text
```

We first tested a word frequency-based method that considers documents as a list of words or phrases and ignores the meaning, structure, and order of words. Then, we employed the TF-IDF and RAKE methods. As explained in Section 2.1, while the RAKE method is suitable for finding meaningful phrases, the TF-IDF method is suitable for finding key words. In this project, our system needs key words and short phrases rather than long phrases, leading me to choose the TF-IDF method. Although the TF-IDF is a conventional and old weighting method, it is still widely used for keyword extraction (Beel *et al.*, 2016).

Python codes 3.4. RAKE

```
def RAKE_method(self):
    r = Rake(self.stop_words)
    a=r.extract_keywords_from_text()
    b=r.get_ranked_phrases()
    c=r.get_ranked_phrases_with_scores()
```

Python codes 3.5. TF-IDF

```
def TFIDF_method(self, ngram):
    vectorizer = CountVectorizer(ngram_range = (ngram,ngram))
    vectorizer.fit_transform(self.corpus)
    features = (vectorizer.get_feature_names())

    # Applying TFIDF
    vectorizer = TfidfVectorizer(ngram_range = (ngram,ngram))
```

```
X2 = vectorizer.fit_transform(self.corpus)
scores = (X2.toarray())

# Getting top ranking features
sums = X2.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0,col] ))
ranking = pd.DataFrame(data1, columns = ['term','rank'])
words = (ranking.sort_values('rank', ascending = False))

if ngram == 1:
    words_num = words.head(self.UNI_NUM_WORDS)
else:
    words_num = words.head(self.NUM_WORDS)
return words_num
```

Table 1 shows how the TF-IDF weighted value is multiplied by the TF and IDF (Aizawa, 2003). As explained, the TF value refers to the frequency of words appearing in a document and we need the IDF value to prevent meaningless common words from being incorrectly emphasized. Then, the TF-IDF method can extract keywords.

Table 1. TF-IDF weighting model

TF value	$tf_{i,\ j} = \frac{n_{i,\ j}}{\sum_k n_{k,\ j}}$ $n_{i,j}$: The number of occurrences of word in document $\sum_k n_{k,j}$: The total number of occurrences of all words in document K : The number of keywords D : The number of documents
IDF value	$idf_i = \log \frac{\lvert D \rvert}{\lvert \{d_j \rvert\ t_j \in d_j\} \rvert}$ $\lvert D \rvert$: The total number of documents $\lvert \{d_j \rvert t_j \in d_j\} \rvert$: The number of documents that keyword occurs
TF-IDF weighted value	$TFIDF_{i,\ j} = tf_{i,j} \times idf_i$

However, since most high-ranked keywords extracted by the TF-IDF method are unigrams, we tried to also extract bigrams and trigrams by using a threshold value. In many cases, bigrams and trigrams are more meaningful than a single word. For instance, when 'process' is an extracted keyword, it is not clear whether it is used in relation to markets, sciences or something else. It is more straightforward to determine this when the keyword is 'software process' or 'software process improvement'. Thus, we first extract a unigram, then extract bigrams and trigrams that contain the unigram. The number of unigrams, bigrams, and trigrams can be set using a configuration file. For bigrams and trigrams, we introduced a threshold value to manipulate the number of them. Bigrams and trigrams whose TF-IDF weighted scores are higher than the threshold value are extracted as key concepts.

These key concepts are automatically connected to the predefined OER services (currently, Google, Wikipedia, MOOC, Google Scholar, Research Gate, and the Durham University library). A user can choose to link each key concept to all or some of OER services. This choice can be made implicitly by observing a user's interaction with our system. However, this is possible only when there is already enough data for a system to analyze and predict the behavior of a user. In other words, it is not suitable for an initial system like this one. Thus, our current system explicitly made a choice to allow users to create a personal OER service list.

Python codes 3.6. OER services

```
def message_window(self, event, arg, func_chat):
        self.win = tk.Toplevel()
        tk.Label(self.win, text="What do you want?").pack()
        if 'Wikipedia' in selected_cont:
                tk.Button(self.win, text='Wikipedia', command=lambda: self.open_url(arg[1])).pack()
        if 'Google' in selected_cont:
                tk.Button(self.win, text='Google', command=lambda: self.open_url(arg[2])).pack()
        if 'Mooc' in selected_cont:
                tk.Button(self.win, text='Mooc', command=lambda: self.open_url(("url")).)).pack()
        if 'Google Scholar' in selected_cont:
                tk.Button(self.win, text='Google Scholar', command=lambda:
self.open_url(("url")).)).pack()
        if 'Research Gate' in selected_cont:
                tk.Button(self.win, text='Research Gate', command=lambda: self.open_url("url")).pack()
        if 'Durham Library' in selected_cont:
                tk.Button(self.win, text='Durham Library', command=lambda:
self.open_url(("url")).)).pack()
```

For similar reasons, we explicitly ask three questions at the beginning of the service to gather user data based on user modeling:

> *Question 1: What is you major? How close is your major to this subject?*
>
> *Answer choices: Not close, Close, Very close*
>
> *Question 2: Which year are you in?*
>
> *Answer choices: 1st year, 2nd year, 3rd year*
>
> *Question 3: How familiar are you with this subject?*
>
> *Answer choices: Not at all, Somewhat, Average, Extremely, Expert*

In this project, we used only minimal questions for user modeling, but the number of questions and answer choices can be modified by operator through a configuration file.

Configuration file

SCORE_INTENSITY means threshold score value for 2, 3-gram keywords
UNI_NUM_WORDS means the number of 1-gram keywords
NUM_WORDS means the number of 2, 3-gram keywords

SCORE_INTENSITY= 0.0
UNI_NUM_WORDS= 5
NUM_WORDS= 20

What is the title, object and prerequisite?
TITLE= Software Quality Management
OBJECT= We learn software quality management from this lecture
PREREQUISITE= OOP, Software Design, Foundations of Computer Science, Algorithm, Operating Systems

How many questions?
QUESTIONS= 3
Question= What is your major? How close is your major to this subject?
Levels= Not close, Close, Very close
Question= Which year are you in?
Levels= 1st, 2nd, 3rd
Question= How familiar are you with this subject?
Levels= Not at all, Somewhat, Average, Extremely, Expert

Based on the answers, users are grouped into several clusters using k-means clustering. To determine the number of clusters (k), we used an elbow method, the oldest and most popular heuristic method for determining k (Syakur, 2018). The method starts with k = 1 and calculates the Sum of Square Error (SSE) from each point to its assigned center (Eq. 2). Then, it increases k by 1 and calculates the SSE until an inflection point of the elbow curve is visible.

$$SSE = \sum_{k=1}^{K} \sum_{x \in S_k} dist^2(c_k, x) \quad (2)$$

$$\arg\min \|c - x\|_2^2 \quad (3)$$

where k is the number of clusters, C the assigned center of each cluster, and x_i data point in each cluster S. In this project, we applied 30 virtual samples for testing

Python codes 3.7. Elbow method

