Natural Language Generation for Intelligent Tutoring Systems

by

Do Dung VU

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, MAY 16th, 2022

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Prof. Sylvie Ratté, Thesis supervisor
Département de génie logiciel et des TI, École de technologie supérieure

Prof. Tan Pham, President of the board of examiners
Département de génie mécanique, École de technologie supérieure

Prof. Luc Duong, Member of the jury
Département de génie logiciel et des TI, École de technologie supérieure

Prof. Thang Le Dinh, External independent examiner
Département Marketing et systèmes d'information, Université du Québec à Trois-Rivières

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON APRIL 5th, 2022

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

I had committed to pursuing my Ph.D. when I left South Korea after working there for 18 months in the machine learning field. I came to Canada with great zeal and a hopeful future. However, the first company that promised to give me the fund for my Ph.D. project with the application research of computer vision suddenly ceased after a few months without any reason. My supervisor Prof. Sylvie Ratté gave me a concept of the applications of Natural Language Processing. Although I did not understand this conception much, I figured out that it was an amazing technology. It will avail human and machine communication and be the future of many genuine-world applications.

Getting a research challenge without any fund in Canada in parallel with taking care of my own family was at once not only a prodigiously exhilarating but also an affrighting time. I had never been to Canada before, did not know anyone there, and did not realize the French language barrier and the winter weather. Auspiciously, the supervisors and students in the lab were very welcome and subsidiary. They had originated from abroad with many scenarios, hence we could communicate easily and apportion the positive energy to each other. After culminating the research time in the lab, I tried to contact many companies to obtain the fund for my Ph. D, there of them contacted me. Unfortunately, most of the company's focuses were out of my research. However, I met Iulian Serban who was the CEO/CTO of Korbit Inc. The mission of Korbit is to implement the Artificial Intelligence tutor. Hence, we had a great discussion about my research and the way of obtaining funds. Through our discussion, I decided to fixate on my research of natural language generation for intelligent tutoring systems. With my supervisor support, a few months later, I obtained the grant from Mitacs which is a nonprofit national research organization that, in partnerships with Canadian academia, private industry, and regime, operates research and training programs in fields cognate to industrial and convivial innovation. The Mitacs grant has coalesced between Korbit Inc and Government fund. I joined Korbit as an intern to study and work under the supervisor of professor Sylvie Ratté and Iulian Serban. I'd relish thanking each of them for their time, work, and support throughout the years. Without them, my Ph.D. degree would never have become true.

In the beginning, I mentally conceived that a Ph.D. fixates on reading the paper, doing the experiment, and inscribing paper. However, I deduced that a Ph.D. is not only a long journey of revelation, and exploration, but it is a hard training period in both academic and industrial areas with many critical questions and answers, where both of them are always obnubilated. During the Ph.D. program, I learned many things to answer the most consequential questions *Why* and *How* when I solve a given quandary. By doing the literature survey, asking, and applying the many research methodology(e.g., understanding and being proficiently adept at the dependencies, software libraries, indite a scientific paper, etc.,), I found the solution for this problem. It was a long and hard journey. I would relish thanking all of my collaborators whom I have worked with during my Ph.D. I would relish to thank preceptor Sylvie Ratté who fortifies, edifies, engages, and mentors me throughout the years. I would like to appreciate Iulian Serban, and Korbit members who give me a great internship environment, a chance of working and researching in Natural Language Processing. I would like to thank my lab mates at ÉTS who always give me feedback when I prepare the presentation. Last, I would relish thanking the thesis jury members for reading my thesis and providing their comments.

Conclusively, and most of all, I would relish thanking my profound appreciation, my wife and partner in life, Trang Nhung, our daughter Ha An, our son Minh Huy. Trang Nhung has stood by my side for 10 years, and she has always helped me to shoulder the inevitably ineluctable ups and downs of my life. Ha An and Minh Huy have given me the happiness, potency, and purport which avail me to pursue the scientific achievement.

Génération de langage naturel pour les systèmes de tutorat intelligents

Do Dung VU

RÉSUMÉ

Ce projet aborde le problème de la génération automatique de texte pour un système de tutorat intelligent conversationnel (ITS). L'ITS conversationnel pose des questions à l'étudiant, puis analyse sa réponse à l'aide d'algorithmes d'apprentissage automatique. Lorsque la réponse de l'élève est classée comme étant incorrecte, le système doit donner un indice approprié pour l'aider à répondre à la question ou à comprendre le problème. Le système est alimenté par une base de données de questions, réponses et astuces. Le composant générant les conseils est appelé le modèle de « génération de conseils ». À cette fin, nous proposons un modèle de génération de feedback et d'astuces capable de générer des astuces pédagogiques significatives, personnalisées. Pour rendre le cours plus engageant et efficace pour chaque étudiant, nous proposons en outre un modèle de « génération de contenu » capable de générer un nouveau contenu, y compris des questions, des réponses et des conseils, automatiquement à partir de texte non structuré (tel que Wikipédia) ou de manière interactive avec l'aide des enseignants. En outre, nous étudions également comment le système peut être capable d'adapter efficacement son contenu et ses stratégies de tutorat à chaque élève.

Mots-clés: Intelligent Tutor, Artificial Intelligence, Natural Language Processing

Natural Language Generation for Intelligent Tutoring Systems

Do Dung VU

ABSTRACT

This project tackles the problem of automatically generating text for a conversational intelligent tutoring system (ITS). The conversational ITS gives questions to the student and then analyzes their answer using machine learning algorithms. When the student's answer is classified as being incorrect, the system must give an appropriate hint to help them answer the question or understand the problem. The system is powered by a database of questions, answers, and hints. The component generating the hints is called the "hint generation" model. To this end, we propose a feedback and hint generation model capable of generating meaningful, personalized, pedagogical hints. To make the course more engaging and effective for each student, we propose a "content generation" model capable of generating new content, including questions, answers, and hints, automatically from unstructured text (such as Wikipedia) or interactively with assistance from teachers. In addition, we also investigate how the system may be capable of efficiently adapting its content and tutoring strategies to individual students.

Keywords: Intelligent Tutor, Artificial Intelligence, Natural Language Processing

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

4IR	Fourth Industrial Revolution
EdTech	Education Technology
ITS	Intelligent Tutor System
API	Application Programming Interface
QA	Question and Answer
KB	Knowledge Bases
IDF	Inverse Document Frequency
TF	Term Frequency
RNN	Recurrent Neural Network
RNNLM	Recurrent Neural Network Language Model
LSTM	Long-Term Short-Term
GRU	Gated Recurring Units
LSA	Latent Semantic Analysis
SVM	Support Vector Machine
ID3	Iterative Dichotomiser
CART	Classification and Regression Tree
CHAID	Chi-squared Automatic Interaction Detector
IG	Information Gain
HMMs	Hidden Markov Models

EM	Expectation-Maximization
OOV	Out of Vocabulary
UNK	Unknown
NLP	Natural Language Processing
NLU	Natural Language Understanding
AI	Artificial Intelligence
SMOTE	Synthetic Minority Oversampling Technique
AV	Autonomous Vehicle
CAV	Connected and Autonomous Vehicle
STEM	Science, Technology, Engineering, and Mathematics
MOOCs	Massive Open Online Courses
MEMMs	Maximum Entropy Models
CPU	Central Processing Unit
GPU	Graphics Processing Unit
HPC	High-Performance Computing
C.I	Confidence Interval
NLI	Natural Language Inference
SNLI	The Stanford Natural Language Inference (SNLI) Corpus
MNLI	The Multi-Genre Natural Language Inference (MultiNLI) Corpus

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

$\mathbb{N}$ denotes the positive integer set

$P(x)$ denotes the probability of x

$\{.\}$ denotes a set of items

$(.)$ denotes a sequence of items

$\{x_i\}_{i=1}^{I}$ denotes a set of items $x_1, x_2, \cdots, x_I$

$\mathbf{w} = (\omega_1, \cdots, \omega_M)$ denotes a sequence of M discrete symbols

$\mathbb{R}$ denotes the set of real-valued numbers.

$\mathbb{R}^n$ denotes the set of real-valued numbers in n dimensions.

$a \in \mathbb{R}$ denotes a real-valued variable named a.

$[a, b]$, where $a, b \in \mathbb{R}$ and $b > a$, denotes the closed set of real-valued numbers between a and b, including a and b.

(a, b), where $a, b \in \mathbb{R}$ and $b > a$, denotes the open set of real-valued numbers between a and b, excluding a and b.

$\mathbf{a} \in \mathbb{R}^n$ denotes a real-valued vector of n dimensions.

$A \in \mathbb{R}^{n \times m}$ denotes a real-valued matrix of n × m dimensions.

A^T and A^{T} both denote the transpose of the matrix A

$i = 1, \cdots, n$ means that i will take integer values 1, 2, 3, 4, 5 and so on until and including integer n.

$A \times B = (a, b) | a \in A, b \in B$, where A and B are sets of items, denotes the Cartesian product of A and B.

n-gram is a sequence of n consecutive words (or tokens).

$w \in U$, where U is a sequence of tokens, denotes a token inside U

$exp(x)$ and e^x denotes the exponential function of the value x.

$\log(x)$ and $\ln(x)$ denotes the natural logarithm function of the value x.

$\tanh(x)$ denotes the hyperbolic tangent taken of value x.

$f'(x)$ denotes the derivative of the function f w.r.t. variable x.

$\frac{\delta}{\delta(x)} f(x)$ denotes the derivative of the function f w.r.t. variable x.

$\nabla_\theta f_\theta(x)$ denotes the derivative of the function f w.r.t. parameters θ. If θ is a vector, then it denotes the Jacobian matrix

$x \cdot y$, where x and y are vectors or matrices, denotes the element-wise product between x and y.

x often denotes an input variable (e.g., a real-valued variable or an input sequence of string tokens).

y often denotes an output variable (e.g., an output label, such as a user intention label).

ψ and $\hat{\psi}$ usually denote model parameters

θ and $\hat{\theta}$ usually denote model parameters

$P_\theta(.)$ usually denotes the probabilistic model parametrized by parameters θ

$x \sim P_\theta(x)$ denotes a sample of the random variable x following the probabilistic model parametrized by parameters θ

$\mathcal{N}_x(\mu, \sum)$ denotes the probability of variable x under a multivariate normal distribution with mean μ and covariance matrix $\sum$

$x \sim \text{Uniform}(a, b)$ denotes that x is an integer random variable sampled at uniformly random from the set $\{a, a+1, \cdots, b-1, b\}$, with $a, b \in \mathbb{N}$ and $b > a$

$x \sim \text{Uniform}(A)$, where A is a finite discrete set, denotes that x is a random variable sampled at uniformly random from the set A

$\mathbb{E}_{x \sim P_{(x)}}[f(x)] = \sum_x P(x) f(x) = \int P(x) f(x) dx$ is the expectation of the function $f(x)$ w.r.t the random variable x following the distribution given by the probability or density function P

$\text{KL}[Q||P] = -\int Q(x) \log(\frac{Q(x)}{P(x)} dx$ is the Kullback-Leibler (KL) divergence between the two probability distribution $P(x)$ and $Q(x)$

$1_{(.)}$ denotes a Dirac-delta function, which equals one if the statement $(.)$ is true and otherwise equals zero

$\forall$ denotes the *for all* operator

the phrase *s.t* is the abbreviation of the phrase *subject to*

the phrase w.r.t is the abbreviation of the phrase *with respect to*

INTRODUCTION

0.1 Motivation

Over the past decades, computers and the internet have become ubiquitous and essential for connecting society. With the evolution of technology, many applications have been deployed to create a new industrial revolution that helps improve productivity and production capacity in almost life aspects such as finance, health care, education, manufacturing, entertainment, transportation, and communication. Technology has a significant impact on students in their study, such as: enhancing their engagement and enabling them to learn and retain more information. Educational technology (EdTech) also works as the prime force to enhance student motivation. Different factors relevant to EdTech like user-friendliness, users' curiosity for new learning tools, and psychological satisfaction effectively enhance students' intrinsic motivation in the long run rather than increasing extrinsic rewards, which are effective on a short-term basis. Unfortunately, understanding and generating educated content is a very difficult problem. Therefore, it is not surprising that these technologies are still in their infancy. This thesis is motivated by these technology challenges that generate content for higher education. In particular, the thesis focus on generating the natural language hint for the Intelligent Tutoring System (ITS) by using the large text corpora. This thesis proposes a state-of-the-art to quickly scale up personalized education on the ITS with a hint creation effort at low costs.

0.2 Central Hypothesis

By considering many relevance scientific research assumptions, the work in this thesis is built based on several hypotheses. These hypotheses constitute the foundations underlying and motivating the work presented in this thesis. Some are well-established in the field, while others might be more contestable. This section provides an overview and discussion of these hypotheses.

The first key hypothesis of this thesis is that communication between humans and machines is collaborative and be beneficial to all parties. Two interlocutors might have a conversation because they both believe that something to be gained through the conversation. In other words, each interlocutor believes that there exists an alignment between their own goals and the goals of the other party and that by conducting a conversation, they may both benefit from it. Let's consider the example of a dialogue system selling spa tickets. In addition to its primary goal of finding a suitable ticket for a human customer, the system may have a secondary goal to maximize profits by selling the most expensive ticket or adding services with the human customer's spending budget. This secondary goal will directly conflict with the customer if the human customer has a secondary goal of purchasing the cheapest ticket.

The second key hypothesis is that, in general, the human and machine interlocutors only have access to partial information about the state of the world, about the other interlocutor's information and goals, and even about their own goals. For example, the dialogue system selling spa tickets cannot know the goals of a human customer beforehand, such as their skin, their health, hair, or even their spending budget. On the other hand, the human customer does not know which spa tickets are available and at what prices. The human customer may not even know their health or their exact budget. Hence, customers might decide based on the options presented by the dialogue system (e.g., the face, body, or hand and foot treatments, or all of them).

Although these two key hypotheses may appear evident to the avid reader, they go against some of the hypotheses implied by some of the literature on goal-driven dialogue systems. In particular, on *voice command system* research, the assumption implies that a goal-driven dialogue system should work as converting the human speech to an appropriate query and submitting it to an application or a service API. For example, consider the case of a voice-controlled dialing system on the vehicle. This system might only expect the human user to mention a contact (e.g., the

name or phone number), and based on the received command, the dialing service calls the contact. This is not a collaborative dialogue where both parties stand to benefit. This simple system further assumes that the human user has access to all relevant information, including their own goal (e.g., the exact contact phone number).

The final key hypothesis of this thesis is predicated on the premise that humans learn the world and about how to communicate through natural language by interacting with others. For example, consider a student studying the topic *machine learning*, whose task is supervised learning model implementation. She might search and research the cyber world for the kindred implementations and the germane discussion thread on the convivial media network, forum (e.g., Stack Overflow, Facebook, etc.). Suppose that on this discussion thread, another person exposes a solution to a homogeneous quandary and get feedback from others about the bugs, error, issues in the deployment. By reading and understanding the discussion, she might learn and apply it to her implementation. By utilizing her knowledge and experience with what she learned, she might decompose the task into sub-tasks to solve and consummate her project. During the conversation, she may get avail by asking a relevant question. The premise is that a paramount amplitude of information is being learned by visually examining and interacting with others. Hence, humans learn a substantial amount about the world and how to communicate by interacting with others; we suggest the final key posit of this thesis. The postulation is that a machine can additionally learn a paramount quantity of the world. After that, it can communicate with a human by natural language based on its knowledge to help humans understand the world. However, in particular, it is arduous to deploy genuine-world machine learning systems, to keep the last postulation as the further.

0.3 Research objectives and methodology

0.3.1 Objectives

The hint generation module is helpful to teach the student to find the correct solution step-by-step, including how to write an answer, what to pay attention to, and how to express and understand a concept. A major benefit of automatic hint generation is that it can be personalized for each student and fix their knowledge gaps. Moreover, it is also a useful exploration for the future task of automatic generation of the content of courses. Additionally, these generated hints are beneficial for teachers. The teacher could select one hint and refine it, which makes the generated hints more friendly to students.

First, it should ensure the contextual relevance between questions, answers, and the considered concepts. This involves task complex contextual information, such as definitions which generally contain many aspects (e.g., concepts, ideas, equations). Second, it requires generating diversified hints for each concept, making the task more challenging. The key to the conversational system is to answer factual questions from the users, especially for the intelligent system. Most of the current systems for Question Answering (QA) are based on the structured Knowledge Bases (KB) such as Freebase (Bollacker, Evans, Paritosh, Sturge & Taylor (2008)) and Wikidata (Vrandečić & Krötzsch (2014)). Here, the question is converted to a logical form using semantic parsing, which can be queried by the KB to obtain the answer (Fader, Zettlemoyer & Etzioni (2014)).

On the other hand, KBs support only certain types of answer schema; constructing and maintaining them is expensive. In addition, there is an enormous of unstructured knowledge available in the content from Wikipedia, textbooks, social media. In this problem, the relevant materials are utilized to extract the right content. This retrieval-based approach (Voorhees & Tice (2000)) provided a much wider coverage over questions and is not limited to specific answer

schema. Here we argue that depending on the type of question and its' knowledge base may be more appropriate and introduce a new system architecture to automatically generate the assumptive questions and answers for each exercise from unstructured data, including the answer evaluation measurement method.

0.3.2 General Methodology

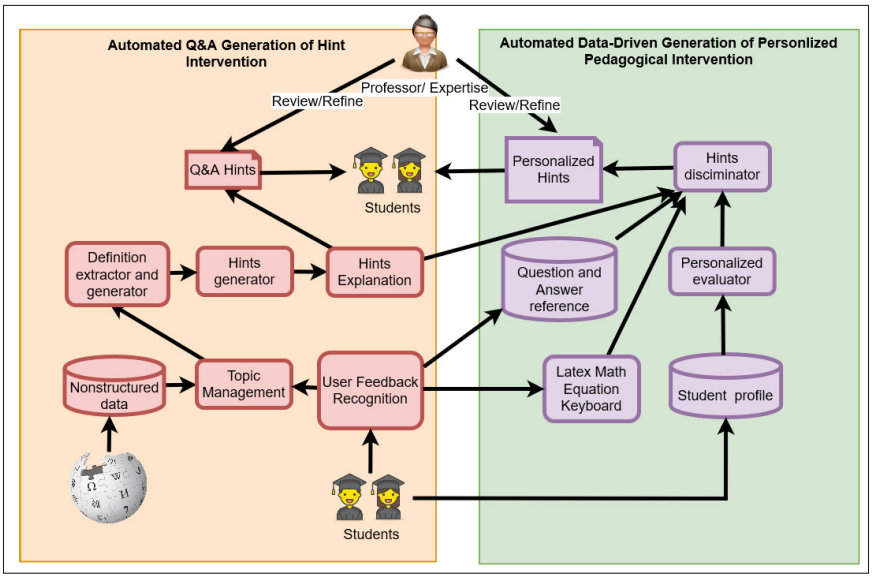

Figure 0.1 The general system architecture

A method to generate new hints is proposed. The hint generator is built on an encoder-decoder framework, and we introduce the attention mechanism and relevance control to boost it. To evaluate our method, we crawl the text stream from non-structured data such as Wikipedia, stack overflow, etc., then perform experiments on it. Automatic evaluation with a score proves that the generated new hints are close to human hints. We generate diversified hints with different topics and different degrees of relevance by utilizing a random sample and the relevance control. The hints discriminator is considered based on the student answer and profile, which is one of the features to make the right hint for the learning student. The hints are generated based on the keyword. A scoring model is proposed to evaluate the value of the keyword with

many features such as the co-reference resolution, the (Term-Frequency - Inverse Document Frequency) TF-IDF, the length of the keyword, etc., Figure 0.1 represents the workflow of the hint generation model.

0.4 Thesis structure

The thesis structure is organized as follows:

Chapter 1 covers background theory to machine learning, natural language processing, and dialogue systems. The chapter is split into three components. The first part presents the probabilistic generative models which form the substratum and act as a framework for some work presented in this thesis. Neural network models are addressed here. The second part introduces the natural language processing techniques with the given frameworks, which are utilized in the thesis. The third part discusses dialogue systems in detail, including system components, methods for optimizing system components, and methods for system evaluation.

Chapter 2 seeks the automatically generate, from an ontology, a personalized, accurate, and syntactically correct paraphrase based on the user's questions and answers in the ITS. State-of-the-art generating hints and their results are presented.

Chapter 3 proposes the automatically generated feedback in a data-driven way methodology and evaluates how the personalizing of feedback can lead to improving the student learning gains. The state-of-the-art machine learning and natural language processing techniques are leveraged to provide hints to students. On the other hand, the experiments were implemented to prove the effectiveness of the feedback in higher education based on real students in a large-scale dialogue-based ITS.

Chapter 4 investigates a head-to-head study comparison of learning outcomes for two online learning platforms: Coursera and Korbit. The evaluation was based on the normalized learning

gain by using pre- and post-assessment quizzes with participants taking courses on an introductory data science topic.

Chapter 5 presents Korbit, a large-scale, open-domain, dialogue-based ITS which has been designed to easily scale to thousands of subjects by automating, standardizing, and simplifying the content creation process.

Chapter 6 conclude the results and contributions of this thesis and discussion more about the further research.

CHAPTER 1

TECHNICAL BACKGROUND

1.1 Probabilistic generative models

1.1.1 *n*-Gram Models

An important class of probabilistic models a the n-gram models for discrete sequences where $n \in \mathbb{N}$. Let $\mathbf{w} = (\omega_1, \cdots, \omega_M)$ be a sequence of M discrete symbols, where $\omega_M \in V$ for discrete set of V. For example, the variables can be the words of a paper, web, communications, etc., represented by their indices. The n-gram model, with parameter θ, assumes the distribution over variables factorizes:

$$\begin{aligned} P_\theta(\mathbf{w}) &= P_\theta(\omega_1, \cdots, \omega_M) \\ &= P_\theta(\omega_1) P_\theta(\omega_2|\omega_1) \cdots P_\theta(\omega_{n-1}|\omega_1 \cdots \omega_{n-2}) \prod_{m=n}^{M} P_\theta(\omega_m|\omega_{m-n+1}, \cdots, \omega_{m-1}) \end{aligned} \tag{1.1}$$

The key approximation is that the probabilities over each variable can be computed utilizing only the anterior $n-1$ tokens:

$$\begin{aligned} P(\omega_m|\omega_1, \cdots, \omega_{m-1}) &\approx P_\theta(\omega_m|\omega_{m-n+1}, \cdots, \omega_{m-1}) \\ &= \theta_{\omega_m, \omega_{m-n+1}, \cdots, \omega_{m-1}} \end{aligned} \tag{1.2}$$

where $\theta_{\omega_m, \omega_{m-n+1}, \cdots, \omega_{m-1}} \in [0, 1]$ is the probability of observing token v given the $n-1$ previous token $\omega_{m-n+1}, \cdots, \omega_{m-1}$ which must sum to one: $\sum_{v \in V} \theta_{v, \omega_{m-n+1}, \cdots, \omega_{m-1}} = 1$. The 2-grams model is designated as bigram and probabilistic generative model, which can apply a probability for any sequence of variables $\omega_1, \cdots, \omega_M$ and it can generate any such sequence by sampling one variable at a time. This model is utilized in many natural language processing applications ((Goodman (2001)). The bigram model is shown in Table 1.1.

Table 1.1 Probabilistic graphical model for bigram (2-gram) model

.

Text	Token sequence	Token value
[My name] is Vu	1	My name
My [name is] Vu	2	name is
My name [is Vu]	3	is Vu

Let $\omega^i{}_{i=1^I}$ be a set of I example sequences, which is denominated as the training dataset. We postulate that all the sequences are independent and identically distributed. The model parameter θ is learned by maximizing the log-likelihood on the training set:

$$\theta = \underset{\theta'}{\operatorname{argmax}} \sum_i \log P_{\theta'}(\omega^i) \tag{1.3}$$

By setting $\theta_{v,\omega_{m-n+1,\cdots,\omega_{m-1}}}$ to be proportional to the number of times token v was observed after tokens $\omega_{m-n+1,\cdots,,\omega_{m-1}}$ in the training data which are often normally regularized or learned with Naïve Bayes methodology. As the value of n grows, and the variables are discrete, the number of possible amalgamations of n variables is $|V|^n$, which grows exponentially with n. Therefore, in practice, n is conventionally a minuscule number around 3 or 4.

1.1.2 Recurrent Neural Networks

I observed that the Recurrent Neural Networks Language model (RNNLM) (Mikolov, Karafiát, Burget, Cernocký & Khudanpur (2010)) has many applications in Natural Language Processing field. Other variants have been applied to diverse sequential tasks, including verbalization synthesis (Chung, Kastner, Dinh, Goel, Courville & Bengio (2015)), handwriting generation (Graves (2013)) and music composition (Boulanger-Lewandowski, Bengio & Vincent (2012)). As afore, let $\omega_1, \cdots \omega_M$ be a sequence of discrete variables, such that $\omega_m \in V$ for a set V. V is called the lexicon, and each discrete ω_m is the token. The RNNLM is a probabilistic generative model with parameters θ, which decomposes the probability over tokens. We opted to fixate on the well Recurrent Neural Networks Language model (RNNLM) (Mikolov *et al.*

(2010)). Other variants have been applied to diverse sequential tasks, including verbalization synthesis (Chung *et al.* (2015)), handwriting generation (Graves (2013)) and music composition (Boulanger-Lewandowski *et al.* (2012)). As afore, let $\omega_1, \cdots \omega_M$ be a sequence of discrete variables, such that $\omega_m \in V$ for a set V. V is called the lexicon, and each discrete ω_m is the token. The RNNLM is a probabilistic generative model, with parameters θ, which decompose the probability over tokens:

$$P_\theta(\omega_1, \cdots \omega_M) = \prod_{m=1}^{M} P_\theta(\omega_m | \omega_1, \cdots, \omega_{m-1}) \tag{1.4}$$

Unlike the n-gram models, the RNNLM does not make a hard postulation restricting the distribution over a token to only depend on the $n-1$ anterior tokens. Instead, it parametrizes the conditional output distribution over token as:

$$\begin{aligned} P_\theta(\omega_{m+1} = v | \omega_1, \cdots, \omega_m) &= \frac{exp(g(h_m, v))}{\sum_{v' \in V} exp(g(h_m, v'))} \\ g(h_m, v) &= O_v^T h_m, \\ h_m &= f(h_{m-1}, I_{\omega_m} \end{aligned} \tag{1.5}$$

where $h_m \in \mathbb{R}^{d_n}$, for $m = 1, \cdots, M$ are the genuine-valued vectors called obnubilated states with dimensionality $d_h \in \mathbb{N}$. The function f is a non-linear smooth function called the obnubilated state update function. For each time step (each token) it coalesces the precedent obnubilated state h_{m-1} with the current token input ω_m to output the current obnubilated state h_m. The obnubilated state h_m acts as a summary of all the tokens visually examined so far, which efficaciously makes it an ample statistic from a statistical perspective. The matrix $I \in \mathbb{R}^{d_e \times |V|}$ is the input word embedding matrix, where column j contains the embedding for word (token) index J and $d_e \in \mathbb{N}$ is called the word embedding dimensionality. Similarly, the matrix $O \in \mathbb{R}^{d_e} \times |V|$ is called the output word embedding matrix. By Equation 1.4 and Equation 1.5, the probability distribution over token ω_{m+1} is parametrized as a *softmax* function over the dot products between the obnubilated state and the output word embeddings for each word in the lexicon. Consequently, the more kindred an output word embedding vector O_v is to the obnubilated state vector h_m (e.g.,

the more diminutive the angle between the two vectors) the higher the probability assigned to token v.

Unlike the n-gram models discussed earlier, the RNNLM does not parametrize a different probability value for every possible cumulation of tokens. Instead, it embeds words into genuine-valued vectors utilizing the word embedding matrices, thereby sanctioning the rest of the model to utilize the same set of parameters for all words visually examined. This was the key innovation of the Neural Network Language Model, and it is utilized by the RNNLM and its extensions, which gained state-of-the-art performance on several machine learning tasks (Mikolov *et al.* (2010), Devlin, Chang, Lee & Toutanova (2019a), Jozefowicz, Vinyals, Schuster, Shazeer & Wu (2016)). In addition to , by utilizing the RNN to compute the obnubilated state, which parametrizes the output distribution, the RNNLM can potentially capture an illimitable amplitude of context (unlike both n-gram models and the earlier Neural Network Language Models).

The model parameters are learned by maximum likelihood. However, unlike the n-gram models discussed above, no closed-form solution exists. Consequently, the parameters are customarily learned utilizing stochastic gradient descent on the training set. Let $\{w^i\}_{i=1}^I$ be the training dataset. An example sequence w^i is a sample at arbitrary, and the parameters are updated:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log P_\theta(\omega_1^i, \cdots, \omega_M^i) \tag{1.6}$$

where $\alpha > 0$ is the learning rate and M_i is the length of sequence i. In practice, any first-order optimization method can be used. For example, the method was developed by (Kingma & Ba (2015)) tends to work well. For the large model, in practice, the gradient w.r.t each parameter can be computed efficiently using Graphics Processing Unit (GPUs) and parallel computing in combination with the *backpropagation algorithm*, a type of dynamic programming (Goodfellow, Bengio & Courville (2016))

There exist different parametrizations of the function f. One of the simplest and most popular parametrizations is the hyperbolic tangent one-layer neural network:

$$f(h_{m-1}, I_{\omega_m}) = \tanh(H^i I_{\omega_m} + H h_{m-1}) \tag{1.7}$$

where $H \in \mathbb{R}^{d_h \times d_h}$ and $H^i \in \mathbb{R}^{\approx \times}$ are its parameters. Usually, a constant, called the bias or intercept, is also added before applying the hyperbolic tangent transformation, but to keep the notation simple, we will omit this. Another popular variant is the Gated Recurrent Unit (GRU) proposed by (Cho, van Merriënboer, Gulcehre, Bahdanau, Bougares, Schwenk & Bengio (2014)):

$$\begin{aligned} r_m &= \sigma(I^{\gamma}_{\omega_m} + H_{\gamma} h_{m-1}), \text{ (reset gate)} \\ u_m &= \sigma(I^{u}_{\omega_m} + H_u h_{m-1}), \text{ (update gate)} \\ \bar{h}_m &= \tanh(H_i I_{\omega_m} + H(\gamma_m \cdot h_{m-1})), \text{ (candidate update)} \\ h_m &= (1 - u_m) \cdot h_{m-1} + u_m \cdot \bar{h}_m, \text{ (update)} \end{aligned} \tag{1.8}$$

where $\cdot$ is the element-wise product and σ is the element-wise logistic function:

$$\sigma(x) = \frac{1}{1 + exp(-x)} \tag{1.9}$$

and where $I, I^{\gamma}, I^{u} \in \mathbb{R}^{d_h \times |V|}, H, H_{\gamma}, H_u \in \mathbb{R}^{\approx \times \approx}$ and $H^i \in \mathbb{R}^{d_h \times d_e}$ are the parameters. The motivation for this parametrization is that the *reset gate* and *update gate* equations control whether or not the model reuses the anterior obnubilated state when computing the current obnubilated state. If the precedent state is useless (i.e., its value will not determine future tokens in the sequence), γ_m should be proximate to zero, and the candidate update will be predicated mainly on the current input ω_m. If the anterior state is utilizable (i.e., h_{m-1} may avail prognosticate future tokens in the sequence), then γ_m should not be zero. If the antecedent state h_{m-1} is utilizable. Still, the current input is frivolous; then, the update gate should set u_m to zero,

ascertaining that minimal information is stored from the current input. When u_m is proximate to zero, the update is linear, and these avails propagate the gradients in the training procedure. This parametrization appears to be superior to the hyperbolic tangent one-layer neural network across several machine learning quandaries (Greff, Srivastava, Koutník, Steunebrink & Schmidhuber (2015)).

A third, also very popular, parametrization is the Long-Term Short-Term Unit (LSTM):

$$\begin{aligned} i_m &= \sigma(I^{\gamma}_{\omega_m} + H^{ih} h_{m-1} + H^{ic} c_{m-1}) \\ f_m &= \sigma(I^{f}_{\omega_m} + H^{fh} h_{m-1} + H^{fc} c_{m-1}) \\ c_m &= f_m c_{m-1} + i_m \tanh I^{c}_{\omega_m} + H^{ch} h_{m-1} \\ o_m &= \sigma(I^{o}_{\omega_m} + H^{oh} h_{m-1} + H^{oc} c_{m-1}) \\ h_m &= o_m \tanh(c_m) \end{aligned} \tag{1.10}$$

where $h_m, c_m \in \mathbb{R}^{d_h}$, for $m = 1, \cdots, M$ are genuine-value vectors, $I^{\gamma}, I^{f}, I^{c}, I^{o} \in \mathbb{R}^{d_h \times |V|}$ and $H^{ih}, H^{ic}, H^{fh}, H^{fc}, H^{ch}, H^{oh}, H^{oc} \in \mathbb{R}^{d_h \times d_h}$ are the parameters. The variables c_m and h_m can be folded into a single vector by concatenation and re-inscribed as an obnubilated state update function. The motivation abaft the LSTM parametrization is kindred to that of the GRU parametrization. In practice, the LSTM unit appears to yield marginally more stable training compared to the GRU unit, although in terms of performance they appear to perform well (Greff *et al.* (2015), and Lipton (2015))

The Deep Bidirectional LSTMs (BiLSTM)is utilized in many applications of NLP. BiLSTM is an extension of the LSTM models in which two LSTMS are applied to the input data. A LSTM is applied to the input sequence (i.e., forward layer). In the second round, the inversion form of the input sequence is victualed into the LSTM model (i.e., rearward layer). Applying the LSTM twice leads to amend learning long-term dependencies and thus consequently will improve the precision of the model.

1.1.3 Learning Word, Phrase and Sentence Embeddings with Probabilistic Generative Models

In this section, we will introduce some probabilistic graphical models. The conception of learning distributed embedding of linguistic units is that each linguistic unit can be mapped into a genuine-valued, distributed vector are representing its semantic and syntactic components. For example, suppose two linguistic units are proximate in this vector space. In that case, they may likely that they have a homogeneous semantic or syntactic component (e.g., topic information). An early and popular method for learning distributed word presentations is Latent Semantic Analysis (LSA) (Deerwester, Dumais, Furnas, Landauer & Harshman (1990)), and another approach can be found in (Ferrone & Zanzotto (2020) , Li & Yang (2018), Camacho-Collados & Pilehvar (2018)). One recent and widely approach is the Skip-Gram model (Mikolov *et al.* (2013), Mikolov *et al.* (2010)). Let $\{\mathbf{w}^i\}_{i=1}^{I}$ be a set of I example sequences of word tokens, called the training dataset, and surmise that each word token emanates from the lexicon V. These might be extracted from an astronomically immense corpus of news articles or Wikipedia articles in many genuine-world applications. The Skip-Gram model aims to learn representations of words, which soothsay their circumventing words (additionally called context words). This approach is incentivized by the *distributional hypothesis*, which states that words that occur in the same contexts incline to have homogeneous construals. During training, the Skip-Gram model will sample a sequence at uniform desultory $\mathbf{w}^i$. Then, it will sample a word pair at uniform desultory from this sequence, $\omega_t, \omega_{t'} \in \mathbf{w}^i$, under the condition that the two words are within c distance of each other (i.e., $|t - t'| \leq c$). The parameter c is called the training context and is conventionally set somewhere in the range between 3 and 12. Following this, the Skip-Gram model prognosticates words $\omega_{t'}$ conditioned on the word ω_t by:

$$P_\theta(\omega_{t'}|\omega_t) = \frac{exp(I^T_{\omega_{t'}}, I_{\omega_t})}{exp(\sum_{\omega \in V} I^T_\omega I_{\omega_t})} \tag{1.11}$$

with word embedding parameters $\theta = I \in \mathbb{R}^{V \times d_e}$ and word embedding dimensionality $d_e \in \mathbb{N}$. These word embedding parameters represent the mapping from a word (e.g., a word index) to its corresponding real-valued, distributed vector representation. The simplest variant of the

model updates its parameters by maximizing the log-likelihood for that particular sample with stochastic gradient descent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log P_\theta(\omega_{t'}|\omega_t) \tag{1.12}$$

When conditioned on a given world, the Skip-Gram model can be interpreted as a probabilistic graphical model. Conditioned on one observed word (i.e., an observed random variable taking values in the set V), the probabilistic graphical model predicts the set of surrounding words independently.

Table 1.2 Examples of the closest tokens given by Skip-Gram model trained on 30 billion training words. This table was adapted from Mikolov *et al.* (2013)

Query Token	**Redmond**	**Havel**	**Ninjutsu**	**Graffiti**
Closest Tokens	Redmond Wash Redmond Washington Miscrosoft	Vaclav Havel president Vaclav Havel Velvel Revolution	ninja martial arts swordsmanship	spray paint graffiti taggers

A simple extension of the Skip-Gram model, called the Skip-Phase model, enables the cognition of distributed representations for phrases such as *Montreal* and *Air Canada* Mikolov *et al.* (2013). In this case, frequently co-occurring tokens are mapped together to compose a single token. For example, since *Air* and *Canada* co-occur together frequently, the Skip-Phase model might supersede them with the amalgamated token *Air_Canada*. Examples of the word and phrase embeddings learned by the Skip-Phrase model are shown in Table 1.2.

The Skip-Gram model belongs to a broader class of models kenned as Word2Vec word embedding models. These models have benefited from many natural language processing tasks due to their performance and computational efficiency. Much work has been done in learning distributed representations for words. A very cognate model predicated on co-occurrence statistics is the Glove model (Pennington, Socher & Manning (2014a)). Other pertinent work are Gaussian word embeddings (Vilnis & McCallum (2015)), and contextualized word embedding (Pennington, Socher & Manning (2014b), McCann, Bradbury, Xiong & Socher (2017), Li & Yang (2018)).

The models discussed so far are capable of representing words and phrases. However, various methods and models exist for learning distributed sentence representations.

The Skip-Thought Vectors model (Kiros, Zhu, Salakhutdinov, Zemel, Torralba, Urtasun & Fidler (2015)) is one of the neural networks which learns to embed sentences into authentic-valued, distributed vectors. Analogous to the Skip-Gram model, this model aims to learn the sentence embeddings by soothsaying neighboring sentences. Let $\{(w_p^i, w^i, w_n^i)\}_{i=1}^I$ be a set of I example triple, called the training dataset. For each example i, let w_p^i, w^i and w_n^i represent the sequence of word tokens in three consecutive sentences inside a document. As afore, surmise that each word token emanates from the lexicon V. Conditioned on a sentence w^i, the Skip-Thought Vectors model presages the precedent sentence words (w_p^i) and the next sentence words (w_n^i) independently:

$$P_\theta(w_p^i), w_n^i|w^i) = P_\theta(w_p^i|w^i)P_\theta(w_n^i|w_i) \tag{1.13}$$

where the probability distributions on the right-hand are given by:

$$P_\theta(w_p^i|w_i) = P_\theta(\omega_{p,1}^i|w_i) \prod_{m=2}^{M_{p,i}} P_\theta(\omega_{p,m}^i|w^i, \omega_{p,1}^i, \cdots, w_{p,m-1}^i) \tag{1.14}$$

$$P_\theta(w_n^i|w_i) = P_\theta(\omega_{n,1}^i|w_i) \prod_{m=2}^{M_{n,i}} P_\theta(\omega_{n,m}^i|w^i, \omega_{n,1}^i, \cdots, w_{n,m-1}^i) \tag{1.15}$$

where sentence w_p^i contains $M_{p,i}$ words, sentence w_n^i contains $M_{n,i}$ words, and where θ are the model parameters. The probability distributions above are parametrized as variants of the RNNLM with the GRU hidden state update function, but where the token word predictions are excluded for the conditioning sentence w^i. The model is presented in detail in (Zhu, Kiros, Zemel, Salakhutdinov, Urtasun, Torralba & Fidler (2015))

Akin to the Skip-Gram model, the training dataset for the Skip-Thought Vector model might be extracted from an astronomically immense corpus of news articles or Wikipedia articles. However, unlike the Skip-Gram model, the structure of the training dataset is a triple of three

sentences. One caveat, which is paramount to mention, is contestable how much learned sentence representations, such as those learned by the Skip-Thought Vector model, capture higher-level sentence structure (such as word order and lexical dependencies). For example, Arora, Liang & Ma (2017) demonstrate that across several natural language processing tasks, the Skip-Thought Vector models, and other models, which postulate to learn sentence embeddings capturing word order, can be outperformed by simpler bag-of-words models. Nevertheless, the field is perpetually dynamic, and it is likely that developing approaches, such as those proposed in Devlin, Chang, Lee & Toutanova (2019b), maybe capture higher-level sentence structure.

1.1.4 Supervised Learning

1.1.4.1 Support Vector Machine

Support Vector Machines (SVM) are a machine learning tool that analyzes data and recognizes patterns or decision boundaries within the dataset used mainly for classification or regression analysis.

