Question classification via machine learning techniques

Ho, Mun Kit

2020

Ho, M. K. (2020). Question classification via machine learning techniques. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/145449>

<https://doi.org/10.32657/10356/145449>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 18:15:05 SGT

Question Classification via Machine Learning Techniques

Ho Mun Kit

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Master of Engineering

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

02-08-20

.....

Date

.....

Ho Mun Kit

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

02-08-20

.....

Date

Andy Khong

.....

A/P Andy W. H. Khong

Authorship Attribution Statement

This thesis contains material from a paper accepted at one conference in which I am listed as an author.

Chapter 4 is published as Mun Kit Ho, Sivanagaraja Tatinati, Andy W. H. Khong, “A Hierarchical Architecture for Question Quality in Community Question Answering Sites,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2020.

The contributions of the co-authors are as follows:

- A/Prof Khong provided the inspiration for this research direction and edited the manuscript draft.
- Dr. S. Tatinati provided valuable advice on the comparison analysis against other baseline algorithms and edited the manuscript draft.
- I came up with the idea. The architecture was realized and coded by myself. I also conducted the experiments and prepared the manuscript draft.

02-08-20

.....

Date

.....

Ho Mun Kit

Acknowledgments

I wish to express sincere appreciation to my supervisor, Assoc. Prof. Andy W. H. Khong, who has been very kind in guiding me through the research for this thesis. He has been patient in walking me through the development of all research ideas, while being methodological in his feedback to develop my research skills that will last a lifetime. Without his words of wisdom, this journey would have been way more challenging.

I would also like to pay special regards to our postdoctoral researcher, Dr. Sivanagaraja Tatinati, who provided helpful insights in our technical discussions and shared valuable lessons from his experiences. This made our publication process a breeze. In addition, I would like to express my gratitude to my teammates Kelvin Ng Hongrui, Liu Kai, S. Supraja, Cao Zhen, Darin Tao Liran, Tan Zhi Wei, Dr. Nguyen Quang Hanh, Nguyen Hai Trieu Anh and Qiu Wei. It has been a wonderful experience working with everyone in our research projects, where we inspired and motivated one another either through rigorous technical brainstorming sessions or simply casual conversations.

Last but not least, I wish to acknowledge the unwavering support and love from my beloved partner, Poh Huey Ching; my father, Ho Peng Kin; my mother, Kwan Lai Kuen; and my siblings, Ho Mun Khar and Ho Mun Han. Thanks for always believing in me and supporting me along the paths that I take.

“A prudent question is one-half of wisdom.”

—Francis Bacon

To my dear family

Contents

Statement of Originality	iii
Supervisor Declaration Statement	v
Authorship Attribution Statement	vii
Acknowledgments	ix
Summary	xv
List of Figures	xvi
List of Tables	xix
Symbols and Acronyms	xxi
1 Introduction	1
1.1 Motivation	2
1.2 Organization of the thesis	4
1.3 Contributions of the thesis	5
2 Literature Review	7
2.1 Taxonomies for question classification	7
2.1.1 Classification of assessment questions in terms of cognitive levels	8
2.1.2 Classification of user-generated questions in terms of quality	9
2.1.2.1 Noise in user-generated questions	10
2.2 Feature extraction for question classification	12
2.2.1 Feature engineering for machine learning algorithms	13
2.2.2 Topic models	15
2.2.3 Neural networks	17
2.2.3.1 Distributional semantics in neural language models	18
2.2.3.2 Sequence encoder	20
2.2.3.3 Neural networks for question classification/quality .	23
2.3 Chapter summary	24

3	Classification of Questions by Cognitive Complexity	27
3.1	Question classification using bi-directional GRU and attention mechanism	28
3.2	Dataset	31
3.3	Experiment setup	32
3.4	Results	34
3.4.1	Comparison analysis	34
3.4.2	Qualitative analysis	35
3.5	Quiz generation system (QGS)	37
3.6	Chapter summary	39
4	Classification of Question Quality in Learner Questions	41
4.1	The proposed tHAN architecture	42
4.1.1	The proposed two-stage hierarchical attention net- work (HAN) with topic-weighted attention (TwAtt)	43
4.1.2	Sentence importance selection	46
4.2	Dataset	47
4.3	Experiment setup	49
4.4	Evaluation	50
4.5	Ablation analysis	53
4.6	Chapter summary	55
5	Specificity for Classifying Question Quality	57
5.1	Introduction	58
5.2	Proposed method	59
5.2.1	Software-specific NER (SNER)	59
5.2.2	s-tHAN	60
5.3	Dataset and pre-processing	63
5.3.1	Dataset	63
5.3.2	Pre-processing	65
5.4	Experiment setup	66
5.5	Results and discussion	67
5.5.1	NER tagging performance	67
5.5.2	QQ classification comparison analysis	69
5.5.3	Feature ablation	71
5.6	Chapter summary	73
6	Conclusion and Recommendations	75
6.1	Conclusion	75
6.2	Recommendations for future research	77
	List of Author's Awards, Patents, and Publications	79
	Bibliography	81

Summary

Questions are indispensable tools in our daily communication and for the process of acquiring information and knowledge. Recent developments in technology and the internet has also brought about many social sites where community members engage in knowledge-building discussions. These technologies have also been translated to online-learning platforms, and increasingly, these have become scalable tools where students across the globe interact and learn. Understanding the cognitive complexities and quality of questions in such learning settings provide additional insights for educators to monitor achievement of learning outcomes and administer intervention when required. This thesis therefore aims to propose automated solutions using machine learning methods to address this pedagogical need.

Questions in online-learning platforms are commonly found in assessments authored by instructors to assess learners' understanding on the subject. As online-learning platform scales up, it becomes increasingly laborious to manually create assessments comprising questions of various difficulties for students. However, existing question classification models are limited in terms of modeling semantics. Labeling assessment questions by cognitive complexity not only involves the detection of keywords that discriminate between complexities, but also requires consideration of contextual semantic features. A neural network-based machine-learning model is proposed with attention mechanism to direct the creation of a question representation for this purpose. Experiments on university-level digital signal processing questions demonstrate improved performance against other keyword feature machine learning models when detecting patterns resembling Bloom's taxonomy learning outcome templates. In addition, the proposed classifier is integrated into a web-based quiz generation system to support retrieval practice among students with a desired mixture of questions at different complexity levels.