```
def elbow_method(self, item_name):
        squared_distances = []
        K = range(1,10)
        #for user data
        if(item_name == "student_cluster"):
                item_val = self.stu_data_ans
                for k in K:
                        km = KMeans(n_clusters=k, random_state=14)
                        km = km.fit(item_val)
                        if km.inertia_ < self.stu_data_val:
                                ret_val = k
                                return ret_val
                        squared_distances.append(km.inertia_)
        #for key concepts
        else:
                item_val = self.key_concepts_list
                for k in K:
```

```
km = KMeans(n_clusters=k, random_state=14)
km = km.fit(item_val)
if km.inertia_ < self.key_data_val:
        ret_val = k
        return ret_val
squared_distances.append(km.inertia_)
```

After determining an adequate k, the k-means method is employed through Eq. (3) to cluster user data into k groups. Then, our system can obtain the best clustered group for a current user. In addition, we cluster key concepts based on their distances, to find best key concept group for a current user. Each user has four kinds of key concept lists, which are an original key concept list automatically extracted from our system and added, core, and deleted key concept lists. When a user decides certain concepts are key concepts while studying, the user can add them to the added key concept list, and then they will be regarded as key concepts although they are not automatically extracted as key concepts. The added concepts may or may not be shown in the lecture notes. A user may highlight the most important key concepts during the class. Then, the highlighted key concepts are listed in the core key concept list; in the same manner, a user can delete key concepts. Using these key concepts, our system computes the distance of each key concept using the TF-IDF method. Through this method, we transform key concepts into numbers (vectors), so our system can understand and calculate the distances between them. Then, we apply the elbow method again to find a proper k before applying the k-means algorithm.

Python codes 3.8. User clustering

```
def student_cluster(self, selected_ans, fig_opt):
        num_cluster = self.elbow_method("student_cluster")
        kmeans = KMeans(n_clusters=num_cluster, max_iter=600, algorithm = 'auto',
random_state=14)
        kmeans.fit(self.stu_data_ans)
        self.kmeans_label = kmeans.labels_
        print("\nStudent cluster: \n", self.kmeans_label)

        #finding the most simliar cluster for a current user based on the user's answers
        selected_ans = np.reshape(selected_ans, (1, -1))
        predicted_student = kmeans.predict(selected_ans)
        self.predicted_student_group = predicted_student[0]
        print("\nCurrent studuent predicted cluster: \n", self.predicted_student_group)

        cluster_map = pd.DataFrame()
        cluster_map['cluster'] = kmeans.labels_
```

```
        self.pred_stu_cluster_group = []
        self.pred_stu_cluster_group.append(cluster_map.cluster == self.predicted_student_group)
```

Python codes 3.9. Key concept clustering

```
def key_concept_cluster(self, fig_opt = 0):
        tf_idf = self.tf_idf_vectorizor.fit_transform(self.stu_data_key_del_AK)
        tf_idf_norm = normalize(tf_idf)
        self.tf_idf_array = tf_idf_norm.toarray()

        pd.DataFrame(self.tf_idf_array, columns=self.tf_idf_vectorizor.get_feature_names())
        #print(self.tf_idf_vectorizor.get_feature_names())

        #reduce dimension to remove non-essential parts
        sklearn_pca = PCA(n_components = 3)
        self.key_concepts_list = sklearn_pca.fit_transform(self.tf_idf_array)

        num_cluster = self.elbow_method("key_concept_cluster")
```

```
        kmeans = KMeans(n_clusters=num_cluster, max_iter=600, algorithm = 'auto',
random_state=14)
        kmeans.fit(self.key_concepts_list)
        self.keyconcept_label = kmeans.labels_
        print("\nKeyconcepts cluster: \n", self.keyconcept_label)

        cluster_map = pd.DataFrame()
        cluster_map['cluster'] = kmeans.labels_

        self.key_concept_rm_duplicate(num_cluster, cluster_map)
```

Now we have two clustering results: user clustering and key concept clustering, and each user belongs to one user cluster and one key concept cluster. Obviously, users in the same user cluster can have different key concept clusters. Then, when a current user is assigned to a specific user cluster, the majority key concept clusters in the user cluster will be matched to the current user. For example, as illustrated in Table 2, there are 30 users and three user clusters (1, 2, and 3) and four key concept clusters (A, B, C, and D). There are nine users in user cluster 1; seven of these users' key concept cluster is A; the other two users' key concept cluster is B. User cluster 2 has 13 users, and so on. When a current user is assigned to user cluster 1, our system finds the most common key concept clusters in that user cluster. In this example, the majority key concept cluster is A, so our system matches the current user with key concept cluster A.

Table 2. Number of key concept groups for each user group

User group	Number of Key concept group
9 users in user cluster 1	7 of 9 users in key concept cluster A 2 of 9 users in key concept cluster B
13 users in user cluster 2	3 of 13 users in key concept cluster A 3 of 13 users in key concept cluster B 3 of 13 users in key concept cluster C 4 of 13 users in key concept cluster D
8 users in user cluster 3	5 of 8 users in key concept cluster C 3 of 8users in key concept cluster D

However, this method has one problem. Every user assigned to user cluster 1 will be matched with key concept cluster A, because we have ignored minority clusters. To handle this problem and give users a choice, we propose a prior knowledge–based key concept clustering. As we explained earlier in this section, there is a deleted key concept list; here, we have divided this list in two: one is concepts a user deletes because the user Already Knows (DAK) the key concepts, and the other is concepts deleted because the user thinks that they are Out of Bounds (DOB). Using a DAK key concept list, we can assume the user's prior knowledge. Thus, instead of using all key concepts, we use only a DAK key concept list for key concept clustering.

After clustering, we present the user with the DAK key concept list of all key concept clusters for a specific user cluster. Thus, when a user is assigned to user cluster 1, if the key concept clusters are DAK key concept clusters in Table 2, the lists of DAK key concept cluster A and B would be presented to the user. Then, the user is asked to choose the list that contains more key concepts the user already knows. Although the majority of DAK key concept cluster is A in user cluster 1, if the user chooses DAK list B, the user is matched to key concept cluster B. This way our system can find the best key concept cluster for a current user according to the user's prior knowledge.

After matching a key concept list to the current user, our system automatically generates a lesson plan using the key concept lists, and makes use of added and core key concepts

in the lesson plan to encourage or critique as a critical friend. The added key concept list can be used to recommend further study. This list contains key concepts added by previous users who had similar prior knowledge. Thus, these added key concepts may also be interesting to the current user. We can also use the core key concept list for other purposes: a user can use the list to prepare for a class, because the list represents important concepts of the lesson; the system can use the list to check whether the user is following the class well by asking questions; finally, the list can be used for review or as a summary at the end of the class.

Python codes 3.10. Lesson plan