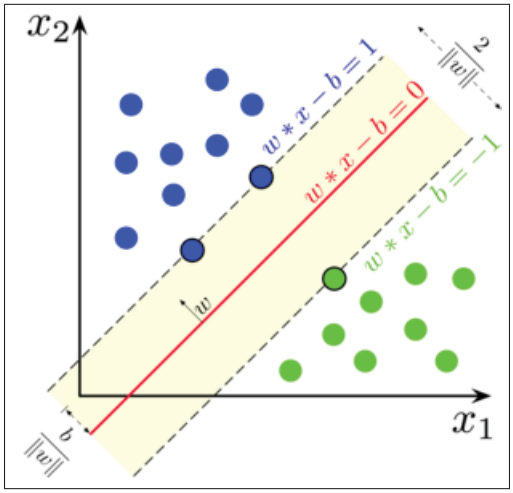

Figure 1.1 Support Vector Machine

SVM constructs the hyper-planes in a multidimensional space that separates different class boundaries and the number of dimensions is called the feature vector of the dataset. SVM can

handle multiple continuous and categorical variables. The goal of the SVM is to separate the two types into classes based on the features. The model consists of three lines. One is $w * x - b = 0$ that is the margin, the line $w * x - b = 1$ and $w * x - b = -1$ represents the position of the closest data points of both the classes. The objective of the SVM is to maximize the perpendicular distance between the two edges of the hyper-plane to minimize the occurrence of generation error. Since the hyper-plane depends on the number of support vectors, the generalization capacity increases with decreasing support vectors. The algorithm is realized as follows: Set the training sample $\{x_i, y_i\}_{i=1}^{N}$, where x_i is the sample of input models, $y_i \in \{-1, 1\}$, classification problem correctly distributes the categorization tags for each sample in order to find the equation $f(x)$. The positive data and negative data are separated through the separating hyper-plane $(w, b) \in \gamma$, where γ is the free margin, which supports vector to meet the conditions of data point (x_i, y_i)

$$
\begin{aligned}
w.x_i + b = -1,\ y_i = -1 \\
(or) \\
w.x_i + b = 1,\ y_i = 1 \\
\min \|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} \xi_i^2 \\
\text{subject to } \ y_i(\langle w.x_i \rangle + b) \geq 1 - \xi_i,\ i = 1, \cdots, n \\
\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle - \frac{1}{2C} \alpha_i^2 \\
\text{subject to } \sum_{i=1}^{n} y_i \alpha_i = 0,\ \alpha_i \geq 0,\ i = 1, \cdots, n \\
\text{after the replacement of the kernel function} \\
\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \frac{1}{2C} \alpha_i^2 \\
\text{Finally, the discriminant function is:} \\
f(x) = sgn(\sum_{i=1}^{n} \alpha_i y_i K(x_i.x) + b)
\end{aligned}
\tag{1.16}
$$

In solving non-linear relegation quandaries, the non-linear kernel function $K(x_i, x_j)$ supersedes the dot product $\langle x_i, x_j \rangle$ and $K(x_i, x_j)$ is presented at (Dioşan, Rogozan & Pécuchet (2008)). Hence, SVM can be acclimated to solve sundry quandaries in the genuine-world. SVM is subsidiary in text and hyper text categorization (Pradhan, Ward, Hacioglu, Martin & Jurafsky (2004)). SVM achievement shown the significantly higher search precision than traditional query refinement schemes [10]. Relegation the satellite data (SAR) (Maity (2016)), or hand-inscribed characters can be apperceived (Decoste & Schölkopf (2002)) (Maitra, Bhattacharya & Parui (2015)), or text relegation (Lakhotia & Bresson (2018)). Hence, SVM has been widely applied in the sciences with relegation tasks.

1.1.4.2 Decision Tree

Decision Tree Algorithm belongs to the family of supervised learning algorithms, which can solve regression and relegation quandaries. The goal of utilizing a Decision Tree is to engender a training model to presage the class or value of the target variable by learning simple decision rules inferred from training data. In Decision Trees, for presaging a class label for a record, we commence from the tree's root. We compare the values of the root attribute with the record's attribute. We follow the branch corresponding to that value on the substratum of comparison and jump to the next node. There are two types of Decision Trees:

- Categorical Variable Decision Tree: Decision Tree which has a categorical target variable.
- Continuous Variable Decision Tree: Decision Tree has a continuous target variable

The Figure 1.2 illustrates the visualization of Decision Tree, where

- Root Node: it represents the entire population or sample, which gets divided into two or more homogeneous sets.
- Splitting: It is a process of dividing a node into two or more sub-nodes.
- Decision Node: When a sub-node splits into further sub-nodes, it is called the decision node.
- Leaf/Terminal Node: Nodes do not split is called Leaf or Terminal Node
- Pruning: When we remove sub-nodes of a decision node, this process is called pruning which can be explained as the opposite process of splitting

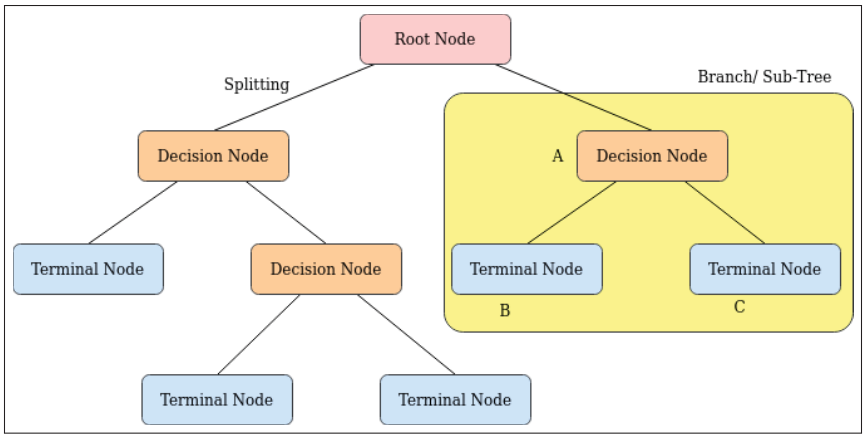

Figure 1.2 Decision Tree Diagram

- Branch/Sub-Tree: A subsection of the entire tree is called branch or sub-tree
- Parent and Child Node: A node, divided into sub-nodes is called a parent node, whereas sub-nodes, are the child of a parent node.

Decision Trees classify the examples by sorting them down the tree from the root to some leaf, with the leaf node providing the relegation of the example. Each node in the tree acts as a test case for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is recursive and is reiterated for every sub-tree rooted at the incipient node.

The decision to do strategic splits heavily affects a tree's precision. The decision criteria are different for relegation and regression trees. Decision trees use multiple algorithms to split a node into two or more sub-nodes. The generation of sub-nodes increases the homogeneity of resultant sub-nodes. The algorithm cull is withal predicated on the type of target variables such as ID3 (Iterative Dichotomiser), C4.5, CART (Relegation and Regression Tree), CHAID (Chi-squared Automatic Interaction Detector), which were mentioned in (Fürnkranz (2010)). Suppose the data consists of N attributes, then deciding which attribute to place at the root or different calibers of the tree as internal nodes is a perplexing step. Just arbitrarily culling any node to be root can't solve the issue. If we follow a desultory approach, it may give us

lamentable results with low precision. Hence, for solving this attribute cull quandary, some criteria are utilized as:

- Entropy measures the randomness in the information being processed. The higher the entropy, the harder it is to draw conclusions from that information. Hence, the entropy for one attribute is represented as:

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{1.17}$$

  where S is the current state, and p_i is the probability of an event i of state S or the percentage of class i in a node of state S. The entropy for multiple attributes is represented as:

$$E(T, X) = \sum_{c \in X} P(c)P(X) \tag{1.18}$$

  where T is the current state, and X is the culled attribute. ID3 abides by the rule, such as a branch with an entropy of zero is a leaf node, and a branch with entropy more than zero needs further splitting.
- Information gain (IG) is a statistical property that measures how well a given attribute dissevers the training examples according to their target relegation. Constructing a decision tree involves finding an attribute that returns the highest information gain and most little entropy. Information gain is a decrementation in entropy. It computes the distinction between entropy afore split and average entropy after the dataset split predicated on given attribute values. ID3 decision tree algorithm utilizes the IG.

$$IG = E(before) - \sum_{j=1}^{K} E(j, after) \tag{1.19}$$

  where $(before)$ is the dataset before the split, K is the number of subsets generated by the split and $(j, after)$ is subset j after the split.

- Gini index is a cost function used to evaluated splits in the dataset. It is caculated by

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2 \tag{1.20}$$

  Gini index works with the categorical target variable "Prosperity" or "Failure". It performs only binary split. The higher value of the Gini index implicatively insinuates higher inequality higher heterogeneity. CART utilizes the Gini index method to engender split points.
- Gain Ratio is a modification of information gain (IG) that abbreviates its in-equitableness and is conventionally the best option. The gain ratio surmounts the quandary with information gain by considering the number of branches that would result in the split. It rectifies information gain by taking intrinsic information into account. Gain Ratio is utilized in C4.5, the advantages version of ID3.

$$GainRatio = \frac{Information\ Gain}{Split\ Info} \tag{1.21}$$

- Truncation in Variance is an algorithm utilized for perpetual target variables (regression quandaries). This algorithm utilizes the standard formula of variance to operate the best split. The split with lower variance is culled as the criteria to split the population:

$$Variance = \frac{\sum (X - \bar{X})^2}{n} \tag{1.22}$$

  where $\bar{X}$ is the mean of the values, X is actual, and n is the number of values.
- Chi-Square is one of the oldest tree relegation methods (CHAID). It ascertains the statistical paramountcy between the distinctions between sub-nodes and parent-node. We quantify it by the sum of squares of standardized distinctions between visually examined and expected frequencies of the target variable. It works with the categories target variable "Prosperity" and "Failure". It can perform two or more splits. The higher the value of Chi-Square, is higher the statistical paramountcy of distinctions between sub-node and parent node.

$$\chi^2 = \sum \frac{(O - E)^2}{E} \tag{1.23}$$

where χ^2 is Chi-Square obtained, $\sum$ is the sum of, O is the observed score, E is the expected score.

The common problem with Decision Trees is overfitting. There are two ways to remove overfitting:

- Pruning Decision Tree: In pruning, you trim off the tree's branches, i.e., abstract the decision nodes starting from the leaf node such that the overall precision is not perturbed. This is done by segregating the genuine training set into two sets: training data set, D, and validation data set, V. Prepare the decision tree utilizing the segregated training data set, D. Then perpetuate trimming the tree accordingly to optimize the precision of the validation data set, V.
- Random Forests are an ensemble learning method for relegation, regression, and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (relegation) or mean/average presage (regression) of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their precision is lower than gradient boosted trees. However, data characteristics can affect their performance

1.1.4.3 Logistic Regression

Logistic regression (Tolles & Meurer (2016)) is a probabilistic linear classifier, parameterized by a weight matrix $\mathbf{w}$ and a bias b. It enables the system to estimate categorical results with the help of a group of independent variables. The classifier equation for logistic regression model is given as:

$$y = sgn(\mathbf{w}^T\mathbf{x} + b) \tag{1.24}$$

where $y \in \{-1.1\}$ denotes the output class recognized for the input $\mathbf{x}$ fed to the system. The classifier equation 1.24 is rewritten using augment weight matrix θ as:

$$y = sgn(\boldsymbol{\theta}^T\mathbf{x}) \tag{1.25}$$

The augmented weight matrix $\boldsymbol{\theta}$ is obtained during the training phase of logistic regression model. Let $\{x_j, y_j\}$ for $j \in [1, m]$ denote the training dataset, where y_j is the target output for training data x_j. The weight matrix is first initialized to 1, i.e., $\boldsymbol{\theta} = \mathbf{1}$. The equation for weight updates is given as:

$$\boldsymbol{\theta}_j(\boldsymbol{n}) = \boldsymbol{\theta}_j(\boldsymbol{n-1}) + \alpha \cdot \vartheta_j \tag{1.26}$$

where α is the learning rate of the model and ϑ_j is given as:

$$\vartheta_j = \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}) \tag{1.27}$$

where m denotes the number of samples available in training dataset, $j \in [1, m]$ and $h_theta(x)$ is the logistic function given as:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (\xi(h_\theta(x^{(i)}), y^{(i)}) \tag{1.28}$$

where $\xi(h_\theta(x)), y)$ is given as:

$$\xi(h_\theta(x)), y) = \begin{cases} -log(h_\theta(x)) & \text{if y = 1} \\ -log(1 - h_\theta(x)) & \text{if y = 0} \end{cases} \tag{1.29}$$

To obtain minimum average cost for the logistic regression model designed, the Gradient Descent method is employed to obtain the iterative expression for $J(\theta)$ as:

$$J(\theta) = J(\theta) + (-\frac{1}{m})(\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)} \log(1 - h_\theta(x^{(i)}))) \tag{1.30}$$

Let ϵ denote the acceptable threshold for cost function such that the iteration in equation 1.30 is terminate when $\epsilon \geq |J(\theta)|$. Logistic regression can be binomial, ordinal, or multinomial. Binomial or binary logistic regression deals with situations in which the visually examined outcome for a dependent variable can have only two possible types, "0" and "1". Multinomial logistic regression deals with situations where the outcome can have three or

more possible that are not authoritatively mandated. Ordinal logistic regression deals with dependent variables that are authoritatively mandated. Logistic regression is use for relegating text (Pranckevičius & Marcinkevičius (2016)) (Fesseha, Xiong, Emiru & Dahou (2020)), does the prognostication of instauration from astringent hemorrhagic shock (Lucas, Williams & Cabrales (2019)), analysis on learning deportment and learning effect (Ran, Zhang, Zheng & Wang (2018)), and so on. Hence, logistic regression is utilized in sundry fields which included machine learning, medical, and convivial sciences.

1.1.5 Latent Variable Models

Many probabilistic graphical models supplementally contain latent (obnubilated) stochastic variables, i.e., stochastic variables which are not visually examined in the original data. The Obnubilated Markov Models (HMMs) and Kalman filters (withal kenned as linear state-space models) have two consequential classes. These models posit a sequence of latent stochastic variables, with precisely one latent stochastic variable for each visually examined token, which explains all the dependencies (e.g., correlations) between the tokens. Importantly, the latent stochastic variables comply with the Markov property: each latent stochastic variable depends only on the anterior latent stochastic variable. This is akin to the bigram model discussed in section 1.1.1. It is instructive to understand the HMM and Kalman filter, and the role that latent variables may play in probabilistic graphical models. Hence, we perpetuate by giving a formal definition for these two models.

As before, let $\omega_1, \cdots, \omega_M$ be a sequence of discrete variables, such that $\omega_M \in V^\omega$ for $m = 1, \cdots, M$ for a vocabulary V^ω. Let $s_1, \cdots, s_M$ be a sequence of discrete latent variables, such that $s_m \in V^s$ for$m = 1, \cdots, M$ for a discrete set V^s. The HMM, with parameter θ, factorizes the probability over variables as:

$$\begin{aligned} P_\theta(\omega_1, \cdots, \omega_M, s_1, \cdots, s_M) &= P_\theta(s_1) \prod_{m=2}^{M} P_\theta(s_m|s_{m-1}) \prod_{m=1}^{M} P_\theta(\omega_m|s_m) \\ &= \theta^0_{s_1} \prod_{m=2}^{M} \theta^s_{s_m s_{m-1}} \prod_{m=1}^{M} \theta^\omega_{\omega_m, s_m} \end{aligned} \tag{1.31}$$

where $\theta^0 \in \mathbb{R}^{|V^s|}$, $\theta^s \in \mathbb{R}^{|V|^s \times |V^s|}$ and $\theta^\omega \in \mathbb{R}^{|V|^\omega \times |V^s|}$ are no-negative parameters, which defined probability distributions. The corresponding graphical model is shown in Figure 1.3

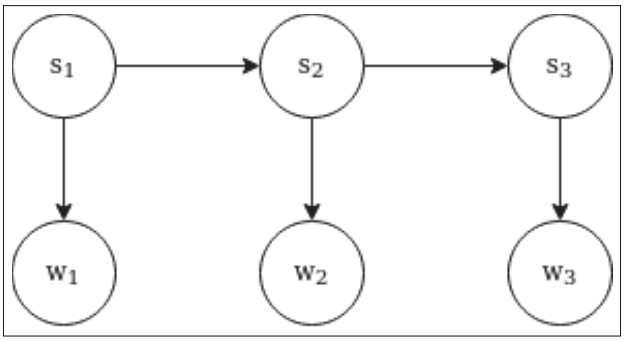

Figure 1.3 Probabilistic graphical model for Hidden Markov Model and Kalman filter model

It is straightforward to derive that the observed tokens are independent conditioned on the latent variables:

$$\begin{aligned} P_\theta(\omega_1, \cdots, \omega_M, s_1, \cdots, s_M) &= P_\theta(s_1) \prod_{m=2}^{M} P_\theta(s_m|s_{m-1}) \prod_{m=1}^{M} P_\theta(\omega_m|s_m) \\ &= \mathcal{N}_{s_1}(\boldsymbol{\theta}_\mu^0, \theta_\Sigma^s) \prod_{m=2}^{M} \mathcal{N}_{s_m}(\theta_\mu^s s_{m-1}, \theta_\Sigma^s) \prod_{m=1}^{M} \frac{exp(\theta_{\omega_m}^{\omega\ T} s_m)}{\sum_{\omega'} exp(\theta_{\omega'}^{\omega\ T} s_m)} \end{aligned} \tag{1.32}$$

where $\mathcal{N}_x(\boldsymbol{\mu}, \sum)$ is the probability of variable x under the multivariate normal distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\sum \in \mathbb{R}^{d \times d}$. The parameters defining the generation process over latent stochastic variables are $\boldsymbol{\theta}_\mu^0 \in \mathbb{R}^d$ and $\theta_\mu^s, \theta_\Sigma^s \in \mathbb{R}^{d \times d}$. The parameter defining the generation process over observed variables is $\theta^\omega \in \mathbb{R}^{d \times |V^\omega|}$, used in a similar way to the RNN parameters. The graphical model is the same as the HMM model, shown in Figure 1.3

For the HMM, the diminutive lexica V^s and V^ω the model parameters may be learned utilizing the stochastic gradient descent procedure described earlier by simply summing out the latent variables. For the Kalman filter and the HMM with astronomically immense lexica, the exact gradient updates are generally intractable, and instead, other procedures must be utilized. Two such training procedures are the prospect-maximization algorithm (EM) and the variational

learning procedure (Bishop (2006a)). The variational learning procedure assumes that a posterior distribution $Q_\psi(s_1, \cdots, s_M|\omega_1, \cdots, \omega_M)$ is estimated with parameters ψ, which approximates $P_\theta(s_1, \cdots, s_M|\omega_1, \cdots, \omega_M)$ by a multivariate normal distribution. It utilizes a lower-bound on the log-likelihood based on Jensen's inequality:

$$\begin{aligned} \log(P_\theta(\omega_1, \cdots, \omega_M) &= \log \sum_{s_1, \cdots, s_M} P_\theta(\omega_1, \cdots, \omega_M, s_1, \cdots, s_M) \\ = \log \sum_{s_1, \cdots, s_M} Q_\psi(s_1, \cdots, s_M|\omega_1, \cdots, \omega_M) &\frac{P_\theta(\omega_1, \cdots, \omega_M, s_1, \cdots, s_M)}{Q_\psi(s_1, \cdots, s_M|\omega_1, \cdots, \omega_M)} \end{aligned} \tag{1.33}$$

The distribution $Q_\psi(s_1, \cdots, s_M|\omega_1, \cdots, \omega_M)$ depends on $\omega_1, \cdots, \omega_M$. Given a set of data examples, it is possible to maintain a Q distribution with separate parameters ψ over each example. A more recent approach utilized in the neural network literature for perpetual latent stochastic variables is to have a neural network parametrize the posterior, where all data examples share the same parameters (Kingma & Welling (2014), Rezende, Mohamed & Wierstra (2014)). Here, the approximate posterior factorizes are:

$$\begin{aligned} Q_\psi(s_1, \cdots, s_M|\omega_1, \cdots, \omega_M) &= \prod_{m=1}^{M} Q_\psi(s_m|\omega_1, \cdots, \omega_M) \\ &= \prod_{m=1}^{M} \mathcal{N}_{s_m}(\mu_m^\psi(\omega_1, \cdots, \omega_M), \theta_\Sigma^\psi(\omega_1, \cdots, \omega_M)) \end{aligned} \tag{1.34}$$

where $\mu_m^\psi \in \mathbb{R}^d$ and $\theta_\Sigma^\psi \in \mathbb{R}^{d \times d}$ are functions of $\omega_1, \cdots, \omega_M$, defined by the approximate posterior parameters ψ and where θ_Σ^ψ is a positive diagonal matrix. The functions μ_m^ψ and θ_Σ^ψ are typically parameterized as neural networks. The procedure now requires a re-parameterization in order to obtain samples $(s_1, \cdots, s_M) \sim Q_\psi(s_1, \cdots, s_M|\omega_1, \cdots, \omega_M)$. Let $\epsilon_m \sim \mathcal{N}(0, 1)$, for $m = 1, \cdots, M$ (i.e., a sample from the multivariate normal distribution with zero mean, identity covariance matrix and dimensionality d). It is then possible to rewrite s_m as:

$$s_m = f_m(\epsilon_m, \omega_1, \cdots, \omega_M) = \mu_m^\psi(\omega_1, \cdots, \omega_M) + \sqrt{\text{diag}(\theta_\Sigma^\psi(\omega_1, \cdots, \omega_M))\epsilon_m} \tag{1.35}$$

where $\sqrt{\text{diag}(\theta^{\psi}_{\sum}(\omega_1, \cdots, \omega_M))}$ is a diagonal matrix with diagonal elements equal to the square roots of the diagonal elements in $\theta^{\psi}_{\sum}(\omega_1, \cdots, \omega_M)$. This re-paramenterization method sanctions the taking of gradient w.r.t parameter ψ. Predicated on these gradients, the training procedure can utilize approximate stochastic gradient descent to learn model parameters. As before, let $\{\text{w}^i{}_i\}$ be the training dataset. An example i is sampled together with $\epsilon^i_1, \cdots, \epsilon^i_M \sim \mathcal{N}(0, 1)$ and the parameters are updated by:

$$\theta \leftarrow \theta + \alpha\nabla_\theta \log P_\theta(\omega^i_1, \cdots, \omega^i_M | s^i_1, \cdots, s^i_M) - \alpha\nabla_\theta \log(\frac{Q_\psi(s^i_1, \cdots, s^i_M | \omega^i_1, \cdots, \omega^i_M)}{P_\theta(s^i_1, \cdots, s^i_M)})$$
$$\psi \leftarrow \psi + \alpha\nabla_\psi \log P_\theta(\omega^i_1, \cdots, \omega^i_M | s^i_1, \cdots, s^i_M) - \alpha\nabla_\psi \log(\frac{Q_\psi(s^i_1, \cdots, s^i_M | \omega^i_1, \cdots, \omega^i_M)}{P_\theta(s^i_1, \cdots, s^i_M)}) \tag{1.36}$$

where $s^i_m = f_m(\epsilon^i_m, \omega^i_1, \cdots, \omega^i_M)$ for $m = 1, \cdots, M$. It is straightforward to compute the gradients w.r.t θ, and since s^i_m have been re-paramenterized in term of ϵ^i_m, it is also straightforward to compute the gradients w.r.t ψ. Furthermore, it is possible to compute the gradient of the exact Kullback Leibler divergence, which corresponds to the negative term in both equations which is presented in details at (Kingma & Welling (2014), Rezende *et al.* (2014))

1.1.6 Cosine Similarity

The kindred attribute is the kindred attribute and is often utilized in data relegation that has homogeneous characteristics. Kindred attribute measure can be exploited to calculate the distance of the homogeneous attribute of the two things being compared and ameliorate precision of information retrieval (Gomaa & Fahmy (2013)). Cosine homogeneous attribute is the method used to quantify the degree of a kindred attribute. This method is a traditional method that is often cumulated with the TF-IDF in section 1.1.7. Cosine homogeneous attribute is a kindred attribute rate the calculation obtained from the cosine 0^o is 1 and less than 1 to the value of the homogeneous attribute of the two vectors are verbalized to be homogeneous when the value of

cosine homogeneous cosine is 1 which performed by the following equation:

$$\cos\theta = \frac{A \times B}{|A| \times |B|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} \tag{1.37}$$

The concept of a two-way degree of similarity states that have similarities. As the equation above, where A is the weight of each feature of the vectors A (Item 1) and B (Item 2) is each character in the vector B. Principles cosine similarity, the greater angle formed between two coordinate vector comparison documents, the smaller degree of document similarity. Conversely, the smaller the degree of cosine similarity level, the greater the degree of similarity.

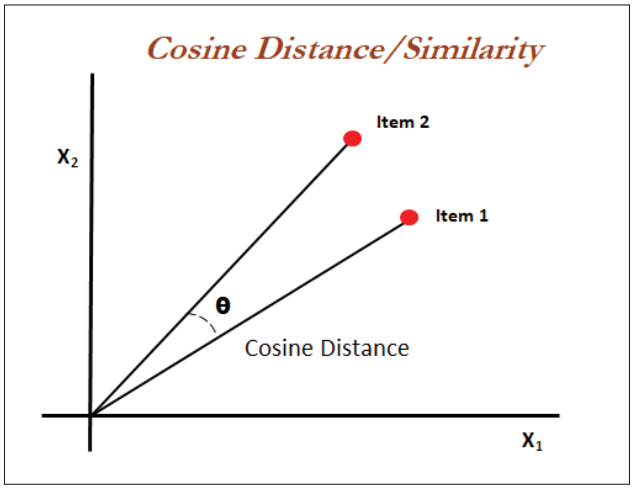

Figure 1.4 The cosine distance/similarity between two items

1.1.7 Term frequency - inverse document frequency

Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is often utilized in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in an amassment or corpus. The paramountcy increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Search engines often utilize variations of the Tf-idf

weighting scheme as a central implement in scoring and ranking a document's pertinence given a utilizer query. (Rajaraman & Ullman (2011)). Tf-idf value increases proportionally to the number of times a word appears in the document. It is offset by the number of documents in the corpus containing the word, which allows adjusting that some words appear more frequently in general. Tf-idf is one of the most popular term-weighting schemes today. A survey conducted in 2015 showed that 83 % of text-predicated recommender systems in digital libraries use Tf-idf (Beel, Gipp, Langer & Breitinger (2016)). The first part of the formula (term frequency (tf)) $\mathrm{tf}(t, d)$ is simply to calculate the number of times each term t appeared in each document. Of course, as with prevalent text mining methods: stop words like *"a", "the"*, punctuation marks will be abstracted beforehand, and words will all be converted to lower cases.

$$\mathrm{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{1.38}$$

where $f_{t,d}$ is the raw count of a term in a document, i.e., the number of times that term t occurs in a document d. The inverse document frequency (idf) is a quantification of how much information the word provides. It is the logarithm scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term and then taking the logarithm of that quotient:

$$\mathrm{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{1.39}$$

where, D is inferring to our document space. It can also be seen as $D = \{d_1, d_2, \cdot, d_n\}$ where n is the number of documents in the collection. $|d \in D : t \in d|$ implies the total number of times in which term t appeared in all of your document d (the $d \in D$ restricts the document to be in your current document space). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to add 1 to avoid the division-by-zero. So the denominator is constructed as $1 + |\{d \in D : t \in d\}|$. Then tf-idf is calculated as:

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D) \tag{1.40}$$

Multiplying these two numbers results in the tf-idf score of a word in a document. The higher the score, the more pertinent that word is in that particular document.

1.2 Natural Language Processing

Natural Language Processing (NLP) aims to train computers to understand natural language, which maps a text to linguistic structures encode its meaning Smith (2011).

1.2.1 Bag-of-words

In some contexts, we will represent a text as a *bag-of-words* (BOW) $x \in \mathbb{R}^{|V|}$ where V is the vocabulary. Each entry x_i corresponds to the number of occurrences of the i-th word in the vocabulary in the text. This frequency may be weighted with *term frequency-inverse document frequency* (tf-idf), which additionally reflects how important a term is in a corpus, a collection of texts. An n-gram is a contiguous sequence of n words in a text. In the standard BOW model, we consider only *unigrams*, sequences of one word. We may additionally consider *bigrams*, sequences of two words. The BOW ignores grammar and word order.

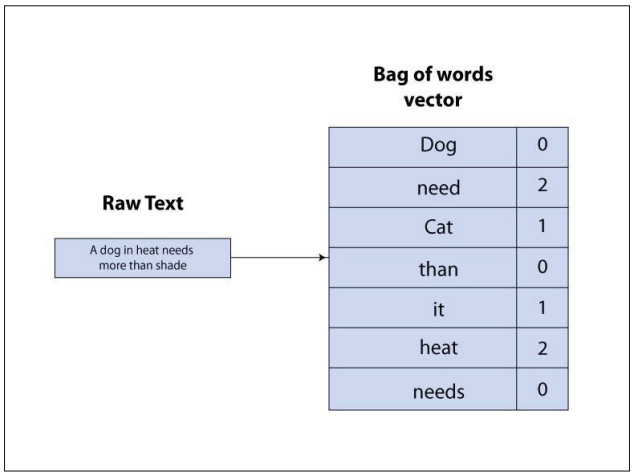

Figure 1.5 Bag of word example

1.2.2 Evaluation metrics

NLP systems are typically evaluated with regards to their performance on the test set of the specific task. For binary classification, accuracy is the common evaluation measure, which is defined as follows:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.41}$$

where TP, TN, FP, FN are the number of true positives, true negatives, false positives, and false negatives, respectively. Intuitively, the number of true predictions is divided all predictions. For multi-class classification, the F score is used Debole & Sebastiani (2004)

$$\begin{aligned} F_\beta &= \frac{(\beta^2 + 1)P \times R}{\beta^2 P + R} \\ P &= \frac{TP}{TP + FP} \\ R &= \frac{TP}{TP + FN} \end{aligned} \tag{1.42}$$

where $\beta = 1$ to obtain teh F_1 score. The F_1 metric balances precision, the fraction of correctly predicted instances and recall, the fraction of correctly predicted instances out of all instances of the category. It provides a good estimate of the overall quality of a model. Certain tasks may use specialized evaluation metrics, which will be introduced with the corresponding task. The metric has evolved to be used for three difference averages, namely *micro, macro*, and *weighted*.

- Let y be the set of predicted (sample, label) pairs,
- $\hat{y}$ be the set of true or gold standard (sample, label) paris,
- L the set of labels,
- y_l the subset of y with label l
- $P(A, B) = \frac{|A \bigcap B|}{|A|}$
- $R(A, B) = \frac{|A \bigcap B|}{|B|}$, where $R(A, B) = 0$ and $P(A, B) = 0$ for $B = \emptyset$,
- $F_\beta(A, B) = (1 + \beta^2)\frac{P(A,B) \times R(A,B)}{\beta^2 P(A,B) + R(A,B)}$

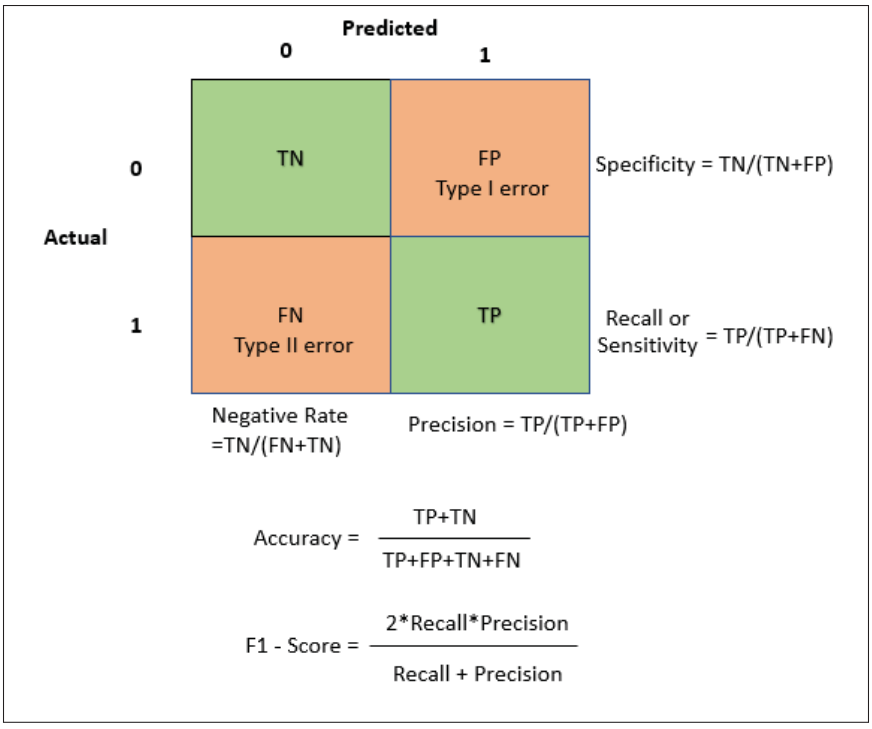

Figure 1.6 Metrics example

Then the metrics are defined[1] as:

$$
\begin{aligned}
F_{\beta-micro} &= F_\beta(y, \hat{y}) \\
F_{\beta-macro} &= \frac{1}{L} \sum_{l \in L} F_\beta(y_l, \hat{y}_l) \\
F_{\beta-weighted} &= \frac{1}{\sum_{l \in L} |\hat{y}_l|} \sum_{l \in L} |\hat{y}_l| F_\beta(y_l, \hat{y}_l)
\end{aligned}
\tag{1.43}
$$

1.2.3 Part-of-speech (POS) tagging

POS tagging is the task of tagging a word in a text with its corresponding part-of-verbalization. A POS is a category of words with homogeneous grammatical properties. Mundane English POS are entity, verb, adjective, adverb, pronoun, preposition, conjunction, etc. POS tagging is

[1] https://scikit-learn.org

arduous as many words can have multiple components of verbalization. POS can be arbitrarily fine-grained and are typically predicated on the culled tag set. The most prevalent tag set utilized by the Penn Treebank comprises 36 tags [2]. However, POS tags vary greatly between languages due to cross-lingual differences. The creation of a "universal" tag set has been an ongoing development: Petrov, Das & McDonald (2012) proposed a tag set of 12 coarse-grained categories, while the current tag set of the Universal Dependencies 2.0 [3] contains 17 tags Nivre, de Marneffe, Ginter, Goldberg, Hajic, Manning, McDonald, Petrov, Pyysalo, Silveira, Tsarfaty & Zeman (2016).

Table 1.3 The POS Tag list

POS	**Description**	**Examples**
ADJ	adjective	*big, old, green, incomprehensible, first*
ADP	adposition	*in, to, during*
ADV	adverb	*very, tomorrow, down, where, there*
AUX	auxiliary	*is, has (done), will (do), should (do)*
CONJ	conjunction	*and, or, but*
CCONJ	coordinating conjunction	*and, or, but*
DET	determiner	*a, an, the*
INTJ	interjection	*psst, ouch, bravo, hello*
NOUN	noun	*girl, cat, tree, air, beauty*
NUM	numeral	*1, 2017, one, seventy-seven, IV, MMXIV*
PART	particle	*'s, not,*
PRON	pronoun	*I, you, he, she, myself, themselves, somebody*
PROPN	proper noun	*Mary, John, London, NATO, HBO*
PUNCT	punctuation	*., (,), ?*
SCONJ	subordinating conjunction	*if, while, that*
SYM	symbol	*$, %, α, β, θ, +, -, $\times$, = *
VERB	verb	*run, runs, running, eat, ate, eating*
X	other	*sfpksdpsxmsa*
SPACE	space	

[2] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[3] https://universaldependencies.org/u/pos/

1.2.4 Chunking

Chunking, also known as shallow parsing, aims to identify continuous spans of tokens that form syntactic units. Chunks are different from parts of speech as they typically represent higher-order structures such as noun phrases or verb phrases. Approaches typically use BIO notation, differentiating the beginning (B) and the inside (I) of chunks. O is used for tokens that are not part of a chunk. Both POS tagging and chunking act mostly on the grammatical and syntactic level, compared to the following tasks, which capture more of the semantic and meaning-related aspects of the text.

1.2.5 Textual Entailment or Natural Language Inference Process

Textual entailment is the task of determining whether a "hypothesis" is true, given a "premise". The cognition holds whenever the truth of one text fragment follows from another text. In the TE framework, the entailing and entailed texts are termed text (p) and hypothesis (h), respectively. Textual entailment is not equipollent to pristine logical entailment – it has a more simple definition: *"p entails h"* $(p \rightarrow h)$ if, typically, a human reading t would infer that h is most likely true. Alternatively: $p \rightarrow h$ if and only if, typically, a human reading p would be justified in inferring the proposition expressed by h from the proposition expressed by p. The cognition is directional because even if *"p entails h"*, the inversion *"h entails p"* is much less certain.

A sentence's meaning can be inferred from its lexical and syntactic composition. Features like negation or synonyms can be acclimated to compare sentences. Dependency graph can withal be habituated to understand how different entities interact with each other. But because of the variability and ambiguity of natural language, semantics cannot be ignored. With the advent of SNLI [4] and MNLI [5] corpus, it became possible to run deep learning models for Natural Language Inference, (NLI). Astronomically big data, which comprises many examples

[4] https://nlp.stanford.edu/projects/snli/

[5] https://cims.nyu.edu/~sbowman/multinli/

of differing compositions, sanctions revelation of features that would otherwise be arduous to identify. Before deep learning, most of the approaches utilized hand-crafted features, mostly distance metrics, to train the NLI models. Additionally, since different languages have a different composition of sentences, manually extracting sundry language features is not feasible. As a result, most recent developments have relied upon learned features utilizing deep neural networks.

- SVM approach: is one of the popular ways of classifying a set of features into one of the target classes. It tries to find a hyperplane that separates the input into distinct classes. Malakasiotis & Androutsopoulos (2007) applied SVM on ten lexical and shallow syntactic features, namely Levenshtein distance, Jaro-Winkler distance, Soundex Manhattan distance, Euclidean distance, Cosine similarity, N-gram distance, Matching coefficient, Dice coefficient, and Jaccard coefficient. Castillo & Alemany (2008) uses lexical and semantic features along with some hand-crafted rules. Textual entailment challenges attracted many SVM-based solutions, but even the best-performing system had less than 65 % accuracy.
- LSTM approach: is engendered in an endeavor to dispense the evanescent gradient quandary that was subsisting in RNN and recollecting contexts for a longer period. In simple, RNN, as more and more words are input, antecedently learned network weights start fading as they are superseded by more recently visually perceived words. This is because the cognition of weights transpires predicated on the gradient of loss, and with time gradient goes on truncating since it has to be back-propagated through time. Bowman, Angeli, Potts & Manning (2015) proposed the first deep learning approach to learn the NLI task. The input premise and hypothesis are encoded utilizing an LSTM into two 100 dimension vectors. Concatenation of these two vectors is then utilized for learning the 3-class relegation. The network consists of a stack of three tanh layers followed by a softmax layer.

  LSTMs processes sentences sequentially to learn a concise representation. This seems akin to a natural approach since humans learn to read text sequentially, word-by-word. But as we become more adept at reading, we are inclined to focus more on certain words in the sentence, which are more paramount than the others. This deportment can be learned by fixating on consequential words. Bahdanau, Cho & Bengio (2014) proposed an attention

mechanism to learn what words to fixate on. It amalgamated the overall representation engendered by LSTM with individual word vectors. Predicated on the loss calculated, each word vector gets its weight-adjusted determining how much the word contributes to the final representation. Rocktäschel, Grefenstette, Hermann, Kočiský & Blunsom (2015) applied an attention mechanism to identify words in the premise that can be influential in deciding the overall relegation. They proposed three models for comparison. The first one uses two LSTMs, one for the premise and one for the hypothesis, arranged such that the initial state of the second LSTM that processes the hypothesis is set to the final state of the first LSTM. This is a conditional encoding since the hypothesis is encoded conditioned on the premise. Zhao, Huang & Ma (2016) mimicked the same for tree representation of sentences. The intuition is that natural language sentences are inherently recursive and can be represented as a tree. Representing premise and hypothesis as a binary tree and then utilizing the attention mechanism to find alignment between nodes of these trees gives a way to find entailment cognation recursively. Neural Tree Indexer (NTI) Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser & Polosukhin (2017) is a way to capture the compositionality of a sentence. Compared to other tree-predicated methods, they did not engender two trees for premise and hypothesis. Instead, they engendered only one tree predicated on the words in hypothesis, and cumulation of these words becomes the tree's nodes.

- WordNet: Knowledge-based Inference Model (KIM) Chen, Zhu, Ling, Inkpen & Wei (2018) involved the utilization of external knowledge in the neural network. Generally, neural networks for Entailment include encoder, attention, local entailment, and sentence level entailment. This model does the same but uses External Knowledge in the form of WordNet. This addition of WordNet is beneficial to help recognize the word or phrase-level relations. It also helps when the training data is limited, and the model cannot learn much from the given data. At first, sentences are encoded by encoders as context-dependent representations. Second, the co-attention between premise and hypothesis was calculated to obtain word-level alignment. After that, the local inference information for entailment/non-entailment prediction was collected. Finally, the composition component aggregates all sentences and predicts the label. WordNet relations need to be converted to a numerical representation.

Semantic relations among the words are determined using synonymy, antonymy, hypernymy, hyponymy, etc. All these relations or features are converted to real numbers. Positive relations like synonymy, hypernymy and hyponymy help capture the entailment, whereas negative relations antonymy co-hyponymy (words with the same hypernymy) helps in determining non-entailment. The Co-attention component uses a co-attention matrix and softly aligns word pairs between the premise and hypothesis with the help of external knowledge. The inference collection component computes local entailment between words or phrases by comparing premise and hypothesis alignment vectors. Inference composition uses the BiLSTM layer and determines sentence level entailment between a premise and a hypothesis.