User-generated questions have, on the other hand, become increasingly popular on social media sites for inquiring about specific knowledge outside academic

settings. These questions, as opposed to assessment questions, are authored casually, which are error-prone and usually not as sophisticated. To overcome problems of noise such as misspellings, it is important to progressively interpret the question by filtering out the noise and pick out only the salient features. This is achieved via a hierarchical architecture with a new topic-weighted attention mechanism that provides context-aware attention on the question. Furthermore, the proposed approach performs well in the chosen evaluation metrics against other baseline models without assistance from community features. The efficacy of this approach is verified on the Stack Overflow questions dataset. This approach is found to be effective at finding contextual information in the sub-divided texts to form an effective overall representation.

Studies on human-authored texts have found that specific information included in a piece of text improve comprehension. In education and on websites, this helps to increase the overall quality of information being communicated. In the previous model, the attention scheme was data-driven and may not make use of granular entities for extracting features. Using entity embeddings from a named-entity recognizer, the markers give hints to the attention to focus the feature extraction around the entities, thus enhancing performance in its discrimination of *very good* vs *bad* questions. Results on the Stack Overflow question dataset indicate that the tag embeddings enhanced its performance over the predecessor, especially with finer categories of tags used, instead of binary indicators. The entity tags were shown to work well with the proposed topic-weighted attention mechanism, thus creating a structural bias to focus on specificity-related features at these crucial locations.

List of Figures

2.1	Attention distribution while reading a question snippet as reported by a volunteer.	11
2.2	Steps in text classification.	12
2.3	Single artificial neuron.	17
2.4	Prediction of surrounding words using center word via a window size $E = 3$ with a skip-gram model of <i>Word2vec</i> . The variables i and o denote, respectively, the input and output words within the window.	18
2.5	Architecture of an unfolded recurrent neural network.	20
2.6	Sequence learning using a GRU.	22
3.1	Flowchart of variables in training a bi-directional GRU classifier.	29
3.2	Attention visualizations of three exemplar DSP questions. The color depth is proportional to the attention neuron activation a_j during inference of the question's label $\hat{y}$. Contiguous dark spots indicate important segments for the class label.	35
3.3	Schematic diagram of the proposed quiz generation system (QGS).	38
4.1	Sentences identified as highly-discriminative by the proposed model and how its predicted label compares against human and true labels. (left) Example of a <i>very good</i> question, and (right) Example of a <i>bad</i> question.	42
4.2	Architecture of proposed tHAN network.	44
4.3	Topic-weighted attention (TwAtt) mechanism.	45
4.4	Effect of number of trained topics on F1 (%) for HAN and tHAN.	53
5.1	A CRF infers NE tag at each step with the highest probability (in red) by using features extracted from the input sequence of words. The full list of NE tags is given in Section 5.3.1. Words marked with tags other than 'O' indicate mentions of entities, e.g., PyPy (API) and PostgreSQL (Framework).	59
5.2	Architecture of the proposed s-tHAN, of which tHAN from Section 4.1 is highlighted in gray.	60
5.3	Post-processing is applied after NER tagging to mimic the tokenizer in tHAN.	61
5.4	Confusion matrix of entity tags by SNER.	68

- 5.5 Comparison of attention patterns at the sentence attention and *TwAtt* between 3 models for a *very good* Stack Overflow question. Darker squares indicate higher attention activations. 72

List of Tables

2.1	Example of questions under Bloom’s Taxonomy.	9
2.2	Features for text classification.	13
3.1	Dataset statistics of each complexity class.	31
3.2	Classification performance on DSP question dataset. Scores are expressed in percentages (%).	34
4.1	Comparison analysis for all three quality classes. Metrics are expressed in percentages (%).	51
4.2	Top topics learned using the best model.	55
5.1	Statistics of tags	64
5.2	QQ classification results comparing s-tHAN against other baseline models. Metrics are expressed in percentages (%).	69
5.3	QQ results when a subset of NE tags or when TwAtt mechanism are removed from the s-tHAN ₆ model. Metrics are expressed in percentages (%).	71

Symbols and Acronyms

Symbols

$\mathbf{x}$	generic input vector to model
y	true class label
$\hat{y}$	predicted class label
Q	a sequence of sentences, i.e., a single question data sample
$\mathbf{q}$	question vector
S	a sequence of word tokens, i.e., a sentence
$\mathbf{s}$	sentence vector
$\tilde{w}$	a scalar word token
$\mathbf{w}$	word embedding vector
v	a scalar topic word
$\mathbf{v}$	topic embedding vector
t	a scalar named-entity tag
$\mathbf{t}$	named-entity tag embedding vector
$\mathbf{u}$	attention parameter vector
$\mathbf{W}$	parameter matrix for linear transformation of model inputs
$\boldsymbol{\theta}$	model parameters
$\mathbf{h}$	hidden state vector of an RNN
b	bias scalar
$\mathbf{b}$	bias vector
$[\cdot; \cdot]$	concatenation operator
σ	activation function
$\odot$	Hadamard product
$\cdot$	dot product
$\times$	Cartesian product
i	index for i -th sentence for models with sentence-level inputs

j	index for j -th word
k	index for k -th topic word

Acronyms

API	Application programming interface
CBOW	Continuous bag-of-words
CNN	Convolutional neural network
CQA	Community question answering
CRF	Conditional random field
DFT	Discrete Fourier transform
DSE	Domain-specific embeddings
DSP	Digital signal processing
DTFT	Discrete-time Fourier transform
Fram	Tool-library-framework
GRU	Gated recurrent unit
GTSM	Global topic structure model
HAN	Hierarchical attention network
i.i.d.	Independent and identically distributed
LDA	Latent Dirichlet allocation
LSA	Latent semantic analysis
LSTM	Long short-term memory
MOOC	Massive open online courses
NER	Named-entity recognition/recognizer
NLP	Natural language processing
Plat	Platform
POS	Parts-of-speech
PL	Programming language
QC	Question classification
QGS	Quiz generation system
QQ	Question quality
RNN	Recurrent neural network
SNER	Software-specific named-entity recognizer