```
def lesson_plan(self):
        if self.selected_ans == []:
                not_sel_msg = tk.Toplevel()
                tk.Label(not_sel_msg, text="Please finish 'File -> Initial Set' first").grid(row=0)
                return

        self.lesson_added_concepts = self.data_analysis.recommand_further_study()
        self.lesson_core_concepts = self.data_analysis.make_question_for_summary()

        lesson_plan_pop = tk.Toplevel()

        pop_lb = tk.Text(lesson_plan_pop, bd=0, bg="#FCF3FF", font="Arial")
        pop_lb.pack()
        pop_lb.insert(tk.END, "+++ Lesson Plan +++\n\n")
        pop_lb.insert(tk.END, "Lesson Title: \n")
        pop_lb.insert(tk.END, self.lesson_title)
        pop_lb.insert(tk.END, "\n\nLesson Object: \n")
```

```
        pop_lb.insert(tk.END, self.lesson_object)
        pop_lb.insert(tk.END, "\n\nPrerequisite: \n")
        pop_lb.insert(tk.END, self.lesson_prerequisite)
        pop_lb.insert(tk.END, "\n\nPreview: we can make questions using following core key concepts\n")
        pop_lb.insert(tk.END, self.lesson_core_concepts)
        pop_lb.insert(tk.END, "\n\nAssessment: we can make assessments using following core and added key concepts\n")
        pop_lb.insert(tk.END, self.lesson_core_concepts)
        pop_lb.insert(tk.END, " ")
        pop_lb.insert(tk.END, self.lesson_added_concepts)
        pop_lb.insert(tk.END, "\n\nReview and Summary: we can review and summarize using following core key concepts \n")
        pop_lb.insert(tk.END, self.lesson_core_concepts)
        pop_lb.insert(tk.END, "\n\nExpected Duration Time: \n")
        pop_lb.insert(tk.END, self.lesson_duration_time)
        pop_lb.insert(tk.END, " hours")
        pop_lb.config(state=tk.DISABLED)
```

Our system also gathers information on duration by asking users how much time they spent on the lesson at its end. Duration information, as well as all user and key concept data, are stored in our database.

The automatically generated lesson plan contains a lesson title, objectives, prerequisites, preview, assessment, review, and expected duration.

The system can also communicate with users through its input and chat windows. Currently our system can respond to greetings and answer simple questions. If the user wants to know more about a certain concept while studying, they can ask about the concept using the input window and the system will answer using related link and chat windows. A user can find open educational content by clicking the links in the related link window. To make our system understand a question, the user must currently use the special character '@', as illustrated in Figure 7.

Figure 7. Use of special character '@' usage

Python codes 3.11. Simple chatting

```
def send(self, input_win, chat_win, html_label):
        msg = input_win.get("1.0",'end-1c').strip()
        input_win.delete("0.0",tk.END)
        if msg != '':
                        chat_win.config(state=tk.NORMAL)
                        chat_win.insert(tk.END, "You: " + msg + '\n')
                        chat_win.config(foreground="#442265", font=("Verdana", 10))
                        res = self.chatbot_response(msg, html_label)
                        #time.sleep(0.5)
                        chat_win.tag_config("bot", foreground="blue", justify="right")
                        chat_win.insert(tk.END, "Tutor: " + res + '\n', ("bot"))
                        chat_win.config(state=tk.DISABLED)
                        chat_win.yview(tk.END)

#It is needed for answers for question using '@'
def chatbot_response(self, text, html_label):
        r_text = self.is_contained(text)
```