- Transformer Network: Vaswani *et al.* (2017) proposed to process the input in a non-sequential manner so that the computations could be parallelized. The proposed model was called a Transformer network. It follows a similar approach to other sequence-to-sequence models. An encoder layer converts the input into an intermediate representation, which is then decoded to output value by the decoder layer. But it differs from recurrent architectures because it does not need the encoder layer to process the input sequentially before the decoder can generate the output. The encoder layer produces the output for each word in parallel. The decoder learned to attend to output from the encoder. The input to the model is a combination of word embedding and position vector. Position vector allows using features related to the position of words in the sentence. There are multiple encoder and decoder layers, and each layer has multiple sub-layers. In the case of an encoder, there are two sub-layers: the first is the multi-head attention layer, and the second is a fully connected neural network layer. The attention layer learned a weighted combination of inputs that helps focus on the relevant words. Multi-head attention learns these attention weights in multiple representation sub-spaces and then combines them. The decoders have a similar structure except for the third layer of multi-head attention that attends to the outputs from the encoder layer.

Table 1.4 The example of textual entailment

Premise	Hypothesis	Entailment	Contradiction	Neutral
If you help the needy, God will reward you.	Giving money to the poor has good consequences.	77.6%	0.3%	22.1%
Two women are wandering along the shore drinking iced tea.	Two women are sitting on a blanket near some rocks talking about politics.	0.1%	98.7%	1.2%
A large, gray elephant walked beside a herd of zebras.	The elephant was lost.	0.3%	16.1%	83.6%

1.2.6 Name Entity Recognition

Denominated Entity Recognition (NER) Tjong Kim Sang & De Meulder (2003) is one of the tasks of automatic natural language processing (NLP). The task is to automatically identify soi-disant *named entities*: words or sequences of words that denote unique denominations, locations, organizations, and so on. These entities are withal sometimes called *proper names* in natural language. The task typically involves relegating of these identified entities into a set of predefined classes. Consequently, the task of NER is sometimes subtasks: *named entity identification* (the task is to correctly identify a denominated entity span without further relegation, that is, to retrieve all designated entity tokens) and *name entity relegation* (the task is to correctly relegate the retrieved designated entity tokens into a set of predefined classes). Hence, the NER customarily stands for both of the tasks performed jointly. NER is an often solved task of NLP as a pre-processing step of more involute tasks. Figure 1.7 is an example of NER in practice.

Figure 1.7 The example of NER

The categories culled for a particular NER may be dependable on its requisites. If numerical relegation plays a vital role in a particular field, then the categories describing numerical data may need more refinement. NER can be defined as a word-level tagging quandary where each word in a sentence is either mapped to a designated entity-tag or is relegated as a conventional prevalent word Yadav & Bethard (2019a). Hence, the most common approaches Li, Sun, Han & Li (2018) to NER are followings:

- Rule-predicated approaches: Early NER systems were predicated on handcrafted pattern-predicated rules, lexicons, orthographic features and ontologies (Callan & Mitamura (2002) Sekine & Nobata (2004)). This approach proposes to learn extraction rules that rely on linguistic, syntactic, or document format patterns that are homogeneous and consistent across a group of documents. Rules can be engendered predicated on syntactic-lexical patterns and domain-concrete gazetteers. Rule predicated systems work very well, provided the lexicon is exhaustive. In general, precision inclines to be generally high for rule-predicated NER systems because of the lexicons. Still, the recall may often be low due to domain and language-categorical rules and incomplete dictionaries. Another drawback of rule-predicated NER systems is the desideratum of domain experts for constructing and maintaining the knowledge resources and the fact that the system cannot be transferred to other domains Yadav & Bethard (2019b).
- Unsupervised Learning approaches: A typical unsupervised learning approach is clustering. Clustering-predicated NER systems extract denominated entities from the clustered groups predicated on context homogeneous attribute Nadeau & Sekine (2007). The primary concept is that lexical resources, lexical patterns, and statistics computed on a sizably voluminous corpus can be acclimated to infer occurrences of denominated entities.
- Feature-predicated Supervised learning approaches: Supervised machine learning models learn directly from the training data and can be habituated to supersede human-curated rules. This data must be labeled, which implies that there is an expected output defined to each input. The application of supervised learning necessitates feature engineering. Predicated on these features, many machine learning algorithms have been applied in supervised NER, including Obnubilated Markov Models, Decision Trees, Maximum Entropy Models, Support

Vector Machines, and Conditional Arbitrary Fields Li *et al.* (2018). Such systems propose applying a rule system over feature vectors defined on the word level. They describe word cases, punctuation, numerical value, and special characters. Supplementally, resources in the form of gazetteers, lexicons, and dictionaries were victualed to the classifier to enable lookup techniques.

- Deep learning approaches: In recent years, DL-predicated NER models have become the prevailing approach to apperceive denominated entities. Compared to feature-predicated approaches, deep learning is propitious in discovering obnubilated features automatically and thus distributing state-of-the-art results. Deep learning is one of the fields of machine learning that utilizes the growing volume and availability of data to train models efficaciously by utilizing incremented computational processing puissance. It fixates on training artificial neural networks, which compose multiple processing layers and learning representations of data with multiple levels of abstraction. Contemporary neural architectures for NER can be predominantly relegated into categories that depend on their representation of the words in a sentence. For example, the form of representation may be predicated on words, characters, other sub-word units, or any aggregate.
  - Word level architectures: In this architecture, the words of a sentence are given as input to a Recurrent Neural Network (RNN). Each word is represented by its word embedding. Huang, Xu & Yu (2015) experiment with a variety of LSTM predicated models for sequence tagging. They presented a word LSTM model and showed that integrating a Conditional Arbitrary Field (CRF) layer to the top of the word LSTM ameliorated performance, achieving 84.26% F1-score on the English CoNLL 2003 dataset.
  - Character level architectures: A sentence is perceived to be a sequence of characters in the character level architecture. This sequence is passed through an RNN, prognosticating labels for each character. Character labels are transformed into word labels during a post-processing step. Character-level representation has been shown to exploit explicit sub-word-level information such as prefix and suffix. Another advantage of character-level representation is that it handles the out-of-lexicon quandary well due to its intrinsic

properties. Thus the character-predicated model is capable of inferring representations for unseen words and apportion information of morpheme-level regularities Li *et al.* (2018)

1.2.7 Language Model

Language models compute the probability of occurrence of several words in a particular sequence. The probability of a sequence of m words $\{\omega_1, \cdots, \omega_m\}$ is denoted as $P(\omega_1, \cdots, \omega_m)$. Since the number of words coming before a word ω_i varies depending on its location in the input document. $P(\omega_1, \cdots, \omega_m)$ is usually conditioned on a window of n previous words rather than all previous words:

$$P(\omega_1, \cdots, \omega_m) = \prod_{i=1}^{i=m} P(\omega_i|\omega_1, \cdots, \omega_{i-1}) \approx \prod_{i=1}^{i=m} P(\omega_i|\omega_{i-n}, \cdots, \omega_{i-1}) \tag{1.44}$$

In machine translation, the model chooses the best word ordering for an input phrase by assigning a goodness score to each output word sequence alternative.

1.2.7.1 N-gram Language model

Let's begin with the task of computing $P(\omega|h)$, the probability of a word ω given some history h. Suppose the history h is "its water is so transparent that" and we want to know the probability that the next word is the:

$$P(the|its\ water\ is\ so\ transparent\ that) \tag{1.45}$$

One way to estimate this probability emanates from relative frequence counts: take a profoundly and astronomically immense corpus, count the number of times we visually perceive *its dihydrogen monoxide is so transparent that*, and count the number of times this is followed by *the*. This would be answering the question "Out of the times we visually perceived the history h,

how many times was it followed by the word ω" as follows:

$$P(the|its\ water\ is\ so\ transparent\ that) = \frac{C(its\ water\ is\ so\ transparent\ that\ the)}{C(its\ water\ is\ so\ transparent\ that)} \tag{1.46}$$

Let's start with a little formalizing of notation. To present the probability of a particular random variable X_i taking on the value "the", or $P(X_i = \text{"}the\text{"})$, or $P(the)$. A sequence of N words is presented as $\omega_1, \cdots, \omega_n$ or $\omega_{1:n}$. The join probability of each word is $P(\omega_1, \cdots, \omega_n)$ which is computed by the chain rule:

$$P(\omega_{1:n}) = \prod_{k=1}^{n} P(\omega_k|\omega_{1:k-1}) \tag{1.47}$$

It shows the link between computing the joint probability of a sequence and computing the conditional probability of a word given previous words. The Equation 1.47 suggests that we could estimate the joint probability of an entire sequence of words by multiplying together a number of conditional probabilities. The intuition of the n-gram model is that instead of computing the probability of a word given its entire history, we can approximate the his few words. The **bigram** model, for example, approximates the probability of a word given all the previous words $P(\omega_n|\omega_{1:n-1})$ by using only the conditional probability of the preceding word $P(\omega_n|\omega + n - 1)$. When we use a bigram model to predict the conditional probability of the next word, we are thus making the following approximation:

$$P(\omega|\omega_{1:n-1}) \approx P(\omega_n|\omega_{n-1}) \tag{1.48}$$

The assumption that the probability of a word depends only on the previous word is called a **Markov** assumption. Markov models are the class of probabilistic models that assume we can predict the probability of some future unit without looking too far into the past. We can generalize the bigram (looks on word into the past) to trigram (looks two words into the past) and thus to the **n-gram** (looks $n - 1$ words into the past). Thus, the general equation for this n-gram approximation to the conditional probability of the next word in a sequence is:

$$P(\omega_n|\omega_{1:n-1}) \approx P(\omega_n|\omega_{n-N+1:n-1}) \tag{1.49}$$

Given the bigram assumption for the probability of an individual word, we can compute the probability of a complete word sequence by:

$$P(\omega_{1:n}) \approx \prod_{k=1}^{n} P(\omega_k|\omega_{k-1}) \tag{1.50}$$

To estimate these bigram or n-gram probability, a Maximum Likelihood Estimation (MLE) methodology is used. We get the MLE estimate for the parameters of an n-gram model by getting counts from a corpus, and normalizing the counts so that they are between 0 and 1.

$$P(\omega_n|\omega_{n-N+1:n-1}) = \frac{C(\omega_{n-N+1:n-1}\ \omega_n)}{C(\omega_{n-N+1:n-1})} \tag{1.51}$$

The Equation 1.51 estimates the n-gram probability by dividing the observed frequency of a particular sequence by the observed frequency of a prefix. Let's work through an example using a mini-corpus of three sentences. A special symbol ⟨ s ⟩ is added to the beginning of the sentence as the first word. The end-symbol is added to the end of sentence, too. Hence, the example is presented as:

⟨ s ⟩ *I am Vu* ⟨ s ⟩
⟨ s ⟩ *Vu I am* ⟨ s ⟩
⟨ s ⟩ *I do not like green eggs and ham* ⟨ s ⟩

Here are the calculations for some of the bigram probabilities from this corpus: $P(I|\langle s\rangle) = \frac{2}{3} = 0.67$; $P(Vu|\langle s\rangle) = \frac{1}{3} = 0.33$; $P(am|I) = \frac{2}{3} = 0.67$; $P(\langle s\rangle|Vu) = \frac{1}{2} = 0.5$; $P(do|I) = \frac{1}{3} = 0.33$
On the other hand, what about words we simply have never seen before? In such a closed vocabulary system, the test set can only contain words from this lexicon, and there will be no unknown words. This is a reasonable assumption in some domains, such as speech recognition or machine translation, where we have a pronunciation dictionary or a phrase table fixed in advance. So the language model can only use the words in that dictionary or phrase table. In other cases, we have to deal with words we haven't seen before, which we will call **unknown**

words, our **out of vocabulary** (OOV) words. The OOV rate is the percentage of OOV words that appear in the test set. An open vocabulary system is one in which we model these potential unknown words in the test set by adding a pseudo-word or tag $< UNK >$. The exact choice of the $< UNK >$ model does affect on metrics like perplexity. A language model can achieve low perplexity by choosing a small vocabulary and assigning the unknown word a high probability. For this reason, perplexities should only be compared across language models with the same vocabularies (Buck, Heafield & van Ooyen (2014)).

Hence, there are two main issues with n-gram Language Models:

- Sparsily problems with these models arise due to two issues:
  - If the ω_1, ω_2 and ω_3 never appear together in the corpus, the probability of ω_3 is zero. To solve this, a small σ could be added to the count for each word in the vocabulary, called *smoothing*.
  - If ω_1 and ω_2 never occurred together in the corpus, then no probability can be calculated for ω_3. To solve this, we could condition on ω_2 alone. This is called *backoff*.

    Increasing n makes sparsity problems worse. Hence, $n \leq 5$.
- Storage problems: when n increases, the model size increases as well.

1.2.7.2 RNN Language Model

A large-scale deep learning for natural language processing model was proposed at Bengio, Ducharme, Vincent & Janvin (2003). This models learns *a distributed representation of words*, along with the probability function for word sequences expressed in terms of these representation. A simplified version of this model can be seen in Figure 1.8, where the blue layer signifies concatenated word embeddings for the input words $\mathbf{x} = [x^{(1)}; x^{(2)}; x^{(3)}; x^{(4)}]$ the red layer signifies the hidden layer: $\mathbf{h} = f(\mathbf{We} + \mathbf{b_1})$, and the green output distribution is a softmax over the vocabulary: $\hat{\mathbf{y}} = \text{softmax}(\mathbf{Uh} + \mathbf{b_2})$.

The architecture of RNN is introduced in Figure 1.9, where each vertical rectangular box is a hidden layer at a timestep t. Each such layer holds several of neurons, each of which performs a linear matrix operation on its inputs followed by a non-linear operation (e.g., tanh()). At each

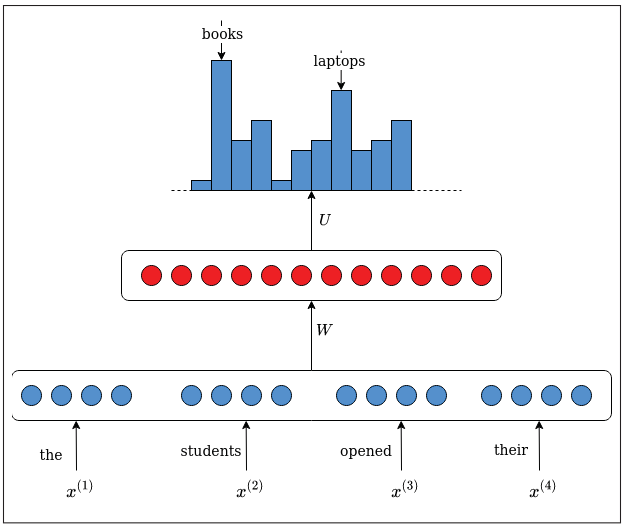

Figure 1.8 The first deep neural network architecture model for NLP

timestep, there are two inputs to the hidden layer: the output of the previous layer h_{t-1}, and the input at the time step x_t. The former input is multiplied by a weight matrix $W^{(hh)}$ and the latter by a weight matrix $W^{(hx)}$ to produce output features h_t, which are multiplied with a weight matrix W^S and run through a softmax over the vocabulary to obtain a prediction output $\hat{y}$ of the next word.

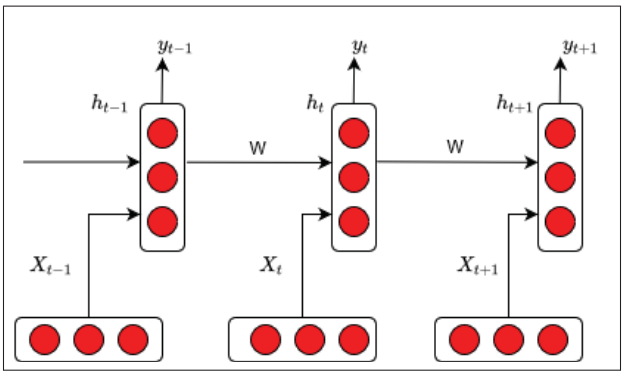

Figure 1.9 A Recurrent Neural Network (RNN). Three time-steps are shown

$$h_t = \sigma(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$
$$\hat{y} = \text{softmax}(W^{(S)} h_t) \tag{1.52}$$

The inputs and outputs of each single neuron are illustrated in Figure 1.10.

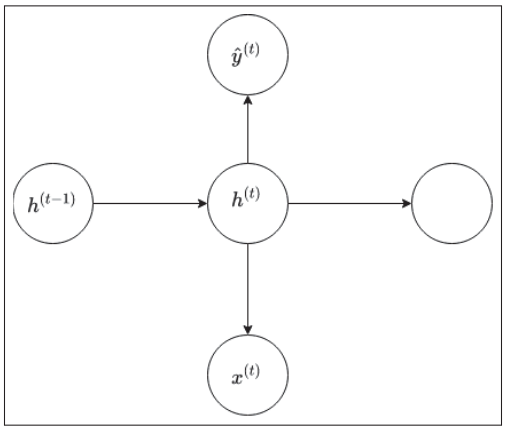

Figure 1.10 The inputs and outputs to a neuron of an RNN

What is interesting here is that the same weights $W^{(hh)}$ and $W^{(hx)}$ are applied repeatedly at each timestep. Thus, the number of parameters the model has to learn is less, and most importantly, is independent of the length of the input sequence-thus defeating the curse of dimensionality. Their parameters in the network are as follows:

An example of an RNN language model is shown in Figure 1.11.

- $x_1, \cdots, x_{t-1}, x_t, x_{t+1}, \cdots, x_\tau$: the word vectors corresponding to a corpus with τ words
- $h_t = \sigma(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$: the relationship to compute the hidden layer output features at each time-step t
  - $x_t \in \mathbb{R}^d$ input word vector at time t
  - $W^{(hx)} \in \mathbb{R}^{D_h \times d}$: weights matrix used to condition the input word vector x_t
  - $W^{(hh)} \in \mathbb{R}^{D_h \times D_h}$: weights matrix used to condition the output of the previous time-step h_{t-1}

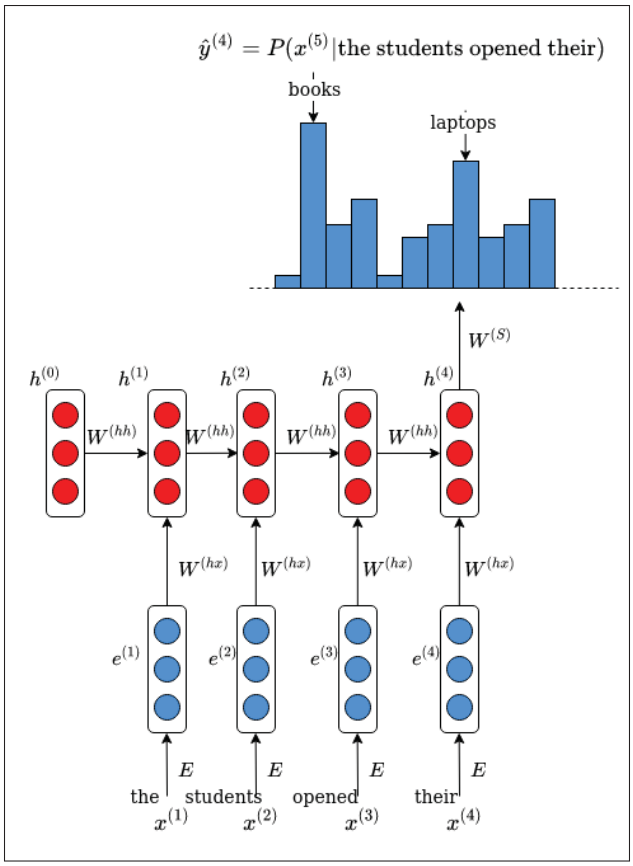

Figure 1.11 An RNN Language Model

- $h_{t_1} \in \mathbb{R}^{D_h}$: output of the non-linear function at the previous time-step, $t-1$. $h_0 \in \mathbb{R}^{D_h}$ is an initialization vector for the hidden layer at time-step $t = 0$
- $\sigma()$: the non-linearity function (sigmoid)

- $\hat{y} = \text{softmax}(W^{(S)} h_t)$: the output probability distribution over the vocabulary at each time-step t. Essentially, $\hat{y}_t$ is the next predicted word given the document context score so far (i.e., h_{t-1}) and the last observed word vector $x^{(t)}$. Here $W^{(S)} \in \mathbb{R}^{|V| \times D_h}$ and $\hat{y} \in \mathbb{R}^{|V|}$ where $|V|$ is the vocabulary.

An example of an RNN language model is shown in Figure 1.11.

1.2.8 Co-reference Resolution Evaluation

1.3 Dialogue Systems

Intelligent Tutoring Systems (ITS) appeared in the early 1970s (Brown & Burton (1975)). One of the goals of AI technologies is to create educational tools that can be adapted to the learner. Because an ITS can play different roles such as "information presentation tools", "knowledge processing tools", "communication tools", it succeeds through each of these roles in stimulating learning and cognitive development of the learner:

- "Information presentation tools": By using these tools, information becomes more accessible, more up-to-date, and can be presented in different formats. It can be a good stimulant for the learner's imagination, memorization, and comprehension.
- "Knowledge processing tools": These tools support for supervision of the learner better, more personalized, and adaptive learning. The learner can do their self-evaluations get recommendations and feedback faster.
- "Communication tools": These tools are utilized for human and machine communication projects through intelligent agents. The better communication is, the better teaching will be, and hence the more effort learning is.

Communication is one of the basic pillars of education, some intelligent tutoring systems proposed in the past were based on conversational dialogs (Freedman (1999), Nkambou, Mizoguchi & Bourdeau (2010)). On the other hand, constructing a complete system including all the necessary components (e.g., representing the domain, modeling the pedagogical knowledge, monitoring the students' progress to offer appropriate help, and constructing the appropriate user interface) is a hot challenge. However, ITS seem effective for improving students' learning outcomes (Bowen, Chingos, Lack & Nygren (2014), Pane, Griffin, McCaffrey & Karam (2014)). One of the key aspects of these systems is the provision of instructional feedbacks or hints that are helpful and appropriate. Many systems are based on production rules, or constraints Nesbit, Adesope, Liu & Ma (2014) where the students' responses are matched against pre-organized solutions. In practice, this often returns in a high development and

maintenance cost. However, appropriate feedback (hint) is a major challenge for ITS. With recent advances in AI and Big data technologies, data-driven tutors are now proposed in Koedinger, Cunningham, Skogsholm & Leber (2008). The personalized supports students du Boulay (2016) and often used in conjunction with conversational agents Lane, Menzies, Ennis & Bezdek (2013) are considered deeply. When a student fails to answer the question, which is issued by the system, the system may respond one of several tactics, such as short feedback (i.e., positive, neutral, negative), pumps (e.g., "Uh, tell me more"), prompts, suggestions, clues, assertions, scaffolding, corrections, and summaries Graesser, Wiemer-Hastings, Wiemer-Hastings, Harter, Group & Person (2000). In early systems, such as AutoTutor, the hints were created by experts or professors manually. Naturally, this is a time-consuming and hard work process, which cannot be scaled to many domains. In some early systems, hints have been generated by applying pre-defined templates to generate textual hints (Hume, Michael, Rovick & Evens (1996a), Graesser, Wiemer-Hastings, Wiemer-Hastings & Kreuz (1999)), which reduced some of the manual efforts, but they remained a time-consuming process. In later systems, hints have also been generated using a domain ontology, such as a knowledge base (Tsovaltzi, Fiedler & Horacek (2004), Tosun (2006)). However, domain ontologies are also expensive to build and maintain. On the other hand, a dialogue system can be a program that communicates with a human user throughout the natural language. The dialogue system provides an associate interface between the user and a computer-based application that permits interaction with the appliance relatively naturally. Analysis on the dialogue system put together stated as interactive conversational, virtual assistants, or usually chatbots have been started from the mid-60s with Weizenbaum's proposed program named ELIZA that was a rule-based system mimicking a Rogerian expert by persistently either recasting statements or asking Weizenbaum (1966). A few years once ELIZA, another chatbot with a clinical science focus, PARRY (Colby (1981)), was accustomed study psychosis. In addition to ELIZA-like regular expressions, the PARRY system was a model of its condition, impacting variables for the agent's levels of concern and anger; certain topics of oral communication could lead PARRY to become further angry or suspicious. However, neither of these a pair of systems used data-driven learning approaches. Later work, just like the MegaHal system by Hutchens and Alder in 1998, began to use data-driven ways (Shawar & Atwell

(2007)). They planned to model the dialogue as a random sequence of separate symbols (words) mistreatment. Figure 1 illustrates the standard design for dialogue systems. It incorporates an Automatic Speech Recognizer, Natural Language Interpreter, Dialogue State Tracker, Dialogue Response Selection, Natural Language Generator, and Text-to-Speech Converter, which we can intergrade our system to make it more effectively.

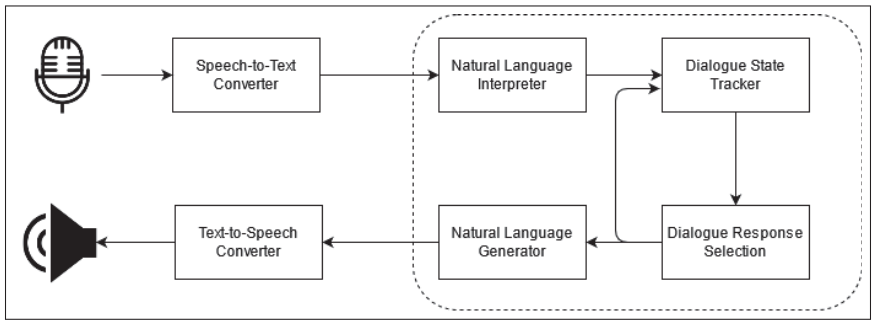

Figure 1.12 An overview of the Dialogue System

Motivated by the recent developments, we focus on generating hints, questions, and answers based on non-structured data in this project. We propose several models, largely inspired by recent neural machine translation models, and we use an approach for dealing with the problem of keywords. We'd like to generate hints using as little supervision as possible (e.g., knowing that the type of a question is (a) "definition", the system can recognize that hints eliciting the definition should be provided, such as asking the student about the attributes/properties of the concept. The properties should be deduced from the input material (e.g., non-structure data as Wikipedia, textbook). Alternatively, we can compile and maintain some sort of ontology of key concepts with their typical attributes. We use the additional questions that do not coincide with the expected answer but help point the student in the right direction. The auxiliary question types are different in their structure and aimed at eliciting different types of information as an answer (e.g., definition, explanation, definition & explanation, two or more definitions contrasting each other, list of properties, and deeper understanding). Even working with the auxiliary question, if the student answers a question different from the original one, we propose a gap reference solution, where the system "masks" the key bit of the answer it is expecting (i.e., given the variety of reference solutions in the dataset, the system can select one at random) and present the student

with the gap solution. We consider the question and answer generator as a transduction problem starting from a Freebase fact, represented by a triple consisting of a subject, a relationship, and an object, which is transduced into a question about the subject, where the object is the correct answer. We evaluate the produced questions and answer in the experiment as well as concerning automatic evaluation metrics, including the well-established machine translation metrics BLEU (Papineni, Roukos, Ward & Zhu (2002)), METEOR (Lavie & Agarwal (2007)), ROUGE (Lin (2004)), ADEM (Lowe, Noseworthy, Serban, Angelard-Gontier, Bengio & Pineau (2017)), and a scoring table with many features (e.g., idf, length of words, similarity of each word) respects with considering the value of co-reference resolution. This suggests that the produced question-answer pairs are of high quality, and therefore they will be useful for training Question - Answer systems. Finally, a confidence model is proposed to generate content interactively with the assistance from teacher.

CHAPTER 2

AUTOMATED QUESTION AND ANSWER GENERATION OF HINT INTERVENTIONS IN INTELLIGENT TUTORING SYSTEM

2.1 Abstract

Intelligent Tutoring Systems (ITS) have proven to be very effective in supporting e-learning when compared to traditional computerized pedagogical approaches. However, many ITS use experts to design and develop rules and exercises, which makes them difficult to build and transfer between domains, and also limits their effectiveness. Additionally, many educational critiques of ITS point to the inadequate immediate feedback and cue sequences that are integrated to make the system intelligent. Indeed, critics highlight their failure to promote personalized learning for students. In this research, we seek to automatically generate, from an ontology, a personalized, precise and syntactically correct paraphrase based on the user's questions and answers in the ITS. We present a methodology that builds the hints and the results collected to show the efficiency of the proposed model. Thus, we propose a machine learning approach to generate personalized hints that have a close relationship with the interests of users and their variety of explanations of the concepts used. Our approach thus considers the individual needs of students while minimizing the need for expert intervention in designing craft rules.

2.2 Introduction

The Personalized tutoring is the service with the goal of helping students in their education process effectively Anania (1983b); Bloom (1984); Burke (1983); Hrastinski, Stenbom, Benjaminsson & Jansson (2019a) that will push them to do their best while enjoying the process. Traditionally, such personalized tutoring service are using the human tutors as teachers and lecturers which content are generated, taught, and evaluated by teachers in a undrirectional fashion and based on the teacher's experience in a specific area of knowledge. On the other hand, the effective state of students will be understood by the human tutors, and thus they will provide the personalized feedback by considering and adapting instructions accordingly.

However, generally, one-on-one tutorial costs too much to conduct on a large scale over the world, hence it is not readily available. Intelligent Tutoring Systems (ITS) is a computer system that aims to provide immediate and customized instruction or feedback to learners which attempt to mimic human tutoring with a low-cost Anderson, Boyle & Reiser (1985b); Nye, Graesser & Hu (2014b). An ITS typically tries to replicate the demostrated benefits of one-to-one, personalized tutoring, in contexts where students would stay at home (online) or at the school (in the class). ITS provides step-by-step guidance for problem solving, progress tracking student's skills and knowledge improvement, and selecting the invdividual basis problems. On the other hands, by comparing to other computer-based learning environments, ITSs are more effective in promoting learning Kulik & Fletcher (2016b); VanLehn (2011b). ITS might compare with Massive Open Online Courses (MOOCs) which are low-cost, computer-based environment with hundreds of millions of students enrolled over the world. However, previous research shows that the rate of student dropout in MOOCs often greater than 90% Jona & Naidu (2014); Rieber (2017). The main reason of the big dropout rate is the poor interaction between system and its users in parallel with the lack of support during the learning period Hone & El Said (2016). However, ITS can deal with the poor interaction by providing a great adaptive and interactive environment to the users. One weakness major to scale-up the ITS is the expensive and laborious process of creating content and pedagogical interventions. Many ITSs use the experts to design the curriculum with hand-crafted rules to generate system interventions, which make the limitation of building and transferring across domains, and their potential scablability Folsom-Kovarik, Schatz & Nicholson (2010b); Olney & Cade (2015b).

2.3 Related work

Many ITS have been successfully deployed to improve students' learning experience in the different domain and application areas in over past decades. There some technical subjects are taught by ITS such as: helping students in mathematics Büdenbender, Frischauf, Goguadze, Melis, Libbrecht & Ullrich (2002b); Goguadze, Palomo & Melis (2005); Hrastinski *et al.* (2019a); Melis & Siekmann (2004b); Passier & Jeuring (2006), algorithms AbuEl-Reesh & Abu-Naser

(2018); Al-Nakhal & Abu-Naser (2017); Leelawong & Biswas (2008); real-world applications Agha, Jarghon & Abu-Naser (2018); Al Rekhawi & Abu-Naser (2018); Qwaider & Abu-Naser (2018). On the other hand, some ITSs are able to provide the general assistance and feedback on student performance (e.g., student characteristics Graesser, Cai, Morgan & Wang (2017), the process of cognition Wu & Looi (2010). The different students have the different aptitudes and knowledge, respectively, so personalized feedback is critical for effective learning. The penalisation of ITS not only help student in independent learning, but also help teachers feedback well Baker (2016); Holstein, McLaren & Aleven (2017, 2019); Al-Dahdooh & Abu-Naser (2017); Al-Nakhal & Abu-Naser (2017); Albacete, Jordan, Katz, Chounta & McLaren (2019b); Chi, Koedinger, Gordon, Jordan & Vanlehn (2011); Lin, Yeh, Hung & Chang (2013b); Munshi & Biswas (2019b); Rus, Stefanescu, Baggett, Niraula, Franceschetti & Graesser (2014b); Rus, Stefanescu, Niraula & Graesser (2014d). In this area of *dialogue-based*, ITSs have been proved to be some of the most effective tools for learning Ahn, Chang, Watson, Tejwani, Sundararajan, Abuelsaad & Prabhu (2018); Nye *et al.* (2014b); Ventura, Chang, Foltz, Mukhi, Yarbro, Salverda, Behrens, Ahn, Ma, Dhamecha et al. (2018), which help to improve student motivation and experience. In the real world, ITS interaction tries to mimic the student-human tutor interactions which present the list of questions and engage students to work on their own problems, pose their questions, and other types of communication with tutor to approach the knowledge via solving problems. In addition, the selection of questions is critical for curriculum structures with both of humans tutor or ITSBoaler & Brodie (2004); Jiang (2014). The main limitation in providing students feedback is the multiple scenarios in student-to-system interactions based on the incorporation or inspiration of teacher instructional scaffolding, where questions as the hints are important to help student Van de Pol, Volman & Beishuizen (2010); Wood (2003). In this paper, we focus on delivering the feedback as the hints in a dialogue-based ITS during problem solving exercises. These feedback include hints, explanations, questions of relevance concept. We utilize the large amount of open data in creating educational content with the given domains such as Wikipedia Brunskill, Mu, Goel & Bragg (2018); Dinan, Roller, Shuster, Fan, Auli & Weston (2019); Guo, Kulkarni, Kittur, Bigham & Brunskill (2016); Liu, Calvo & Rus (2012b); Willis, Davis, Ruan, Manoharan, Landay & Brunskill (2019). By using

NLP techniques, the pedagogical concepts are generated which identify the main knowledge of a topic and help student to study better.

2.4 Automated explanations of hint interventions

To support our research, we utilize the Chat platform named Zulip. The communications on Zulip[1] occurs in streams (which are like channels). Each stream can have several topics, so it fits with the E-learning class where many topic might be considered and discussed. Its' features a unique threading model, in which each message also has a topic, along with the content. Zulip claims that this improves productivity by "making easy to catch up after a day of meetings". Apart from this, Zulip offers standard features found in collaboration apps like message reactions, message search history, polls, private messaging, group messaging etc. Zulip streams can be private or public - only people invited to a private stream can view messages in it, while anyone within an organization can join a public stream. Messages in Zulip can be sent in plain-text or formatted using markdown, along with images, links, and file attachments. Zulip supports native integrations with hundreds of services, which can extend its functionality. Hence, we expect to use Zulip as our platform to embed the MediatorBot which has the mockup at the Figure 2.1 to help users who have the conflict and want to solve a given problem within a topic of *linear regression*.

2.4.1 Features generations

Bloom's taxonomy LW, DR, PW, KA, Mayer, PR, Raths & MC (2001) comprises six categories of cognitive skills, from low to high levels, namely, remember, understand, apply, analyze, evaluate, and create. Remembering and understanding the meaning of a concept are the two fundamental skills in knowledge acquisition. To explore the missing acknowledgement of user, the system analysis the sentences which are interacted such as: questions and answers. Then by giving the hints like the different presentation of concept, we expect that users will be aware of the concepts to solve their problems. An aide gathers background information on a topic:

[1] https://zulip.com/

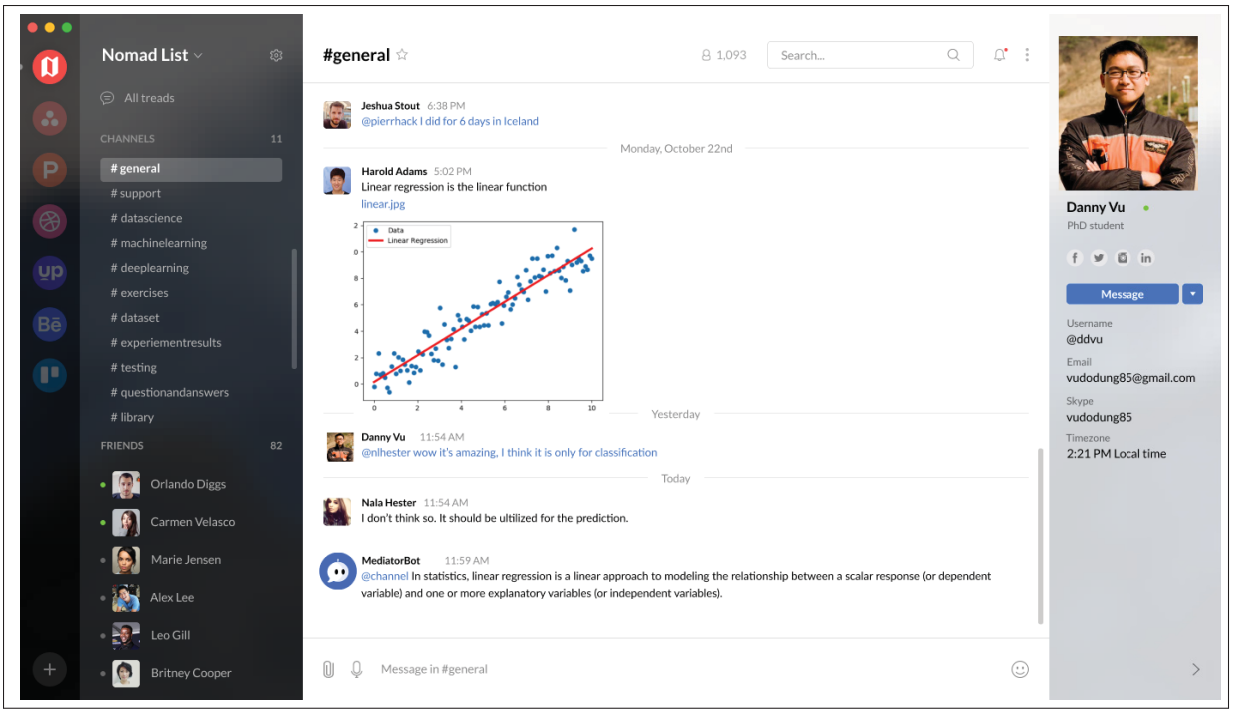

Figure 2.1 The Mockup of Question and Answer system

Wikipedia has available broad subject matter. A wide variety of information on the subject. With over 6 million articles containing over 3.5 billion words, English Wikipedia provides extensive material for the NLP components of our system. The hierarchical structure of the hyperlinks imposed by the Wikipedia format facilitates identification of the sets of pages related to the topic. In addition, the format adopted for Wikipedia articles themselves, where the first sentence typically provides the definition of the title concept and the first paragraph presents a concise description of the topic Kapugama, Lorensuhewa & Kalyani (2016), makes information extraction easier. Hence, it will be great source for us to do the research of generating hints with the given domain which named Wikipedia-Based Explanations such as Table 2.1. The process of generating the Wikipedia-Based Explanations is presented at the Fig 2.2 Given a domain (e.g., *"Machine learning"*, the process is as the following:

Get the list of candidate keyword – Wikipedia offers several ways to group articles: categories, list articles (including item lists, as well as topical glossary, index, outline, and timeline articles), other lists including embedded lists, and navigation templates (of which article series boxes are

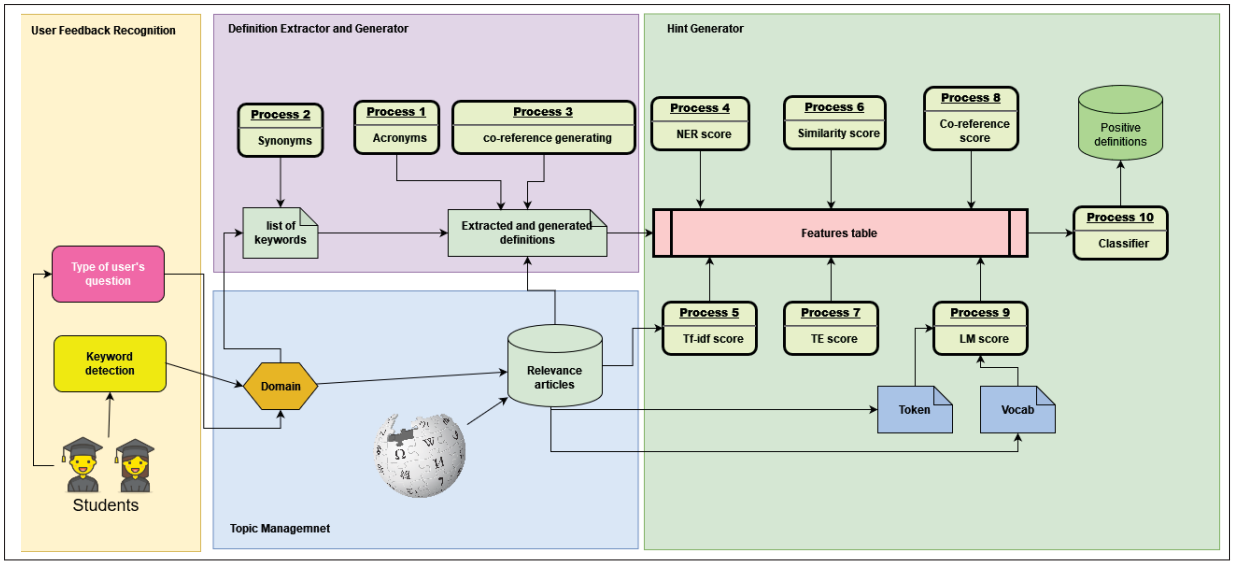

Figure 2.2 Wikipedia-Based Explanations

one type). The grouping of articles by one method neither requires nor forbids the use of the other methods for the same informational grouping. Instead, each method of organizing information has its own advantages and disadvantages, and is applied for the most part independently of the other methods following the guidelines and standards that have evolved on Wikipedia for each of these systems. Hence, by using the given domain as the keyword of the categories in the Wikipedia[2], we create a *recursive_function* to query all the relevance articles within the categories based on a given of depth recursive level (e.g., depth recursive level is 3), the store them to the *raw_dataset D*. To deal with the depth of recursive level, we follow section "See also" of Wikipedia article which contains list of relevance articles as the reference. So the list of technical keywords within the domain is generated based on the title of relevance articles in both horizontal and vertical direction.