SOLO	Structure of observed learning outcomes
Stan	Software standards
s-tHAN	Specificity-enhanced topical hierarchical attention network
SVM	Support vector machine
tHAN	Topical hierarchical attention network
TREC	Text retrieval conference
TwAtt	Topic-weighted attention mechanism
VLE	Virtual learning environment
WL	Weighted cross-entropy loss

Chapter 1

Introduction

Recent developments in information communication technology has transformed our concept of learning and communication of knowledge. Virtual learning environments (VLEs) of education institutions have been developed to host a variety of rich content, and these sophisticated sites are continuously expanding in terms of the suite of tools on these platforms (such as forums, wikis and chat-rooms) to enhance learning effectiveness. These platforms enable learners across the globe to engage in learning discourse in the virtual space, effectively scaling up education. On the other hand, social question-answering websites gather users and experts to build upon highly-specific procedural knowledge. The knowledge-building and acquisition scenarios above share some commonalities. Firstly, users on these sites share a common quest for knowledge. Secondly and more importantly, the process is initiated with the use of questions, e.g., starting a discussion thread or solving an assignment. Since questions are important in determining the quality of subsequent interactions, this thesis focuses on the development of machine learning algorithms to classify questions (based on cognitive levels and quality) in knowledge-based applications.

1.1 Motivation

Interrogatives, or questions, are fundamental instruments for communication, and is particularly important for knowledge acquisition [1]. As defined in [2], a question can broadly be defined as “a speech act that is either an inquiry, an interrogative expression, i.e., an utterance that would be followed by a question in print, or both”. Both written styles of “What is factorial design?” and “Tell me what factorial design is” are considered as questions. Questions can originate from both educators and learners but the generation mechanisms and their intent differ.

In pedagogical research, it has been found that questions play a central role in student learning. From the educators’ perspective, assessment questions are critical to evaluate a student’s degree of mastery on a subject matter. By formulating questions in different complexity levels, educators may gain valuable insights into a student’s achievement with respect to the intended learning outcomes based on the questions they are capable of answering. Generating an appropriate mix of question complexities according to the students’ capabilities is one of the key goals of adaptive learning technologies.

On the other hand, questions are also raised by students during instruction and have been found useful to probe into their learning process. It has been found that the act of question generation involves a deep cognitive process, because this operates at a fundamental level that requires the comprehension of text and social action, learning of complex material, problem solving and creativity. Higher quality questions are often characterized by those that involve inferences, multi-step reasoning, and also high specification. For example, a short yes-no question such as “*Is the answer 5?*” constitutes an intent of answer verification from a surface learning approach; whereas those that involve complex inference skills, while providing some contextual information, such as “*What happens when the temperature*

of ice decreases?” are considered better questions that involve deeper comprehension prior to question-posing. Therefore, examining quality of these questions will provide insight into the students’ existing conceptual understanding [3]. There is also empirical evidence that supports the view that training students to ask good questions improves comprehension, learning and memory [4].

Due to the significant role of questions in examining the cognitive processes in learning, an automated mechanism is ideal for evaluating the quality of questions posed by learners as part of the development effort of scalable virtual learning tools. With this additional insight, instructors can administer suitable interventions for learners to better achieve intended learning outcomes. Nonetheless, since instructor-generated assessments and student-initiated questions involve different surface characteristics and intents, the analysis of text will require different approaches.

Recently, the growth of machine learning algorithms in the area of natural language processing (NLP) has enabled machines to comprehend human texts and provide assistance in many applications. Usefulness of these tools led to increasing adoption of analytics and artificial intelligence (AI) technologies into learning tools to aid learners [5]. It is important to note that, questions differ from conventional declarative texts semantically since there is a gap in information. Moreover, being composed of fewer tokens makes analysis challenging due to the lexical gap and the limited context in the text. This is a result from the presupposed background knowledge between the asker and the answerer, which most likely originates from external knowledge source (e.g., textbook or article), but are not explicitly provided. Furthermore, commercial technologies encounter challenges when being directly applied to the educational context. Challenges such as relevance to education objective, user-experience that concerns a pedagogical workflow, require researchers to take special considerations into account in order to create the product. The development process of such tools should therefore undergo careful iterative

developments to ensure that the technology meets the pedagogical need that is consistent with relevant intervention theories [6].

Individuals regularly engage in knowledge-seeking activities in the virtual space for problem-solving, thus taking on the role of learners not only on VLEs, but also on social networking sites. By evaluating the quality and subsequently encouraging them to produce better ones, products from the subsequent knowledge-building interactions e.g. discussion thread or achievement of learning outcomes will greatly be enhanced. This thesis will focus on addressing the above-mentioned challenges of classifying question quality in the context of knowledge-based interactions by applying machine learning methods.

1.2 Organization of the thesis

This thesis addresses the problem of automatically classifying question quality in the communication of knowledge. This is performed on two sources, namely assessment questions authored by subject matter experts, and user-generated questions commonly found in question-answering websites or discussion forums.

Chapter 2 reviews existing literature on question quality and the broader field of question classification. The taxonomies of questions are introduced, together with the underlying learning theories that supports the segregation between these labels. In addition, feature extraction techniques and machine learning algorithms are introduced to provide a technical foundation for the proposed methods in subsequent chapters. A survey on existing works in the application areas is also presented.

Chapter 3 presents a neural model that interprets short assessment questions and thereafter predicts the quality label. The question could be given concisely due to common background knowledge for the course, which relies on prescribed

study materials. These technical engineering questions are authored by educators and are therefore usually error-free.

The research problem is then switched to user-generated questions in Chapter 4. As novices, questions generated by students usually contain noise and errors which prohibit existing methods from efficiently interpreting the question. These questions are also generally longer to provide more context. The problem of determining question quality is addressed via a hierarchical architecture with a new attention mechanism. Its performance is validated with experiments performed on community question and answering (CQA) questions.