```
        if r_text == "":
                return self.chat.respond(text)
        else:
                u_text = self.make_urls(r_text)
                html_label.set_html('<a href='+ u_text[1] +'>Wikipedia</a>'+','+'<a href='+ u_text[2]
+'>Google </a>')
                f_text = 'Click above link \n' + 'Or try below:     \n' + '" ' + u_text[3] +' "'
                return f_text
```

3.4 Test

Since our proposal is a fairly new approach, no existing test data was available. Thus, we have to collect test data to check the feasibility of our system. The best way for collecting data is to gather data from students who are actually taking a class; however, due to time limitations, we created an artificial dataset for testing.

We picked one lecture note from an actual class on software engineering and made some assumptions about the virtual user's data, as well as the user's added, core, and deleted key concept data.

> *Assumption 1: The lecture, software engineering, is mainly for second- or third-year university students majoring in computer science*
>
> *Assumption 2: Students majoring in computer science have more knowledge of the lecture than those who do not major in computer science*
>
> *Assumption 3: Students whose major and knowledge are similar add, delete, and highlight similar key concepts*
>
> *Assumption 4: First-year students and those with low knowledge of the topic add and highlight fewer key concepts than those in their second- or third-year*
>
> *Assumption 5: Third-year students have the most prior knowledge, followed by second- and first-year students*

Based on these assumptions, we created 30 sample users. When a current user starts our tutoring system, our system analyzes the sample data, clusters them, and then prepares for

the user's input. After the user's DAK list selection, our system provides a tutoring service to a current user using an automatically generated lesson plan.

In addition, we tested several contents written in English from the KFTC English website. Using the documents, we tested preprocessing and key concept extraction. In order to perform further test, we need other assumptions for user data and key concept data. However, since we already made artificial data for testing the feasibility of our tutoring system, it is not necessary to do the same process again. Hence, we tested the data up to the key concept extraction process.

Nevertheless, when our tutoring system can collect users' data and added, deleted and core key concept data from employees in KFTC using this system, our system can provide all the tutoring services described in this project to them.

Chapter 4

Results

The test results and evaluation of our system are described in this section. Our enhanced tutoring system based on key concepts shows that although there are still some restrictions, the system can deliver personalized educational content to a user.

4.1 Key Concept Extraction

In order to determine suitable key concepts for our system, we tested two of the most popular methods for keyword extraction: the TF-IDF and RAKE methods. We applied each to the same lecture notes (randomly selected from among real lecture notes from software engineering in the computer science department) and compared the results (Table 3). Because of the characteristics of each method, the TF-IDF method extracted one or a few words and the RAKE method extracted key phrases. Since our system generally needs key concepts consisting of a few words rather than long phrases, the TF-IDF method is more suitable.

However, if we apply the TF-IDF method without any

constraints, most high-ranked key concepts are single words, or unigrams, as shown in Table 3. As mentioned in Section III.C, we also need to extract bigrams and trigrams that contain the unigram. This led me to develop constraints that allow a user can control the number of each n-gram. Users can set how many unigrams, bigrams, and trigrams are needed by setting values in a configuration file. Table 4 shows that how the setting values in a configuration file are affecting the number of key concepts. A user sets the number of unigrams first, and then sets two more values: first, how many bigrams and trigrams should be extracted; and second, a threshold value that controls the final number of n-grams. Only bigrams and trigrams with scores higher than the threshold value are selected as key concepts. If we set a low threshold, many n-grams are extracted, and many are likely not important concepts, meaning the user must manually delete them. On the other hand, if the threshold is too high, there would be too few bigrams and trigrams, meaning a user may need to add many key concepts manually. Therefore, it is important to set a proper threshold value. There are no clear rules; this value depends on the type of lecture notes. Based on my experiments, I have assumed that it is better to have no more than 30 key concepts for a one-hour lecture.

Table 3. Comparison between TF-IDF and RAKE

Rank	TF-IDF	RAKE
1	process	articles software project cost estimates using cocomo ii
2	cmmi	using good practice whilst higher levels require
3	improvement	capability maturity model integrated cmmi process improvement framework
4	level	constructive cost model ii cocomo ii
5	process improvement	capability maturity model integration cmmi framework