In fact, many acronyms technical keywords are used within the articles given a domain. Hence, We disambiguate abbreviations frequently used for technical terms Schwartz & Hearst (2003) (e.g., *'SVM'* is the acronym of *'Support Vector Machine'* in Table 2.2 with a domain *"machine*

[2] https://www.mediawiki.org/wiki/API:Main_page

Table 2.1 Examples of keywords and relevant articles

Domain	Keyword	Relevant articles
“Machine Learning" “Estimation theory" “Deep learning" “Robotics"	“Autonomous car”	“Glossary of artificial intelligence” “Microbotics” “Autonomous things” “Self-driving car”
	“Linear regression”	“Statistical learning theory” “Deep learning” “Linear regression” “Pattern recognition”

learning". On the other hand, this feature is utilized to identify the abbreviated keyword within the users' questions or answers.

Given the extracted keyword, the synonyms are collected by using Wiki Synonyms API[3] with a given domain (e.g, the synonyms of *'autonomous car'* is *'self-driving car'* at Table 2.3. These synonyms is utilized to paraphrase the definition or detect the user technical terms within the dialogue.

Co-reference resolution generating – Given the 1st paragraph of the articles within the domain which contain the considered keyword, the co-reference resolution sentences Clark & Manning (2016) are generated based on the couple of consequence sentences to explain the concept in a new aspect. However, the accuracy of generated sentences are not good enough (e.g., typos, not meaningful, etc.,) which means some sentences are meaningful, the others are not meaningful (e.g., Table 2.4). Hence, we marked them as the generated definitions.
The extracted definition is the good concept which is extracted from the 1st sentence of Wikipedia Article summary via Wikipedia API to explain the concept's meaning clearly.[4]

After getting the extracted and generated definitions, we would like to classify the good definitions and not good definitions to explain to the students. Hence, we propose a table features to support

[3] https://rapidapi.com/ipeirotis/api/wikisynonyms

[4] https://pypi.org/project/wikipedia/

Table 2.2 The example of acronyms

Acronyms	**Full name**
SVM	Support Vector Machine
SRL	Statistical relational learning
JAIR	Journal of Artificial Intelligence Research
WSNs	wireless sensor networks
AI	artificial intelligence
NLP	Natural language processing
AGI	artificial general intelligence
HMM	hidden Markov models
SVM	support vector machine
FNNs	feedforward neural networks
CNNs	convolutional neural networks
RNNs	recurrent neural networks
LSTM	long short-term memory

Table 2.3 The example of synonyms

Keywords	**Synonyms**
Self-driving car	Autonomous vehicle', 'Driverless Car', 'Robotic cars', 'CyberCars', 'Robot Car', 'Autonomous automobile', 'Robot car', 'Computer-driven car', 'Driver-less cars', 'Autonomous driving'
Linear regression	'Linear modeling', 'Regression line', 'Multiple linear regression', 'Linear trend', 'Best fit line', 'Linear weights', ' Least squares regression', 'Regression coefficient', 'Regression Coefficient', 'Multi-linear regression'

the classification algorithm with high accuracy to obtain the good definition which is named as Wikipedia-base explanation. To make the features table, we considers some features which are presented as the numerical scores.

Term frequency–inverse document frequency (Tf-idf) feature: Before generating this feature, the pre-processed data stage for Tf-idf was implemented to construct the index of document collections. Indexing is the process of building a document representation by giving an identifier to text items. Pre-procesesd stage has these steps as below:

Table 2.4 The example of Extracted and Generated Wikipedia Explanations

Keywords	Wikipedia Explanations	Label
Artificial Intelligence	Artificial intelligence (AI), sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals.	Extracted
Artificial Intelligence	Artificial intelligence is breaking into the healthcare industry by assisting doctors.	Generated

- Tokenizing: Cutting the input strings into separate sentences into words
- Filtering: removing the unnecessary words in the text (e.g., remove stop-words)
- Cleaning: removing the uni-code words in the text

The Tf-idf gives weight to the relationship of a word (term) to a documents. This method combines two concepts for calculating weights: the frequency of a word in a particular document and the inverse frequency of documents containing the word. The frequency of words in the document provided shows how important the word is. The frequency of decrements containing the word indicate how common the word is. So the weight of the relationship between a word and a document will be high if the word frequency is high in the document and the overall frequency of the document containing the word is low. This feature was created by using the *raw_dataset D*. Given a collection of terms $t \in D$ that appear in a set of N documents $d \in D$, each of length n_d, $tf - idf$ weighting is computed as follows:

$$\begin{aligned} tf_{t,d} &= \frac{f_{t,d}}{n_d} \\ idf_t &= log\frac{N}{df_t} \\ W_{t,d} &= tf_{t,d} \times idf_t \end{aligned} \tag{2.1}$$

where $f_{t,d}$ is the frequency of term t in document d, and df_t is the document frequency of term t, that is, the number of documents in which term t appears, and $W_{t,d}$ is the weight of feature Tf-idf.

Language Model score: Before computing the language model score, the pre-processed data stage for Language model score was implemented with the raw dataset as followings:

- Tokenizing: Cutting the input strings into separate sentences into words
- Filtering: removing the unnecessary words in the text (e.g., remove stop-words)
- Cleaning: removing the uni-code words in the text
- Creating Vocabulary dictionary: Get the top 220000 common words in the raw dataset
- Creating Token dictionary: Make the index for each tokenizing of the clean raw-dataset, if the word is not mentioned in the vocabulary dictionay, it will be replaced by tag $<unk>$, and each token is identified with an given index
- Creating train, validation, and test dataset: the train, validation, and test dataset were created by spliting the raw dataset with the ration 70%, 10%, and 20%, respectively.
- Creating Corpus: The corpus of each sentences with the index of token was created with tag $<eos>$ which identifies the end of sentence.

To train the language model, we apply a multi-layer long short-term memory (LSTM) RNN to an input sequence. For each element in the input sequence, each layer computes the following function:

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + be_{ii}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{hf}h_{t-1} + b_{ho}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{2.2}$$

where h_t is the hidden state at time t, c_t is the cell state at time t, x_t is the input at time t, h_{t-1} is the hidden state of the layer at time $t-1$ or the initial hidden state at time 0, and i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, respectively. σ is the sigmoid function, and $\odot$ is the Hadamard product. In a multiplayer LSTM, the input x_t^l of the l layer ($l \geq 2$ is the hidden

state $h_t^{(l-1)}$ of the previous layer multiplied by dropout $\sigma_t^{(l-1)}$ where each $\sigma_t^{(l-1)}$ is a Bernoulli random variable which is 0 with probability dropout. After finishing the training language model, we turn to the computing language model score to evaluate the quality of the sentence. The weight of each term was computed from output layer which is the tensor of containing the output feature h_t from the last layer of the LSTM, for each t. Then the score of sentence was computed as the log-likehood of the norm of word weight tensor as following:

$$LS = -log(\frac{\sum(s_{ti})}{||S||}) \tag{2.3}$$

where LS is the language model score of the considered sentence S, s_{ti} is the score of each term in the sentence S, and $||S||$ is the size of sentence S.

Named Entity Recognition (NER) score:

The spaCy[5] tool, offered by Explosion AI, utilizes a hybrid of Hidden Markov Models (HMM,) Maximum Entropy Models (MEMMs), and Decision Tree Analysis, which are all eventually coated with a convolutional neural network to handle extremely large and diverse sets of data, as well as incorporate new training data at the user's request [8]. It is beyond the scope of this paper to thoroughly derive all of these mathematical models, but it is worth briefly mentioning what each of these statistical techniques accomplish. HMMs are generative models that assign joint probabilities to paired observations and label sequences. The parameters are then trained to maximize the joint likelihood of training sets. MEMMs are conditional probabilistic sequence models that can represent multiple features of a word and can handle long term dependency. Lastly, a decision tree model has very high computational efficiency and is understandable and has high computational effeciency. Hence, the vector of name entity of each term in the sentence is generated to recognize the objects within the sentence. For example, a good definition of a math or an abstract concept should not contain $< MONEY >$ or $< ORG >$, respectively in Table 2.5. The Named Entity Recognition (NER) scores are defined as the binary vector of

[5] https://spacy.io/

Entity within the considered sentence.

$$NER = [\vartheta_i], i \in (1, n),\ n \text{ is the size of NER vector}$$
$$\text{where } \vartheta_i = \begin{cases} 1, & \text{if NER label in the sentence} \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

Table 2.5 The example description of Name Entity Recognize features

Type	Description
PERSON	People, including fictional
NORP	Nationalities, religious, political groups
ORG	Companies, agencies, institutions
GPE	Contries, cities, states
LOC	Non GPE locations, mountains, bodies of waters
FACILITY	Building, airports, highways
PRODUCT	Objects, vehicles, foods
EVENT	Wars, sports, battles
LAW	name of document in law
DATE	Date time
MONEY	Money value including unit
ORDINAL	first, second, third, etc
LANGUAGE	Any named languages

Textual Entailment score:

Textual Entailment recognizing is a probability decision problem i.e., the process indicates how many percentages text τ entails hypothesis $\hbar$. There are three kind of features named entailment ϵ, neutral ν, and contradiction ϕ. These features come with their probability relationship between τ and $\hbar$. The entailment ϵ is the probability of the scenario if the $\hbar$ is the entailment of τ which means $\hbar$ might be utilized to explain the τ. The contradiction ϕ is the probability of the scenario if the $\hbar$ contradicted with the τ which means $\hbar$ is false if τ is true. Moreover, the neutral ν is the probability of the scenario which is not entailment or contradiction. If the feature score of hypothesis is greater than 0.7, it will be considered as the label of hypothesis. The example of

textual entailment features are presented in the Table 2.6.

$$
\begin{aligned}
TE &= [\epsilon, \nu, \phi] \\
\epsilon &= P_{entailment}(\hbar, \tau) \\
\phi &= P_{contradiction}(\hbar, \tau) \\
\nu &= P_{neutral}(\hbar, \tau)
\end{aligned}
\tag{2.5}
$$

$$
\text{where, } \begin{cases} \tau, & \text{the premise sentence (extracted definition)} \\ \hbar, & \text{the hypothesis sentence (generated definition)} \end{cases}
$$

Co-reference score:

The Co-reference score is generated by a neural mention-ranking modelClark & Manning (2016). Mention-ranking models score pairs of mentions for their likelihood of co-reference rather than comparing partial co-reference clusters. Therefore, they operate in a simple setting where co-reference decisions are made independently. Our co-reference feature is the maximum probability of generating new sentence based on couple of given sentence which contain the relation condition grammar. The example of co-reference generation sentences are present in Table 2.7. In this table, the input sentences are sentence 1, sentence 2 which were the consequence couple sentences within the article, and the Generate 1,2,3 are the generated sentences with the probability of pronoun replacement are 8.4%, 30.5%, and 30.4%, respectively. Hence, the generate 2 is considered as the generated definitions which co-reference score (CR) equals 0.305.

$$
CR = max(P(s_i, s_j)) \tag{2.6}
$$

where, $P(s_i, s_j)$ is the probability of pronoun replacement of each sentence s_i and s_j within an article

Table 2.6 The example of textual entailment features

Type	Description	Description	Description
Premise	Bayesian hierarchical modeling is a statistical model written in multiple levels (hierarchical form) that estimates the parameters of the posterior distribution using the Bayesian method.	Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.	In probability theory and statistics, the coefficient of variation (CV), also known as relative standard deviation (RSD), is a standardized measure of dispersion of a probability distribution or frequency distribution.
Hypothesis	Bayesians argue that relevant information regarding decision making and updating beliefs cannot be ignored and that hierarchical modeling has the potential to overrule classical methods in applications where respondents give multiple observational data.	Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance.	CV measures are often used as quality controls for quantitative laboratory assays.
Probability	$P_{entailment}$ **= 0.86** $P_{neutral}$ = 0.12 $P_{contraction}$ =0.02	$P_{entailment}$ =0.03 $P_{neutral}$ =0.19 $P_{contraction}$ **= 0.78**	$P_{entailment}$ =0.02 $P_{neutral}$ **= 0.70** $P_{contraction}$ = 0.28
Label	Entailment	Contraction	Neutral

Similarity score:

The similarity score is computed by considering the percentage matching ratio between two

Table 2.7 The example of generated co-reference sentence

Label	Description	Probability
Sentence 1	A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.	
Sentence 2	It is one way to display an algorithm that only contains conditional control statements.	
Generate 1	a decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility	8.4%
Generate 2	a decision tree is a decision support tool one way to display an algorithm that only contains conditional control statements uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility	**30.5%**
Generate 3	a decision tree is a decision support tool an algorithm that only contains conditional control statements uses a tree-like model of decisions and their possible consequences , including chance event outcomes, resource costs, and utility.	30.4%

sentences by using cosine similarity methodology Lahitani, Permanasari & Setiawan (2016). Hence, we will compute the cosine similarity score between generated definition and extracted definition. After the features table is generated, we turn to the classify step which help us to get the right definitions or good Wikipedia-base explanations from our data Table 2.8. In this data, the number of generated definitions is much larger than the number of extracted definitions, the sample weights data is 23.55. Hence the sampling method likes synthetic minority oversampling technique (SMOTE) which was proposed in Mathew, Pang, Luo & Leong (2018) is considered to fix this issue. In the training data, we label the first sentence of each

article is the extracted Wikipedia-based explanation, the generated sentence by considering co-reference resolution and reconstructed definition are generated Wikipedia-based explanations. We experimented with a number of models, including Decision Tree classifiers Breiman, Friedman, Olshen & Stone (1984), Random Forests Breiman (2001), Logistic Regression Bishop (2006b), and Support Vector Machines Smola & Schölkopf (2004), using the `scikit-learn` implementation. Among those, the Decision Tree classifier performed best in a 50-fold cross-validation experiment, yielding an average accuracy of 81.87% ± 3.2% and average F1-score of 96.97% ± 0.6% in Table 2.9. After that, we obtain the good generated and extract Wikipedia-based explanations as the hints which can be used for explaining in another way of definitions for students. Hence, the hints are stored into the hints dataset with a given domain Figure 2.3. There are two types of hints such as explanations and Q&A (questions and answers). The hints generations are presented in the Section 2.4.2.

Table 2.8 The training dataset

Description	**Quantity**
Number of extracted Wikipedia-based explanation	3223
Number of generated Wikipedia-based explanation	75895
Sample weights	23.55

Table 2.9 The algorithm performance comparison in scenario of all consideration features

Algorithm	**F1-score**	**Accuracy**
Random Forests	48.95% ± 0.5%	95.87% ± 0.2%
Support Vector Machines	49.35% ± 1.4%	95.89% ± 0.7%
Logistic Regression	77.35% ± 7.8%	96.42% ± 0.84%
Decision Trees	**81.87**% ± **3.2**%	**96.97**% ± **0.6**%

2.4.2 Conceptual Hint Generations

Identify keyword – Given a *question* and *answers* of an exercise, the nouns and nouns phrases within the question are extracted and labeled as the keyword of the question. After tokenization

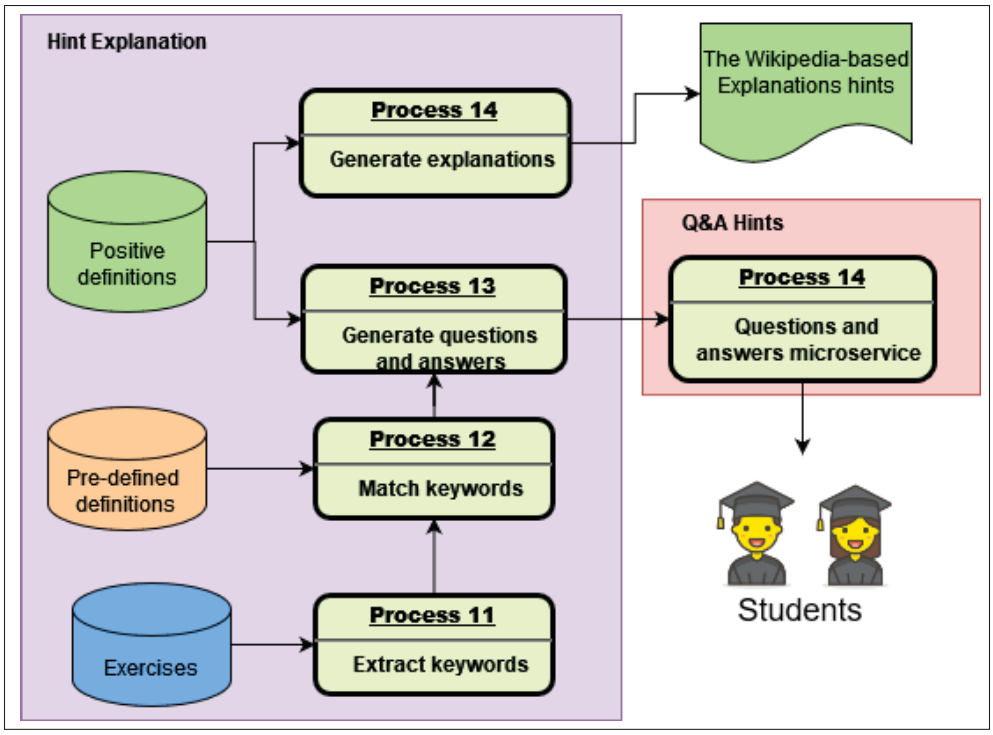

Figure 2.3 Conceptual Hints Generations

step, a state-of-the-art dependency parsing `spaCy`[6] makes a prediction of which tag or label. In the examples in Table 2.10 such keywords and phrases are marked with text boxes: for instance, *autonomous car*, *0* , *Zero*, *No one*, and *No body* are automatically identified as the noun phrase by using `spaCy`. The explanation hints are generated by matching between the keywords within the question of exercises, given answers, and the Wikipedia-based explanation, and constructing with the given prefix (e.g., *Think about the following:, Note that, Consider that, Recall that, Observe that, Think about the following definition:, Here's a hin for you:*) which are presented in Table 2.10.

The system will extract the candidate keywords for each exercie, making user confuse or misunderstand in questions and answers. Then, the system generate the relevance question to help student remember the misconceptions. Hence, we will have the background truth of knowledge which is very important for knowing. Then we propose the context of matching

[6] https://spacy.io

Table 2.10 The example of Wikipedia-based explanation generation

Exercise	Expectation	Generated Explanation Hints
How many human drivers would be needed to drive an **autonomous car**?	**0** **Zero** **No one** **Nobody** is needed	Think about the following definition: A **self-driving car**, also known as an autonomous vehicle (AV), connected and autonomous vehicle (CAV), a driverless car, robot car, or robotic car is a vehicle that is capable of sensing its environment and moving safely with little or no human input
		Here's a hint for you: Different methods and levels of **autonomy** can be achieved through monitoring and remote control from a nearby manned ship, an onshore control center or through artificial intelligence and machine learning, letting the vessel itself decides the course of action.
		Observe that **nobody** is needed

keywords and type of questions to help user step by step to approach the problem of exercise. By using the micro-services API, we generate questions and answers for guiding the users. The couple of questions and answers solution will be shown to make sure that users understand the concept and can be back to solve their problem with the exercises which is presented in the Figure 2.3. To make it more intuitively, Table 2.11 shows the example of couples generated questions and answers which related to the exercise in Table 2.10. Because there is only one keyword *autonomous car* appears in the exercise, so the system only generate the questions for this keyword and the respective answers. Hence, users can understand the concept *autonomous car* in some different aspects before they make the right answers.

Table 2.11 The example of conceptual question and answer generations

Hint Question	Hint Answer
What is the meaning of autonomous car ?	A self-driving car, also known as an autonomous vehicle (AV), connected and autonomous vehicle (CAV), driverless car, robo-car, or robotic car, is a vehicle that is capable of sensing its environment and moving safely with little or no human input
Can you explain the definition of autonomous car ?	Regulatory, safety, legal and security challenges are viewed as the largest obstacles in making autonomous cargo ships a reality
What is the autonomous car ?	The technologies needed to make remote and autonomous ships a reality exist. We will see a remote controlled ship in commercial use by the end of the decade.
What is the definition of autonomous car ?	Autonomous cargo ships, also known as autonomous container ships or maritime autonomous surface ships (MASS), are seaborne vessels that transport either containers or bulk cargo over navigable waters with little or no human interaction

2.5 Experiment

2.5.1 Evaluation algorithms

CUDA has been developed as an integrated development environment for GPUs. GPUs can achieve high performance by executing massively parallel threads simultaneously. For using GPUs effectively, a deep knowledge about them has been required. For example, data should be transferred between CPU and GPU, several GPU memories with different access speeds and sizes should be used according to the cases of processing, and a lot of threads should be managed. Hence, we use NVIDIA® V100 Tensor Core which is the most advanced data center GPU ever built to accelerate AI, high performance computing (HPC), data science and graphics and is integrated on the Google cloud[7] to train our Language model. The learning curve of language model training is presented at Fig 2.4 When the model is over fitting, we stop the training process and get the tensor of weight to compute the Language model score which was mentioned in the Section 2.4. After generating the features table which was mentioned

[7] https://cloud.google.com/

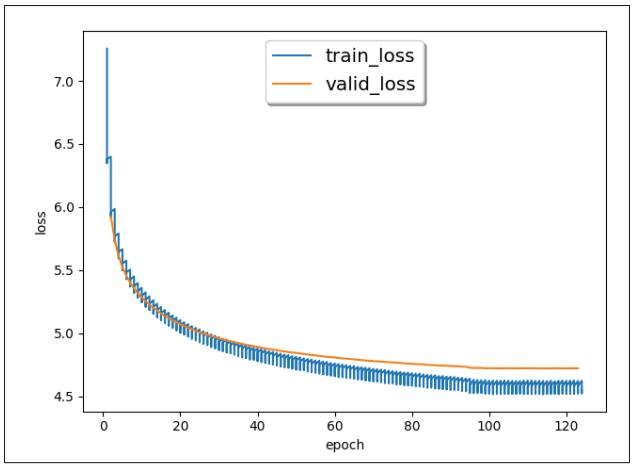

Figure 2.4 Learning curve of language model

in the Section 2.4, we set up the classifications scenarios to optimize the performance of each relevance feature. We define the classifications scenarios as the Table 2.12. To deal with many scenarios and algorithms in the Table2.12, we use the multiprocessing technology which supports spawning processes using an API similar to the threading module. The multiprocessing offers both local and remote concurrency, effectively side-stepping the Global Interpreter Lock by using sub-processes instead of threads. Due to this, the multiprocessing module allows us to fully leverage multiple processors on a given machine. The performance of each algorithm

Table 2.12 The Scenario of classification strategy

Basic	**Basic + LM**	**Basic + TE**	**Basic + LM + TE**
tf-idf, co-reference, length_definition, length_keyword, #_words_definition, #_words_keyword, key_equal_title, NER_score	tf-idf, co-reference, length_definition, length_keyword, #_words_definition, #_words_keyword, key_equal_title, NER_score, language_model _score	tf-idf, co-reference, length_definition, length_keyword, #_words_definition, #_words_keyword, key_equal_title, NER_score, text_entailment _score	tf-idf, co-reference, length_definition, length_keyword, #_words_definition, #_words_keyword, key_equal_title, NER_score, language_model_score, text_entailment_score

(F1-score) with each scenario is presented at Table 2.13. We observed that the performance of Decision Tree with the scenario basic + TE + LM had the biggest value of F1-score. Hence, this

algorithm is chosen for our classification methodology. Moreover, by considering the result of Cohen-kappa Wang, Yang & Xia (2019) $\kappa = 0.61$ which means the expert has the substantial agreement the generated explanations are useful.

Table 2.13 The classification scenarios with the F1-score value of each algorithm

	Decision Tree	**SVM**	**Random Forest**	**Logistic Regression**
Basic + TE + LM	**81.12%**	68.47%	80.46%	77.35%
Basic + TE	78.92%	64.38%	77.25%	77.34%
Basic + LM	79.46%	67.23%	77.43%	77.29%
Basic	77.88%	62.95%	77.31%	77.25%

2.5.2 Experiment result]

Participants To support our claims, we ran experiments involving 796 annotated student–system interactions, collected from 183 students enrolled for free and studying the machine learning course on the platform A between January and February, 2020. To evaluate the Wikipedia-based explanations, we conduct a second experiment. When the student gives an incorrect solution, the system shows two randomly-selected Wikipedia-based explanations (one extracted and one generated) and asks the student to select *the most helpful one*, or to select if *both are equally helpful*, or if *neither of them is helpful*. The system then asks the student to attempt the exercise again, based on which the student's learning gain is measured. It should be noted that since the student receives two hints at once, the observed learning gains are influenced by both hints shown. The results are given in Table 2.14. As would be expected, students find the originally extracted explanations more helpful on average: they are selected as helpful 55.66% of the time, while the automatically generated explanations are selected 44.44% of the time. However, when both types of explanations are shown, at least one of them is rated as helpful 83.33% of the time, with this difference in results between both types and each individual type being significant at a 95% confidence level.

This proves that, although generated explanations are slightly less helpful on average, students are far more likely to rate the feedback as helpful when both types of explanations are shown to them (as compared to only showing extracted explanations).

Lastly, as shown in Table 2.14 where students were shown two explanations (an extracted one and a generated one) and asked which one they found most useful. Afterwards, their $learning\_gain$ was determined by whether they solved the exercise in their next attempt. * indicates statistical significance compared to all other explanation preference classes at a 95% confidence level.

$$learning\_gain = post\_quiz - pre\_quiz \tag{2.7}$$

The student learning gains appear to be highly similar for both extracted and generated explanations, with no statistically significant difference between the two types. Taken together, these results strongly support the hypothesis that generated Wikipedia-based explanations can provide helpful feedback.

Table 2.14 Student preferences and learning gains for Wikipedia-based explanations.

Explanation	**Student Preference**		**Student Learning Gains**	
	Mean	95% C. I.	Mean	95% C. I.
Extracted	55.56%	[43.37%, 67.28%]	16.00%	[4.54%, 36.08%]
Generated	44.44%	[32.72%, 56.63%]	16.67%	[3.58%, 41.42%]
Extracted, Generated or Both Preferred	**83.33%***	**[72.70%, 91.08%]**	**17.65%**	[**6.76%**, **34.53%**]

On the other hand, we run our pilot with FPT[8] employees for two months with personalized and flexible Data science courses which utilized the hints above to help users understand and solve their problems. The study ran over 2 months from 21-December-2020 to 21-February-2021 with many roles (levels) of users such as: student, junior engineer, experience engineer, senior

[8] https://www.fpt.com.vn/en/

engineer, manager, other based on 200 users the respective levels. average commitment and actual studying time are 4.9 and 2.03 hours per week, respectively in Figure **??**.

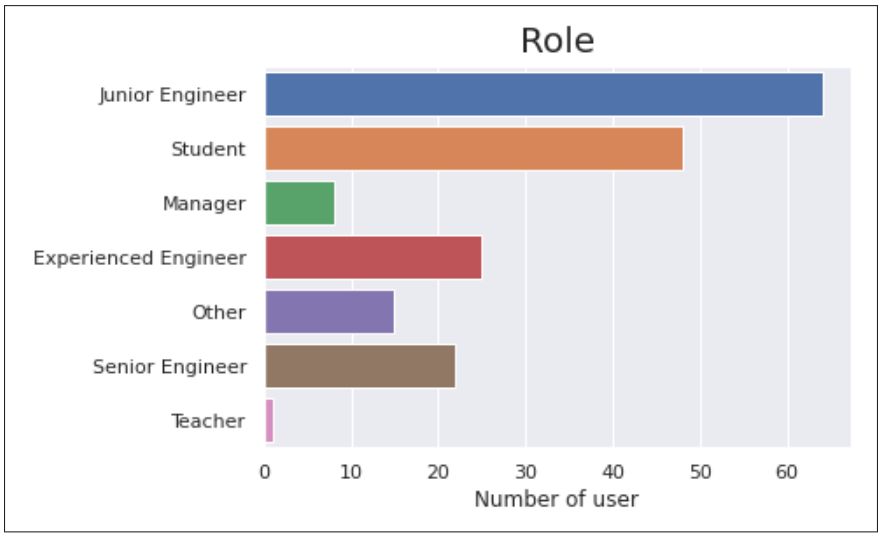

Figure 2.5 Role of users

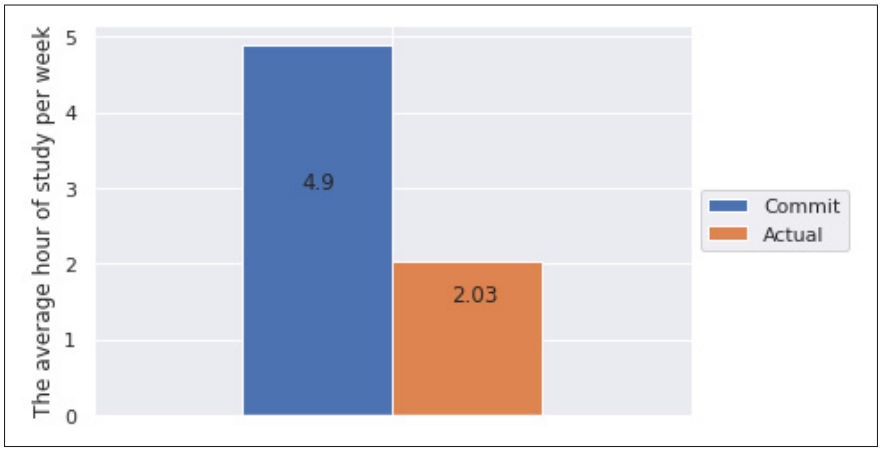

Figure 2.6 Commit vs Actual studying time

All participants were required to take an assessment quiz before the course pre_quiz and after the course $post\_quiz$, where pre_quiz and $post\_quiz$ are calculated as the percentage of right answers. Using the pre_quiz and $post\_quiz$ score, we measure the $normalized\_learning\_gain$

in Equation 2.8 to quantify how efficiently each participant has learnt.

$$normalized\_learning\_gain = \frac{post\_quiz - pre\_quiz}{100\% - pre\_quiz} \tag{2.8}$$

The numerator gives the absolute improvement in the student's score on the concept inventory. The denominator is a correction factor that takes into account the available 'headroom'. This acknowledges that it is easier for an initially low-scoring student to have a larger absolute improvement than an initially high-scoring student, since they have more opportunity to change from incorrect to correct answers. In essence, the normalised gain is the answer to the question, 'What fraction of the questions that the student got wrong before instruction did they get right after instruction?' Having calculated the individual gain for each student, the mean normalised gain, is simply the mean for all students in the class. The maximum possible normalised gain is 1.0. This occurs when a student answers all questions correctly on the post-test. If the student makes no improvement between tests their g will be 0.0, and a half-way improvement is 0.5.

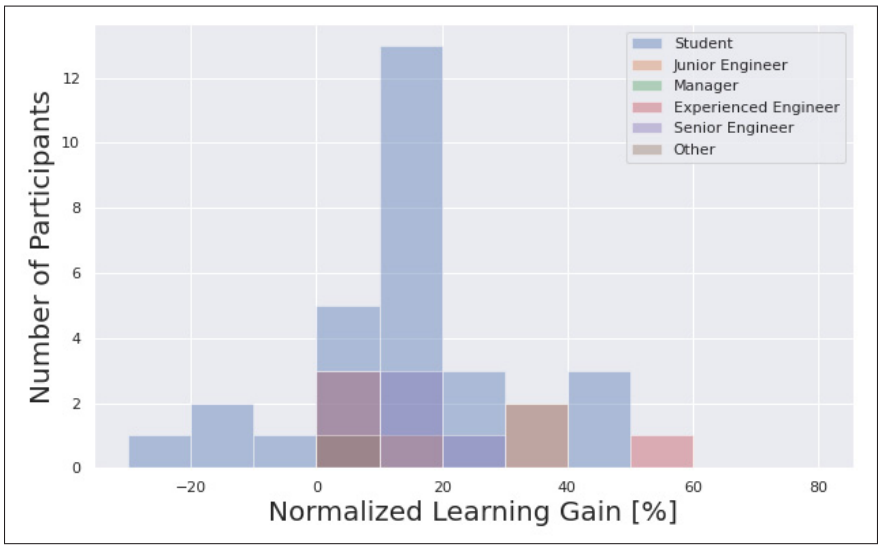

Figure 2.7 Normalized Learning gain of all employee with C.I = 95%

Of course, it is possible for a student's performance to deteriorate. In this case, the normalised gain is negative. The nature of the calculation makes this particularly acute for students who perform well on the pre-quiz: since their 'headroom' is low, even a modest deterioration can

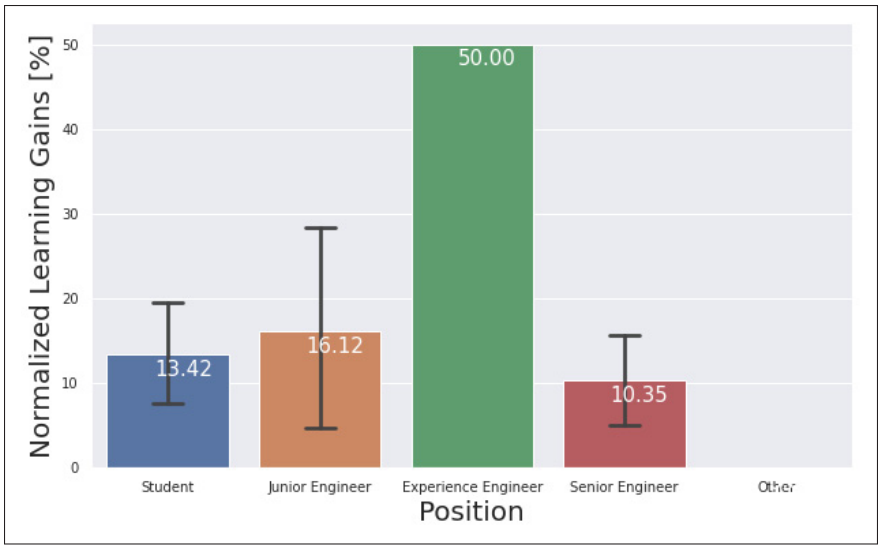

Figure 2.8 Normalized Learning gain of each employee with C.I = 95%

result in large negative gains. (In fact, g has an upper bound of +1.0 but a lower bound of) This asymmetry is one of the main criticisms of mean normalised gain as a measure, but its conceptual simplicity, ease of calculation and widespread existing use maintain its position as the leading measure of learning gain. When users get the trouble with his problem, the hints are shown by the system, then based on the hints, users try to find the solution. Hence, we defined the hint_result in Equation 2.9, which showed the effect of hint to the users.

$$hint\_result = \begin{cases} 1, \text{ user answer correctly after getting hint } (\varepsilon) \\ 0, \text{ user does not answer correctly after getting hint } (\bar{\varepsilon}) \end{cases} \tag{2.9}$$

$$hint\_result\_ratio = \frac{\sum \varepsilon_i}{\sum \varepsilon_i + \sum \bar{\varepsilon}_i}, i \in [1, n], n \text{ the number of users}$$

Figure 2.9 and Figure 2.10 show the effect of hint_result with the employees and each role, respectively. We observe in the Figure 2.9 that in the general, the $hint\_result\_ratio = 47.83\%$ which is the percentage users solving their problem with the hints. This ratio is impressive since the users are doing well with the hints generation.

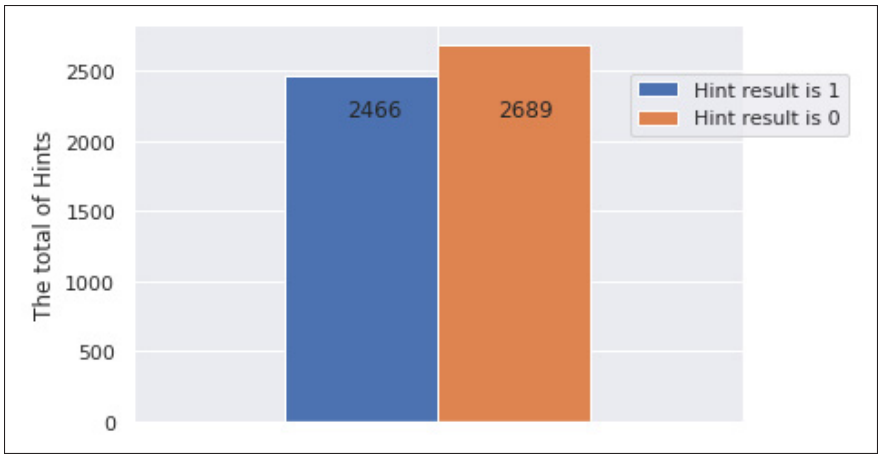

Figure 2.9 Hint result of All employee

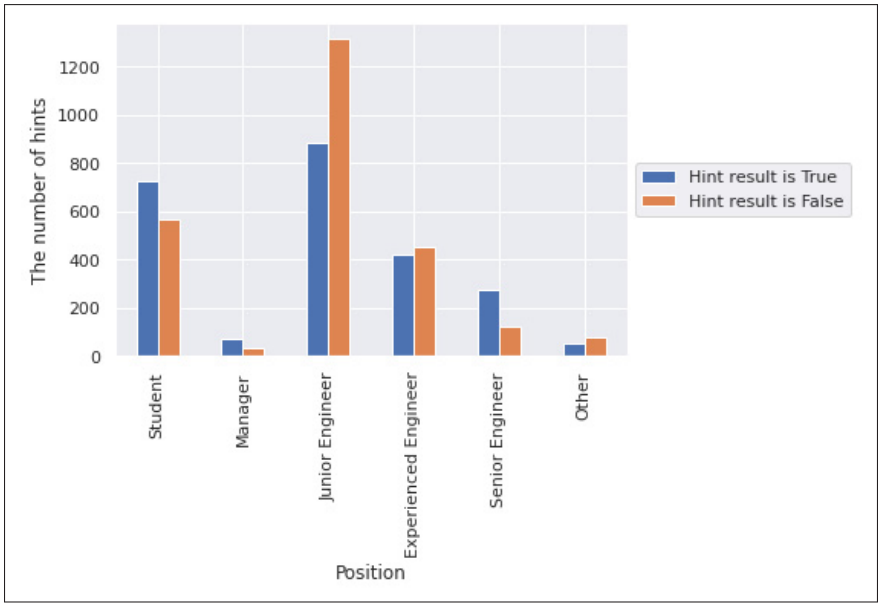

Figure 2.10 Hint result of each employee

For each role of user, the *hint_result_ratio* is different in Figure 2.9. We observed that the Student, Manager and Senior Engineer have the value of $\sum \varepsilon_j$ greater than the value of $\sum \bar{\varepsilon_j}$. On the other hand, the value of $\sum \varepsilon_j$ smaller than the value of $\sum \bar{\varepsilon_j}$ for the role Junior, Experience Engineer and Other. Most of the times, the hints are useful to help users to solve their problems. After users completed the course, we make the quick survey which is used for evaluating the

quality of the system and the feedback of the intelligence tutor which is presented in the Figure 2.11.

Table 2.15 Mean and bound of result survey with C.I = 0.95

Question	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
Mean	3.78	1.33	3.72	1.49	3.78	4.04	3.80	3.71	1.71
Lower bound	3.45	1.16	3.57	1.32	3.53	3.78	3.59	3.48	1.50
Upper bound	4.10	1.5	3.86	1.65	4.02	4.31	4.01	3.93	1.91

The Question 8 focus on evaluating the feedback quality with the scale $[1, 5]$, where 1 is unhelpful and 5 is great helpful feedback, respectively. The result of question 8 is presented in the Table 2.15 for the mean and its' bound of result feedback with the C.I = 95%. The mean value is 3.71 within the bound $[3.48, 3.93]$. On the other hand, the table 2.16 presents the details of survey results based on 50 users who got the certificate. We observe that most of users satisfy with the feedback results. This mean, the feedback (hint) helps users to improve their knowledge and follow the course well.

Table 2.16 Feedback distribution of each role

Role	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	# users
Student	3.56	1.47	3.50	1.41	3.62	4.16	3.75	3.66	1.90	32
Junior Engineer	4.29	1.38	4.14	1.88	4.14	4.14	4.00	4.00	1.12	8
Experienced Engineer	3.00	1.00	3.00	1.00	3.00	4.00	3.00	4.00	1.00	1
Senior Engineer	4.62	1.00	3.88	1.62	4.12	3.88	4.00	4.00	1.75	8
Other	3.00	1.00	3.00	1.00	4.00	3.00	4.00	3.00	1.00	1

2.6 Conclusion

In conclusion, the result results strongly support the AI tutor system to improve student learning outcomes. Future work should investigate how Wikipedia-based explanations and mathematical

hints may improve student learning outcomes, as well as interplay with student learning profiles and knowledge gaps.

2.7 Acknowledgement

The research presented in this paper was financially supported by MITACS (The Mathematics of Information Technology and Complex Systems) IT12816 and approved by the ethical committee of École de technologie supérieure (H20190508).