As previously highlighted, higher quality student questions typically involve higher degree of specificity using elements of subject matter. In Chapter 5, the hierarchical architecture is extended to incorporate specification features. Inspired from works from aspect-level sentiment analysis, the proposed method leverages on the use of named-entities that can be identified with common semantic extraction tools that is commonly found in each expert domain.

Chapter 6 concludes the thesis and proposes several future research directions.

1.3 Contributions of the thesis

This section summarizes the contributions made by the author as described in Chapter 3, Chapter 4 and Chapter 5.

In Chapter 3, a sequential neural network encoder with attention is proposed for classifying questions that are labeled with Bloom’s taxonomy. In addition, a question generation system is developed around the classifier backend to generate assessment questions for learners’ retrieval practice. This system promotes learners’ engagement and scaffold their individual practice sessions with questions

categorized in accordance to the cognitive complexity required to solve them. The system is deployed for a university-level digital signal processing (DSP) course and feedback from the instructor indicates that the proposed system is effective in aiding learners comprehend high-order DSP concepts by abstracting the relationship between the generated questions at various complexity levels.

User-generated questions in forums and CQA sites are often noisy. For quality classification under such cases, a hierarchical architecture is proposed in Chapter 4 to address the problem. This architecture avoids the use of social network indicators as features by predicting quality via a full semantic evaluation on the text. The proposed architecture employs a new attention mechanism that extracts meaningful features from the noisy question text at different granularities while filtering redundant information for such classification tasks. The efficacy of this network is validated on the Stack Overflow dataset. This work has been published in the paper Mun Kit Ho, Sivanagaraja Tatinati, Andy W. H. Khong, “A Hierarchical Architecture for Question Quality in Community Question Answering Sites,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2020.

In Chapter 5, specificity is incorporated to the hierarchical network to enhance quality classification. Question texts are first annotated via a software-specific named-entity recognition (NER) tagger to mark the relevant entities. Experiments show that when all six entity categories are used, the entity embeddings guide the attention mechanisms so that key sentence segments are utilized for feature extraction. Inclusion of these embeddings therefore indicates the degree of specification for the supporting points in the question. Hence this approach also successfully demonstrates that the utility of lower-level semantic extraction tools can yield significant improvements for the higher-level tasks, i.e., question quality (QQ) classification.

Chapter 2

Literature Review

Questions play a central role in human discourse, and more so in education instruction. In this thesis, questions are analyzed by applying state-of-the-art natural language processing (NLP) techniques to classify questions. As a string of text shorter than generic documents, questions contain less information content [7–10] as they serve to only address an information gap with presupposed background facts. Hence, designing a model that automatically selects salient features within the text, and the transfer of prior information from external sources are key for this task. This chapter surveys the latest developments of the techniques used in question classification (QC), particularly those generated in knowledge-building activities.

2.1 Taxonomies for question classification

Question quality (QQ) evaluation has gained interest from parties for different purposes. These include, but not limited to, evaluating question strings from automated question generation systems [11], education [2, 12], evaluating quality

of user-generated questions on community question answering (CQA) sites and responses from autonomous dialogue agents [13]. Closely related to this is the task of QC.

Early emerging works in QC follow the taxonomy proposed by Li and Roth [7] on TREC questions [14]. The main purpose of this task is to enable downstream automatic question-answering systems to constrain answers to only the labeled answer types e.g. LOCATION, HUMAN, ENTITY [15]. On the other hand, education researchers found utility in classifying questions according to expected performance from the student with respect to a set of defined learning outcomes. In this thesis, the application of QC in education takes the latter definition. This correlates with a mixture of the question difficulty and the degree of cognitive complexity involved in generating or answering the question. In creating assessment questions, subject matter experts (cf. Section 2.1.1) undergo a different generation mechanism compared to that when a non-expert user asks a question (cf. Section 2.1.2). Hence, the concept of quality is further sub-divided by source.

2.1.1 Classification of assessment questions in terms of cognitive levels

Under this labeling scheme, questions are generated by experts. Deeper questions involve higher cognitive processing such as multi-step reasoning and domain-transfer skills in order to provide a corresponding answer. These questions are generally perceived as more challenging and require greater mastery in the subject matter. Two major schemes to measure competency in this area are the structure of observed learning outcomes (SOLO) [16] and Bloom’s taxonomy [17]. In these works, labeling by Bloom’s taxonomy is favored over SOLO since the verb-centric categorization scheme corresponds well with observable learning outcomes

Category	Example
Knowledge	<u>Define</u> the concept of inheritance
Comprehension	<u>Explain</u> the structure of a method in a program
Application	<u>Demonstrate</u> the relationship between packages, classes and methods
Analysis	<u>List</u> the advantages of using a container-type class
Synthesis	<u>Create</u> a Java program showing the concept of overload
Evaluation	<u>Justify</u> the concept of inheritance and write a sample source code

TABLE 2.1: Example of questions under Bloom’s Taxonomy.

from the student. Examples corresponding to each level of Bloom’s taxonomy are shown in Table 2.1.

Automating the labeling of question complexities can assist examiners in striking a balance between progressive amount of questions that assess basic versus those that assess higher levels of learning. By organizing questions into varying complexities, an examiner is able to gain insights into the student’s performance and administer suitable interventions. Works have been done in this area to evaluate questions from higher education courses, predominantly in engineering and sciences [18–20].

2.1.2 Classification of user-generated questions in terms of quality

As opposed to classifying assessment questions based on *cognitive levels*, the *quality* of student-initiated questions are generally evaluated qualitatively by researchers by studying classroom interactions [2, 4]. These conversational questions, however, were not collected to train an automated QC machine to achieve the same outcome. A relevant work in this area is applied on the iSTART reading comprehension system [21] where questions were labeled according to the scheme in [22].

In the study, users were instructed to pose a relevant question after reading news article to evaluate their comprehension [12, 23].