6	software	specific software vendor empirical model based
7	cmmi level	process improvement cycle involves process measurement
8	technique	support fact based decision making
9	estimate	revised maturity framework cmmi introduced
10	key point	promote software technology transfer particularly

Table 4. Configuration test

Case	Unigram	Bigram	Trigram	Total number of key concepts
NumU:2 Num:20 Threshold: 3	process, cmmi	process improvement (total: 1)	(total: 0)	3
NumU:2 Num:20 Threshold: 1	process, cmmi	process improvement, cmmi level, etc. (total: 6)	process capability assessment, cmmi level optimized, etc. (total: 6)	14
NumU:5 Num:20 Threshold: 3	process, cmmi, improvement, level, software	process improvement (total: 1)	(total: 0)	6
NumU:5 Num:20 Threshold: 2	process, cmmi, improvement, level, software	process improvement, cmmi level (total: 2)	(total: 0)	7
NumU:5 Num:20 Threshold: 1	process, cmmi, improvement, level, software	process improvement, cmmi level, etc. (total: 9)	process capability assessment, process improvement stage, etc. (total: 7)	21

NumU: Number of unigram
Num: Number of n-gram

Table 5. KFTC documents test

Documents	Unigram	Bigram	Trigram
KFTC Chairperson attends the ICN Annual Conference to discuss the role of competition authorities in the digital economy	kftc, competition, policy, platform, chairperson	digital economy, online platform, chairperson sungwook, sungwook joh, competition authority, annual conference, policy development, coming year, commerce act, community grasp	chairperson sungwook joh, grasp competition authority, coming year opportunity, commerce act protect, community grasp competition, sungwook joh attendance, competition authority police, share achieved policy, conference great korea, confirmed authentic english
KFTC commences the consent order process for Apple Korea	mobile, measure, corrective, apple, consent	consent order, mobile carrier, corrective measure, order process, detail corrective, opinion stakeholder, apple korea, consumer smes, business operation, carrier share	consent order process, detail corrective measure, order process june, benefit consumer smes, mobile carrier share, mutual benefit consumer, program developer consumer, smes program developer, initiate consent order, abuse superior bargaining
Tesla Revised Unfair Terms upon the Korea Fair Trade Commission's Order	damage, order, tesla, delivery, vehicle	tesla korea, bad faith, delivery period, prior revision, act negligence, intentional act, cancel order, revision tesla, order fee, caused intentional	intentional act negligence, revision tesla korea, caused intentional act, delivery period including, damage caused intentional, car delivery elapsed, damage period car, period car delivery, vehicle order agreement, responsibility damage period

KFTC holds joint symposium to discuss data monopoly, competition, and consumer issues	data, professor, issue, monopoly, competition	data monopoly, monopoly competition, digital economy, consumer issue, competition consumer, university professor, big data, market dominance, information protection, national university	data monopoly competition, monopoly competition consumer, competition consumer issue, digital economy data, chairperson sungwook joh, seoul national university, law challenge professor, joo presentation consideration, kftc continuously communicate, kftc participated panelis

Website: http://ftc.go.kr/eng/cop/bbs/selectBoardList.do?key=515&bbsId=BBSMSTR_000000002402&bbsTyCode=BBST18

In addition, we tested several documents from KFTC English website. We collected five unigrams and ten bigrams and trigrams. Table 5 shows the results. Depending on the purpose, we can remove unnecessary words such as kftc by adding them in stopwords list.

4.2 OER Services

Our system can provide a lesson using the extracted key concepts that link to OER online materials as an AHS service. Currently, there are six OER online services in our OER service list, but more can be added. A user can make a personal OER service list by editing the list, as illustrated in Figure 8. These selected services are shown when a user clicks one key concept in the key concept window. Then, one of OER services is reached by clicking on its name. According to target user groups, OER list can be different. For instance, when KFTC employees use this system, KFTC homepage can be added in the OER service list.

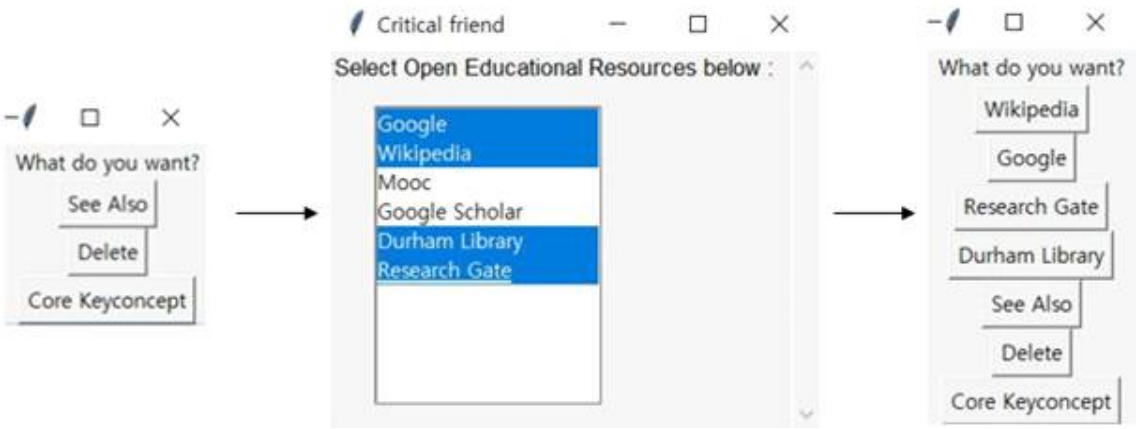
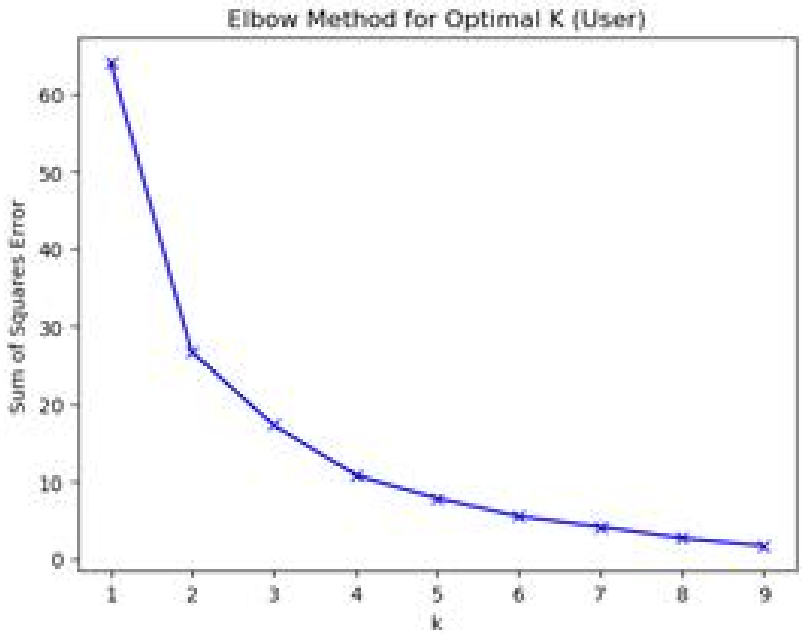


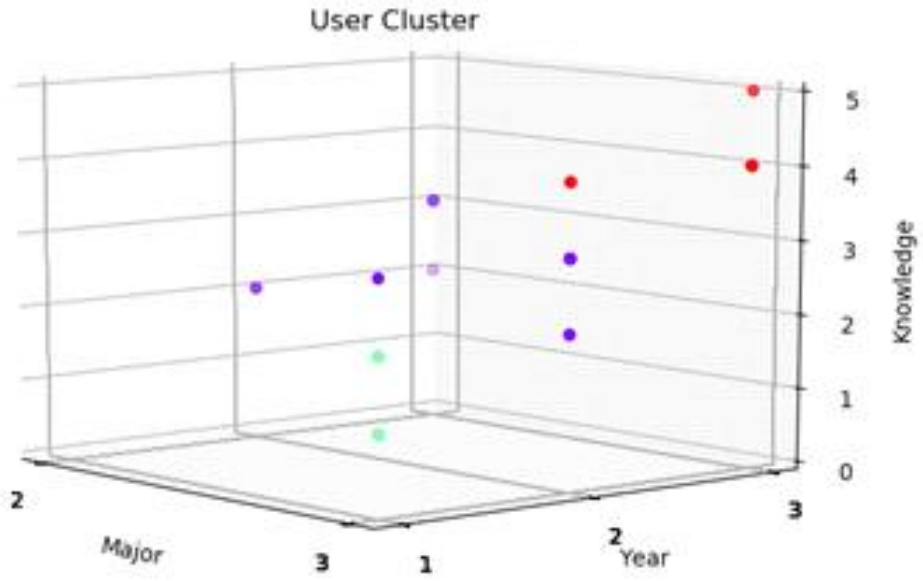
Figure 8. Personalizing which open educational resources are displayed

4.3 Clustering

As described in Section Ⅲ.C, this project uses two kinds of clusters: user and key concept clusters. We first made a user model. Then, we collect user data based on the user model by asking questions. These questions can be added, deleted or modified through a configuration file. The current system uses the three questions described earlier. After collecting data, our system applies the elbow method.



(a) Elbow method

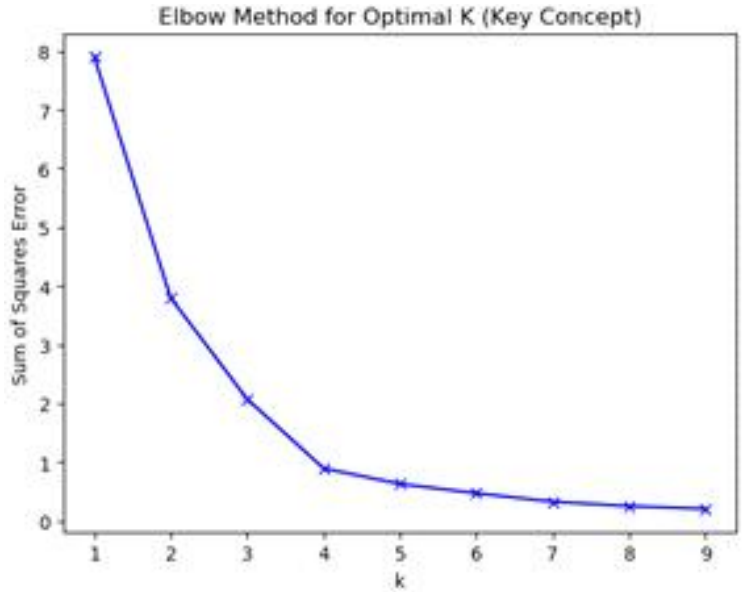


(b) User cluster

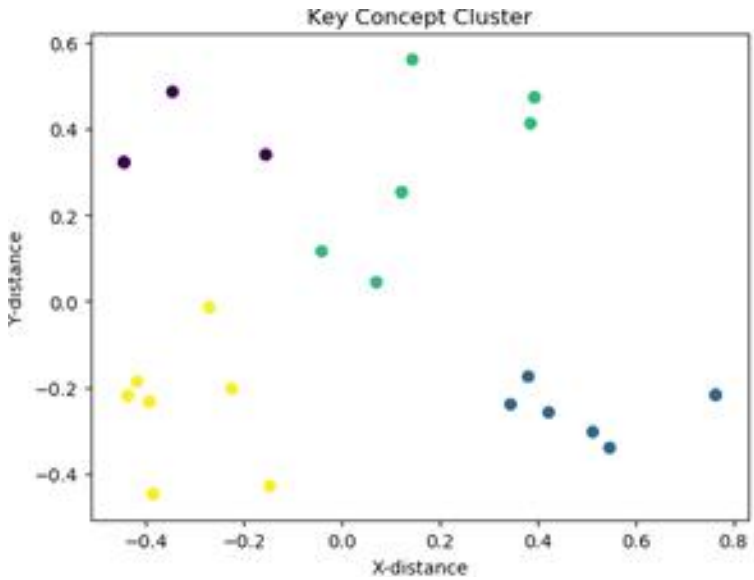
Figure 9. Elbow method and cluster for user data