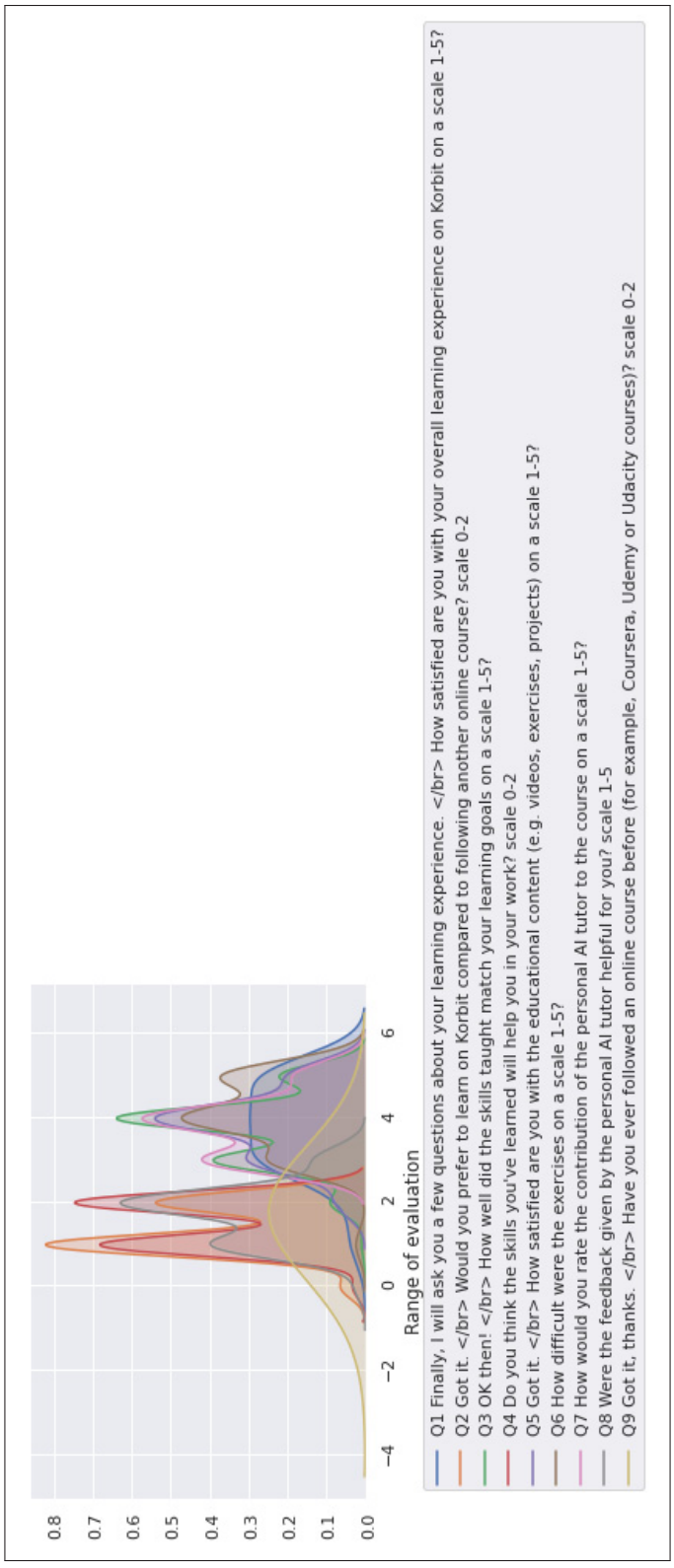

Figure 2.11 User feedback

CHAPTER 3

AUTOMATED DATA-DRIVEN GENERATION OF PERSONALIZED PEDAGOGICAL INTERVENTIONS IN INTELLIGENT TUTORING SYSTEMS

An adapt version of this chapter has been published in International Journal of Artificial Intelligence in Education 2021.

3.1 Introduction

Personalized tutoring helps students achieve their learning goals effectively (Anania (1983a), BLOOM (1984), BURKE (1980), Hrastinski, Stenbom, Benjaminsson & Jansson (2021), Hume, Michael, Rovick & Evens (1996b)). Traditionally, such personalized tutoring has been provided by human tutors. The benefits of having a human tutor include a tutor's ability to understand the effective state of the student, and thus provide personalized feedback by adapting instructions accordingly. Conventional setting, such as teaching each student's personal needs, however one-on-one tutoring is generally seen as too costly to be conducted on a large scale in most societies, an is thus not readily available.

Intelligent Tutoring System (ITS), *"computer-based instructional systems with models of instructional content that specify what to teach, and teaching strategies that specify how to teach"* (Wilson (1990), attempt to mimic personalized human tutoring in a computer-based environment and are a low-cost alternative to human tutors (Anderson, Boyle & Reiser (1985a), Nye, Graesser & Hu (2014a)). ITS are capable of providing step-by-step guidance during problem solving, tracking students' skills and knowledge development, and selecting problems on an individual basis. When compared to other computer-based learning environment (e.g., Massive Open Online Courses), ITS have been shown to be more effective in promoting learning, with the particular strength of ITS lying in their ability to deal with the interactive and personalized aspects of individual learning effectively (Hone & El Said (2016), Kulik & Fletcher (2016a), Vanlehn (2011)).

However, one major bottleneck to a wider-spread use of ITS is the expensive and laborious process of creating content and pedagogical interventions. Many ITS reply heavily on expert design and hand-crafted rules to generate system intervention, which makes them difficult to build and transfer across domains, and limits their potential efficacy and scalability (Folsom-Kovarik, Schatz & Nicholson (2010a), Olney & Cade (2015a)). In this paper, we address this major bottleneck in ITS development, and make two significant contributions.

Frist, we describe how state-of-the-art machine learning (ML) and natural language processing (NLP) techniques can be used to automatically generate data-driven *personalized hints* and *Wikipedia-based explanations*. Feedback generated this way takes the individual needs of students into account, does not require expert intervention or hand-crafted rules, and is easily scale-able and transferable across domains. Second, we demonstrate that the personalized feedback leads to substantially improved student learning outcomes and improved subjective feedback evaluation in pratice.

To support our claims, we utilize the personalized feedback models in `Korbit`, a large-scale dialogue-based ITS, which was launched in 2019 and today has over 15000 students enrolled in courses on machine learning and data science. We present the results of the experiments run on the `Korbit` learning platform remotely between January and February, 2020, involving 796 annotated student-system interactions collected from 183 students enrolled for free. We measure student success rate as the proportion of instances where a student provides a correct solution after receiving a hint or explanation from our ITS. The results show that personalized feedback provided on our platform significantly increases learning outcomes, as it leads to an average success rate of 60.47% at solving exercises on the platform. Moreover, we observe the substantial improvement in subjective feedback evaluation provided by the students.

3.2 Related Works

In this section, we first overview previous work related to the development of ITS in various domains, and then we discuss applications of NLP techniques in ITS for adaptivity, personalization and automated feedback generation.

3.2.1 Intelligent Tutoring System

Over the past two decades, many ITS have been successfully deployed to enhance teaching and improve students' learning experience in a number of domains and application areas. In particular, ITS have been actively used to teach technical subjects: from helping students acquire knowledge about theoretical concepts, mathematics (Büdenbender *et al.* (2002b); Goguadze *et al.* (2005); Hrastinski *et al.* (2019a); Melis & Siekmann (2004b); Passier & Jeuring (2006)), and logic (Abel, Chang & Pfenning (2001); Andrews, Brown, Pfenning, Bishop, Issar & Xi (2004); Kaliszyk, Wiedijk, Hendriks & Raamsdonk (2007); Stamper, Eagle, Barnes & Croy (2011b)) algorithms (AbuEl-Reesh & Abu-Naser (2018); Al-Nakhal & Abu-Naser (2017); Leelawong & Biswas (2008); to assisting students in knowledge and skill acquisitions in natural science (Makatchev, Vanlehn & Jordan (2006b); Zhang & VanLehn (2016, 2017)); to teaching real-world applications (Agha *et al.* (2018); Al Rekhawi & Abu-Naser (2018); Qwaider & Abu-Naser (2018). Apart from providing students with general assistance and feedback on their performance, ITS are able to address individual student characteristics (Graesser *et al.* (2017)) and cognitive processes (Wu & Looi (2010)). Since students differ in terms of their aptitudes and knowledge, personalized instruction in education is critical for effective learning. Personalization and adaptability of ITS to individual student needs have been shown to not only help students in independent learning, but also help teachers personalize feedback and instruction, in particular in blended and flipped-classroom environments (Baker (2016); Holstein *et al.* (2017, 2019)).

Many ITS incorporate explicit student models and consider the development of a personalized curriculum and personalized feedback (Al-Dahdooh & Abu-Naser (2017); Al-Nakhal & Abu-Naser (2017); Albacete *et al.* (2019b); Chi *et al.* (2011); Lin *et al.* (2013b); Munshi & Biswas

(2019b); Rus *et al.* (2014b,d)). In this respect, *dialogue-based* ITS have been shown to be some of the most effective tools for learning (Ahn *et al.* (2018); Graesser, Chipman, Haynes & Olney (2005); Graesser, VanLehn, Rosé, Jordan & Harter (2001b); Nye *et al.* (2014b); Ventura *et al.* (2018)), as they simulate the familiar learning environment of student–tutor interaction, which helps improve student confidence and motivation and leads to a better learning experience. In particular, dialogue-based ITS excel in their ability to ask students questions and present them with problem-solving exercises, while also providing students with the opportunity to pose their own questions, request hints and explanations, and engage in other types of communication with the tutor.

The tradition to structure tutoring around active dialogue and, in particular, in the manner of asking questions and eliciting answers related to the subject material dates as far back as the Socratic method and Plato's academy (Mills, Rice, Berliner & Rosseau (1980)). Previous research shows that when students attempt to provide answers, they get involved in such constructive activities as reflecting on the taught material, explaining material to themselves as well as to others, self-assessing and understanding the level of their knowledge, and connecting different areas of the subject, among others (Graesser & Person (1994); Hrastinski *et al.* (2019a)). Such activities are central to reasoning and understanding (Ram (1991); Webb (1989)). In addition, the selection of questions to present students with and the analysis of their performance in answering these questions is critical for curriculum structuring itself, both for human tutors and in ITS (Boaler & Brodie (2004); Jiang (2014)). Here, ITS can structure their curriculum appropriately by selecting the questions according to each student's individual development.

At the same time, the main bottleneck in providing students with personalized feedback in ITS is the ability of such systems to address the multitude of possible scenarios in student–system interactions, and this is where methods of automated, data-driven feedback generation are of critical importance. Much of the work investigating personalized feedback incorporates or takes inspiration from research on student–teacher instructional scaffolding (Van de Pol *et al.* (2010); Wood (2003)).

In this paper, we focus on delivering personalized feedback in a dialogue-based ITS during problem-solving exercises. Such feedback includes hints, explanations, elaborations, and prompts, among other pedagogical interventions. Following up on the promising results from past research, we investigate how we can leverage large amounts of open-access data in creating educational content. Of particular relevance here is the line of related work, where researchers have investigated how machine learning and large-scale, open-access resources such as Wikipedia can be utilized to generate various types of educational content and interactions with the aim of scaling up computer-based learning systems and addressing the needs of their students (Brunskill *et al.* (2018); Dinan *et al.* (2019); Guo *et al.* (2016); Liu *et al.* (2012b); Willis *et al.* (2019)). In particular, it has been shown that the use of NLP techniques in application to Wikipedia may help to generate pedagogically motivated concept maps to be used within an ITS (Lahti (2009)); identify pre-requisite relations and sequencing among learning objects to better model the learning path of the student and assess gaps in student's understanding of the subject (De Medio, Gasparetti, Limongelli, Sciarrone & Temperini (2016); Ramírez-Noriega, Juárez-Ramírez, Jiménez, Martínez-Ramírez & Figueroa Pérez (2018); Talukdar & Cohen (2012)); and generate a variety of pedagogical interventions ranging from open questions (Liu, Calvo, Aditomo & Pizzato (2012a); Shah, Shah & Kurup (2017)) to multiple-choice quizzes (Guo *et al.* (2016); Tamura, Takase, Hayashi & Nakano (2015)) across a number of subject domains.

3.2.2 Natural Language-based Interactions in ITS

A number of previous approaches designed dialogue-based ITS using natural language interface and allowing students to provide unrestricted input to the system (Stamper *et al.* (2011b); Makatchev, VanLehn, Jordan & Pappuswamy (2006a)). Previous research shows that such unrestricted interaction helps support meta-cognitive processes in students, while also helping the system identify misconceptions in student's reasoning. Since such systems provide students with an opportunity to interact with the tutor in an unrestricted manner, this leads to further challenges related to natural language understanding on the one hand, and to natural language-based

generation of interactive and personalized feedback and interventions on the another hand. This paper primarily focuses on the latter task.

In a tutorial dialogue, where one participant represents a teacher, an expert on the subject, or a more knowledgeable partner (in particular, such a partner may be represented by a human or an AI tutor) and another participant is a less knowledgeable partner (i.e., a student), hinting is a widely-used tactic (Hume *et al.* (1996b)). (Hume, Michael, Rovick & Evens (1996c)) defined a hint as *"a rhetorical device that is intended to either: (1) provide the student with a piece of information that the tutor hopes will stimulate the student's recall of the facts needed to answer a questions, or (2) provide a piece of information that can facilitate the student's making an inference that is needed to arrive at an answer to a question or the prediction of system behavior"*. Hints are aimed at encouraging students to engage in active cognitive processes that are thought to promote deeper understanding and long-term retention. It is important to note that while hints are widely used by teachers to prompt students to correct their errors, they normally do not provide the full information the students need to solve a particular problem (Hume *et al.* (1996b)). It identify hints that convey information needed to arrive at an answer and those that point students to the relevant information that they already possess as the two main types of hints used in practice. They further distinguish between hints in the form of *explanations, summaries, questions,* and *negative acknowledgements*. In this work, we focus on generating hints in the form of *explanations*, pointing students at the relevant information and conveying related facts without revealing the actual answer.

Previous work investigated the impact of data-driven hints on educational outcomes in terms of learning and persistence. In particular, (Stamper, Eagle, Barnes & Croy (2011a)) augment their Deep Thought logic tutor with a Hint Factory that generates data-driven, context-specific hints for an existing computer aided instructional tool. The results show that students who receive hints attempt and complete significantly more problems than the control, no-hint group. Moreover, students who receive hints early in the learning process outperform all other students in the post-test. These results suggest that data-driven hints are effective in promoting learning. However, the data-driven component in the Hint factory is primarily concerned with the

automated generation of hint sequences of varying complexity. In contrast, our work addresses the NLP-based generation of hints in a natural language.

There is a growing body of research on automated hint generation for programming exercises (McBroom, Koprinska & Yacef (2021), Price, Dong, Zhi, Paaßen, Lytle, Cateté & Barnes (2019)). Most work in this area is concerned with *generation of a suitable sequence of hints* to provide to students at specific points during their learning, and hints are mainly generated using templates combining mixed-language input (Rivers (2017)). This line of work is related to ours. However, we note that hints related to programming exercises are mostly concerned with procedural knowledge, whereas our platform addresses both procedural and declarative knowledge. In addition, interactions on our platform are more open-ended and involve more unrestricted language.

NLP techniques have also been widely used to model ITS's natural language understanding (NLU) components. For instance, Benzmüller, Horacek, Kruijff-Korbayova, Pinkal, Siekmann & Wolska (2005) introduce a dialogue system into a mathematical assistance tool, where a student builds a proof by producing natural language utterances, and the system provides them with domain-specific hints produced when the student is stuck or shows non-understanding of domain concepts. The NLU modules in Benzmüller *et al.* (2005) uses a specialized syntactic parser and relies on an in-domain semantic interpretation. Similar to this work, Aleven, Popescu & Koedinger (2001) are mostly concerned with the challenges in NLU and the interpretation of mixed language input from the student, rather than with the natural language-based generation of pedagogical interventions.

Finally, Zhang & Vanlehn (2016) and Zhang & VanLehn (2017) consider the use of NLP techniques in automated generation and adaptation of questions on biology to learner profiles using a semantic network, thus alleviating the need for domain expert intervention. They show that students with adaptive question selection have larger learning gains than those with adaptive question selection.

To summarize, in contrast to the previous work, we apply NLP techniques to generate hints expressed in a natural language. Since we do not rely on hand-crafted rules or templates, our methodology can be applied to any input domain and potentially address both declarative and procedural knowledge.

3.3 Korbit Learning Platform

`Korbit` is a large-scale, open-domain, mixed-interface, dialogue-based ITS, which uses machine learning, natural language processing (NLP), and reinforcement learning (RL) to provide interactively, personalized learning online. The ITS has over 7,000 students enrolled worldwide, including students from educational institutions and professionals from industry partners. `Korbit` is capable of teaching topics related to data science, machine learning, and artificial intelligence. The platform is highly modular and will soon be expanded with many more topics.

Students enroll on the `Korbit` website by selecting either a course or a set of skills they would like to study. Students may also answer a few questions about their background knowledge. Based on these, `Korbit` generates a personalized curriculum for each student. Following this, `Korbit` tutors the student by alternating between short lecture videos and interactive problem-solving exercises. The `outer-loop` system decides on which lecture video or exercise to show next based on the personalized curriculum. The initial curriculum currently determines the ordering of videos and exercises, but work is underway to adapt the curriculum during the learning process.

During the exercise sessions, the `inner-loop` system manages the interaction. First, it shows the student a problem statement (e.g.,, a question). The student may then attempt to solve the exercise, ask for help, or skip the exercise. If the student attempts to solve the exercise, their solution attempt is compared against the expectation (i.e. reference solution) using an NLP model. If their solution is classified as incorrect, then the `inner-loop` system will select one of a dozen different pedagogical interventions. The pedagogical interventions include textual

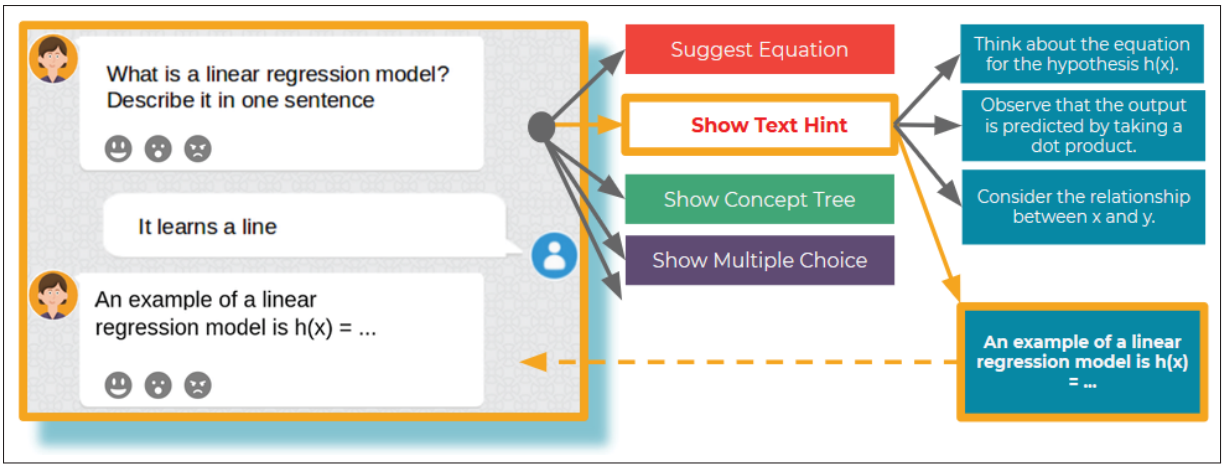

Figure 3.1 An example of how the `Korbit` ITS inner-loop system selects the pedagogical intervention. The student gives an incorrect solution and afterwards receives a text hint.

hints, mathematical hints, elaborations, explanations, concept tree diagrams, and multiple-choice quiz answers. The pedagogical intervention is chosen by an ensemble of machine learning models based on the student's profile and last solution attempt. Depending on the pedagogical intervention, the `inner-loop` system may either ask the student to retry the initial exercise or follow up on the intervention (e.g., with additional questions, confirmations, or prompts).

The `Korbit` ITS is closely related to the line of work on dialogue-based ITS, such as the pioneering AutoTutor and the newer IBM Watson Tutor (Ahn *et al.* (2018); Graesser *et al.* (2005, 2001b); Nye *et al.* (2014b); Ventura *et al.* (2018)). Although `Korbit` is highly constrained compared to existing dialogue-based ITS, a major innovation of `Korbit` lies in its modular, scalable design. The `inner-loop` system is implemented as a finite-state machine. Each pedagogical intervention is a separate state, with its own logic, data, and machine learning models. Each state operates independently of the rest of the system, has access to all database content (including exercises and lecture videos), and can autonomously improve as new data becomes available. This ensures that the system gets better and better, that it can adapt to new content, and that it can be extended with new pedagogical interventions. Furthermore, the transitions between the states of the finite-state machine are decided by a reinforcement learning

model, which is agnostic to each state's underlying implementation and continues to improve as more and more data becomes available.

3.4 Methodology

The `Korbit` ITS utilizes different types of data sources to generate a large variety of personalized feedback automatically. In this section, we describe in detail the automatic generation process for *personalized hints and explanations*, *Wikipedia-based explanations*, and *mathematical hints*. These constitute three of the many intervention types employed by the `Korbit` ITS. We also present three personalized hints and explanations selection models.

3.4.1 Personalized Hints and Explanations

Personalized hints and explanations are generated using NLP techniques and assessed according to several metrics related to the quality of the hint or explanation and the past interaction of the system with the student.

3.4.1.1 Hints and Explanations Generation

The system generates many of hints and explanations by applying linguistic patterns to all expectations (i.e., reference solutions) available in our database. Table 3.1 demonstrates some examples of hints generated using our 3-step algorithm detailed below:

1. *Identification of keywords and keyphrases*: These include nouns and noun phrases within the question. In the examples in Table 3.1 such keywords and phrases are marked with text boxes: for instance, *overfitting*, *underfitting* and *logistic regression* are automatically identified as keywords and phrases.
2. *Identification of an appropriate sentence span*: It would seem likely the best hints should not include keywords, keyphrases, and related words as they may reveal the solution to the student. We apply state-of-the-art dependency parsing with `spaCy`[1] to eliminate parts of

[1] https://spacy.io

the expectation sentences that contain keywords and phrases: for instance, in Table 3.1 the clause *A model is underfitting* is filtered out, while *it has a high bias* is considered as a candidate for hint generation.

3. *Generation of a grammatically correct hint* is done automatically using discourse-based modifications (e.g., *Think about the case*) and the partial hint extracted from an expectation in step (2) (e.g. *when it has a high bias*).

Table 3.1 Text-based hint generation. Keywords and phrases are marked with boxes, discourse-based modifications are underlined.

Question	Expectation	Generated hint
What is the difference between overfitting and underfitting?	A model is underfitting when it has a high bias.	Think about the case when it has a high bias.
Would you use linear regression or logistic regression to model a classification problem?	I would use logistic regression, because the outputs are discrete.	Think about the following: the outputs are discrete.

3.4.1.2 Personalized Hints and Explanations Selection

Once hints and explanations are generated with the algorithm described above, they are ranked based on their quality and appropriateness for each student. The appropriateness of this ranking determines the quality of the personalized feedback provided to the student. We employ a machine learning approach and utilize the Random Forest classifier from the `scikit-learn`[2] suite (Breiman (2001). The algorithm considers various sets of features described below. The sets of features considered define the complexity of the feedback selection model.

1. BASELINE MODEL relies on the use of **linguistic features**, which assess the quality of the hint or explanation from the linguistic perspective only. These features do not take into account personal aspects of the student–system interaction and only assess feedback (hint or

[2] https://scikit-learn.org

explanation) in isolation. This set contains a total of 14 features to capture various aspects of the generated feedback, including its quality, grammaticality, and appropriateness to the question. We describe some of the features below.

- We measure the *length* of the hint / explanation in terms of the number of words, which helps the algorithm learn how comprehensive suggested feedback is.
- *Completeness of the parse tree* is measured using the proportion of sentences in the hint / explanation that contain subject-verb structure: For instance, this feature would penalize incomplete sentences like *"Note that grow with the size of the dataset"*, which would be generated by the hint-generation algorithm described in Section 3.4.1.1 using a combination of a discourse-based modification *"Note that"* and the partial hint *"grow with the size of the dataset"* extracted from an expectation after the keyphrase *"non-parametric models"* is eliminated.
- *Perplexity score* is estimated with a Wikipedia-based language model, and it helps the algorithm assess the quality, fluency, and grammaticality of generated feedback.
- *Keyword overlap* and *topic overlap* between the hint / explanation and the question help assess the fit of generated feedback for the question: the more related feedback is to the question, the higher is the overlap between the two in terms of words and topics. Here, we define "topics" narrowly as the titles of the Wikipedia articles that contain possible definitions of the keywords and phrases (see Section 3.4.2 for more details on our Wikipedia-based approach).
- *Average uniqueness score* of the keywords in the hint / explanation is estimated as an average of the inverse-document frequencies of the keywords according to their use across reference solutions. This feature helps the algorithm estimate how informative a keyword or phrase is: the more frequently it occurs in reference solutions to various questions in our database, the less specific it is about any given question. An example of such a generic keyword is the *model*.

- *Ambiguity of the keywords* is further estimated as the number of senses associated with a word in WordNet,[3] which we access via the `NLTK` interface.[4]
- Features based on the *proportion of lexical items of a certain type* (for instance, pronouns and named entities) are used as proxies for specificity of the hint's / explanation's content.

2. SHALLOW PERSONALIZATION MODEL relies on the combination of **linguistic features** pertaining to the hint / explanation **and performance-based features**. Performance-based features consider past student performance and include the total number of attempted questions, proportion of correctly and incorrectly answered questions, and the total length of the student–system dialogue interaction, among other features. As a result, this model uses a total of 22 features.
3. DEEP PERSONALIZATION MODEL, in addition to the 22 features described above, takes into account up to 4 previous interaction turns between the student and the system (selected to maximize overall coverage of interactions that are currently available on our platform) and analyzes them from the linguistic point of view by taking the proportion of keywords and topics overlapping between the question and each of the student statements. This final model relies on 49 features in total and combines **linguistic features** about the hint / explanation, **performance-based features, and linguistic features** applied to the past student–system interactions.

Thus, our feedback selection models get increasingly more complex in terms of the amount of personalization involved – from no personalization in the BASELINE MODEL based on the linguistic quality of the hint / explanation only, to the SHALLOW PERSONALIZATION MODEL that adds high-level, quantitative student performance metrics, to the DEEP PERSONALIZATION MODEL that also takes into account dialogue-based interactions between the student and the platform.

[3] https://wordnet.princeton.edu

[4] http://www.nltk.org

The models are trained and evaluated on a collection of 450 previously recorded student–system interactions of up to 4 turns in length.[5] The models are trained in a binary classification setting to predict if a student given a specific hint or explanation will correctly solve the exercise in their next attempt.

Table 3.2 Accuracy and F1 scores of different hint and explanation selection models (with 95% confidence intervals) calculated based on cross-validation with $k = 50$ folds. * indicates statistical significance compared to baseline model at a 95% confidence level.

Model	Accuracy	F1-score
RANDOM	53.64% ± 3.99%	48.21% ± 3.63%
BASELINE (No Personalization)	60.57% ± 4.45%	54.90% ± 4.74%
SHALLOW PERSONALIZATION	68.75% ± 4.06%	62.23% ± 4.49%
DEEP PERSONALIZATION	**86.71**%* ± **3.34**%	**84.81**%* ± **3.97**%

Table 3.2 shows the results. The RANDOM model achieves an accuracy of 53.64% ± 3.99% and an F1 score of 48.21% ± 3.63%. The BASELINE system that relies on the linguistic features only to select the best matching explanation reaches slightly higher performance. Taking individual performance measures into account brings considerable improvements in the results, with the SHALLOW PERSONALIZATION model achieving an accuracy of 68.75% ± 4.06% and an F1 score of 62.23% ± 4.49%. The best performing model overall uses DEEP PERSONALIZATION and achieves 86.71% ± 3.34% accuracy and 84.81% ± 3.97% F1 score, which are statistically significant improvements at a 95% confidence level over all other models. Therefore, we should expect the DEEP PERSONALIZATION model to select the most appropriate personalized feedback.

3.4.2 Wikipedia-Based Explanations

Wikipedia-based explanations provide alternative ways of helping students to understand and remember concepts more effectively. With over 6 million articles containing over 3.5 billion

[5] These students–system interactions were recorded from an earlier version of the `Korbit` ITS, which selected the hints to show uniformly and randomly, i.e., without any consideration for the student performance or the hint quality.

words, English Wikipedia provides extensive material for the NLP component of our system. The hierarchical structure of the hyperlinks imposed by the Wikipedia format facilitates the identification of the sets of pages related to the topic. In addition, the format adopted for Wikipedia articles themselves, where the first sentence typically defines the title concept and the first paragraph presents a concise description of the topic (Kapugama *et al.* (2016), makes information extraction easier. Thus, by generating a large set of Wikipedia-based explanations for the subject domain (e.g., hundreds of explanations for each exercise in `Korbit`), a personalized feedback model may be able to target a larger set of student knowledge gaps and provide more effective help.

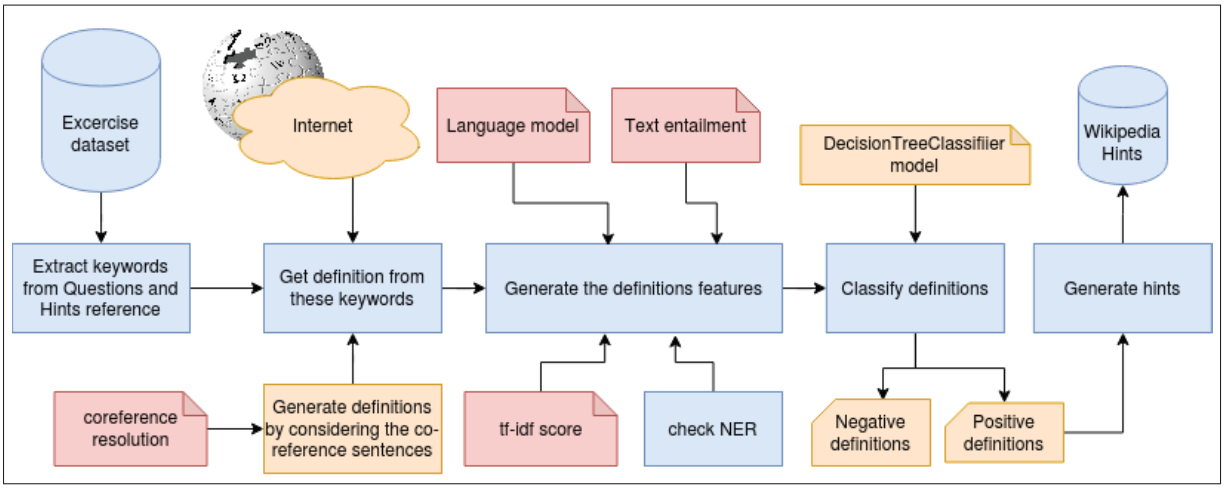

Figure 3.2 The *Wikipedia explanations* multi-stage generation pipeline. *"Positive definitions"* refer to the high-quality explanations, while *"negative definitions"* are low-quality ones.

To generate Wikipedia-based explanations, we use a multi-stage generation pipeline. This is illustrated in Figure 3.2. Five major stages in this pipeline are:

1. Extracting keywords from questions and expectations (reference solutions)
2. Generating candidate explanations based on these keywords
3. Extracting features for candidate explanations classification
4. Evaluating candidate explanations with respect to their quality level
5. Selecting all relevant Wikipedia explanations

In the first stage, all relevant domain keywords are extracted from the reference questions and solutions by extracting noun phrases and pronouns using `spaCy` (some examples of extracted keywords and relevant articles are presented in Table 3.3). We disambiguate abbreviations frequently used for technical terms (Schwartz & Hearst (2003),collect synonyms for the keywords using the WikiSynonyms API,[6] and apply co-reference resolution to substitute pronouns (Clark & Manning (2016).

Table 3.3 Examples of keywords and relevant articles

Domain	Keyword	Relevant articles
“Machine Learning" “Estimation theory" “Deep learning" “Robotics"	“Autonomous car”	“Glossary of artificial intelligence” “Microbotics” “Autonomous things” “Self-driving car”
	“Linear regression”	“Statistical learning theory” “Deep learning” “Linear regression” “Pattern recognition”

Next, using NLP techniques, we create a set of *extracted Wikipedia-based explanations* based on the first sentence in each article and generate *candidate Wikipedia-based explanations* using the rest of the article. We consider the *extracted Wikipedia-based explanations* to be “high quality” explanations since they are grammatically correct and describe the article topic clearly and succinctly. On the other hand, the majority of *candidate Wikipedia-based explanations* may be expected to be of “low quality” (e.g., containing irrelevant, off-topic information or being grammatically incorrect) since they were generated from different snippets of each Wikipedia article. Thus, to select the most appropriate *candidate Wikipedia-based explanations*, we train several binary classification models to label an explanation as being either “high quality” or “low quality” based on its linguistic features. The best performing classification model is then used to select the set of best *candidate Wikipedia-based explanations*, which we will refer to as the *generated Wikipedia-based explanations* (since our pipeline generated them).

[6] https://rapidapi.com/ipeirotis/api/wikisynonyms

A 2GB in-domain ("Machine learning") dataset was crawled from Wikipedia and re-sampled using the SMOTE algorithm (Mathew *et al.* (2018) to tackle the oversampling problem. We extracted several features using NLP techniques, including co-reference resolution score (Clark & Manning (2016), language model score using a state-of-the-art LSTM neural network (Al-Rfou, Choe, Constant, Guo & Jones (2018), textual entailment-based relations using a state-of-the-art attention-based neural network (Parikh, Täckström, Das & Uszkoreit (2016), TF-IDF scores, and named entity classes (Nothman, Ringland, Radford, Murphy & Curran (2013), among others. We experimented with a number of models, including Decision Tree classifiers (Breiman *et al.* (1984), Random Forests (Breiman (2001), Logistic Regression (Bishop (2006b), and Support Vector Machines (Smola & Schölkopf (2004), using the `scikit-learn` implementation. Among those, the Decision Tree classifier performed best in a 50-fold cross-validation experiment, yielding an average accuracy of 81.87% ± 3.2% and average F1-score of 96.97% ± 0.6%. Examples of originally extracted and automatically generated Wikipedia explanations are shown in Table 3.4.

Table 3.4 Examples of Wikipedia-based explanations. Identified keywords are marked with boxes, and information that helps guide a student is highlighted in italics.

Question	Wikipedia explanations	Label
How many human drivers would be needed to drive an autonomous car?	A self-driving car, also known as an autonomous vehicle (AV) connected and autonomous vehicle (CAV), driverless car, robot car, or robotic car, is a vehicle that is capable of sensing its environment and moving safely *with little or no human input*.	Extracted
	Different methods and levels of autonomy can be achieved through monitoring and remote control from a nearby manned ship, an onshore control center or through artificial intelligence and machine learning, *letting the vessel itself decide* the course of action.	Generated

3.4.3 Mathematical Hints

One of the key novel features of the `Korbit` ITS is its ability to evaluate free-form math equations, written using LaTeX. Math equations are particularly challenging to evaluate and give feedback because equivalent mathematical expressions can have different string representations. Moreover, the notation between students may vary, and the notation itself can be ambiguous. For example, the equation "$y(x+5)$" has two interpretations, as shown in Figure 3.3: y could be a function or a term multiplied by $x+5$. To evaluate students' equations, we first convert their LaTeX string containing the equation into multiple parse trees, where each tree represents a possible interpretation of the equation. We train a scoring function that looks at parse tree features such as number of variables, number of operators, number of functions, and number of repeating variables and functions to assign an energy value to each candidate parse tree. The lowest energy tree is selected as the most likely intended parse tree.

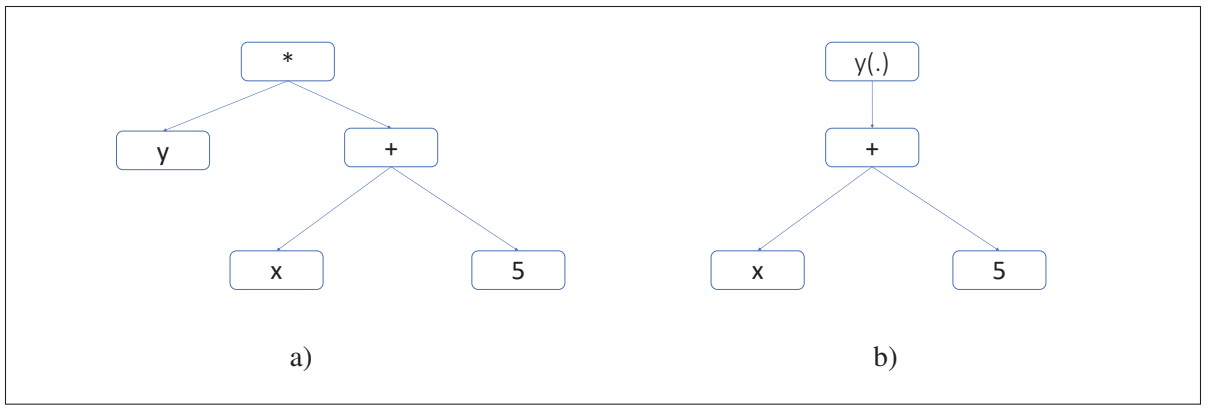

Figure 3.3 Example of Math Equation

To determine if an equation is correct, the ITS compares the student's most likely parse tree to the parse trees formed by equations in the expectations (i.e. reference solutions). If a match is found, then the solution is accepted.

There are two ways in which students receive mathematical hints from the `Korbit` ITS. First, when an exercise has a LaTeX equation inside an expectation, the system will provide a mathematical hint in the form of four suggested equations, where some terms (such as variables

and operators) are hidden. One of these equations is generated from the expectation, and the remaining three are generated from incorrect solutions. The student must select the appropriate masked equation and fill it in with the correct symbols.

Second, suppose a student inputs an incorrect equation. In that case, the system will attempt to match it to a similar reference equation and provide a mathematical hint on what the student needs to change in their equation. The hint is based on the edits that the student's equation parse tree must undergo to become the parse tree of an expectation (e.g., changing the contents of a node, adding nodes, or removing nodes from the equation parse tree). This method only compares student equations to reference equations and is agnostic to everything else. This allows us to provide hints for equations in any domain. However, the hints are also domain-independent – they can only tell a student *in what way* their equation is wrong, and what to change to correct it, but not identify conceptual errors. Only this second type of mathematical hints is personalized to each student and will be considered in the experiments section.

Being fully automated, feedback provided on the `Korbit` platform differs from that on other ITS teaching mathematics, where hand-crafted test cases typically generate feedback (Büdenbender *et al.* (2002b); Hennecke (1999).

3.5 Experiment

This section presents the results obtained with the `Korbit` ITS using personalized feedback. The experiments involve 796 annotated student–system interactions, collected from 183 students enrolled for free and studying the machine learning course on the `Korbit` learning platform remotely between January and February 2020.

3.5.1 Personalized Hints and Explanations

To evaluate the personalized hints and explanations, a hint or explanation is selected at uniform random from one of the personalized feedback selection models when a student gives an incorrect solution. Afterward, the student learning gain is measured as the proportion of instances where a

student provides a correct solution after receiving a personalized hint or explanation. Since it is possible for the ITS to provide several pedagogical interventions for a given exercise, we separate the learning gains observed for all students from those for students who received a personalized hint or explanation before their second attempt at the exercise. To accurately measure student learning gains, all student solutions are manually annotated by domain experts as being either correct or incorrect.

The results are given in Table 3.5. In line with the results from Table 3.2, the DEEP PERSONALIZATION MODEL leads to the highest student learning gains at 48.53% followed by the SHALLOW PERSONALIZATION MODEL at 46.51% and the BASELINE MODEL at 39.47% for all attempts. Furthermore, the difference between the learning gains of the DEEP PERSONALIZATION MODEL and BASELINE MODEL for the students before their second attempt is statistically significant at 95% confidence level based on a z-test (p=0.03005). These results strongly support the hypothesis that automatically generated personalized hints and explanations lead to substantial student learning gains.

Table 3.5 Student learning gains for personalized hints and explanations with 95% confidence intervals (C.I.). After being shown a hint or explanation, their learning gain was determined by whether they solved the exercise in their next attempt. * indicates statistical significance compared to baseline model at a 95% confidence level.

	All Attempts		**Before Second Attempt**	
Model	**Mean**	**95% C.I.**	**Mean**	**95% C.I.**
BASELINE (No Personalization)	39.47%	[24.04%, 56.61%]	37.93%	[20.69%, 57.74%]
SHALLOW PERSONALIZATION	46.51%	[31.18%, 62.34%]	51.43%	[33.99%, 68.62%]
DEEP PERSONALIZATION	**48.53%**	[**36.22%**, **60.97%**]	**60.47%***	[**44.41%**, **75.02%**]

3.5.2 Wikipedia-Based Explanations

To evaluate the Wikipedia-based explanations, we conduct a second experiment. When the student gives an incorrect solution, the system shows two randomly-selected Wikipedia-based explanations (one extracted and one generated) and asks the student to select *the most helpful one*, or to select if *both are equally helpful*, or if *neither of them is helpful*. The system then asks the student to attempt the exercise again, based on which the student's learning gain is measured. It should be noted that since the student receives two hints at once, the observed learning gains are influenced by both hints shown.

The results are given in Table 3.6. As expected, students find the originally extracted explanations more helpful on average: they are selected as helpful 55.66% of the time, while the automatically generated explanations are selected 44.44% of the time. However, when both types of explanations are shown, at least one of them is rated as helpful 83.33% of the time, with this difference in results between both types and each type being significant at a 95% confidence level. This proves that, although generated explanations are slightly less helpful on average, students are far more likely to rate the feedback as helpful when both types of explanations are shown to them (compared to only showing extracted explanations). Lastly, as shown in Table 3.6, the student learning gains appear to be highly similar for both extracted and generated explanations, with no statistically significant difference between the two types. Taken together, these results strongly support the hypothesis that generated Wikipedia-based explanations can provide helpful feedback.