In addition to the above, CQA sites have recently become popular platforms for knowledge-seeking activities. These websites encourage community members to post high-quality questions that hold long-term value. Using Stack Overflow as an example, the community guidelines consider a question as good if it is clearly-written and describes a specific answerable programming problem. In constructing their textual content, all users can utilize fields provided— title, body and tags to describe their problems or solutions concisely to capture attention of other users for the purpose of providing feedback or solutions. Since these questions are generated by non-experts and also resembles forum discussions on VLEs, the question texts can be evaluated as learner-initiated questions, using degree of positive votes as an indication of quality. It is useful to note that, the definition of QQ varies across works as each adopts an arbitrary measure. For instance, Ravi et al. [24] defined QQ as a ratio of average score awarded by voters to the number of votes obtained for that question. Ponzanelli et al. [25] and Hodgins [26], on the other hand, divide the quality into $\{very\ good, good, bad, very\ bad\}$ classes using arbitrary thresholds on votes and rules, while Zheng et al. [27] defined quality as a real-valued score as a function of total votes, answers and views.

2.1.2.1 Noise in user-generated questions

User-generated texts especially those found in CQA sites and forums typically contain a lot of noise arising from the process of organizing their thoughts and their own knowledge. Such variation across askers and within each asker presents challenges for existing NLP methods. This issue is observed in forums ranging from generic forums such as Qatar Living [28] to highly technical ones such as Stack Overflow. Researchers observe that users do not follow linguistic rules strictly

Title: passing default parameter value vs not passing parameter at all?
Body: (s1) here 's what i want to do given a table xxcode and a stored procedure xxcode is there a way ...
(s2) in other words i want to tell within the stored procedure if the default value was used because it was ...

FIGURE 2.1: Attention distribution while reading a question snippet as reported by a volunteer.

and spelling mistakes are widespread for infrequent technical vocabulary [29]. For example, “JavaScript” is often misspelled as “javasript”. A straightforward solution is to employ rule-based or dictionary-lookup spellcheckers to fix these errors as a pre-processing step before analysis. However the problem is exacerbated by the large number of words falling into the heavy-tailed distribution of technical texts, especially in knowledge discussion forums that involves software, sciences or medicine. In the context of Stack Overflow, several factors cause the determination of QQ challenging compared to other domains. For instance, some software entities take the form of common words e.g. Java, Python. In addition, users can define custom entities with identical lexical and syntactic formats as popular libraries resulting in the introduction of polysemy. Lastly, the informal nature introduces many variations for the same entity. For instance, JavaScript can be expressed as js, JS or javascript.

In knowledge-based technical discussions, tools employed for English texts and NLP tools for generic texts will, therefore, require adaptations to address the above challenges. Firstly, it is beneficial to pre-train a language model on domain-specific corpora to learn the constrained semantic features and avoid confusion with other English texts seldom found in the same context. For example, by learning the semantics of ‘Java’ from Stack Overflow texts, it will be less likely to be confused with ‘Java coffee’. This will also increase vocabulary coverage, thus reducing out-of-vocabulary words. This approach has been adopted by language models of domain-specific education and professional texts, giving rise to Edu2Vec [30], BioBERT [31]

FIGURE 2.2: Steps in text classification.

etc., which have benefited downstream tasks.

Secondly, some structural modifications can be added to existing models to create inductive biases to patterns. This enables the structures to selectively extract features conditional on the context while discarding the noise highlighted above, such that selection of features can be performed by mimicking human attention as shown in Fig. 2.1.

2.2 Feature extraction for question classification

The task of QC is a restricted sub-task within a broader scope of document classification. The steps taken to build a question classifier is identical to that in a text classification workflow [32], as shown in Fig. 2.2. The question string first undergoes pre-processing that typically involves the removal of stopwords and special characters, e.g. mathematical symbols and diagrams. Representative lexical, syntactic or semantic features can then be engineered for each class. Optionally, each feature can be given different priorities using selection techniques. Finally, a rule-based [33] or machine learning algorithm is applied to compute a label for the question string. The model's performance can then be evaluated on test data using metrics such as accuracy or the F1 score.

Text features	Examples
Lexical Syntactic Semantic	bag of words, n -grams, WordNet synsets [34] POS tags, constituency parse tree Dependency parse tree, neural-based sentence embeddings, LDA topics

TABLE 2.2: Features for text classification.

2.2.1 Feature engineering for machine learning algorithms

Machine learning approaches are generally preferred over time-consuming process of figuring out rule-based heuristics for QC. Given the availability of training data nowadays, a high-performance classification program that leverages on discriminative features between classes in the order of thousands can be constructed. However, the design of appropriate features for the task may still require domain-specific expertise in order to be effective. This is formally described as follows: Suppose a vector of input features $\mathbf{x}$ of size M is computed to represent each question Q . A machine learning algorithm serves to determine a hypothesis function $f(\cdot)$ parameterized by $\boldsymbol{\theta}$. Given a true class label y , the label for Q is computed via

$$\hat{y} = f(x_1, x_2, \dots, x_M; \boldsymbol{\theta}).$$

where $\hat{y}$ is defined as the model output. The parameters are then optimized against a given objective function L that compares $\hat{y}$ against the true label y in order to obtain the optimal set of parameters $\boldsymbol{\theta}^*$ for the final trained model, i.e.,

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} L(\hat{y}, y).$$

As seen above, selection of features $\mathbf{x} = (x_1, x_2, \dots, x_M)$ is paramount for obtaining the best performance out of machine learning models. Text-based features used for text classification can broadly be categorized as lexical, syntactic and semantic features, examples of which are shown in Table 2.2. The most widely-used

bag-of-words (BoW) representation assumes the lexical tokens are independent. Contiguous tokens can form collocations in the form of n -grams to add towards the BoW features. In addition, a feature selection procedure can assign weights to the word features using statistical methods such as term frequency-inverse document frequency (tf-idf) [35]. While the computation of these statistical lexical features are straightforward, they fail to model context since they neglect dependencies between expressions, which can be better achieved with syntactic and semantic features.

Furthermore, syntactic features in the question can be exploited to enhance the word features. Haris and Omar [19] employed parts of speech (POS) tag templates which were mined from observations to predict the complexity of questions in Bloom’s taxonomy. Zhang and Lee [8] used tree kernels in conjunction with support vector machines to classify questions. This helps to identify weights of the tree fragments based on their depth while looking for question’s focus. Other complex semantic features are elaborated in Section 2.2.2 and Section 2.2.3.