Figure 9(a) illustrates the result of applying the elbow method to 30 sample user data. When k is 3 or 4, the inflection point of the elbow curve becomes visible. Figure 9(b) shows the k-means cluster result for user data when k is 3. It appears to include fewer than 30 samples because some user data overlapped multiple times.

For key concept data, we used a DAK key concept list. We first perform the TF-IDF method to transform DAK key concepts into number vectors. Then, we use the k-means algorithm to cluster DAK key concepts in terms of distances among them. Figure 10(a) displays the elbow method result for DAK key concept data, and shows that the inflection point of the elbow curve appears when k is 4. Figure 10(b) presents the DAK key concept cluster result for a k of 4.



(a) Elbow method



(b) Key concept cluster

Figure 10. Elbow method and cluster for DAK key concept data

The user cluster and key concept cluster for each user is shown in Table 6. Using this information, our system can match the best user and key concept clusters with the current user.

Table 6. User cluster and key concept cluster for each user

UserID	User cluster	Key concept cluster	UserID	User cluster	Key concept cluster
1	1	A	16	2	C
2	1	A	17	2	D
3	1	A	18	2	C
4	1	A	19	2	D
5	1	A	20	3	C
6	2	A	21	3	D
7	1	B	22	3	C
8	1	A	23	2	D
9	1	A	24	2	D
10	1	B	25	2	A
11	2	A	26	3	D
12	2	B	27	3	C
13	2	B	28	3	D
14	2	B	29	2	C
15	3	C	30	3	C

4.4 Lesson plan

After the matching process, our system generates a lesson plan for a current user. To do so, the system needs input data from a lecturer as well as user cluster and key concept information. There is a limit to the system's ability to generate a lecture title or object using only extracted key concepts, especially in its initial stages. Hence, the lecturer providing the lecture notes was asked to enter a lecture title, object, and prerequisite in a configuration file. The system can automatically generate the preview, assessment, and review and summary, using key concepts, as illustrated in Figure 11. As explained in Section Ⅲ.C, the system can provide core key concepts before the class begins, so a user can preview the lecture in advance, and later ask questions using the key concepts to check whether the user understand the lecture correctly. The key concepts can be used for review or summary as well. In addition, the system

can use added key concepts to suggest assessments for further study. However, the actual use of these core and added key concepts is not implemented in the current version of the system because it is outside the scope of this project. Apart from the use of key concepts, our system estimates the duration of the lesson by analyzing time data from the user cluster.

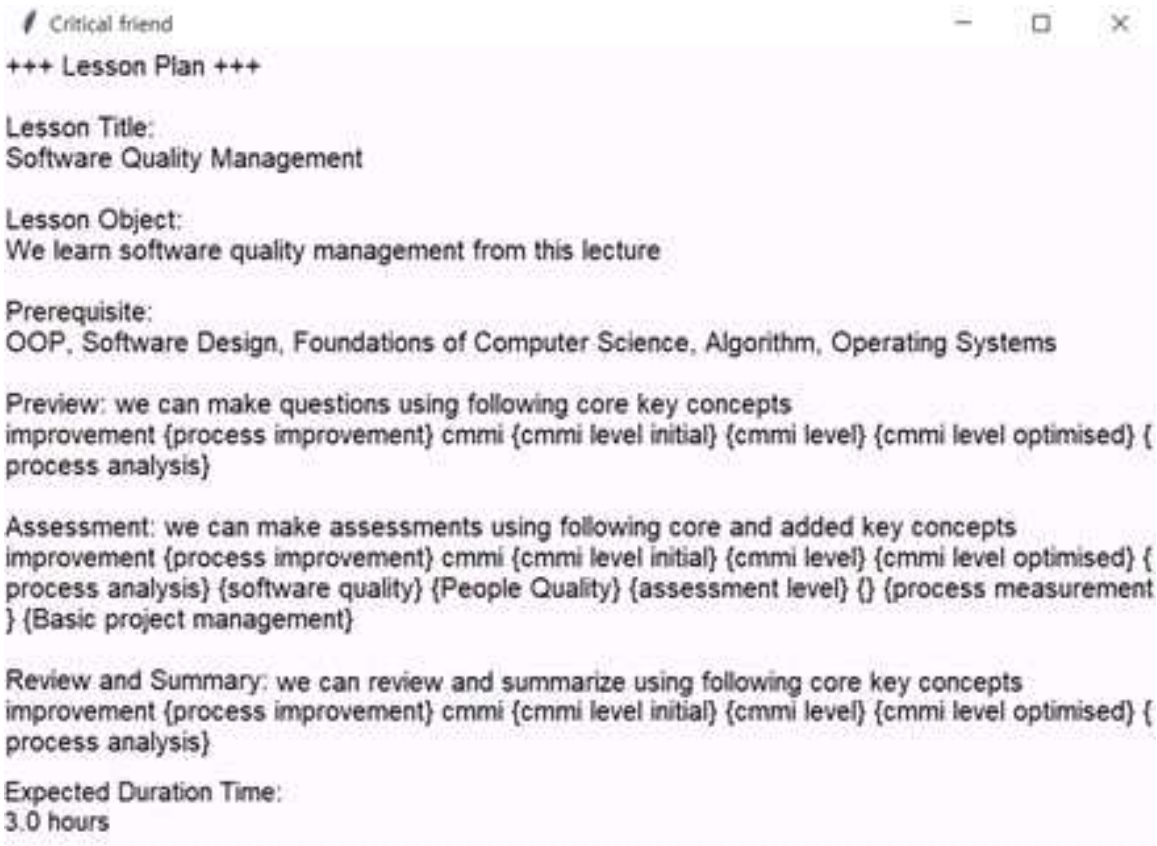


Figure 11. Sample lesson plan

4.5 Database

To store all user data and key concepts, our system uses a csv file format which is plain-text file format commonly used for data analysis. We also developed a database using sqlite3 engine for expandability based on the ERD in section Ⅲ.A.

Chapter 5

Evaluation

In this subsection, I evaluate the strengths and limitations of our solution and briefly discuss how to further improve our system.

5.1 Strengths

I presented our enhanced tutoring system and showed that it can deliver OER services using key concepts. Our system mainly uses automatically extracted key concepts for tutoring. Key concepts are core content of the lecture, so it is reasonable to use them for tutoring a user on the topic. Key concept extraction can be easily developed because many kinds of keyword extraction libraries are available; depending on the materials, we can choose an appropriate extraction approach and extract key concepts. Our system uses k-means clustering to assign each user to the most appropriate user group and key concept group. This clustering method is easy to implement but a powerful way to divide data points into a number of groups, a capability our system needs to group user and key concept data.

Our system provides an OER list. Instead of searching for concepts from individual resources, a user can access several OERs presented by our system. Currently, six OERs are available, and this number can be increased according to users' needs.

Our tutoring system allows a user to make an individualized key concept list and OER list. Initial key concepts are given to every user, from which the user can add and delete concepts. A user also can highlight some key concepts to mark them as core key concepts. By doing this, a user can develop a personalized key concept list. Moreover, every user has different preferences for OERs, so our system lets users choose OER options from an editable list.

Lastly, our system can generate a lesson plan by using extracted key concepts. This lesson plan is provided to a user and also used by the system to tutor the user.

5.2 Limitations

The major limitation of our project is that artificial test data was used for its testing. Since our system is new and in an initial version, no test data was available. Due to time constraints, it was difficult to find volunteers to use this system. Thus, we developed data for 30 artificial test cases using the rules explained in Section III.D.