3.5.3 Mathematical Hints

For this last experiment, we used two different methods to evaluate the quality of the provided mathematical hints. We first collected 86 previous student–system interactions with the ITS, where the student's solution attempt was incorrect and contained a mathematical equation. Two domain experts then independently labeled the feedback from the ITS as “very useful",

Table 3.6 Student preferences and learning gains for Wikipedia-based explanations. Students were shown two explanations (an extracted one and a generated one) and asked which one they found most useful. Afterward, their learning gain was determined by whether they solved the exercise in their next attempt. * indicates statistical significance compared to all other explanation preference classes at a 95% confidence level.

	Student Preference		**Student Learning Gains**	
Explanation	**Mean**	**95% C. I.**	**Mean**	**95% C. I.**
Extracted	55.56%	[43.37%, 67.28%]	16.00%	[4.54%, 36.08%]
Generated	44.44%	[32.72%, 56.63%]	16.67%	[3.58%, 41.42%]
Extracted, Generated or Both Preferred	**83.33%***	**[72.70%, 91.08%]**	**17.65%**	[**6.76%**, **34.53%**]

"somewhat useful" or "not useful".[7] The results, presented in Table 3.7, show that the majority of the mathematical hints were labeled as "very useful", with over 90% labeled as either "very useful" or "somewhat useful".

We also measure the learning gains of the feedback on student data collected between May 1st, 2020, and June 21st, 2020. We observe a learning gain of 41.38% with a 95% confidence interval of [23.50%, 61.10%]. Although the learning gains from the mathematical hints do not reach the level of the deeply personalized hints, we note that these two values cannot be directly compared. Indeed, the data for this experiment was collected from different students than the ones from the previous experiments. In addition, the problem of understanding math equations is different from the problem of understanding natural language and thus requires a completely different approach to solve.

Some examples of feedback are provided in Table 3.8. The useful feedback states what the student must change to get the correct solution. Most of the "not useful" feedback is due to students writing syntactically incorrect LaTeX equations, such as forgetting to use "_" to subscript indices.

[7] The inter-annotator agreement between experts had a Spearman's rank correlation of 0.623.

Table 3.7 Quality of feedback provided by mathematical hints.

Feedback Quality	**Mean**	**95% C.I.**
Not useful	9.3%	[4.1%, 17.51%]
Somewhat useful	39.53%	[29.15%, 50.66%]
Very useful	51.16%	[40.14%, 62.10%]
Learning gains	41.38%	[23.50%, 61.10%]

We also note that the majority of students prefer to fill in the equations given by our automatically generated mathematical hints (described in Section 3.4.3) instead of writing their own equations. This means that their equations are often close to an expectation's equation, which makes it easy to determine the appropriate feedback to give.

Finally, as stated earlier, the main limitation of the mathematical hints is that they do not capture any of the conceptual information described by the equations and can only tell the student *how* to correct their equations, but not *why* they need to be corrected. Therefore, in the future, we plan to expand this system with explanatory formative feedback focusing on why the equation is conceptually incorrect.

Table 3.8 Examples of mathematical feedback provided

Feedback Quality	**Student Solution**	**Feedback Provided**
Very Useful	$\sum_i \lvert h(x_i) - y_i \rvert$	Hmm... I expected to see the expression $(h(x_i) - y_i)^2$ in your answer.
Somewhat Useful	$J^*(h_1) < J^*(h_2)$	Let's see... I think you should change the $<$ in $J^*(h_1) < J^*(h_2)$ into $>$.
Not Useful	$\phi(y) = x1 * x2$	Hmm... I think something is incorrect in $x * 1 * x * 2$. In particular, you should not be multiplying $x * 1$ and x.

3.6 Discussion

In this paper, we have proposed methods for automated generation and personalization of feedback in an intelligent tutoring system (ITS). In particular, we have focused on generation and personalization of text-based hints and extraction and generation of Wikipedia-based explanations leveraging large amounts of potentially useful data available for learner needs on Wikipedia. We generate each of these types of feedback in a fully automated manner, using data-driven approaches and state-of-the-art machine learning and natural language processing techniques. We have conducted several experiments investigating the utility of personalized feedback, including measuring student success rates and students' subjective preferences for each type of feedback. The experiments strongly support our hypothesis that the personalized hints and Wikipedia-based explanations help to improve student learning outcomes. In this work, we have shown that personalized feedback automatically generated in a data-driven way leads to improved learning outcomes measured as the success rate in the students' ability to answer the questions on the material correctly after being provided with an informative hint. This is a crucial step towards solving one of the major bottlenecks for large-scale ITS, which have often relied on expert design and hand-crafted rules in the past. Future work will investigate the scalability and transferability of our personalized data-driven feedback models across multiple domains. One limitation of the current work is that we measure learning outcomes as success in answering the question immediately after personalized feedback is provided. We believe that observed improvements are important as they show that the generated hints and explanations are helpful and guide students in the right direction. However, future experiments on our platform will address student learning gains and their ability to retain knowledge, which can be tested using delayed post-tests on the relevant concepts, as well as to perform near transfer (i.e., testing knowledge of the same concept in a similar context), and far transfer (i.e., testing knowledge of the same concept in a new context). Such future experiments will further support the usefulness of the automatically generated personalized feedback. An additional challenge for ITS that teach technical subjects, such as machine learning, data science, and artificial intelligence, lies in the combination of various modalities and the use of mixed language

involved in generating the pedagogical interventions and the provision of feedback. An ITS in these domains must evaluate answers expressed in a purely textual form and provided by the students in response to the questions that are, likewise, expressed in a natural language (e.g., *"What is a linear regression model?"*). However, ITS focusing on technical domains must also handle other modalities, such as mathematical equations, chemical equations, source code, and so on. For example, an ITS teaching machine learning will often have to evaluate and provide feedback on mathematical expressions. On the one hand, such expressions may be included in student answers: e.g.a mathematical expression would be expected as a response to the question *"Define the sum-of-squares error function"* from an ITS. On the other hand, mathematical expressions may be included in the mixed-modality questions, which may further combine them with textual content, as does a question like "Suppose the output is categorical with ten categories ($y = 1, 2, \cdots, 10$). If $y_i = 9$, then what would its corresponding one-hot vector representation be?". This proved to be particularly challenging in the past, with many systems aiming to provide feedback on mathematical expressions resorting to hand-crafted rules Büdenbender, Frischauf, Goguadze, Melis, Libbrecht & Ullrich (2002a), Goguadze *et al.* (2005), Hennecke (1999) or involving a human tutor Cukurova, Mavrikis, Luckin, Clark & Crawford (2017),Hrastinski, Stenbom, Benjaminsson & Jansson (2019b). In addition, as Benzmüller *et al.* (2005) and Dietrich & Buckley (2008) note, students' responses using mixed language are often characterized by underspecification and ambiguity, with the latter being typical of both natural language and mathematical expressions. Math equations are particularly challenging to evaluate and give feedback because equivalent mathematical expressions can have different string representations. Moreover, the notation between different students may vary, and the notation itself can be ambiguous Dietrich & Buckley (2008). For example, the equation "$y(x + 5)$" has two interpretations, as shown in Figure 3: y could be a function or a term multiplied by $x + 5$. Our ongoing research is concerned with the models capable of analyzing math equations in addition to purely text-based content and providing relevant feedback. Preliminary results show that our data-driven mathematical hints provide students with useful insights. In the future, we also plan to expand the set of hints with those on programming exercises and investigate students' learning outcomes from the feedback that complements textual hints with mathematical

equations and code snippets or instructions relevant for the specific taught concepts. We have shown that students generated hints and Wikipedia-based explanations helpful. Future work should also investigate how and what types of hints and explanations may improve student learning outcomes and their interplay with student learning profiles and knowledge gaps. In particular, we plan to investigate how varying hint complexity and the level of hint transparency can be used in instructional scaffolding. In addition, we will explore how large amounts of available learning material can be leveraged to generate further pedagogical interventions in a data-driven way. Of particular importance for future work is the development of models capable of explanatory formative feedback. Such models can be applied both to mathematical hints, providing students with further insights as to why their equations may be incorrect, and to textual hints and explanations, identifying what is missing or what is conceptually incorrect in the given answer and providing students with the guidance towards fixing the missing or incorrect ideas in their answers. Future work should also investigate the interplay between the granularity of such formative feedback and various student learning profiles. Finally, it should be noted that there has been a massive increase in the use of ITS, and more broadly, online learning platforms, separate from and alongside traditional human teacherstudent interactions (for example, in flipped classrooms and blended learning environments). Therefore, future research must look closely into such aspects of the learning process as student motivation, engagement, and managing students' emotional states. Of particular interest are such questions as to whether tutoring via an ITS should mimic human tutoring or rather provide students with an alternative means of learning and which aspects of the learning process are best addressed with an ITS tutor versus a human one.

CHAPTER 4

A COMPARATIVE STUDY OF LEARNING OUTCOMES FOR ONLINE LEARNING PLATFORMS WILL TRANSFORM ONLINE LEARNING FOR MILLIONS

An adapt version of this chapter has been published in Artificial Intelligence in Education. AIED 2021. Lecture Notes in Computer Science(), vol 12749. Springer and arXiv:2203.03724v1.

4.1 Introduction

In a comparative head-to-head study, we investigate the learning outcomes induced by two popular online learning platforms. Coursera is a widely used learning platform that follows a traditional model for online courses: students on this platform learn by watching lecture videos, reading, and testing their knowledge with multiple-choice quizzes. In contrast, Korbit[1] takes a different approach to online learning, focusing more on active learning and personalization for the student (Serban, Gupta, Kochmar, Vu, Belfer, Pineau, Courville, Charlin & Bengio (2020)). Korbit is powered by an AI tutor, which creates a personalized curriculum for every student and teaches through short lecture videos, interactive problem-solving exercises, mini-projects, and personalized pedagogical interventions mimicking a human tutor. Specifically, Korbit alternates between lecture videos and interactive problem-solving statements, and students attempt to solve the exercises. Students can also pose their questions on the material, ask for help, or even skip exercises. The AI tutor addresses each incorrect attempt and each request for help with one of a dozen pedagogical interventions tailored to students' needs, thus ensuring that interactions are personalized. Figure 4.1 visualizes the differences between two platforms. This study aims is to measure the efficiency with which students learn on each platform. Since the key difference between the platforms is that Korbit supports personalized, active learning and problem-based learning, we aim to investigate to what extent such mode of tutoring on online platforms contributes to learning outcomes. We aim to test the following hypothesis: *Participants who take the courses on Korbit have higher learning gains than those who take*

[1] www.korbit.ai

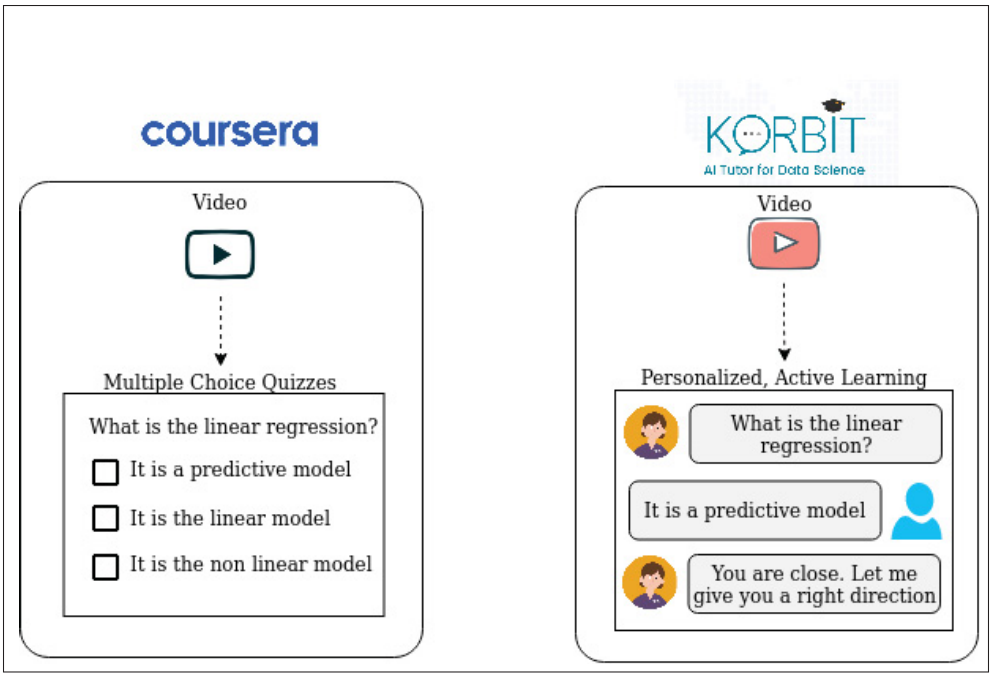

Figure 4.1 Cousera follows a traditional learning approach, while Korbit uses a personalized, active learning approach with problem-solving exercises.

the course on Coursera because Korbit provides a wider and more personalized variety of pedagogical elements to its students.

4.2 Related Works

Online learning platforms providing a massive number of students with access to learning on various subjects have the potential to revolutionize education (Graesser, VanLehn, Rose, Jordan & Harter (2001a), Koedinger & Corbett (2005), Wang, Paquette & Baker (2014)). In particular, such platforms have the capability of bridging the gap and addressing inequalities in the society caused by uneven access to in-person teaching (Hrastinski *et al.* (2021), Tomkins, Ramesh & Getoor (2016)). The current pandemic only exacerbates the need for the high quality online education being accessible to a wide variety of students (Adedoyin & Soykan (2020), Armstrong-Mensah, Ramsey-White, Yankey & Self-Brown (2020), Pokhrel & Chhetri (2021)). Nevertheless, the efficacy of online and distance learning has been and continues to be challenged

by researchers. It was found that the course design and the mode of teaching strongly influence how students progress (Tomkins *et al.* (2016), Vigentini & Clayphan (2015)). Specifically, it may be hard to address the differences in students' learning needs, styles, and aptitudes (Coffield, Learning & Britain) (2004), Stash, Cristea & De Bra (2004), VanLehn, Graesser, Jackson, Jordan, Olney & Rosé (2007)), and this calls for approaches that can be adapted and personalized to the needs of each particular student. Studies confirm that personalization is key to successful online and distance learning (Narciss, Sosnovsky, Schnaubert, Andrès, Eichelmann, Goguadze & Melis (2014), Sampson & Karagiannidis (2002)), such as personalized complexity level and personalized feedback, as it can maximize the learning benefits for each student (Yin, Patikorn, Botelho & Heffernan (2017)). A number of studies have demonstrated that problem-solving is a highly effective approach for learning in various domain (Chow, Yacef, Koprinska & Curran (2017), Kumar (2005b), Wood & Wood (1996), Woolf (2008)). Such problem-solving learning activities can be addressed by intelligent tutoring systems, which are also capable of giving personalized feedback and explanations, incorporating conversational scaffolding, and engaging students into active and problem-solving exercises (Albacete, Jordan, Katz, Chounta & Mclaren (2019a), Büdenbender *et al.* (2002a), Chi (2011),Fossati (2014), Kumar (2005c), Lin, chu Yeh, Hung & Chang (2013a), Melis & Siekmann (2004a), Munshi, Mishra, Zhang, Paquette, Ocumpaugh, Baker & Biswas (2020), Nye *et al.* (2014b)). Many studies have been conducted evaluating the impact of educational technology and online learning platforms on student learning outcomes (Demmans Epp, Phirangee, Hewitt & Perfetti (2020), Kashihara & Hasegawa (2005), Ma, Adesope, Nesbit & Liu (2014), Mark & Greer (1993),Rosé, Moore, VanLehn & Allbritton (2001),RTan. Y.,Quintana (2019),Tomkins *et al.* (2016),Vanlehn, Graesser, Jackson, Jordan, Olney & Rosé (2007). We adopt the well-established pre- post-assessment framework, where students are split into intervention groups and their knowledge of the subject is evaluated before and after their assigned intervention. In contrast to previous studies investigating learning outcomes with intelligent tutoring systems, in this study, the AI-powered learning platform, Korbit, is a fully automated system based on machine learning models (Serban *et al.* (2020)). The system is trained from scratch on educational content to generate automated, personalized feedback for students. It can automatically generalize

to new subjects and improve as it interacts with new students (Kochmar, Do Dung, Belfer, Gupta, Serban & Pineau (2020b), Grenander, Belfer, Kochmar, Serban, St-Hilaire & Cheung (2021)). In the context of online and distance learning, students' ability to self-assess, and to develop self-regulation skills and strategies, plays a crucial role (Barokas, Ketterl & Brooks (2010b),Kashihara & Hasegawa (2005)). However, many studies show that students generally struggle to evaluate their knowledge and skills' level (Brown, Andrade & Chen (2015b))

4.3 Experimental Setup

Participants 48 participants completed a 3-hour long course on *linear-regression* using one of two online platforms. Their learning outcomes were measured before and after the course using pre- and post- assessment quizzes. The experiment was run completely online. Participants completing either course were rewarded a $200 Amazon gift card. We posted ads on social media and sent out emails to student clubs from local universities to recruit participants. Candidates interested in participating had to fill out a questionnaire specifying their field of study, their degree, and whether they have completed courses in *machine learning* or *artificial intelligence*. Candidates were classified as eligible or ineligible based on their answers to this enrollment questionnaire. Specifically, candidates who had or were studying a math-heavy discipline at university (e.g., mathematics, statistics, physics) or who had completed any courses on statistics, machine leanings, or artificial intelligence were deemed ineligible. As a result, out of the 60 applicants, 48 participants were selected. The majority fall into our target audience of undergraduates (89.6%) studying disciplines not centered around mathematics: health sciences (27.7%), computer science (23.4%), cognitive science (12.8%), among others. To ensure an unbiased setting for the experiments on two platforms, we randomly divided participants into two groups. The first group was asked to study the course on *linear regression* in **Coursera** and the second was asked to study the course on the same subject from **Korbit**. **Choice of the Material** Linear regression was selected as the topic of study on both online platforms since it is one of the most fundamental topics that is covered early on in any course on machine learning and data science, and the material covering this topic on both platforms is comparable. Extra care was

taken to yield a fair comparison between the two platforms to ensure that the linear regression courses were as similar as possible. The sub-topics covered, the difficulty level, and both courses' length were carefully aligned. The linear regression course on Korbit was adapted for this study, combining existing and new content specifically created to align with the sub-topics covered in the Coursera course. As a result, the courses on both platforms contain an introductory session and provide short lecture videos, followed by multiple-choice questions in the case of Coursera and interactive problem-solving exercises in the case of Korbit. The sub-topics taught include numerical variables, correlation, residuals, least squares regression, and evaluation metrics. The course on each platform takes approximately 3 hours to complete. **Study Flow** The study ran over four days with strict deadlines set for the participants. The participants received instructions detailing all the steps they would need to complete on first day, and from this point, they had three days to complete the course. If they completed all necessary steps before the third day deadline, they were asked to complete the final post-assessment quizzes by the end of the fourth day. All participants were required to take an assessment quiz on linear regression before the course (*pre-quiz*) and another one after the course (*post-quiz*). Using pre- and post- quiz scores, we measure *learning gains* to quantify how efficiently each participant has learned. The pre- and the post- quizzes both consisted of 20 multiple-choice questions. They were equally adapted to both courses, meaning that any topic or concept mentioned in the quizzes was covered in both courses to an equal extent, ensuring that students mastering the topics using either course would succeed in answering them. In addition, the quizzes went through an independent review process, which ensured that the quiz scores were not inherently biased towards one of the learning platforms. Furthermore, each question of the pre-quiz was isomorphically paired with a question in the post-quiz, meaning that the difficulty of the two quizzes was as similar as possible without any question being identical. This ensured that learning gains were accurately measured without bias from the differing quiz difficulty. **Safeguarding Against Invalid Results** Since the study ran fully online, it was important to take precautionary measures to minimize the participants' chances of cheating or otherwise not following the instructions. We identified the following two potential scenarios:

- A participant might have completed the pre- and the post- quizzes without really studying the course, or going through the course without paying attention, skipping the videos and exercises
- A participant might have used external resources to find the correct answers on the quizzes.

In both cases, the participant's scores would be meaningless because they would be completely or almost completely unrelated to the course and the learning platform. To minimize the chances of this happening, we provided participants with very clear instructions on what they were required and what they were not allowed to do. We required them to upload a completion certificate confirming that they went through the whole course on the corresponding platform. We also expect that scheduling the study over four days, regardless of how quickly the participants could go through the course, helped discourage them from breezing through the quizzes and the course to get their gift card immediately and ensured there was a delay between their learning and assessment. For Korbit involving problem-solving exercises, we further defined a minimum requirement for participants to attempt at least 80% of the exercises. **Learning Gains** To evaluate which of the two online learning platforms teaches the participants more effectively, we compare Coursera and Korbit based on the *average learning gain* and *normalized learning gain*[14] of the participants on each platform. A student's learning gain g is estimated as the difference between their score on the post-quiz and on the pre-quiz as follows:

$$g = post\_score - pre\_score \tag{4.1}$$

where $post\_score$ and pre_score are the score of the post-quiz and pre-quiz, respectively. Both scores fall in the interval $[0\%, 100\%]$. A student's invidual normalized learning gain g_{norm} is calculated by offsetting a particular student's learning gain against the score range in the ideal scenario in which a student achieves a score of 100% in the post-quiz.

$$g_{norm} = \frac{post\_score - pre\_score}{1 - pre\_score} \tag{4.2}$$

4.4 Results and Discussion

25 participants completed the course on Coursera and 23 participants completed the course on Korbit. One participant on Korbit did no satisfy the requirement of attempting at least 80% of the exercises and was therefore excluded from the analysis. It is worth noting that to align the courses on the two platforms closely, the personalized curriculum and the programming exercises offered by Korbit were specifically disabled for this study. On the one hand, this allowed us to compare the two platforms on a fair basis and specifically explore the effects of personalized, active learning and problem-based exercises on the learning outcomes following the formulated hypothesis. On the other hand, given that the personalized curriculum and programming exercises are highlighted on the Korbit website, this may have created a mismatch between students' expectations and their actual learning experience with this platform. Future experiments using the full functionality of Korbit will aim to investigate the effect of other pedagogical elements. **Learning Outcomes** Average learning gains are shown in Figure 4.2

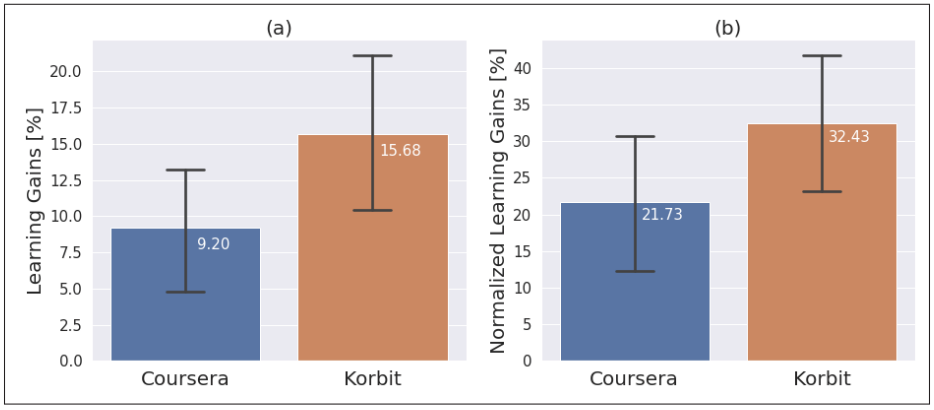

Figure 4.2 (a) Average learning gain g with 95% confidence intervals. (b) Average normalized learning gains g_{norm} with 95% confidence intervals. Here and indicate a statistically significant difference at 95% and 90% confidence level respectively

for the two platform [2]. The average normalized learning gains g_{norm} for Korbit participants are 49.24% higher than the average normalized gains for Coursera participants. The difference is

[2] 95% confidence interval (C.I) are estimated as: $1.96 * \frac{Standard\ Deviation}{\sqrt{Population\ Size}}$

statistically significant at the 90% confidence level ($p = 0.068$). When considering raw learning gains g, the Korbit average is 70.43% higher than the Coursera average with 95%confidence ($p = 0.038$). It should be observed that there are three participants from Coursera and two from Korbit, who showed negative learning gains. These results are hard to interpret because the difficulty level of the pre- and the post-assessment quizzes is the same. A possible explanation for this is that they had little prior knowledge and did not learn the material during the course but managed to pick the correct answers in the pre-quiz by chance and were less lucky in the post-quiz. We also observed that the median normalized learning gain g_{norm} is only slightly higher for Korbit, being at 31.82% as opposed to 28.57% for Coursera. Because the Korbit average is higher substantially than Coursera, this could mean that participants on Coursera and Korbit are well below the Coursera median and well above the Korbit median, respectively. This might stem from the fact that Korbit is better adapted to students with different backgrounds and learning needs by having various pedagogical tools and more active learning. However, we believe that more data and further studies would be needed to confirm this assumption. Further, the medians for raw learning gain g are closer to the respective average values for both platforms: 10% for Coursera and 15% for Korbit. This is in line with our hypothesis but suggests that while the normalization process produces a more meaningful metric for learning gains, it does introduce a lot of variance in the metric due to dividing by the pre-quiz score. Overall, our hypothesis that learning outcomes are higher for participants on Korbit than participants on Coursera is confirmed by the results presented here, with the average learning gains g and normalized learning gains g_{norm} being substantially higher for Korbit at 95% and 90% statistical significance-level respectively. To explain these results, we theorize that the active learning elements of Korbit play a significant role in the participants' learning experience. The key difference between the two platforms is that Korbit supports active learning and problem-based learning, while Coursera is limited in this regard. To give an idea of the extent to which Korbit provides more active learning than Coursera, we estimate how much time participants spent on different learning activities while completing the course on either platform. Table4.1 presents the estimates. For Coursera, the time estimates for videos and readings come directly from the course website. Coursera gives no time estimate for the 13 multiple choice questions in the

Table 4.1 Estimated time spent on different learning activities on each platform (minutes and % of total), excluding time spent on enrollment and assessment

Learning Activity	Coursera	Korbit
Watching lecture videos	46.0 min (33.82%)	29.6 min (24.89%)
Reading material	50.0 min (36.76%)	0.0 min (0.00%)
Solving quizzes and exercises	40.0 min (29.42%)	89.3 min (75.11%)

course for the exercises. We estimate it at 40 minutes by testing it ourselves. For Korbit, we have access to the data from the participants' interactions on the platform, which lets us directly calculate estimates for the average time spent watching videos and solving exercises. Based on these estimates, we conclude that participants on Korbit spent more time on active learning than Coursera participants by a factor of 2.23. We believe that this substantial difference in the active learning time is one of the main reasons for the higher learning gains observed w.r.t Korbit participants. We further investigate the participants' behavior on Korbit using the data collected automatically on this platform and report the most insightful correlations observed between participants' behavioral factors and their learning gains.

Firstly, we establish a correlation between the total amount of time participants spent on solving exercises and their overall performance. Specifically, there is a positive correlation between the time spent on exercises and the rate participants provided correct answers on the first try ($r = 0.34$). At the same time, the time spent on exercises and the average number of attempts participants needed to get a correct answer is negatively correlated ($r = -0.34$). This suggests that participants who took more time working on the exercises and formulating their answers performed better. These participants got the correct answer on the first try more often and, on average, took fewer attempts to answer correctly.

Secondly, we observe a correlation between the participants' performance on exercises and their learning gains. Specifically, the rate of correct answers on the first try positively correlates with both learning gains ($r = 0.44$) and post-quiz results ($r = 0.46$), and the number of exercises completed positively correlates with the post-quiz score ($r = 0.28$). These correlations may

suggest that participants who spent more of their study time on active learning and problem-solving exercises performed better and, as a result, obtained higher post-quiz scores and learning gains. We believe this might indicate the effectiveness of active learning elements on Korbit. One limitation to this hypothesis is that no strong correlation was found between the time spent on exercises and the learning gains or post-quiz scores. We believe this is partly due to the diversity in the participants' background knowledge, leading to high variance in the pace at which participants completed exercises. More data is needed to understand the factors driving the learning gains thoroughly.

Another interesting observation is that participants who scored higher on the pre-quiz were less receptive to the pedagogical intervention from the AI tutor on Korbit. Specifically, for participants who scored over 50% on the pre-quiz, there is a stronger negative correlation between their learning gains and the total number of hints they received during the course ($r - 0.58$). This correlation is weaker for participants who score below 50% on the pre-quiz ($r = -0.32$). This suggests that the pedagogical interventions were more helpful for the students with lower pre-quiz scores. We believe that the negative correlation between the number of hints received and learning gains is partly explained because pedagogical interventions on Korbit mostly occur when a student is already struggling with an exercise. Therefore, participants who received more hints most likely struggled more with the exercises, which, as discussed above, correlates with lower learning gains. **Metacognitive Evaluation** In addition to measure actual learning gains using pre- and post- quizzes, we evaluated various metaconition aspects related to the students' learning experience with the two platforms using a questionnaire where students were asked to report on their experience. Table 4.2 list 3 questions we asked the participants on their *perceived learning gains* There is a strong correlation between how participants rated their comprehension of the topics studied and their actual learning gains (Q1). This can be seen in Figure 4.3 which shows the normalized learning gains depending on this subjective rating. The correlation is higher for Korbit participants ($r = 041$) than for Coursera participants ($r = 0.26$). This suggests that Korbit gave participants a more accurate understanding of their knowledge level and helped improve their metacognition. For Q2 and Q3 on perceived learning, no correlation was found

Table 4.2 Average perceived learning gains (±95% confidence interval)

ID	Question	Coursersa	Korbit
Q1	How would you rate your comprehension of the topics you studied? (1-5)	4.16 ± 0.27	3.65 ± 0.29
Q2	How well do you think you performed on the final quiz (1-100%)	76.7% ± 5.41%	75.2% ± 5.63%
Q3	How capable would you feel in applying the skills you learned in a practical setting (at your job, in a personal project, in you research, etc)? (1-5)	3.68 ± 0.29	3.35 ± 0.38

with the actual learning gains. Even though perceived learning gains are strongly correlated with the actual learning gains for Korbit, on average, participants from Korbit reported lower levels of comprehension in absolute term Q1, and the difference from the Coursera average is significant ($p < 0.5$). This directly contradicts the fact that Korbit participants obtained higher learning gains on average. In addition, the results for Q2 and Q3 suggest that the Korbit participants' perception of their performance on the post-quiz and their perceived capability to apply the skills they learned are slightly lower than for Coursera participants.In line with previous work, we note that this indicates that students generally struggle to evaluate their knowledge and skills level (Crowell (2015a),Brown *et al.* (2015b), ref (2013)). Furthermore, we hypothesize that this contradictory result is the presence of some frustrating elements in Korbit: for instance, one common source of confusion reported for Korbit is the fact that the AI tutor did not always understand participants' answers. On the one hand, the AI tutor on Korbit engaging in a dialogue with participants and providing them with the interactive problem-solving exercises strongly contributes to higher learning gains on Korbit. On the other hand, this aspect makes Korbit more technically challenging to implement than the more traditional approach taken by Coursera. In line with previous research (Lehman, D'Mello & Graesser (2012a),Yin *et al.* (2017)), we hypothesize that improvements in the AI tutors' understanding of students' answers and feedback might contribute to higher perceived learning gains in students. Future research should investigate this hypothesis.

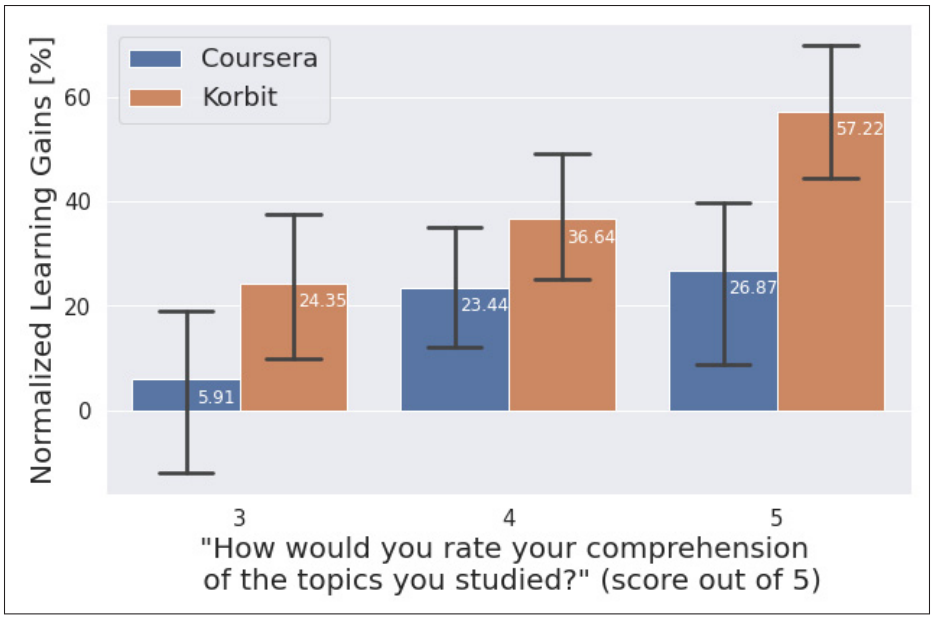

Figure 4.3 Normalized learning gains for each self-assessed comprehension rating with 95% confidence intervals. Only 1 participant gave a score lower than 3 (not shown here)

4.5 Conclusions

This study compared two popular online learning platforms concerning the learning outcomes and metacognition included by the platforms. The first platform, Coursera, is a widely-used learning platform that follows a traditional model for online courses: students learn by watching lecture videos, reading, and testing their knowledge with multiple-choice quizzes. The second platform, Korbit, focuses on active learning and personalization, where each student learns through short lecture videos and interactive problem-solving exercises accompanied by personalized feedback. We assessed the learning gains of 47 participants after a 3-hour long course on *linear regression* topic, with 25 participants taking the course on Coursera and 22 on Korbit. We observed that the average learning gain for Korbit is 70.43% higher than on Coursera ($p < 0.05$). This supports the hypothesis that participants on Korbit have higher learning gains than those who take the course on Coursera because Korbit provides a wider and more personalized variety of pedagogical elements to its students. Furthermore, the average normalized learning gain for Coursera and Korbit participants are 21.73% and 32.43%, respectively. With Korbit producing

49.24% higher ($p < 0.1$) learning gains relative to Coursera, the result support the hypothesis that Korbit teaches more effectively. However, more data should be collected to validate the same hypothesis w.r.t normalized learning gains. In addition, perceived learning gains and metacognition abilities were assessed with a feedback questionnaire filled out by participants after the post-assessment quiz. It was found that Coursera participants report higher perceived learning gains even though their actual learning gains are lower on average. Higher correlations between perceived learning and actual learning gains for Korbit participants suggest that Korbit induces better metacognition. Lower perceived learning gains on Korbit can be explained because this platform involves more complex teaching elements. These elements may contribute to higher actual learning gains; they may at the same time also be a source o frustration and confusion for some participants. Future research should look more closely into the factors that affect the perceived learning for participants and should investigate to what extent the participants benefit from the course being adapted to their learning needs and preferences.

CHAPTER 5

A LARGE-SCALE, OPEN-DOMAIN, MIXED-INTERFACE DIALOGUE-BASED ITS FOR STEM

An adapt version of this chapter has been published in Artificial Intelligence in Education, 2020, Springer International Publishing, p. 387–392.

5.1 Introduction

Intelligent tutoring systems (ITS) are computer programs powered by artificial intelligence (AI), which deliver real-time, personalized tutoring to students. Traditional ITS implement or imitate the behavior and pedagogy of human tutors. In particular, one type of ITS are dialogue-based tutors, which use natural language conversations to tutor students Nye *et al.* (2014b). This process is sometimes called "Socratic tutoring", because of its similarity to Socratic dialogue Rosé *et al.* (2001). Newer ITS have started to interleave their dialogue with interactive media (e.g. interactive videos and web applets) – a so-called "mixed-interface system". It has been shown that ITS can be twice as effective at promoting learning compared to the previous generation of computer-based instruction and that ITS may be as effective as human tutors in general Kulik & Fletcher (2016b).

However, even though ITS have been around for decades and are known to be highly effective, their deployment in education and industry has been extremely limited Olney (2018); Ritter, Anderson, Koedinger & Corbett (2007). A major reason for this is the sheer cost of development Folsom-Kovarik *et al.* (2010b); Olney (2018). As observed by Olney Olney (2018): *"Unfortunately, ITS are extremely expensive to produce, with some groups estimating that it takes 100 hours of authoring time from AI experts, pedagogical experts, and domain experts to produce 1 hour of instruction."* On the other hand, lower-cost educational approaches, such as massive open online courses (MOOCs), have flourished and now boast millions of learners. It is estimated that today there are over 110 million learners around the world enrolled in MOOCs Shah (2019). However, the learning outcomes resulting from learning in MOOCs de-

pend critically on their teaching methodology and quality of content and remains questionable in general Cavanaugh & Jacquemin (2015); Colvin, Champaign, Liu, Zhou, Fredericks & Pritchard (2014); Kirtman (2009); Koedinger, Kim, Jia, McLaughlin & Bier (2015); Koxvold (2014); Otto, Bollmann, Becker & Sander (2018). In particular, recent research indicates that MOOCs with low active learning levels, little feedback from instructors and peers, and few peer discussions tend to yield poor learning outcomes Koedinger *et al.* (2015); Otto *et al.* (2018). Furthermore, it is well-known that student retention in MOOCs is substantially worse than in traditional classroom learning Hone & El Said (2016). By combining the low cost and scalability of MOOCs with the personalization and effectiveness of ITS, we hope `Korbit` may one day help to teach and motivate millions of students around the world effectively.

5.2 The Korbit ITS

`Korbit` is a large-scale, open-domain, mixed-interface, dialogue-based ITS, which uses machine learning, natural language processing (NLP) and reinforcement learning (RL) to provide interactive, personalized learning online. The ITS has over 7,000 students enrolled from around the world, including students from educational institutions and professionals from industry partners. `Korbit` is capable of teaching topics related to data science, machine learning, and artificial intelligence. The platform is highly modular and will soon be expanded with many more topics.

Students enroll on the `Korbit` website by selecting either a course or a set of skills they would like to study. Students may also answer a few questions about their background knowledge. Based on these, `Korbit` generates a personalized curriculum for each student. Following this, `Korbit` tutors the student by alternating between short lecture videos and interactive problem-solving exercises. The `outer-loop` system decides on which lecture video or exercise to show next based on the personalized curriculum. The ordering of videos and exercises is currently determined by the initial curriculum, but work is underway to adapt the curriculum during the learning process.

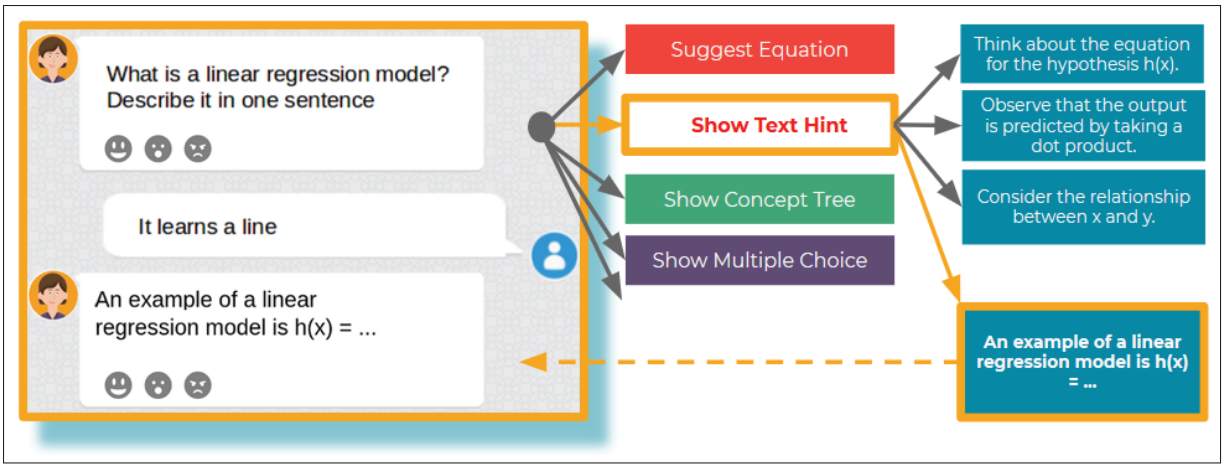

Figure 5.1 An example of how the `Korbit` ITS inner-loop system selects the pedagogical intervention. The student gives an incorrect solution and afterwards receives a text hint.

During the exercise sessions, the `inner-loop` system manages the interaction. First, it shows the student a problem statement (e.g., a question). The student may then attempt to solve the exercise, ask for help, or skip the exercise. If the student attempts to solve the exercise, their solution attempt is compared against the expectation (i.e. reference solution) using an NLP model. If their solution is classified as incorrect, then the `inner-loop` system will select one of a dozen different pedagogical interventions. The pedagogical interventions include textual hints, mathematical hints, elaborations, explanations, concept tree diagrams, and multiple choice quiz answers. The pedagogical intervention is chosen by an ensemble of machine learning models based on the student's profile and last solution attempt. Depending on the pedagogical intervention, the `inner-loop` system may either ask the student to retry the initial exercise or follow up on the intervention (e.g., with additional questions, confirmations, or prompts).