Non-text features can also be employed by leveraging on expertise from other domains. These features include, for example, Linguistic Information and Word Count (LIWC) tool [36] and readability [37], which were subsequently fed to a shallow classification algorithm such as logistic regression, support vector machine (SVM) [38], or random forests [26].

Expert-advised social network features can also be used to model the relationships between user profile information and the quality of questions they produce. Agichtein et al. [39] explored the use of link-analysis features for QQ on Yahoo Answers, i.e., user-item interaction features based on HITS [40] and PageRank [41]. This is based on the assumption that good answerers consistently generate good content. Combined with text linguistic qualities, usage statistics, and graph-based

interaction relationships between user-items, this system then employs stochastic gradient boosted trees to predict *good/bad* questions. Li et al. [42] modeled dependency relationships between user and question items with bipartite graph. Using both question-related and asker-related features in both node groups, the final question qualities and asker expertise are estimated using mutual reinforcement label propagation algorithm. Baltadzhieva and Chrupała [43] explored the use of syntactic features of the layout of Stack Overflow questions via the use of title, body, code snippets and tags, with user reputations. By analyzing coefficients from their ridge regression model, certain text surface patterns and the presence of code snippets are found to be important for determining QQ on Stack Overflow.

2.2.2 Topic models

Initially posed as a method to obtain a low-rank matrix on the document-term matrix, latent semantic analysis (LSA) [44] used singular vector decomposition to produce salient vector representations for word and documents. Later, strong relationships were discovered to occur between words and documents, giving a notion of topics. Taking these further, a hidden distribution of topics is assumed to be present between observed documents and words. By modeling this relationship, probabilistic-latent semantic indexing (pLSI) [45] produces a probabilistic distribution of topics for each document, which then gained widespread adoption with the introduction of latent Dirichlet allocation (LDA) [46].

The motivation behind topic modeling assumes that words in each document is governed by both topic-word and document-topic distributions, where the topics constitutes a hidden variable. The model considers a collection of documents in a dataset D as a mixture of probabilistic distributions as follows

$$P(D|\alpha, \beta) = \prod_{d \in D} \int P(\theta_d|\alpha) \left(\prod_{\tilde{w} \in d} \sum_{\nu} P(\nu|\theta_d) P(\tilde{w}_d|\nu, \beta) \right) d\theta_d$$

where $\nu, \tilde{w}$ denote individual topic and individual word, respectively, and θ_d is the topic probability for document d . α, β are defined as the hyperparameters that determine sparsity of the Dirichlet priors for document-topic distribution and topic-word distributions, respectively.

After training the model using a Monte Carlo sampling method known as Gibbs sampling, topic distributions for each document $P(\theta_d|\alpha)$ and word distributions for each topic $P(w|\nu, \beta)$ can be obtained. These probability values can be used as document’s semantic features as input to a classifier. Such an approach overcomes limitations from previous lexical-based semantic features, and now postulates that words originate from salient semantic generation mechanism of topics and therefore the mixture of topics can be used as the semantic features to enhance the text representation. This is a major breakthrough in the representation of document semantics, since words are no longer independent lexical entities, but globally related via the hidden topics variable. However, since the multinomial distribution assumption does not enforce order between collections of observations, semantics arising from sequential arrangement of words cannot be captured.

Several QQ classification works employs the above feature extraction approach. Supraja et al. [20] trained an LDA topic model on a domain-specific digital signal processing corpus. Using the topic probabilities as features to an SVM and extreme learning machine (ELM), each question is classified according to Bloom’s learning objectives. This approach has also been employed in [24], which combines unigrams and text-length, sentence topic model, global topic model and global topic structure model (GTSM) [47] as features to a linear SVM classifier with non-linear radial basis function kernels. It is argued that the GTSM is a good indicator for quality due to its explicit modeling of discourse between sentences. However, it was reported that the proposed 3-level topic features only marginally improved the accuracy over unigrams and length features.

FIGURE 2.3: Single artificial neuron.

2.2.3 Neural networks

Neural networks serve as general function approximators where, given sufficient dimensions, their parameters can, in theory, model any function. These networks comprise artificial neurons depicted in Fig. 2.3. The output of the neuron can be expressed by

$$\hat{y} = \sigma(\boldsymbol{\theta}^\top \mathbf{x} + b),$$

where each individual neuron is parameterized by weights $\boldsymbol{\theta}$, and bias b to transform the input features $\mathbf{x}$. Subsequently, a non-linear function σ in the form of sigmoid, tanh or rectified linear unit (ReLU) [48] can be used to compute the final output $\hat{y}$. Using a loss function, the parameters can be optimized using gradient descent and backpropagation for multi-layered networks.

When arranged in multiple layers depth-wise, deep neural networks can perform automated feature extraction driven by patterns in the data. When arranged in specialized arrangements and interconnections, these networks can form convolutional neural networks, recurrent neural networks, attention mechanisms etc. which are powerful architectures for extracting features for various applications.

FIGURE 2.4: Prediction of surrounding words using center word via a window size $E = 3$ with a skip-gram model of *Word2vec*. The variables i and o denote, respectively, the input and output words within the window.

2.2.3.1 Distributional semantics in neural language models

Neural networks have also been deployed to address problems associated with language modeling—to predict probability of textual expressions. A probabilistic approach of learning a context-sensitive language model was explored by Bengio et al. [49] that successfully reduced 17,000 full vocabulary feature space of one-hot representations to a dense vector representation of 100 features, thus alleviating the curse of dimensionality that has long faced by NLP problems. Moreover, the learned real-valued vectors are represented in a common feature space with dimensions exhibiting certain semantic similarity between words.