User participation, especially at the initial stage, is crucial to make our system more useful. The current system can only partially anticipate a user' s behaviors, because it is new and does not currently have enough data as a basis for analysis and prediction. Hence, our system needs to encourage users to add and delete key concepts and make a personalized key concept list for future analysis. In addition, for better analysis, sincerity from the user is important.

Another limitation is the quality of extracted key concepts. For the test, we used PowerPoint lecture notes and the TF-IDF method. It is difficult to say that the TF-IDF method is the best method for all kinds of written materials (e.g., textbooks). Depending on the keyword extraction methods and the type of data, the quality of the extraction may vary.

Lastly, since we do not have enough time to develop and test both English and Korean language, our current system can only process English characters.

5.3 Improvement

This project was focused on determining the technical feasibility of a tutoring system that can present meaningful content to a user. Hence, the user interface is relatively crude. User interface is vital for a program that requires user interaction. It is always better to make a system easy for its users, and in this case, that will entail an improved user interface.

Furthermore, when the system gathers enough user data, it can analyze the data to determine the best user cluster without asking questions.

Last but not least, our current system can only extract key concepts from English text data; its expansion to use different types of data (such as audio or video data) and characters (such as Korean), might be more effective.

Chapter 6

Conclusion

The aim of the present project was to provide a proof of concept for an enhanced tutoring system that can deliver a lesson on a given topic using simple data analysis. In this work, we proposed an enhanced tutoring system based on key concepts. The method clusters users according to their answers to simple questions, then assign a current user to the closest user group. At the same time, the system analyzes key concepts selected by each user and then clusters the key concepts in terms of their similarity. After the clustering step, each user is assigned to one clustered user group and key concept group. Consequently, each user in the same user group can have a different key concept group. Then, the system displays the different key concept groups of the current user's user group for selection. After the user's selection, the system matches the current user with the selected key concept group. With this matched key concept group, our system automatically generates a lesson plan. Within the lesson plan, the system can ask questions, encourage, or even offer friendly critique to the user, before, after, and during the class. Furthermore, each key concept is directly connected to open education resources, so a user can access them with one click.

We tested our system using data for 30 artificial users and one set of lecture notes used for a software engineering class. Users were clustered into three groups and key concepts clustered into four groups. A current user is assigned to one of the user groups and then paired up with one of the key concept groups according to the user’ s choice. Using this information, a lesson plan is generated, and this lesson plan is used for tutoring. The user can also make a personalized key concept list and open education resource list by editing the lists. In addition, the system can reply to users’ simple questions by providing online links to open educational resources. We also tested several documents from KFTC English website to prove the feasibility of our tutoring system for enhancing the KFTC's surveillance capabilities.

However, these results may not be applicable to all situations. In order to realize such services, user participation at the initial stage is very important, at least until the system has gathered enough data to analyze and subsequently predict users’ actions. Once the system secures enough data, it can provide better services without much user intervention. To deliver a tutoring service, our system must first extract key concepts, but our current system can extract concepts only from text documents. Moreover, the services can be affected by the quality of key concept extraction methods.

Although there are some limitations and further work is required, I demonstrate that our enhanced tutoring system can provide a lesson for a specific topic based on given data and help users

to learn contents in the big data. Therefore, this system can be used for strengthening KFTC's fair trade surveillance capabilities by using the big data. Further research should be undertaken to investigate methods for extracting key concepts from other resource types, such as audio and video data, rather than just text data, and other characters such as Korean. It is more convenient for users if our system requires less input from the users. Our current system explicitly asks the users questions and about preferences rather than anticipating behavior. It can be improved by providing a service without much user intervention by implicitly observing their interaction with our system.

References

Aizawa A., (2003). “An information-theoretic perspective of tf-idf measures”, Information Processing & Management, Vol 39(1), pp. 45-65.