The `Korbit` ITS is closely related to the line of work on dialogue-based ITS, such as the pioneering AutoTutor and the newer IBM Watson Tutor Ahn *et al.* (2018); Graesser *et al.* (2005, 2001b); Nye *et al.* (2014b); Ventura *et al.* (2018). Although `Korbit` is highly constrained compared to existing dialogue-based ITS, a major innovation of `Korbit` lies in its modular, scalable design. The `inner-loop` system is implemented as a finite-state machine. Each

pedagogical intervention is a separate state, with its own logic, data and machine learning models. Each state operates independently of the rest of the system, has access to all database content (including all exercises and lecture videos) and can autonomously improve as new data becomes available. This ensures that the system gets better and better, that it can adapt to new content and that it can be extended with new pedagogical interventions. Furthermore, the transitions between the states of the finite-state machine is decided by a reinforcement learning model, which itself is agnostic to the underlying implementation of each state and also continues to improve as more and more data becomes available.

5.3 System Evaluation

We have conducted multiple studies to evaluate the `Korbit` ITS. Some of these studies have evaluated the entire system, while others have focused on particular aspects or modules. Taken together, the studies demonstrate that the `Korbit` ITS is an effective learning tool and that it overall improves student learning outcomes and motivation compared to alternative online learning approaches.

In this paper, to keep things short, we limit ourselves and discuss only one of these studies. The study we present compares the entire system (`Full ITS`) against an xMOOC-like system Daniel (2012). The purpose of this particular study is to evaluate 1) whether students prefer the `Korbit` ITS or a regular MOOC, 2) whether the `Korbit` ITS increases student motivation, and 3) which aspects of the `Korbit` ITS students find most useful and least useful. In an ideal world, `Korbit` ITS would be compared against a regular xMOOC teaching students through lecture videos and multiple-choice quizzes in a randomized controlled trial (a randomized A/B testing experiment). However, it is not possible to compare against such a system in a randomized controlled trial because it would create confusion and drastically offset our students' expectations.[1] Therefore, in this study, we compare the `Full ITS` against a reduced ITS, which appears identical to the `Full ITS` and utilizes the same content (video lectures and exercise questions), but defaults to

[1] Indeed, this was attempted in an earlier study. During that study, however, when students found out that they were assigned to the xMOOC system instead of the ITS system, they would complain, logout, and create a new account to access the main ITS system.

multiple choice quizzes 50% of the time. Thus, students assigned to the reduced ITS effectively spend about half of their interactions in an xMOOC-like setting. We refer to this system as the `xMOOC ITS`.

Table 5.1 A/B testing results comparing the `Full ITS` against the `xMOOC ITS`: average time spent by students (in minutes), returning students (in %), students who said they will refer others (in %) and learning gain (in %), with corresponding 95% confidence intervals. The * and ** shows statistical significance at 90% and 95% confidence level respectively.

System	Time Spent	Returning Students	Refer Others	Learning Gain
`xMOOC ITS`	22.98±4.18	26.98%±3.44%	44.83% ± 9.00%	**39.14%±2.35%**
`Full ITS`	**39.86±3.70****	**31.69%±1.92%***	**54.17%±4.05%**	

The experiment was conducted in 2019 with n=612 participants. Students who enrolled online were randomly assigned to either the `Full ITS` (80%) or `xMOOC ITS` (20%). Students came from different countries and were not subject to any selection or filtering process. Apart from bug fixes and minor speed improvements, the system was kept fixed during this time period to limit confounding factors. After using the system for about 45 minutes, students were shown a questionnaire to evaluate the system.

Table 5.1 shows the experimental results. The average time spent in the `Full ITS` was 39.86 min compared to 22.98 min in the `xMOOC ITS`. As such, the `Full ITS` yields a staggering 73.46% increase in time spent. In addition, the percentage of returning students and the percentage of students who said they would refer others to use the system is substantially higher for the `Full ITS` compared to the `xMOOC ITS`. These results were also confirmed by the feedback provided by the students in the questionnaire. Thus, we can conclude that students strongly prefer `Korbit` ITS over xMOOCs and that the `Korbit` ITS increases overall student motivation. Table 5.1 also shows the average student learning gain, which was observed to be 39.14%. The learning gain is measured as the proportion of instances where a student provides a correct exercise solution after having received a pedagogical intervention from the `Korbit` ITS. Thus, the pedagogical interventions appear to be effective.

Finally, in the questionnaire, 85.31% of students reported that they found the chat equally or more fun than learning alone, and 66.67% of students reported that the chat helped them learn better sometimes, many times, or all of the time. For the `Full ITS`, 54.17% of students reported that they would refer others to use `Korbit` ITS. In addition, students reported that the `Korbit` ITS could be improved by more accurately identifying their solutions as being correct or incorrect and, in the case of incorrect solutions, by providing more relevant and personalized feedback.

CHAPTER 6

GENERAL DISCUSSION AND CONCLUSION

This thesis proposes methods for automated generation and personalization of feedback in an intelligent tutoring system. In addition, we run some pilot programs with real users, including high education students and employees, to compare the performance of the systems and the learning outcome between our system and the MOOCs system. In particular, we have focused on the generation and personalization of text-based hints, extraction, and generation of Wikipedia-based explanations leveraging large amounts of potentially useful data available for learner needs on Wikipedia. Each type of feedback is generated in a fully automated manner, using data-driven approaches and state-of-the-art machine learning and natural language processing techniques. In addition, we conducted several experiments to investigate the effectiveness of personalization feedback and measure student success rates and students' preference for each type of feedback.

By running these experiments with more than 200 students and almost 1000 software developers (junior, experienced, and senior developers) within one year, the results strongly support our hypothesis that the personalized hints and Wikipedia-based explanations help to improve student learning. In this work, the personalized feedback is automatically generated in a data-driven way leads to improved learning outcomes measured as the success rate in the students' ability to answer the questions on the material correctly after being provided with an informative hint. This is a crucial step towards solving one of the major problems for large-scale ITS, which have often relied on expert design and hand-crafted rules in the past. Our future research will investigate how to scale and transfer the personalized data-driven feedback models across multiple domains. We will do step by step for the similar domains first, then big differences later. The other limitation is that we measure learning outcomes as the success rate in answering the question immediately after personalized feedback is provided. We compute the normalized learning gain by using the result of post-quizzes and pre-quizzes. Since we believe that generated hints and explanations are helpful and guide students in the right direction.

Moreover, we delayed post-tests on the relevant concepts to evaluate the students' ability to retain knowledge. Future experiments on our platform will address the student self-correcting features by observing the students' improvement via their behavior. An additional challenge for ITS is that they teach technical subjects by using multiple languages or doing the short projects involved in generating the pedagogical interventions and the provision of feedback.

An ITS in these domains must evaluate answers expressed in a purely textual form and provided by the students in response to the questions that are, likewise, expressed in a natural language (e.g., *"What is a linear regression model?"*). On the one hand, such expressions may be included in student answers: e.g.a mathematical expression would be expected as a response to the question *"Define the sum-of-squares error function"* from an ITS. Mathematical expressions may be included in the mixed-modality questions, which may further combine them with textual content, as does a question like "Suppose the output is categorical with 10 categories ($y = 1, 2, \cdots, 10$) If $y_i = 9$, then what would its corresponding one-hot vector representation be?". This proved to be particularly challenging in the past, with many systems aiming to provide feedback on mathematical expressions resorting to hand-crafted rules or involving a human tutor. In addition, students use mixed language under ambiguity with both natural language and mathematical expressions. Sometimes, they forget the math operators or use the different names of the variable to explain the same meaning of the equations. Our ongoing research is concerned with the models capable of analyzing math equations in addition to purely text-based content and providing relevant feedback. In the future, we also plan to work on the set of hints with those on programming exercises and investigate students' learning gain with the specific concepts. We have shown that students and generated hints and Wikipedia-based explanations are helpful. Future work should also investigate how and what type of hints may improve student learning outcomes. In particular, we plan to investigate how varying hint complexity and the level of hint transparency can be used in instructional scaffolding in parallel with optimizing the number of hints for each student to engage him to study. Moreover, the peer-to-peer hint might be considered in our future research; the ranking system will be applied to pick up the good

student answers, suggestions, and explanations to use as the hints for another student based on the student interaction and profile.

Future work should also investigate the interplay between the granularity of such formative feedback and various student learning profiles. Finally, it should be noted that there has been a massive increase in the use of ITS, and more broadly, online learning platforms, separate from and alongside traditional human teacher and student interactions (e.g., in flipped classrooms and blended learning environments). Therefore, future research must look closely into such aspects of the learning process as student motivation, engagement, and managing students' emotional states. In particular, doing the student emotion states analysis strategy will give better feedback of updating the quality of the hints. Of particular interest are such questions as to whether tutoring via an ITS should mimic human tutoring or rather provide students with an alternative means of learning and which aspects of the learning process are best addressed with an ITS tutor versus a human one.

We compared two popular online learning platforms concerning the learning outcomes and metacognition included by the platforms. The first platform, Coursera, is a widely used learning platform that follows a traditional model for online courses: students learn by watching lecture videos, reading, and testing their knowledge with multiple-choice quizzes. The second platform, Korbit, focuses on active learning and personalization, where each student learns through short lecture videos and interactive problem-solving exercises accompanied by personalized feedback. We assessed the learning gains of almost 200 participants after a 3-hour long course *linear regression* topic, with 30% participants taking the course on Coursera, 35% on Korbit with feedbacks, 35% on Korbit without feedback. We observed that the normalized learning gain for Korbit with feedback has significantly higher than Coursera and Korbit without feedback ($p < 0.05$). This supports the hypothesis that Korbit with feedback (hints) teaches more effectively.

In addition, we did the study for both students in university and the fresher developers who do not have experience with *linear regression* or *machine learing*, and we got the same result for

both types of candidates, which means Korbit with hints induces the better meta-cognition. Our experiment with 68 fresh software developers included both Korbit with and without feedback as a baseline shows that the normalized learning gain of Korbit with feedback is better almost 3 times than Coursera with 95% confidence level ($p = 0.04176$). However, Korbit without feedback and personalized learning gains are almost Coursera, demonstrating that feedback and personalization are core components and major drivers of student learning outcomes. Future research should look more closely into the participant's benefit from the course being adapted to their learning needs and prefer

BIBLIOGRAPHY

(2013). SAGE Handbook of Research on Classroom Assessment AU - Brown, Gavin T. L. AU - Harris, Lois R. Thousand Oaks: SAGE Publications, Inc. Student Self-Assessment1, doi: 10.4135/9781452218649.

Abel, A., Chang, B.-Y. E. & Pfenning, F. (2001). Human-Readable, Machine-Verifiable Proofs for Teaching Constructive Logic. *Proceedings of the Workshop on Proof Transformations, Proof Presentations and Complexity of Proofs (PTP'01)*.

AbuEl-Reesh, J. Y. & Abu-Naser, S. S. (2018). An Intelligent Tutoring System for Learning Classical Cryptography Algorithms (CCAITS). *International Journal of Academic and Applied Research (IJAAR)*, 2(2), 1–11.

Adedoyin, O. B. & Soykan, E. (2020). Covid-19 pandemic and online learning: the challenges and opportunities. *Interactive Learning Environments*, 0(0), 1-13. doi: 10.1080/10494820.2020.1813180.

Agha, M., Jarghon, A. & Abu-Naser, S. (2018). An Intelligent Tutoring Systems For Teating SQL. *International Journal of Academic Information Systems Research (IJAISR)*, 1–7.

Ahn, J.-W., Chang, M., Watson, P., Tejwani, R., Sundararajan, S., Abuelsaad, T. & Prabhu, S. (2018). Adaptive Visual Dialog for Intelligent Tutoring Systems. *International Conference on Artificial Intelligence in Education*, pp. 413–418.

Al-Dahdooh, R. & Abu-Naser, S. (2017). Development and Evaluation of the Oracle Intelligent Tutoring System (OITS). *European Academic Research*, 4, 8711-8721.

Al-Nakhal, M. & Abu-Naser, S. (2017). Adaptive Intelligent Tutoring System for Learning Computer Theory. *European Academic Research*, 4, 8770-8782.

Al Rekhawi, H. & Abu-Naser, S. (2018). Android Applications UI Development Intelligent Tutoring System. *International Journal of Engineering and Information Systems (IJEAIS)*, 1–14.

Al-Rfou, R., Choe, D., Constant, N., Guo, M. & Jones, L. (2018). Character-Level Language Modeling with Deeper Self-Attention.

Albacete, P., Jordan, P., Katz, S., Chounta, I.-A. & Mclaren, B. (2019a, 06). The Impact of Student Model Updates on Contingent Scaffolding in a Natural-Language Tutoring System. pp. 37-47. doi: 10.1007/978-3-030-23204-7_4.

Albacete, P., Jordan, P., Katz, S., Chounta, I.-A. & McLaren, B. M. (2019b). The Impact of Student Model Updates on Contingent Scaffolding in a Natural-Language Tutoring System. *International Conference on Artificial Intelligence in Education*, pp. 37–47.

Aleven, V., Popescu, O. & Koedinger, K. (2001). Towards Tutorial Dialog to Support Self-Explanation: Adding Natural Language Understanding to a Cognitive Tutor. 246–255.

Anania, J. (1983a). The influence of instructional conditions on student learning and achievement. *Evaluation in Education*, 7(1), 1-92. doi: https://doi.org/10.1016/0191-765X(83)90002-2.

Anania, J. (1983b). The Influence of Instructional Conditions on Student Learning and Achievement. *Evaluation in Education: An International Review Series*, 7(1), 3–76.

Anderson, J. R., Boyle, C. F. & Reiser, B. J. (1985a). Intelligent Tutoring Systems. *Science*, 228(4698), 456–462.

Anderson, J. R., Boyle, C. F. & Reiser, B. J. (1985b). Intelligent tutoring systems. *Science*, 228(4698), 456–462.

Andrews, P. B., Brown, C. E., Pfenning, F., Bishop, M., Issar, S. & Xi, H. (2004). ETPS: A System to Help Students Write Formal Proofs. *J. Autom. Reason.*, 32(1), 75–92. doi: 10.1023/B:JARS.0000021871.18776.94.

Armstrong-Mensah, E., Ramsey-White, K., Yankey, B. & Self-Brown, S. (2020). COVID-19 and Distance Learning: Effects on Georgia State University School of Public Health Students. *Frontiers in Public Health*, 8, 547.

Arora, S., Liang, Y. & Ma, T. (2017). A Simple but Tough-to-Beat Baseline for Sentence Embeddings. *ICLR*.

Arroyo, I., Beal, C. R., Bergman, A. M., Lindenmuth, M., Marshall, D. & Woolf, B. P. (2003). Intelligent Tutoring for high-stakes achievement tests.

Bahdanau, D., Cho, K. & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation, Consulted at http://arxiv.org/abs/1409.0473.

Baker, R. S. (2016). Stupid tutoring systems, intelligent humans. *International Journal of Artificial Intelligence in Education*, 26(2), 600–614.

Barokas, J., Ketterl, M. & Brooks, C. A. (2010a). Lecture Capture: Student Perceptions, Expectations, and Behaviors.

Barokas, J., Ketterl, M. & Brooks, C. (2010b, 01). Lecture Capture: Student Perceptions, Expectations, and Behaviors. pp. 424-431.

Baxter, J. A. & Haycock, J. (2014). Roles and student identities in online large course forums: Implications for practice. *The International Review of Research in Open and Distributed Learning*, 15(1), 20–40. doi: 10.19173/irrodl.v15i1.1593.

Beel, J., Gipp, B., Langer, S. & Breitinger, C. (2016). Research-paper recommender systems : a literature survey. *International Journal on Digital Libraries*, 17(4), 305–338. doi: 10.1007/s00799-015-0156-0.

Bengio, Y., Ducharme, R., Vincent, P. & Janvin, C. (2003). A Neural Probabilistic Language Model. *J. Mach. Learn. Res.*, 3(null), 1137–1155.

Benzmüller, C., Horacek, H., Kruijff-Korbayova, I., Pinkal, M., Siekmann, J. & Wolska, M. (2005, 01). Natural Language Dialog with a Tutor System for Mathematical Proofs. pp. 1-14. doi: 10.1007/978-3-540-70934-3_1.

Bishop, C. M. (2006a). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag.

Bishop, C. M. (2006b). *Pattern Recognition and Machine Learning*. Springer. Consulted at http://research.microsoft.com/en-us/um/people/cmbishop/prml/.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, 13(6), 4–16.

BLOOM, B. S. (1984). The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher*, 13(6), 4-16.

Boaler, J. & Brodie, K. (2004). The importance, nature, and impact of teacher questions. *Proceedings of the twenty-sixth annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*, 2, 774–782.

Bollacker, K., Evans, C., Paritosh, P. K., Sturge, T. & Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. *SIGMOD Conference*.

Boulanger-Lewandowski, N., Bengio, Y. & Vincent, P. (2012). Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. *Proceedings of the 29th International Coference on International Conference on Machine Learning*, (ICML'12), 1881–1888.

Bowen, W. G., Chingos, M. M., Lack, K. A. & Nygren, T. I. (2014). Interactive Learning Online at Public Universities: Evidence from a Six-Campus Randomized Trial. *Journal of Policy Analysis and Management*, 33(1), 94-111. doi: https://doi.org/10.1002/pam.21728.

Bowman, S. R., Angeli, G., Potts, C. & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642. doi: 10.18653/v1/D15-1075.

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5-32. doi: 10.1023/A:1010933404324.

Brown, G. & Harris, L. (2013, 01). Student self-assessment. pp. 367-393.

Brown, G., Andrade, H. & Chen, F. (2015a). Accuracy in student self-assessment: Directions and cautions for research. *Assessment in Education Principles Policy and Practice*, 22, 444-457. doi: 10.1080/0969594X.2014.996523.

Brown, G. T., Andrade, H. L. & Chen, F. (2015b). Accuracy in student self-assessment: directions and cautions for research. *Assessment in Education: Principles, Policy & Practice*, 22(4), 444-457. doi: 10.1080/0969594X.2014.996523.

Brown, J. & Burton, R. (1975). MULTIPLE REPRESENTATIONS OF KNOWLEDGE FOR TUTORIAL REASONING.

Brunskill, E., Mu, T., Goel, K. & Bragg, J. (2018). Automatic Curriculum Generation Applied to Teaching Novices a Short Bach Piano Segment. *NeurIPS Demonstrations*.

Buck, C., Heafield, K. & van Ooyen, B. (2014). N-gram Counts and Language Models from the Common Crawl. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pp. 3579–3584. Consulted at http://www.lrec-conf.org/proceedings/lrec2014/pdf/1097_Paper.pdf.

Büdenbender, J., Frischauf, A., Goguadze, G., Melis, E., Libbrecht, P. & Ullrich, C. (2002a). Using Computer Algebra Systems as Cognitive Tools. *Intelligent Tutoring Systems*, pp. 802–810.

Büdenbender, J., Frischauf, A., Goguadze, G., Melis, E., Libbrecht, P. & Ullrich, C. (2002b). Using computer algebra systems as cognitive tools. *International Conference on Intelligent Tutoring Systems*, pp. 802–810.

BURKE, A. J. (1980). *STUDENTS' POTENTIAL FOR LEARNING CONTRASTED UNDER TUTORIAL AND GROUP APPROACHES TO INSTRUCTION*. (Ph.D. thesis, Ann Arbor).

Burke, A. J. (1983). *Students' potential for learning contrasted under tutorial and group approaches to instruction*. (Ph.D. thesis, University of Chicago, Joseph Regenstein Library, Department of Photoduplication).

Callan, J. & Mitamura, T. (2002). Knowledge-Based Extraction of Named Entities. 532–537. doi: 10.1145/584792.584880.

Camacho-Collados, J. & Pilehvar, M. T. (2018). From Word to Sense Embeddings: A Survey on Vector Representations of Meaning. *J. Artif. Int. Res.*, 63(1), 743–788. doi: 10.1613/jair.1.11259.

Castillo, J. J. & Alemany, L. A. (2008). An approach using Named Entities for Recognizing Textual Entailment. *Proceedings of the First Text Analysis Conference, TAC 2008, Gaithersburg, Maryland, USA, November 17-19, 2008*. Consulted at https://tac.nist.gov/publications/2008/participant.papers/Sagan.proceedings.pdf.

Cavanaugh, J. K. & Jacquemin, S. J. (2015). A large sample comparison of grade based student learning outcomes in online vs. face-to-face courses. *Online Learning*, 19(2), n2.

Cazden, C. (1979). Peekaboo as an Instructional Model: Discourse Development at Home and at School. *Papers and Reports on Child Language Development, No. 17*, 33–58.

Chen, Q., Zhu, X., Ling, Z.-H., Wei, S., Jiang, H. & Inkpen, D. (2017). Enhanced LSTM for Natural Language Inference. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1657–1668. doi: 10.18653/v1/P17-1152.

Chen, Q., Zhu, X., Ling, Z.-H., Inkpen, D. & Wei, S. (2018). Neural Natural Language Inference Models Enhanced with External Knowledge. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2406–2417. doi: 10.18653/v1/P18-1224.

Chi, M. (2011). Instructional Factors Analysis: A Cognitive Model For Multiple Instructional Interventions.

Chi, M., Koedinger, K., Gordon, G., Jordan, P. & Vanlehn, K. (2011, 01). Instructional Factors Analysis: A Cognitive Model For Multiple Instructional Interventions. *EDM 2011 - Proceedings of the 4th International Conference on Educational Data Mining*, pp. 61-70.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. doi: 10.3115/v1/D14-1179.

Chow, S., Yacef, K., Koprinska, I. & Curran, J. (2017). Automated Data-Driven Hints for Computer Programming Students. *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, (UMAP '17), 5–10. doi: 10.1145/3099023.3099065.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C. & Bengio, Y. (2015). A Recurrent Latent Variable Model for Sequential Data. *NIPS*.

Clark, K. & Manning, C. (2016). Deep Reinforcement Learning for Mention-Ranking Coreference Models. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2256–2262.

Coffield, F., Learning & Britain), S. R. C. G. (2004). *Learning styles and pedagogy in post-16 learning : a systematic and critical review*. London: London : Learning and Skills Research Centre.

Colby, K. M. (1981). Modeling a paranoid mind. *Behavioral and Brain Sciences*, 4(4), 515–534. doi: 10.1017/S0140525X00000030.

Colvin, K. F., Champaign, J., Liu, A., Zhou, Q., Fredericks, C. & Pritchard, D. E. (2014). Learning in an introductory physics MOOC: All cohorts learn equally, including an on-campus class. *The international review of research in open and distributed learning*, 15(4), 263–283.

Crowell, T. L. (2015a). Student Self Grading: Perception vs. Reality. *American Journal of Educational Research*, 3(4), 450–455. doi: 10.12691/education-3-4-10.

Crowell, T. L. (2015b). Student Self Grading: Perception vs. Reality. *American Journal of Educational Research*, 3(4), 450–455. Consulted at http://pubs.sciepub.com/education/3/4/10.

Cui, G., Lu, Q., Li, W. & Chen, Y. (2009, Sep.). Mining Concepts from Wikipedia for Ontology Construction. *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, 3, 287-290. doi: 10.1109/WI-IAT.2009.284.

Cukurova, M., Mavrikis, M., Luckin, R., Clark, J. & Crawford, C. (2017). Interaction analysis in online maths human tutoring: The case of third space learning. *international conference on artificial intelligence in education*, pp. 636–643.

Daniel, J. (2012). Making sense of MOOCs: Musings in a maze of myth, paradox and possibility. *Journal of interactive Media in education*, 2012(3), 18.

De Medio, C., Gasparetti, F., Limongelli, C., Sciarrone, F. & Temperini, M. (2016). Automatic Extraction of Prerequisites Among Learning Objects Using Wikipedia-based Content Analysis. *International conference on intelligent tutoring systems*, pp. 375–381.

Debole, F. & Sebastiani, F. (2004). Supervised Term Weighting for Automated Text Categorization. *Text Mining and its Applications*, pp. 81–97.

Decoste, D. & Schölkopf, B. (2002). Training Invariant Support Vector Machines. *Machine Learning*, 46(1), 161-190. doi: 10.1023/A:1012454411458.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T. & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, 391-407.

Demmans Epp, C., Phirangee, K., Hewitt, J. & Perfetti, C. A. (2020). Learning management system and course influences on student actions and learning experiences. *Educational Technology Research and Development*, 68(6), 3263-3297. doi: 10.1007/s11423-020-09821-1.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019a). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. doi: 10.18653/v1/N19-1423.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019b). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. doi: 10.18653/v1/N19-1423.

Dietrich, D. & Buckley, M. (2008). Verification of human-level proof steps in mathematics education. *Teaching Mathematics and Computer Science*, 6, 345-362.

Dinan, E., Roller, S., Shuster, K., Fan, A., Auli, M. & Weston, J. (2019). Wizard of Wikipedia: Knowledge-powered Conversational Agents. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dioşan, L., Rogozan, A. & Pécuchet, J.-P. (2008). Evolutionary Optimisation of Kernel and Hyper-Parameters for SVM. *Modelling, Computation and Optimization in Information Systems and Management Sciences*, pp. 107–116.

du Boulay, B. (2016). Artificial Intelligence as an Effective Classroom Assistant. *IEEE Intelligent Systems*, 31(6), 76-81. doi: 10.1109/MIS.2016.93.

Fader, A., Zettlemoyer, L. & Etzioni, O. (2014). Open question answering over curated and extracted knowledge bases. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1156—1165.

Ferrone, L. & Zanzotto, F. M. (2020). Symbolic, Distributed, and Distributional Representations for Natural Language Processing in the Era of Deep Learning: A Survey. *Frontiers in Robotics and AI*, 6, 153. doi: 10.3389/frobt.2019.00153.

Fesseha, A., Xiong, S., Emiru, E. D. & Dahou, A. (2020). Text Classification of News Articles Using Machine Learning on Low-resourced Language: Tigrigna. *2020 3rd International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pp. 34-38. doi: 10.1109/ICAIBD49809.2020.9137443.

Folsom-Kovarik, J., Schatz, S. & Nicholson, D. (2010a). Plan Ahead : Pricing ITS Learner Models.

Folsom-Kovarik, J. T., Schatz, S. & Nicholson, D. (2010b). Plan ahead: Pricing ITS learner models. *Proceedings of the 19th Behavior Representation in Modeling & Simulation (BRIMS) Conference*, pp. 47–54.

Fossati, D. (2014). Data driven automatic feedback generation in the iList intelligent tutoring system.

Freedman, R. (1999). Atlas: A Plan Manager for Mixed-Initiative, Multimodal Dialogue.

Fürnkranz, J. (2010). Decision Tree. In Sammut, C. & Webb, G. I. (Eds.), *Encyclopedia of Machine Learning* (pp. 263–267). Boston, MA: Springer US. doi: 10.1007/978-0-387-30164-8_204.

Goguadze, G., Palomo, A. G. & Melis, E. (2005). Interactivity of Exercises in ActiveMath. *ICCE*, pp. 109–115.

Gomaa, W. & Fahmy, A. (2013). A Survey of Text Similarity Approaches. *international journal of Computer Applications*, 68, 13–18. doi: 10.5120/11638-7118.

Gong, Y., Luo, H. & Zhang, J. (2018). Natural Language Inference over Interaction Space. *International Conference on Learning Representations*. Consulted at https://openreview.net/forum?id=r1dHXnH6-.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press.

Goodman, J. T. (2001). A bit of progress in language modeling. *Computer Speech & Language*, 15(4), 403-434. doi: https://doi.org/10.1006/csla.2001.0174.

Graesser, A. C. & Person, N. K. (1994). Question asking during tutoring. *American educational research journal*, 31(1), 104–137.

Graesser, A. C., Wiemer-Hastings, K., Wiemer-Hastings, P. & Kreuz, R. (1999). AutoTutor: A simulation of a human tutor. *Cognitive Systems Research*, 1(1), 35-51. doi: https://doi.org/10.1016/S1389-0417(99)00005-4.

Graesser, A. C., Wiemer-Hastings, P., Wiemer-Hastings, K., Harter, D., Group, T. R. G. T. R. & Person, N. (2000). Using Latent Semantic Analysis to Evaluate the Contributions of Students in AutoTutor. *Interactive Learning Environments*, 8(2), 129-147.

Graesser, A. C., VanLehn, K., Rose, C. P., Jordan, P. W. & Harter, D. (2001a). Intelligent Tutoring Systems with Conversational Dialogue. *AI Magazine*, 22(4), 39. doi: 10.1609/aimag.v22i4.1591.

Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W. & Harter, D. (2001b). Intelligent tutoring systems with conversational dialogue. *AI magazine*, 22(4), 39–39.

Graesser, A. C., Chipman, P., Haynes, B. C. & Olney, A. (2005). AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*, 48(4), 612–618.

Graesser, A. C., Cai, Z., Morgan, B. & Wang, L. (2017). Assessment with computer agents that engage in conversational dialogues and trialogues with learners. *Computers in Human Behavior*, 76, 607 - 616. doi: https://doi.org/10.1016/j.chb.2017.03.041.

Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. *CoRR*, abs/1308.0850, 1–43. Consulted at http://dblp.uni-trier.de/db/journals/corr/corr1308.html#Graves13.

Greer, J. & Mark, M. (2016). Evaluation Methods for Intelligent Tutoring Systems Revisited. *International Journal of Artificial Intelligence in Education*, 26(1), 387-392. doi: 10.1007/s40593-015-0043-2.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J. (2015). LSTM: A Search Space Odyssey. cite arxiv:1503.04069Comment: 12 pages, 6 figures, doi: 10.1109/TNNLS.2016.2582924.

Grenander, M., Belfer, R., Kochmar, E., Serban, I., St-Hilaire, F. & Cheung, J. (2021, 03). Deep Discourse Analysis for Generating Personalized Feedback in Intelligent Tutor Systems.

Guo, Q., Kulkarni, C., Kittur, A., Bigham, J. P. & Brunskill, E. (2016). Questimator: Generating knowledge assessments for arbitrary topics. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16). AAAI Press*.

Hake, R. R. (1998). Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American Journal of Physics*, 66, 64-74.

Hennecke, M. (1999). *Online Diagnose in intelligenten mathematischen Lehr-Lern-Systemen*. VDI-Verlag.

Hevner, A. R., March, S. T., Park, J. & Ram, S. (2004). Design Science in Information Systems Research. *MIS Q.*, 28(1), 75–105.

Holstein, K., McLaren, B. M. & Aleven, V. (2017). Intelligent Tutors as Teachers' Aides: Exploring Teacher Needs for Real-time Analytics in Blended Classrooms. *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, pp. 257–266.

Holstein, K., McLaren, B. M. & Aleven, V. (2019). Designing for complementarity: Teacher and student needs for orchestration support in AI-enhanced classrooms. *International Conference on Artificial Intelligence in Education*, pp. 157–171.

Hone, K. & El Said, G. (2016). Exploring the factors affecting MOOC retention: A survey study. *Computers & Education*, 98, 157–168. doi: 10.1016/j.compedu.2016.03.016.

Hone, K. S. & El Said, G. R. (2016). Exploring the factors affecting MOOC retention: A survey study. *Computers & Education*, 98, 157-168. doi: https://doi.org/10.1016/j.compedu.2016.03.016.

Hone, K. S. & El Said, G. R. (2016). Exploring the factors affecting MOOC retention: A survey study. *Computers & Education*, 98, 157–168.

Hrastinski, S., Stenbom, S., Benjaminsson, S. & Jansson, M. (2019a). Identifying and exploring the effects of different types of tutor questions in individual online synchronous tutoring in mathematics. *Interactive Learning Environments*, 0(0), 1-13. doi: 10.1080/10494820.2019.1583674.

Hrastinski, S., Stenbom, S., Benjaminsson, S. & Jansson, M. (2019b). Identifying and exploring the effects of different types of tutor questions in individual online synchronous tutoring in mathematics. *Interactive Learning Environments*, 29, 1-13. doi: 10.1080/10494820.2019.1583674.

Hrastinski, S., Stenbom, S., Benjaminsson, S. & Jansson, M. (2021). Identifying and exploring the effects of different types of tutor questions in individual online synchronous tutoring in mathematics. *Interactive Learning Environments*, 29(3), 510-522.

Huang, Z., Xu, W. & Yu, K. (2015). Bidirectional LSTM-CRF Models for Sequence Tagging. Consulted at http://arxiv.org/abs/1508.01991.

Hume, G., Michael, J., Rovick, A. & Evens, M. (1996a). Hinting as a Tactic in One-on-One Tutoring. *The Journal of the Learning Sciences*, 5, 23-47.

Hume, G., Michael, J., Rovick, A. & Evens, M. (1996b). Hinting as a Tactic in One-on-One Tutoring. *Journal of The Learning Sciences - J LEARN SCI*, 5, 23-47. doi: 10.1207/s15327809jls0501_2.

Hume, G., Michael, J., Rovick, A. & Evens, M. (1996c). Hinting as a Tactic in One-on-One Tutoring. *Journal of the Learning Sciences*, 5(1), 23-47.

J. Mackness, S. M. & Williams, R. (2010). The ideals and reality of participating in a MOOC. In *Proceedings of the 7th International Conference on Networked Learning 2010* (pp. 266–275). University of Lancaster.

Jabri, S., Dahbi, A., Gadi, T. & Bassir, A. (2018). Ranking of text documents using TF-IDF weighting and association rules mining. *2018 4th International Conference on Optimization and Applications (ICOA)*, 1-6.

Jiang, Y. (2014). Exploring teacher questioning as a formative assessment strategy. *RELC Journal*, 45(3), 287–304.

Jona, K. & Naidu, S. (2014). MOOCs: emerging research. *Distance Education*, 35(2), 141-144. doi: 10.1080/01587919.2014.928970.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N. & Wu, Y. (2016). *Exploring the Limits of Language Modeling*. Consulted at https://arxiv.org/abs/1602.02410.

Kaliszyk, C., Wiedijk, F., Hendriks, M. & Raamsdonk, F. (2007). Teaching logic using a state-of-the-art proof assistant.

Kapugama, K. D. C. G., Lorensuhewa, S. A. S. & Kalyani, M. A. L. (2016, Sep.). Enhancing Wikipedia search results using Text Mining. *2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, pp. 168-175. doi: 10.1109/ICTER.2016.7829915.

Kashihara, A. & Hasegawa, S. (2005). A Model of Meta-Learning for Web-Based Navigational Learning. *Advanced Technology for Learning*, 2, 198-206. doi: 10.2316/Journal.208.2005.4.208-0862.

Kingma, D. P. & Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Kingma, D. P. & Welling, M. (2014). Auto-Encoding Variational Bayes. *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R. & Fidler, S. (2015). Skip-Thought Vectors. cite arxiv:1506.06726Comment: 11 pages, Consulted at http://arxiv.org/abs/1506.06726.

Kirtman, L. (2009). Online versus in-class courses: An examination of differences in learning outcomes. *Issues in teacher education*, 18(2), 103–116.

Knight, K. (1983). Book Reviews : The Manual of Learning Styles Peter Honey and Alan Mumford: Published and distributed by Peter Honey, Ardingley House, 10 Linden Avenue, Maidenhead, Berkshire, SL6 6BH, 1982, 83 pp., £25·20, ISBN 0 9508444 0 3. *Management Education and Development*, 14(2), 147-150.

Kochmar, E., Vu, D. D., Belfer, R., Gupta, V., Serban, I. & Pineau, J. (2020a). Automated Personalized Feedback Improves Learning Gains in An Intelligent Tutoring System. *Artificial Intelligence in Education*, 12164, 140 - 146.

Kochmar, E., Do Dung, V., Belfer, R., Gupta, V., Serban, I. & Pineau, J. (2020b, 05). Automated Personalized Feedback Improves Learning Gains in an Intelligent Tutoring System.

Kochmar, E., Vu, D. D., Belfer, R., Gupta, V., Serban, I. V. & Pineau, J. (2020c). Automated Personalized Feedback Improves Learning Gains in An Intelligent Tutoring System. *Artificial Intelligence in Education*, pp. 140–146.

Kochmar, E., Vu, D. D., Belfer, R., Gupta, V., Serban, I. V. & Pineau, J. (2021). Automated Data-Driven Generation of Personalized Pedagogical Interventions in Intelligent Tutoring Systems. *International Journal of Artificial Intelligence in Education*. doi: 10.1007/s40593-021-00267-x.

Koedinger, K., Cunningham, K., Skogsholm, A. & Leber, B. (2008). An Open Repository and analysis tools for fine-grained, longitudinal learner data. *EDM*.

Koedinger, K. R. & Corbett, A. (2005). Cognitive Tutors. In Sawyer, R. K. (Ed.), *The Cambridge Handbook of the Learning Sciences* (pp. 61–78). Cambridge University Press. doi: 10.1017/CBO9780511816833.006.

Koedinger, K. R., Kim, J., Jia, J. Z., McLaughlin, E. A. & Bier, N. L. (2015). Learning is not a spectator sport: Doing is better than watching for learning from a MOOC. *Proceedings of the second (2015) ACM conference on learning@ scale*, pp. 111–120.

Koxvold, I. (2014). *MOOCs: Opportunities for their use in compulsory-age education*. Department for Education.

Kulik, J. A. & Fletcher, J. D. (2016a). Effectiveness of Intelligent Tutoring Systems: A Meta-Analytic Review. *Review of Educational Research*, 86(1), 42-78. doi: 10.3102/0034654315581420.

Kulik, J. A. & Fletcher, J. (2016b). Effectiveness of intelligent tutoring systems: a meta-analytic review. *Review of educational research*, 86(1), 42–78.

Kumar, A. N. (2005a). Results from the Evaluation of the Effectiveness of an Online Tutor on Expression Evaluation. *SIGCSE Bull.*, 37(1), 216–220. doi: 10.1145/1047124.1047422.

Kumar, A. N. (2005b). Results from the Evaluation of the Effectiveness of an Online Tutor on Expression Evaluation. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, (SIGCSE '05), 216–220. doi: 10.1145/1047344.1047422.

Kumar, A. N. (2005c). Generation of Problems, Answers, Grade, and Feedback—Case Study of a Fully Automated Tutor. *J. Educ. Resour. Comput.*, 5(3), 3–es. doi: 10.1145/1163405.1163408.

Lahitani, A. R., Permanasari, A. E. & Setiawan, N. A. (2016). Cosine similarity to determine similarity measure: Study case in online essay assessment. *2016 4th International Conference on Cyber and IT Service Management*, pp. 1-6. doi: 10.1109/CITSM.2016.7577578.

Lahti, L. (2009). Guided generation of pedagogical concept maps from the Wikipedia. *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, pp. 1741–1750.

Lakhotia, S. & Bresson, X. (2018). An Experimental Comparison of Text Classification Techniques. *2018 International Conference on Cyberworlds (CW)*, pp. 58-65. doi: 10.1109/CW.2018.00022.

Lane, K., Menzies, H. M., Ennis, R. & Bezdek, J. M. (2013). School-wide Systems to Promote Positive Behaviors and Facilitate Instruction. *Journal of Curriculum and Instruction*, 7,

6-31.

Lavie, A. & Agarwal, A. (2007). METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. 228-231.

Leelawong, K. & Biswas, G. (2008). Designing learning by teaching agents: The Betty's Brain system. *International Journal of Artificial Intelligence in Education*, 18(3), 181–208.

Lehman, B., D'Mello, S. & Graesser, A. (2012a). Confusion and complex learning during interactions with computer learning environments. *The Internet and Higher Education*, 15(3), 184-194. doi: https://doi.org/10.1016/j.iheduc.2012.01.002. Emotions in online learning environments.

Lehman, B., D'Mello, S. & Graesser, A. (2012b). Confusion and complex learning during interactions with computer learning environments. *Internet and Higher Education*, 15(3), 184–194. Consulted at https://www.learntechlib.org/p/199222.

Li, J., Sun, A., Han, J. & Li, C. (2018). A Survey on Deep Learning for Named Entity Recognition. Los Alamitos, CA, USA: IEEE Computer Society. doi: 10.1109/TKDE.2020.2981314.

Li, Y. & Yang, T. (2018). Word Embedding for Understanding Natural Language: A Survey. In Srinivasan, S. (Ed.), *Guide to Big Data Applications* (pp. 83–104). Cham: Springer International Publishing. doi: 10.1007/978-3-319-53817-4_4.

Lin, C.-Y. (2004, 01). ROUGE: A Package for Automatic Evaluation of summaries. pp. 10.

Lin, C. F., chu Yeh, Y., Hung, Y. H. & Chang, R. I. (2013a). Data mining for providing a personalized learning path in creativity: An application of decision trees. *Computers & Education*, 68, 199-210. doi: https://doi.org/10.1016/j.compedu.2013.05.009.

Lin, C. F., Yeh, Y.-C., Hung, Y. H. & Chang, R. I. (2013b). Data mining for providing a personalized learning path in creativity: An application of decision trees. *Computers & Education*, 68, 199 - 210. doi: https://doi.org/10.1016/j.compedu.2013.05.009.

Lipton, Z. C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning.

Liu, M., Calvo, R. A., Aditomo, A. & Pizzato, L. A. (2012a). Using Wikipedia and Conceptual Graph Structures to Generate Questions for Academic Writing Support. *IEEE Transactions on Learning Technologies*, 5(3), 251–263.

Liu, M., Calvo, R. A. & Rus, V. (2012b). G-Asks: An intelligent automatic question generation system for academic writing support. *Dialogue & Discourse*, 3(2), 101–124.

Lowe, R., Noseworthy, M., Serban, I. V., Angelard-Gontier, N., Bengio, Y. & Pineau, J. (2017). Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Lucas, A., Williams, A. T. & Cabrales, P. (2019). Prediction of Recovery From Severe Hemorrhagic Shock Using Logistic Regression. *IEEE Journal of Translational Engineering in Health and Medicine*, 7, 1-9. doi: 10.1109/JTEHM.2019.2924011.

LW, A., DR, K., PW, A., KA, C., Mayer, R., PR, P., Raths, J. & MC, W. (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. New York: Longman.