Learning of distributed representations for words in a numerical space follows the distributional hypothesis [50], which is based on the intuition where terms occurring in similar contexts are semantically similar. *Word2vec* was proposed to learn word semantics in an unsupervised manner using this perspective. Using a source corpus, a fixed-size E sliding window moves across each sentence. The input and target of this network are one-hot vectors $w_j = \mathbb{1}(\tilde{w}_j) \in \mathbb{R}^V$, where the numeral 1 is placed at the assigned index j for the word that is a member of the vocabulary $\mathbb{V}$. The main objective of this probabilistic model is to predict

the $\tilde{w}_e^{(o)}$ neighboring words, using the center word $\tilde{w}_{(E+1)/2}^{(i)}$, as shown in Fig. 2.4. Here, $e \neq (E+1)/2$. For each output word, the predicted probability from the set of vocabulary form this single hidden layer neural model such that the output is given by

$$\begin{aligned} y' &= p(\tilde{w}_e^{(o)} | \tilde{w}^{(i)}) \\ &= \text{softmax}(\mathbf{W}' \mathbf{W} \mathbf{w}^{(i)}), \end{aligned} \quad (2.1)$$

where $\mathbf{W}$ and $\mathbf{W}'$ represents the parameter matrices for input-to-hidden and hidden-to-output linear transformations, respectively. Note that the hidden layer has a lower dimension than the vocabulary size to achieve dimensionality reduction. The training objective is then to maximize the negative log-likelihood of $-\log y'$.

After convergence, the model will have learned a parameter matrix $\mathbf{W}$ that encodes context-dependent semantics by predicting the word's neighbors. Each row of the matrix $\mathbf{W}_{j,:}$ corresponds to the dense semantic representation for the word $\tilde{w}_j$. The degree of learned semantic similarity between words can be computed using the dot-product in which the resultant value reflects their proximity in the semantic space. This matrix forms an embedding lookup table to convert tokens $\tilde{w} \in d$ in a document into their corresponding indexed vectors $\mathbf{W}_{j,:}$ before performing analysis. This process provides a semantic prior for the words as input to the text classification model. The above-mentioned steps of creating numerical vectors for words via unsupervised language modeling procedure is called pre-training or transfer learning [51].

The concept of pre-training these word embeddings was highlighted in [52]. It was shown that the deep architecture, when trained on a sufficiently huge dataset, can provide some syntactic and semantic meaning for words. Unsupervised learning

FIGURE 2.5: Architecture of an unfolded recurrent neural network.

of language models offers significant benefits. Most importantly, it mitigates the requirement of expensive annotated data required for training machine learning models. In this aspect, the distributed semantics assumption allow highly-descriptive features of individual words to be extracted from large amount of unannotated corpora available on the internet. The downstream benefits of a good pre-trained language model has led to many state-of-the-art embeddings such as GPT-2 [53] and BERT [54]. While these embeddings provide good representations, they should be carefully applied to the right problem contexts. In Stack Overflow, BERT does not function well because wordpiece sub-word embeddings [55] cannot compose programming entities. Hence under such cases, simpler lexical language models [56] based on *Word2vec* are still employed.

2.2.3.2 Sequence encoder

Following the success of continuous vectors in *word* representations, sequence prediction models in the form of recurrent neural networks (RNN) are employed to generate a representation from a list of ordered tokens in a *sentence*. These networks have also been applied on handwriting and speech recognition tasks which also involve sequential information.

A feedforward network learns from (i) fixed number of inputs and (ii) assumes the input features are i.i.d.. This may not be practical in learning features from the

human language because (i) sentences naturally contain varying number of words; and (ii) word semantic features are dependent on its surrounding context. These issues have been addressed via the use of RNNs, as depicted in Fig. 2.5. The core differentiating feature of an RNN from other learning models is a hidden state vector $\mathbf{h}_j$ that summarizes all inputs it had seen so far. At each time step j , $\mathbf{h}_j$ is updated via

$$\mathbf{h}_j = \tanh(\mathbf{W}_x \mathbf{x}_j + \mathbf{W}_h \mathbf{h}_{j-1}).$$

Since the same set of parameters $\mathbf{W}_x, \mathbf{W}_h$ are used for computation of $\mathbf{h}_j$ at all time steps, significant savings is achieved in terms of the number of parameters learned. The evolving dynamic state vector $\mathbf{h}_j$ models after the updated knowledge states as each word is processed across a sentence. The variable $\mathbf{h}_j$, therefore, plays an important role in using previous hidden state as feedback and updating itself with the latest input. As a result, these models enable learning features from arbitrary-length sequences. By running it along the whole sequence, it is capable of ‘encoding’ surrounding features into a single vector $\mathbf{h}_j$, thus incorporating contextual information at every time step. As such, these models are commonly used as encoder layers for NLP tasks because of their capability to represent arbitrarily-sized sequential inputs into a fixed size vector for subsequent computations.

The RNN has been reported to suffer from the vanishing gradient problem due to back-propagation through time (BPTT) training [57]. Parameters at the initial time steps do not receive as much adjustments because the gradient dwindled over numerous multiplications with small fractions. This problem was addressed with the introduction of parameterized gate neurons that control information flow and prevents gradients from unnecessarily flowing into the state vector [58]. This addition enabled retention of even longer distance features, hence more contextual information can be gained from text representations, leading to higher performance in NLP applications that relies on contextual features. The most notable variant is the long short-term memory (LSTM) architecture [58]. In this work, a simplified

FIGURE 2.6: Sequence learning using a GRU.

variant of the LSTM with reduced number of gates, known as the gated recurrent units (GRU) [59], is used. Its operation is specified by

$$\mathbf{z}_j = \sigma(\mathbf{W}_z \cdot [\mathbf{x}_j; \mathbf{h}_{j-1}] + \mathbf{b}_z), \quad (2.2)$$

$$\mathbf{r}_j = \sigma(\mathbf{W}_r \cdot [\mathbf{x}_j; \mathbf{h}_{j-1}] + \mathbf{b}_r), \quad (2.3)$$

$$\hat{\mathbf{h}}_j = \tanh(\mathbf{r}_j \odot (\mathbf{W}_{hh} \mathbf{h}_{j-1}) + \mathbf{W}_{hx} \mathbf{x}_j + \mathbf{b}_h), \quad (2.4)$$

$$\mathbf{h}_j = (1 - \mathbf{z}_j) \odot \hat{\mathbf{h}}_j + \mathbf{z}_j \odot \mathbf{h}_{j-1}, \quad (2.5)$$

where $\mathbf{z}_j, \mathbf{r}_j$ denote the update gate and reset gate at time j , respectively. These equations can be summarized in a compact expression as

$$\mathbf{h}_j = \overrightarrow{GRU}(\mathbf{x}_j, \mathbf{h}_{j-1}). \quad (2.6)$$

The main difference between the GRU and vanilla RNN lies in the two gates, a reset gate $\mathbf{r}_j$, and an update gate $\mathbf{z}_j$. Information flow across these gates depend on parameterized computation of the current input and previous state. The variable $\mathbf{r}_j$ determines how much of new input should be fused with the previous memory to compute an update, and $\mathbf{z}_j$ defines how much of the previous memory to retain.