Al-Hanjori M. M., Shaath M. Z., and Abu Naser S. S., (2017). “Learning computer networks using intelligent tutoring system”, International Journal of Advanced Research and Development (2), 1.

Astleitner H. and Hufnagl M., (2003). “The Effects of Situation-Outcome-Expectancies and of ARCS-Strategies on Self-Regulated Learning with Web-Lectures”, Journal of Educational Multimedia and Hypermedia, 12(4): 361-376.

Baldoni M., Baroglio C.,and Patti V., (2004). “Web-Based Adaptive Tutoring: An Approach Based on Logic Agents and Reasoning about Actions”, Artificial Intelligence Review 1(22): 3-39.

Beel J., Gipp B., Langer S., and Breitinger C., (2016). “Research-paper recommender systems: a literature survey”, International Journal on Digital Libraries 17(4), pp. 305-338.

Benyon D. and Murray D., (1993). “Applying user modeling to human-computer interaction design”, Artificial Intelligence Review 7, pp. 199-225.

Bird, Steven, Edward L. and Ewan K. (2009), Natural Language Processing with Python. Available at: https://www.nltk.org/ (Accessed: 31 August 2020).

De Bra P., Brusilovsky P, and Houben G. J., (1999). “Adaptive hypermedia: from systems to framework”, ACM Computing Surveys, 31 (4): 12.

Dogan B., and Camurcu A. Y., (2008). “Visual Clustering of Multidimensional Educational Data From an Intelligent Tutoring System”, Computer Applications in Engineering Education, Vol. 18, pp. 375-382.

Hipp R. D., (2020). SQLite. Available at: https://www.sqlite.org/index.html (Accessed: 31 August 2020).

Hu X. and Wu B., (2006). “Automatic keyword extraction using linguistic features,” IEEE International Conference on Data Mining, 6:19-23.

Jong-Pil C., Jang-Mi P., Sun-Gwan H., and Chul-Hwan L., (2002). "Automated Lesson Planner System for ICT Education", International Conference on Computers in Education, Vol.1, pp. 485-489.

Kaur J. and Gupta V., (2010). “Effective Approaches For Extraction Of Keywords”, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 6, pp. 144−148.

Kouno S., Yokoyama S., Nakamura N., Yonezawa N., and Miyadera Y., (2002). "Development of generator for lesson plan making support systems", International Conference on Computers in Education, Vol.2, pp. 1181−1185.

Luckin R. and Cukurova M., (2019). “Designing educational technologies in the age of AI: A learning sciences−driven approach”, British Journal of Educational Technology, 50 (6): 2824− 2838.

Luhn H. P., (1957). “A Statistical Approach to Mechanized Encoding and Searching of Literary Information”, IBM Journal of Research and Development, 1:309−317.

Lundh F., (1999). An introduction to tkinter. Available at: http://www.pythonware.com/library/tkinter/introduction/index.html (Accessed: 31 August 2020).

MacQueen J., (1967). “Some methods for classification and analysis of multivariate observations”, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, University of California Press, Berkeley, Calif., pp. 281−297.

Matsuo Y. and Ishizuka M., (2004). “Keyword Extraction from a Document Using Word Co-Occurrence Statistical Information”, International Journal of Artificial Intelligence Tools, 13(1):157-169.

McKinney W. and others, (2010). “Data structures for statistical computing in python”, In Proceedings of the 9th Python in Science Conference, Vol. 445, pp. 51-56. Available at: https://pandas.pydata.org/ (Accessed: 31 August 2020).

Nagro S. A., Fraser, D. W., Hooks S. D., (2019). "Lesson Planning With Engagement in Mind: Proactive Classroom Management Strategies for Curriculum Instruction". Intervention in School and Clinic. 54 (3):131-140.

Nguyen L., (2014). “User Model Clustering”, Journal of Data Analysis and Information Processing, 2:41-48.

Nguyen L., Do P., and Fr□schl, C., (2008). “Learner Model in Adaptive Learning”, in Proceedings of Word Academy of Science, Engineering and Technology.

Nwana H. S., (1990), “Intelligent Tutoring Systems: an overview”, Artificial Intelligence Review 4, pp. 251-277.

Pedregosa F. *et al.*, (2011). JMLR 12, pp. 2825-2830. Available at: https://scikit-learn.org/stable/ (Accessed: 31 August 2020).

Rajaraman A. and Ullman J., (2011). “Data Mining”, In Mining of Massive Datasets, Cambridge: Cambridge University Press, pp. 1-17.

Rose S., Engel D., Cramer N., and Cowley W., (2010). “Automatic Keyword Extraction from Individual Documents”, Text Mining: Applications and Theory. New York: JohnWiley & Sons, Ltd., 1:1-20.

Sang-Woon K., Joon-Min G., (2019). ”Research paper classification systems based on TF-IDF and LDA schemes”, Human-centric Computing and Information Sciences, 9(1):30-50.

Somyurek S., (2015). “The New Trends in Adaptive Educational Hypermedia Systems”, International Review of Research in Open and Distance Learning, 16(1):221-241.

Syakur M. A., Khotimah B. K., Rochman E. M. S., and Satoto B. D., (2018). “Integration K-Means Clustering Method and Elbow Method For Identification of The Best Customer Profile Cluster”, IOP Conference Series: Materials Science and Engineering, 336(1).

Virvou M., Troussas C., and Alepis E., (2012). “Machine learning for user modeling in a multilingual learning system”, International Conference on Information Society (i-Society 2012), pp. 292-297.

Zhang Q., Wang Y., Gong Y., and Huang X., (2016). “Keyphrase extraction using deep recurrent neural networks on Twitter”, In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 836-845.