Ma, W., Adesope, O. O., Nesbit, J. & Liu, Q. (2014). Intelligent tutoring systems and learning outcomes: A meta-analysis. *Journal of Educational Psychology*, 106, 901-918.

Maitra, D. S., Bhattacharya, U. & Parui, S. K. (2015). CNN based common approach to handwritten character recognition of multiple scripts. *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1021-1025. doi: 10.1109/ICDAR.2015.7333916.

Maity, A. (2016). Supervised Classification of RADARSAT-2 Polarimetric Data for Different Land Features. Consulted at http://arxiv.org/abs/1608.00501.

Makatchev, M., VanLehn, K., Jordan, P. & Pappuswamy, U. (2006a). Representation and Reasoning for Deeper Natural Language Understanding in a Physics Tutoring System. *FLAIRS Conference*.

Makatchev, M., Vanlehn, K. & Jordan, P. W. (2006b). Representation and Reasoning for Deeper Natural Language Understanding in a Physics Tutoring System.

Malakasiotis, P. & Androutsopoulos, I. (2007). Learning Textual Entailment using SVMs and String Similarity Measures. *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 42–47. Consulted at https://aclanthology.org/W07-1407.

Malekzadeh, M., Mustafa, M. & Lahsasna, A. (2015). A Review of Emotion Regulation in Intelligent Tutoring Systems. *Educational Technology & Society*, 18, 435-445.

Mark, M. A. & Greer, J. E. (1993). Evaluation Methodologies for Intelligent Tutoring Systems. *JOURNAL OF ARTIFICIAL INTELLIGENCE IN EDUCATION*, 4, 129–153.

Mathew, J., Pang, C. K., Luo, M. & Leong, W. H. (2018). Classification of Imbalanced Data by Oversampling in Kernel Space of Support Vector Machines. *IEEE Transactions on Neural*

Networks and Learning Systems, 29(9), 4065-4076. doi: 10.1109/TNNLS.2017.2751612.

McBroom, J., Koprinska, I. & Yacef, K. (2021). A Survey of Automated Programming Hint Generation: The HINTS Framework. *ACM Comput. Surv.*, 54(8), 1–27. doi: 10.1145/3469885.

McCann, B., Bradbury, J., Xiong, C. & Socher, R. (2017). *Learned in Translation: Contextualized Word Vectors*. Advances in Neural Information Processing Systems. Curran Associates, Inc.

Melis, E. & Siekmann, J. (2004a). ActiveMath: An Intelligent Tutoring System for Mathematics. *Artificial Intelligence and Soft Computing - ICAISC 2004*, pp. 91–101.

Melis, E. & Siekmann, J. (2004b). ActiveMath: An Intelligent Tutoring System for Mathematics. *Artificial Intelligence and Soft Computing - ICAISC 2004*, pp. 91–101.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J. & Khudanpur, S. (2010). Recurrent neural network based language model. *INTERSPEECH*, pp. 1045-1048. Consulted at http://dblp.uni-trier.de/db/conf/interspeech/interspeech2010.html#MikolovKBCK10.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *CoRR*, abs/1310.4546. Consulted at http://arxiv.org/abs/1310.4546.

Mills, S. R., Rice, C. T., Berliner, D. C. & Rosseau, E. W. (1980). The correspondence between teacher questions and student answers in classroom discourse. *The Journal of Experimental Education*, 48(3), 194–204.

Milne, D. & Witten, I. H. (2013). An open-source toolkit for mining Wikipedia. *Artificial Intelligence*, 194, 222 - 239. doi: https://doi.org/10.1016/j.artint.2012.06.007. Artificial Intelligence, Wikipedia and Semi-Structured Resources.

Munshi, A. & Biswas, G. (2019a). Personalization in OELEs: Developing a Data-Driven Framework to Model and Scaffold SRL Processes. *Artificial Intelligence in Education*, pp. 354–358.

Munshi, A. & Biswas, G. (2019b). Personalization in OELEs: Developing a Data-Driven Framework to Model and Scaffold SRL Processes. *International Conference on Artificial Intelligence in Education*, pp. 354–358.

Munshi, A., Mishra, S., Zhang, N., Paquette, L., Ocumpaugh, J., Baker, R. & Biswas, G. (2020). Modeling the Relationships Between Basic and Achievement Emotions in Computer-Based Learning Environments. *Artificial Intelligence in Education*, pp. 411–422.

Nadeau, D. & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30, 3-26.

Narciss, S., Sosnovsky, S., Schnaubert, L., Andrès, E., Eichelmann, A., Goguadze, G. & Melis, E. (2014). Exploring feedback and student characteristics relevant for personalizing feedback strategies. *Computers & Education*, 71, 56–76.

Nesbit, J. C., Adesope, O. O., Liu, Q. & Ma, W. (2014). How Effective are Intelligent Tutoring Systems in Computer Science Education? *2014 IEEE 14th International Conference on Advanced Learning Technologies*, pp. 99-103. doi: 10.1109/ICALT.2014.38.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R. & Zeman, D. (2016). Universal Dependencies v1: A Multilingual Treebank Collection. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Consulted at http://www.petrovi.de/data/lrec16.pdf.

Nkambou, R., Mizoguchi, R. & Bourdeau, J. (2010). *Advances in Intelligent Tutoring Systems* (ed. 1st). Springer Publishing Company, Incorporated.

Nothman, J., Ringland, N., Radford, W., Murphy, T. & Curran, J. (2013). Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence*, 194, 151–175. doi: 10.1016/j.artint.2012.03.006.

Nye, B., Graesser, A. & Hu, X. (2014a). AutoTutor and Family: A Review of 17 Years of Natural Language Tutoring. doi: 10.1007/s40593-014-0029-5.

Nye, B. D., Graesser, A. C. & Hu, X. (2014b). AutoTutor and family: A review of 17 years of natural language tutoring. *International Journal of Artificial Intelligence in Education*, 24(4), 427–469.

Olney, A. & Cade, W. (2015a, 08). Authoring Intelligent Tutoring Systems Using Human Computation: Designing for Intrinsic Motivation. pp. 628-639. doi: 10.1007/978-3-319-20816-9_60.

Olney, A. M. (2018). Using novices to scale up intelligent tutoring systems. *Interservice/Industry training, Simulation, and Education Conference (I/ITSEC)*.

Olney, A. M. & Cade, W. L. (2015b). Authoring intelligent tutoring systems using human computation: designing for intrinsic motivation. *International conference on augmented cognition*, pp. 628–639.

Otto, D., Bollmann, A., Becker, S. & Sander, K. (2018). It's the learning, stupid! Discussing the role of learning outcomes in MOOCs. *Open Learning: The Journal of Open, Distance and e-Learning*, 33(3), 203–220.

Pane, J. F., Griffin, B. A., McCaffrey, D. F. & Karam, R. (2014). Effectiveness of Cognitive Tutor Algebra I at Scale. *Educational Evaluation and Policy Analysis*, 36(2), 127-144.

Papineni, K., Roukos, S., Ward, T. & Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation.

Parikh, A., Täckström, O., Das, D. & Uszkoreit, J. (2016). A Decomposable Attention Model for Natural Language Inference. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2249–2255. doi: 10.18653/v1/D16-1244.

Passier, H. & Jeuring, J. (2006). Feedback in an interactive equation solver. UU WINFI Informatica en Informatiekunde.

Pennington, J., Socher, R. & Manning, C. (2014a). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. doi: 10.3115/v1/D14-1162.

Pennington, J., Socher, R. & Manning, C. D. (2014b). Glove: Global Vectors for Word Representation. *EMNLP*, 14, 1532–1543. Consulted at https://nlp.stanford.edu/pubs/glove.pdf.

Petrov, S., Das, D. & McDonald, R. (2012). A Universal Part-of-Speech Tagset. *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pp. 2089–2096. Consulted at http://www.lrec-conf.org/proceedings/lrec2012/pdf/274_Paper.pdf.

Pokhrel, S. & Chhetri, R. (2021). A Literature Review on Impact of COVID-19 Pandemic on Teaching and Learning. *Higher Education for the Future*, 8(1), 133-141.

Polyzou, A., Athanasios, N. & Karypis, G. (2019). Scholars Walk: A Markov Chain Framework for Course Recommendation. *Proceedings of the 12th International Conference on Educational Data Mining*, pp. 396–401.

Pradhan, S. S., Ward, W. H., Hacioglu, K., Martin, J. H. & Jurafsky, D. (2004). Shallow Semantic Parsing using Support Vector Machines. *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pp. 233–240. Consulted at https://www.aclweb.org/anthology/N04-1030.

Pranckevičius, T. & Marcinkevičius, V. (2016). Application of Logistic Regression with part-of-the-speech tagging for multi-class text classification. *2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, pp. 1-5. doi: 10.1109/AIEEE.2016.7821805.

Price, T. W., Dong, Y., Zhi, R., Paaßen, B., Lytle, N., Cateté, V. & Barnes, T. (2019). A Comparison of the Quality of Data-Driven Programming Hint Generation Algorithms. *International Journal of Artificial Intelligence in Education*, 29(3), 368-395.

Qwaider, S. R. & Abu-Naser, S. S. (2018). Excel Intelligent Tutoring System. *International Journal of Academic Information Systems Research (IJAISR)*, 2(2), 8–18.

Rajaraman, A. & Ullman, J. D. (2011). *Mining of Massive Datasets*. USA: Cambridge University Press.

Ram, A. (1991). A theory of questions and question asking. *Journal of the Learning Sciences*, 1(3-4), 273–318.

Ramachandran, S., Jensen, R., Ludwig, J., Domeshek, E. & Haines, T. (2018). ITADS: a real-world intelligent tutor to train troubleshooting skills. *International Conference on Artificial Intelligence in Education*, pp. 463–468.

Ramírez-Noriega, A., Juárez-Ramírez, R., Jiménez, S., Martínez-Ramírez, Y. & Figueroa Pérez, J. (2018). Determination of the course sequencing to intelligent tutoring systems using an ontology and Wikipedia. *Journal of Intelligent & Fuzzy Systems*, 34(5), 3177–3185.

Ran, J., Zhang, G., Zheng, T. & Wang, W. (2018). Logistic Regression Analysis on Learning Behavior and Learning Effect Based on SPOC Data. *2018 13th International Conference on Computer Science Education (ICCSE)*, pp. 1-5. doi: 10.1109/ICCSE.2018.8468834.

Rezende, D. J., Mohamed, S. & Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, (ICML'14), II–1278–II–1286.

Rieber, L. P. (2017). Participation patterns in a massive open online course (MOOC) about statistics. *British Journal of Educational Technology*, 48(6), 1295–1304.

Ritter, S., Anderson, J. R., Koedinger, K. R. & Corbett, A. (2007). Cognitive Tutor: Applied research in mathematics education. *Psychonomic bulletin & review*, 14(2), 249–255.

Rivers, K. (2017). Automated Data-Driven Hint Generation for Learning Programming. doi: 10.1184/R1/6714911.v1.

Rocktäschel, T., Grefenstette, E., Hermann, K., Kočiský, T. & Blunsom, P. (2015). Reasoning about Entailment with Neural Attention.

Rosé, C. P., Moore, J. D., VanLehn, K. & Allbritton, D. (2001). *A comparative evaluation of socratic versus didactic tutoring*. Proceedings of the Annual Meeting of the Cognitive Science Society.

RTan. Y.,Quintana, R. M. (2019). What can we learn about learner interaction when one course is hosted on two MOOC platforms? *Companion Proceedings to the International Conference on Learning Analytics and Knowledge (LAK)*.

Rus, V., Stefanescu, D., Baggett, W., Niraula, N., Franceschetti, D. & Graesser, A. C. (2014a). Macro-adaptation in Conversational Intelligent Tutoring Matters. *Intelligent Tutoring Systems*, pp. 242–247.

Rus, V., Stefanescu, D., Baggett, W., Niraula, N., Franceschetti, D. & Graesser, A. C. (2014b). Macro-adaptation in conversational intelligent tutoring matters. *International Conference on Intelligent Tutoring Systems*, pp. 242–247.

Rus, V., Stefanescu, D., Niraula, N. & Graesser, A. C. (2014c). DeepTutor: Towards Macro- and Micro-Adaptive Conversational Intelligent Tutoring at Scale. *Proceedings of the First ACM Conference on Learning @ Scale Conference*, (L@S '14), 209–210. doi: 10.1145/2556325.2567885.

Rus, V., Stefanescu, D., Niraula, N. & Graesser, A. C. (2014d). DeepTutor: towards macro-and micro-adaptive conversational intelligent tutoring at scale. *Proceedings of the first ACM conference on Learning@ Scale conference*, pp. 209–210.

Sampson, D. & Karagiannidis, C. (2002). Personalised Learning: Educational, Technological and Standardisation Perspective. *Digital Education Review*, 24-39.

Schwartz, A. & Hearst, M. (2003). A Simple Algorithm For Identifying Abbreviation Definitions in Biomedical Text. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, 4, 451-62. doi: 10.1142/9789812776303_0042.

Sekine, S. & Nobata, C. (2004). Definition, Dictionaries and Tagger for Extended Named Entity Hierarchy. *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. Consulted at http://www.lrec-conf.org/proceedings/lrec2004/pdf/65.pdf.

Serban, I. V., Gupta, V., Kochmar, E., Vu, D. D., Belfer, R., Pineau, J., Courville, A., Charlin, L. & Bengio, Y. (2020). A Large-Scale, Open-Domain, Mixed-Interface Dialogue-Based ITS for STEM. *Artificial Intelligence in Education*, pp. 387–392.

Sethi, I. K. & YOO, J. H. (1997). Structure-driven induction of decision tree classifiers through neural learning. *Pattern Recognition*, 30(11), 1893 - 1904. doi: https://doi.org/10.1016/S0031-3203(97)00005-8.

Shah, D. (2019). By The Numbers: MOOCs in 2019.

Shah, R., Shah, D. & Kurup, L. (2017). Automatic question generation for intelligent tutoring systems. *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, pp. 127–132.

Shawar, B. & Atwell, E. (2007). Chatbots: Are they Really Useful? *LDV Forum*, 22, 29-49.

Smith, N. A. (2011). *Linguistic Structure Prediction*. Morgan and Claypool.

Smola, A. J. & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3), 199-222. doi: 10.1023/B:STCO.0000035301.49549.88.

Stamper, J., Eagle, M., Barnes, T. & Croy, M. (2011a, 06). Experimental Evaluation of Automatic Hint Generation for a Logic Tutor. 22, 345-352. doi: 10.1007/978-3-642-21869-9_45.

Stamper, J. C., Eagle, M., Barnes, T. & Croy, M. (2011b). Experimental Evaluation of Automatic Hint Generation for a Logic Tutor. *Artificial Intelligence in Education*, pp. 345–352.

Stash, N. V., Cristea, A. I. & De Bra, P. M. (2004). Authoring of Learning Styles in Adaptive Hypermedia: Problems and Solutions. (WWW Alt. '04), 114–123.

Sundermeyer, M., Schlüter, R. & Ney, H. (2012). LSTM Neural Networks for Language Modeling. *INTERSPEECH*.

Talukdar, P. P. & Cohen, W. W. (2012). Crowdsourced Comprehension: Predicting Prerequisite Structure in Wikipedia. *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pp. 307–315.

Tamura, Y., Takase, Y., Hayashi, Y. & Nakano, Y. I. (2015). Generating quizzes for history learning based on Wikipedia articles. *International Conference on Learning and Collaboration Technologies*, pp. 337–346.

Tjong Kim Sang, E. F. & De Meulder, F. (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147. Consulted at https://www.aclweb.org/anthology/W03-0419.

Tolles, J. & Meurer, W. J. (2016). Logistic Regression: Relating Patient Characteristics to Outcomes. *JAMA*, 316(5), 533-534. doi: 10.1001/jama.2016.7653.

Tomkins, S., Ramesh, A. & Getoor, L. (2016). Predicting Post-Test Performance from Online Student Behavior: A High School MOOC Case Study.

Tosun, N. (2006). Determination of optimum parameters for multi-performance characteristics in drilling by using grey relational analysis. *International Journal of Advanced Manufacturing Technology*, 28, 450-455. doi: 10.1007/s00170-004-2386-y.

Tsovaltzi, D., Fiedler, A. & Horacek, H. (2004, 08). A Multi-dimensional Taxonomy for Automating Hinting. 3220, 772-781. doi: 10.1007/978-3-540-30139-4_73.

Van de Pol, J., Volman, M. & Beishuizen, J. (2010). Scaffolding in teacher–student interaction: A decade of research. *Educational psychology review*, 22(3), 271–296.

VanLehn, K., Graesser, A., Jackson, G. T., Jordan, P., Olney, A. & Rosé, C. (2007). When Are Tutorial Dialogues More Effective Than Reading? *Cognitive science*, 31 1, 3-62.

VanLehn, K. (2011a). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4), 197–221. doi: 10.1080/00461520.2011.611369. Funding Information: I am grateful for the close readings and thoughtful comments of Michelene T. H. Chi, Dexter Fletcher, Jared Freedman, Kasia Muldner, and Stellan Ohlsson. My research summarized here was supported by many years of funding from the Office of Naval Research (N00014-00-1-0600) and National Science Foundation (9720359, EIA-0325054, 0354420, 0836012, and DRL-0910221).

VanLehn, K. (2011b). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educational Psychologist*, 46(4), 197-221. doi: 10.1080/00461520.2011.611369.

Vanlehn, K. (2011). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educational Psychologist*, 46, 197-221. doi: 10.1080/00461520.2011.611369.

Vanlehn, K., Graesser, A., Jackson, G., Jordan, P., Olney, A. & Rosé, C. (2007). When Are Tutorial Dialogues More Effective Than Reading? *Cognitive science*, 31, 3-62. doi: 10.1080/03640210709336984.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, pp. 5998–6008.

Ventura, M., Chang, M., Foltz, P., Mukhi, N., Yarbro, J., Salverda, A. P., Behrens, J., Ahn, J.-w., Ma, T., Dhamecha, T. I. et al. (2018). Preliminary evaluations of a dialogue-based digital tutor. *International Conference on Artificial Intelligence in Education*, pp. 480–483.

Vigentini, L. & Clayphan, A. (2015). Pacing through MOOCs: Course Design or Teaching Effect? *Proceedings of the 8th International Conference on Educational Data Mining, EDM 2015, Madrid, Spain, June 26-29, 2015*, pp. 572–573. Consulted at http://www.educationaldatamining.org/EDM2015/proceedings/poster572-573.pdf.

Vilnis, L. & McCallum, A. (2015). Word Representations via Gaussian Embedding. *ICLR*.

Voorhees, E. M. & Tice, D. M. (2000). Building a Question Answering Test Collection. *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (SIGIR '00), 200–207. doi: 10.1145/345508.345577.

Vrandečić, D. & Krötzsch, M. (2014). Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, 57, 78-85. doi: 10.1145/2629489.

Walls, J., Widmeyer, G. & Sawy, O. (2004). Assessing information system design theory in perspective: How useful was our 1992 initial rendition? *Journal of Information Technology Theory and Application*, 6, 43-58.

Wang, J., Yang, Y. & Xia, B. (2019). A Simplified Cohen's Kappa for Use in Binary Classification Data Annotation Tasks. *IEEE Access*, 7, 164386-164397. doi: 10.1109/ACCESS.2019.2953104.

Wang, Y., Paquette, L. & Baker, R. (2014). A Longitudinal Study on Learner Career Advancement in MOOCs. *Journal of Learning Analytics*, 1(3), 203-206. doi: 10.18608/jla.2014.13.23.

Webb, N. M. (1989). Peer interaction and learning in small groups. *International journal of Educational research*, 13(1), 21–39.

Weizenbaum, J. (1966). ELIZA—a Computer Program for the Study of Natural Language Communication between Man and Machine. *Commun. ACM*, 9(1), 36–45. doi: 10.1145/365153.365168.

Willis, A., Davis, G., Ruan, S., Manoharan, L., Landay, J. & Brunskill, E. (2019). Key Phrase Extraction for Generating Educational Question-Answer Pairs. *Proceedings of the Sixth (2019) ACM Conference on Learning@ Scale*, pp. 1–10.

Wilson, J. D. (1990). Artificial Intelligence and Tutoring Systems. *Journal of Research on Computing in Education*, 23(1), 142-144.

Wood, D. (2003). The Why? What? When? and How? of Tutoring: The Development of Helping and Tutoring Skills in Children. *Literacy teaching and learning*, 7, 1–30.

Wood, D. & Wood, H. (1996). Vygotsky, Tutoring and Learning. *Oxford Review of Education*, 22(1), 5-16. doi: 10.1080/0305498960220101.

Woolf, B. P. (2008). *Building Intelligent Interactive Tutors: Student-Centered Strategies for Revolutionizing e-Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Wu, L. & Looi, C.-K. (2010). Agent Prompts: Scaffolding Students for Productive Reflection in an Intelligent Learning Environment. *Intelligent Tutoring Systems*, pp. 426–428.

Yadav, V. & Bethard, S. (2019a). A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. Consulted at http://arxiv.org/abs/1910.11470.

Yadav, V. & Bethard, S. (2019b). A Survey on Recent Advances in Named Entity Recognition from Deep Learning models.

Yin, B., Patikorn, T., Botelho, A. F. & Heffernan, N. T. (2017). Observing Personalizations in Learning: Identifying Heterogeneous Treatment Effects Using Causal Trees. *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, (L@S '17), 299–302. doi: 10.1145/3051457.3054009.

Zhang, L. & VanLehn, K. (2016). How do machine-generated questions compare to human-generated questions? *Research and Practice in Technology Enhanced Learning*, 11(1), 7. doi: 10.1186/s41039-016-0031-7.

Zhang, L. & Vanlehn, K. (2016). How do machine-generated questions compare to human-generated questions? *Research and Practice in Technology Enhanced Learning*, 11, 1–28. doi: 10.1186/s41039-016-0031-7.

Zhang, L. & VanLehn, K. (2017). Adaptively selecting biology questions generated from a semantic network. *Interactive Learning Environments*, 25(7), 828-846. doi: 10.1080/10494820.2016.1190939.

Zhang, Y., Paquette, L., Baker, R. S., Ocumpaugh, J., Bosch, N., Munshi, A. & Biswas, G. (2020). The Relationship between Confusion and Metacognitive Strategies in Betty's Brain.

Zhao, K., Huang, L. & Ma, M. (2016). Textual Entailment with Structured Attentions and Composition. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 2248–2258. Consulted at https:

//aclanthology.org/C16-1212.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A. & Fidler, S. (2015). Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 19-27. doi: 10.1109/ICCV.2015.11.

Appendices

APPENDIX A

PILOT PROGRAM DATA COLLECTION

This appendix describes the pilot program data collection conducted for the Korbit ensemble system to evaluate the performance of this ITS in parallel with the effect of the natural language generation system. Even Korbit has many pilot programs with many industrial and education partners. I would like to describe the pilot programs with FPT-Software (Fsoft) which I was involved 100% as the coordinated project manager. We run three pilot programs with Fsoft, one of the biggest Information Technology companies in Vietnam. The pilot program allows Fsoft employees to learn Data Science Courses on the Korbit platform, then ask them to score how they feel about the system. After three pilots for almost one year, we obtain more than 1000 participants for our experiment with the better feedback pilot by pilot. Our setup asks the different employees from Fsoft to study with and without the Korbit feedback (hints), or to study with another platform (Coursera), then evaluate the systems with the questionnaire tables. We use two ways to collect feedback data as follows:

- On the Korbit platform: we ask users feedback when they finish their module
- On the Typeform platform: to be fair with the other online learning platforms, we use Typeform to gather users' feedback.

A.1 Pilot 1 – Learn Data Science Skills with an AI

Fsoft and Korbit agreed to conduct a test pilot using Korbit's "Learn AI with an AI" course to upskill and reskill Fsoft employees on AI. Fsoft employees usually have a monetary training and development allocation (CAD 25) upon completing Udemy, Coursera, etc. Korbit was added to this list of options and was advertised on the internal Yammer network. The test pilot took place between September and December 2019. The following analysis is an evaluation of the results, an A/B test, the platform survey and written feedback provided by the Fsoft users on the platform.

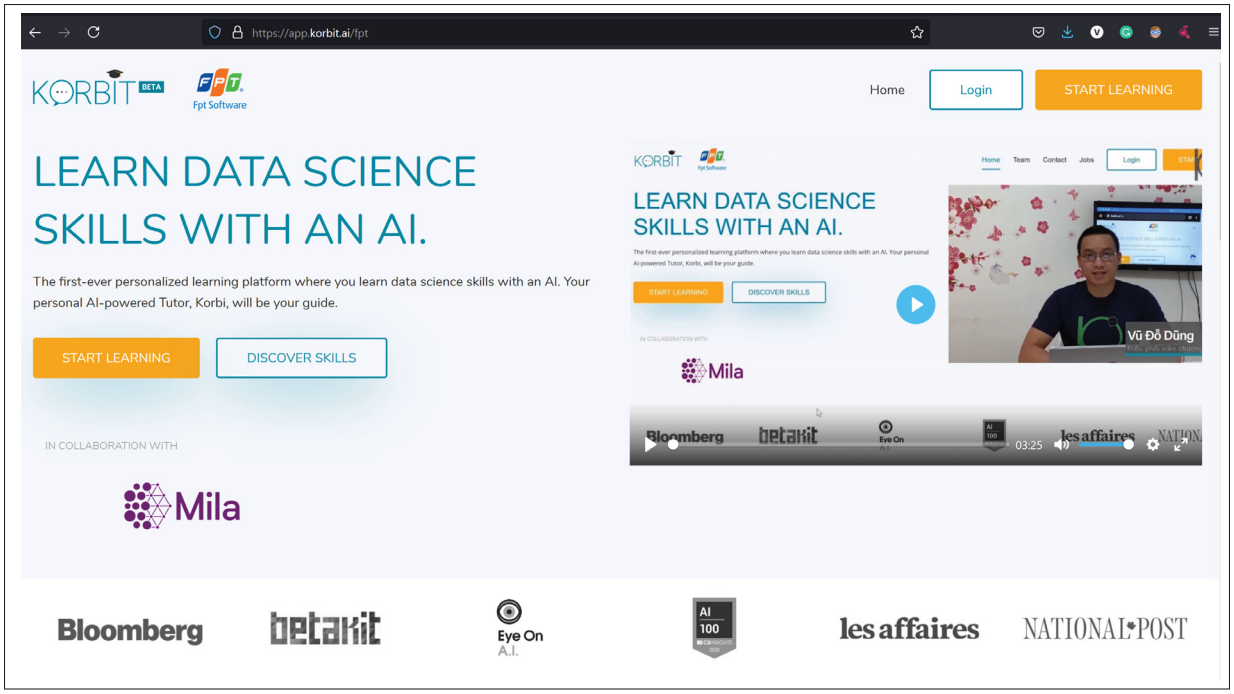

Figure 1.1 Learn Data Science Skills with an AI

A.2 Pilot 2 – Personalized Curriculum

Fsoft will distribute a link to the Korbit platform inside the company to have 2000 employees register and study data science and machine learning on Korbit. We have completed an initial test pilot with FPT, and more than 600 employees registered on the platform for the Learn Data Science skills with an AI course. Many employees skipped exercises and videos, and management finally decided to set a minimum of 45 exercises per employee. There was a lack of communication, and many employees were unaware of this requirement.

A.2.1 Experiment Flow

FPT pilot start (tentative): April 22, 2020 End: after 3 months

Total learning time (included quizzes/ assignments): 10.5 hours Users will be able to personalize their learning path (select skills)

- Pre-quiz: 15 minutes

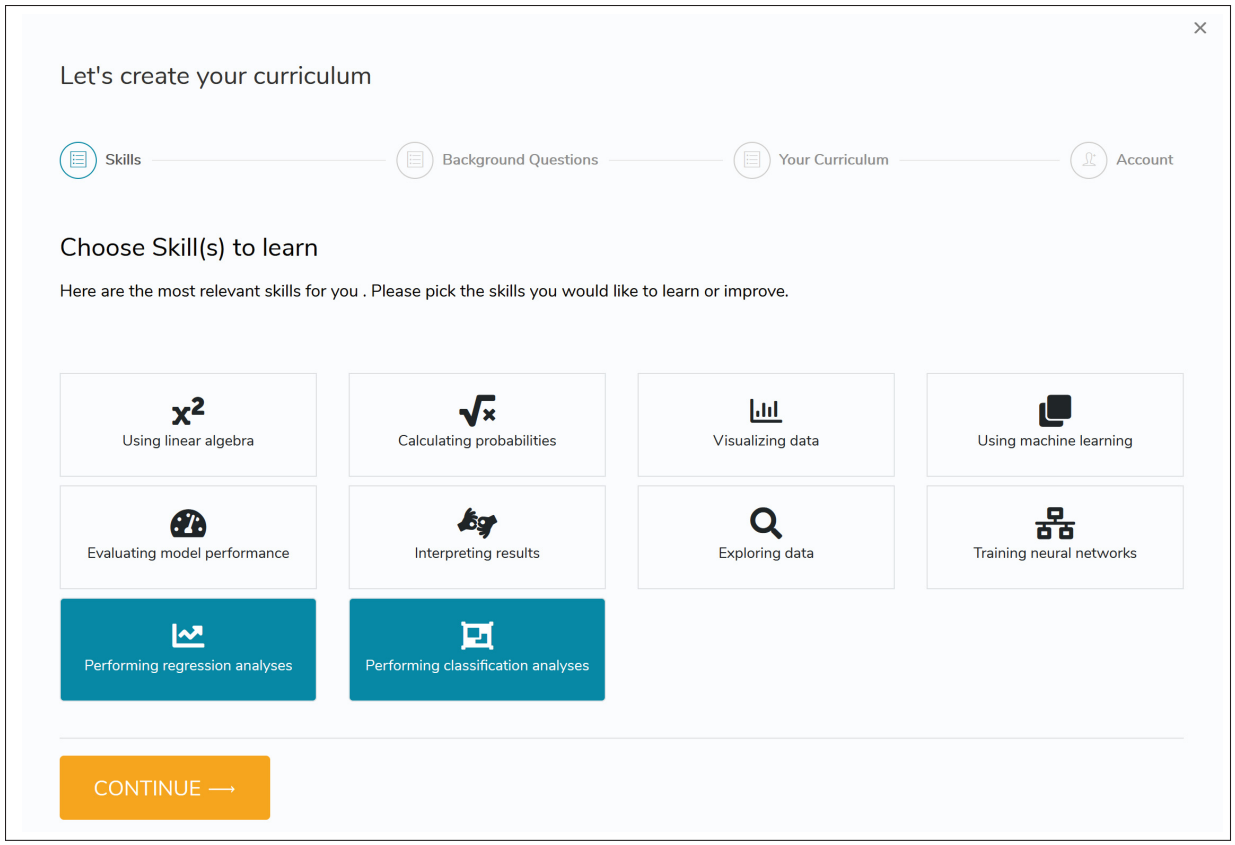

Figure 1.2 Personalized Curriculum

- Post-quiz: 30 minutes

for the multiple-choice questions and 120 minutes for the model applications

Course content, videos (see below, point 4)

Questions and answers (14 modules, 3 exercises/module, 5 minutes/exercise: 210 minutes) Total estimate: (content: 110 + 180 + questions and answers: 210 + pre- and post quizzes: 120 + 15 = 635 minutes 10.5 hours)

Course duration Depends on the skills users select (they can select one or more skills)

- Min: 30 minutes
- Max: 110 minutes

of video (new content) and 180 minutes of video (machine learning content)

Pass criteria (Grade scale: 10)

Diligence score: frequency of learning, time of interaction, questions and answers (20Users watch videos: Number of watched minutes/ total video minutes (minimum 50%)

Users interact with Korbi: questions/answers/hints: how many interactions does a student have with Korbit?

E.g., ten questions -> how many correct answers does a user have? Number of answers/Number of questions (minimum 50%)

E.g., ten questions -> how many interactions does a user have? Number of interactions/Number of questions (minimum 50%)

Knowledge score: post-quizzes (80%) : 8 MCQ

Theory post-quizzes: 60% MCQ

Programming quizzes (models): 40% - MCQ with given dataset (2000 exercises -> sample them)

Total score: Diligence score + Knowledge score (grade scale: 10)

<50%: failed (5) - nothing, ask to study again

>50%: passed (5) - certificate

>75%: excellent (7.5) - certificate + bonus (medals)

The way to get support directly from Korbit. Users: Email (vu@korbit.ai) TeamViewer (Remote) - configure, setup the learning (e.g., register users/passwords)

Access the admin panel (reports) - dashboard

A.3 Pilot 3 – Comparison performance of two online learning platforms for employee internal training

A.3.1 Experiment Flow

FPT pilot: April 2021

Send out a recruitment email with a link to the enrollment survey: https://korbitai.typeform.com/to/ICg5SAWn

Wait a few days for employees to respond.

When enough employees have responded, take the responses from TypeForm and filter out overqualified employees for the study (those who already have experience with AI/stats/math/data science/etc.)

Take the resulting list of employees from step 3 and send them an email asking them to confirm their participation in the study here: https://korbitai.typeform.com/to/gMZRLX65

Download the list of participants who confirmed their participation from TypeForm. This is now the list of employees who will participate in the study.

Randomly divide the list of participants into two groups. Group 1 (33% of participants) for Coursera. And Group 2 (66% of participants) for Korbit. The Korbit group will then randomly be assigned one of the two Korbit variants when they sign-up (with feedback or without feedback). So we will end up with 33% Coursera, 33% Korbit without feedback, 33% Korbit with Feedback.

On May 4th (ideally in the morning Vietnam time), send an email to Group 1 explaining that the study is starting today, and that includes a PDF of the Coursera instructions : https://docs.google.com/document/d/136EgvAxDtLkHBW_2jcj4X94IGV24zG0AXFWsQVKcXN4/edit

On May 4th (ideally in the morning Vietnam time), send an email to Group 2 explaining that the study is starting today, and that includes a PDF of the Korbit instructions : https://docs.google.com/document/d/1uaR3oFJJhvzfhla0JBxVwUoXCkAzbfHE-2moQpB7sDI/edit

Monitor the experiment and respond to questions and issues.

Participants have until May 9th before midnight (gmt+7) to complete the course. On May 9th, during the day (Montreal time), check that every participant from Group 1 has uploaded their "week 1 quiz answers" here: https://korbitai.typeform.com/to/mU5BHbYD

And that Group 2 participants have uploaded their completion certificate here: https://korbitai.typeform.com/to/m37HpsWr

Participants who don't upload those won't continue in the experiment.

From TypeForm, download the list of participants who have uploaded their "week 1 quiz answers" (Group 1) and their completion certificate (Group 2).

On May 10th in the morning (GMT+7), send an email to the participants from step 11 saying something to the effect of: "Congratulations on completing the course. Please complete the first final quiz before the end of the day. The second final quiz will take place on May 24th." with the link to the post-quiz: https://korbitai.typeform.com/to/iUzvSvoz

On May 24th in the morning (GMT+7), send an email to the participants who completed the final quiz, saying something to the effect of: "Congratulations on completing the first final quiz. Please complete the second final quiz before the end of the day. with the link to the delayed quiz:

https://korbitai.typeform.com/to/DFlMOkxT

On May 25th, send an email to the participants who completed the delayed quiz, saying something to the effect of "Thank you for completing the study. You will receive an official email from Korbit stating that you have completed the course and the quizzes. Don't forget to fill out the feedback form:

https://korbitai.typeform.com/to/OUaihHoo\T1\textquotedblright

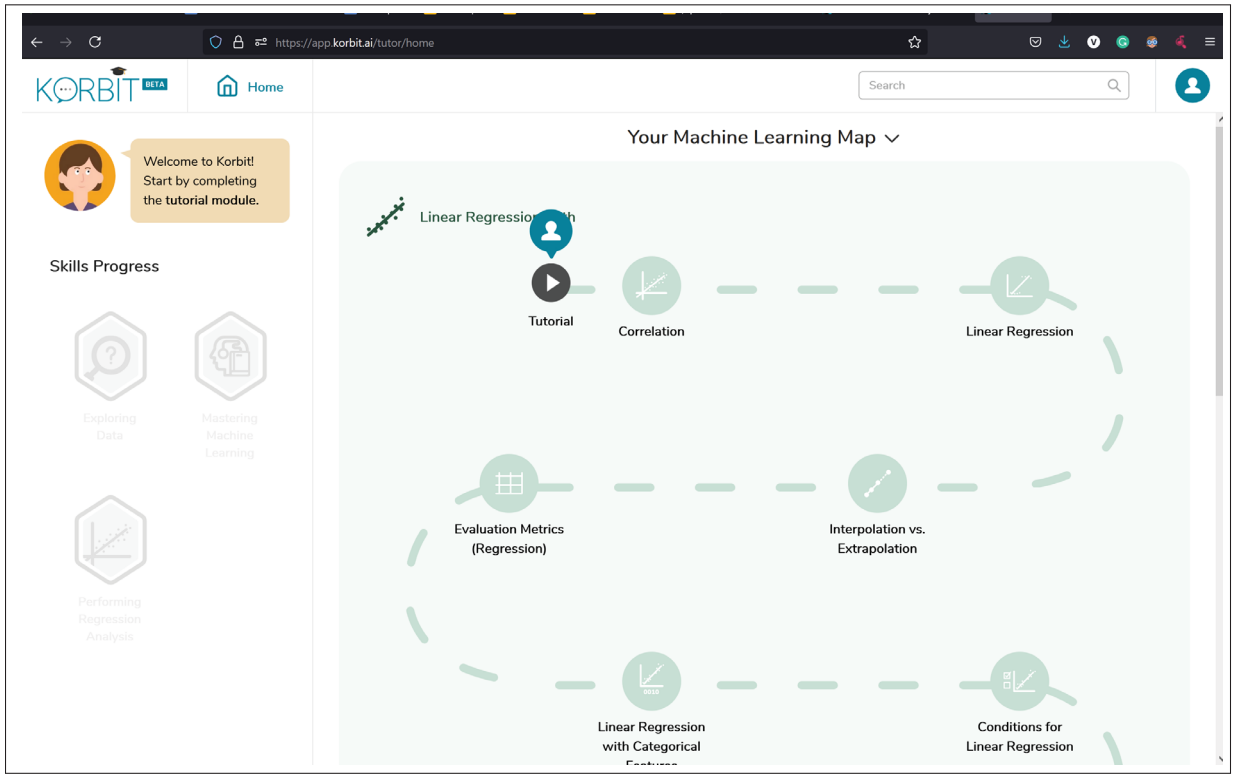

Figure 1.3 Korbit platform

A.3.2 Instruction Document of studying on Korbit

Please make sure to complete each step of this document in order. You will need about 4 hours in total to complete all the steps, which include a preliminary quiz (30 min), an online course on linear regression (2 hours), and a final quiz (1 hour 30 min) Course Subject: Linear Regression Platform: Korbit AI

Complete the preliminary quiz: Quiz link: https://www.korbit.ai/fpt-pre-quiz

Sign-up to Korbit. Go to https://app.korbit.ai/signup/curriculum/linear-regression And complete the sign-up process. You will automatically be assigned with the linear regression course. Create an account with your FPT email address. For a better experience on the platform, please use Chrome or Firefox.

Complete the course before May 23rd at 11:59 pm GMT+7. Get your completion certificate by completing each module on your learning path. While studying on Korbit, please avoid using any external help or resources (e.g., other websites).

Upload your completion certificate before May 23rd at 11:59 pm GMT+7 Upload link: https://www.korbit.ai/fpt-completion-certificate-upload

Complete the final quiz before May 24th at 11:59 pm GMT+7 If you've successfully completed steps 1 through 4, you will receive an email from Korbit on May 24th at 8:00 am GMT+7. This email will contain a link to the final quiz. This quiz contains a coding assessment. You will have 1 hour to complete the coding assessment part.

Fill out this feedback form: Link:https://www.korbit.ai/fpt-feedback-survey

Once step 6 is completed, you will receive a confirmation email from Korbit, attesting that you have completed the study.

A.3.3 Instruction Document of studying on Coursera

Please make sure to complete each step of this document in order. You will need about 4 hours in total to complete all the steps, which include a preliminary quiz (30 min), an online course on linear regression (2 hours), and a final quiz (1 hour 30 min) Course Subject: Linear Regression Platform: Coursera

Complete the preliminary quiz. Quiz link: https://www.korbit.ai/fpt-pre-quiz

Enroll in the Coursera course on linear regression and modeling. Choose the "Audit the course" option. Go to https://www.coursera.org/learn/linear-regression-model and click on the "Enroll for Free" button (ignore the date). Sign-up with the email address you provided us and click on "Join for Free".

Read and accept the consent form for this study. Consent form link: https://www.korbit.ai/fpt-study-consent

Complete only week 1 of the course before May 23rd at 11:59 pm GMT+7. You must:

- Watch all week 1 video
- Complete all the week 1 reading
- Complete and pass the week 1 practice quiz
- Complete the week 1 quiz (no need to submit it on Coursera)
- Avoid using any external help or resources (e.g., other websites)

Upload a screenshot of your week 1 practice quiz result. Upload the screenshot before May 23rd at 11:59 pm GMT+7. Upload link: https://www.korbit.ai/fpt-mooc-week-one-practice-quiz Please make sure your name and quiz grade are visible (see format below).

Complete the final quiz before May 24th 11:59 pm GMT+7. If you've completed steps 1 through 5, you will receive an email from Korbit on May 24th at 8:00 am GMT+7. This email will contain a link to the final quiz.

This quiz contains a coding assessment. You will have 1 hour to complete the coding assessment part.

Fill out this feedback form: https://www.korbit.ai/fpt-feedback-survey

Once step 7 is completed, you will receive a confirmation email from Korbit, attesting that you have completed the study.