The learning mechanism of a GRU is demonstrated with a sample sentence as shown in Fig. 2.6. Substituting the generic input $\mathbf{x}$ in (2.6) are word vectors $\mathbf{w}_j$ which are sequentially fed to the GRU unit at each time step along the sentence.

The state vector $\mathbf{h}_j$ is computed at every time step using (2.2)-(2.5). More specifically, computation of $\mathbf{h}_3$ uses $\mathbf{h}_2$ and ‘parameter’ $\mathbf{w}_2$ as input and as feedback to itself. Since $\mathbf{h}_2$ contains the learned representations of both $\mathbf{w}_1$ and $\mathbf{w}_2$ during the previous recursive computation, $\mathbf{h}_3$ now contains context representations of all three words in it. Therefore at every time step, $\mathbf{h}_j$ contains information pertaining to previous words, hence giving it contextual information. Along the sentence, the GRU unit selectively opens its gates only to permit changes to certain values of the memory, therefore strengthening its capability to retain information over time.

It is worth noting that the recursive computation described above is only performed in a single direction. It has been shown that humans do read-ahead and this improves comprehension ability. Likewise, learning sequences from both directions does improve performance empirically [60]. Therefore it is common to stack two parallel GRUs in both directions to capture context information from both ends. The learned hidden vectors are stacked together to summarize the sequential patterns from both sides of the word.

2.2.3.3 Neural networks for question classification/quality

Most of the neural network-based question classifiers are applied on the TREC dataset. Due to the entity labels, this is commonly approached as a document classification task. Most works employ convolutional filters [61] and RNNs [62] to convert the variable length question tokens into a fixed size feature vector before classification. Structural features of the question text were also used to enhance existing word-based features in the form of dependency tag embeddings [63] or parse trees [64]. A generic transformer [65] based sentence encoding model [66] has also been found to benefit question modeling which currently holds state of the art performance on the TREC dataset.

On the other hand, classification of questions in a learning setting have also been approached with neural models. In a reading comprehension tutoring system, Ruseti et al. [12] explored the use of bi-directional GRU encoders to encode both user questions and their source sentences and, thereafter, using an attentive pooling architecture to classify question complexity into four levels. In terms of CQA QQ classification, Zheng et al. [27] employed CNNs in a weakly-supervised setting to analyze the quality of Stack Overflow questions. Each question is modeled with *Word2vec* features multiplied by the asker’s reputation and the question’s number of answers.

The approaches reviewed above introduce unique neural architectures for creating a question representation. However, it remains to be explored how a suitable network can be constructed to extract features in accordance with cognitive complexities in Bloom’s taxonomy. For user-generated questions, models transferred from benchmarking datasets offer limited performance. This is due to equal treatment of all segments within the question during feature extraction and neglecting the naturally disjoint semantics between sentences. Emphasis placed on specific parts of a question may be helpful in filtering out noisy texts inherent in user-generated texts (e.g., abbreviations and spelling errors as highlighted in Section 2.1.2.1) and prioritize the use of some highly-discriminative sentences. The remaining chapters of this thesis highlight the proposed solutions with specialized learning architectures to address these gaps.

2.3 Chapter summary

This chapter reviews the latest developments in the area of question classification and related taxonomies for labeling questions under different use cases. Specifically, knowledge-based applications are targeted, which requires the use of Bloom’s taxonomy for instructor questions, whereas for community non-expert

questions, arbitrary ‘quality’ measures are employed to gain insight into the value of these questions for knowledge-building. In these applications, existing works have shown that machine-learning based models are more robust to detect changes and are hence preferred over rule-based methods. While machine-learning models offer significant convenience in the design of accurate classifiers, it remains a challenge to create discriminative features for its use, be it lexical, syntactic, semantic or non-textual features. In this direction, recent advancements in neural networks have enabled unsupervised language models in the form of distributional vector representations for words with contextual semantic representations. This enables higher-order semantic features to be used in question classification and are preferred over previous lexical-based features for modeling dependencies between words in a question. To understand the value it offers for the task of question classification, a technical review of the latest neural language models and sequence encoders were provided, followed by related works that employ neural models for knowledge-based question quality applications.

Chapter 3

Classification of Questions by Cognitive Complexity

In this chapter, a model that automatically classifies questions posed by an instructor is proposed. The class labels of these questions are correlated with complex thinking skills, requiring the learner to progress towards higher mastery on a subject matter and creativity in problem-solving. To determine the learner's current understanding of the subject, educators refer to the Bloom's Taxonomy that categorizes learners' capabilities at each cognitive level. By classifying questions according to this scheme, it alleviates the burden of educators by striking a good balance between higher- and lower-level cognitive questions.

Automating this task is non-trivial as it not only involves the detection of keywords that discriminates between complexities, but also requires a soft-matching of semantic features when the keyword features are inadequate for identification of the class label. This is achieved with the proposed bi-directional GRU (BiGRU) model with attention mechanism that selects important parts of the question based

on the context. In addition, the proposed classifier is integrated into a quiz generation system (QGS) to encourage learner retrieval practice. While previously-reported systems have shown effectiveness in increasing learner engagement and retention [67], they only adapt to learners' history of topic exposure and fail to consider the cognitive complexity involved during the process of attempting questions. This system, on the other hand, encourages retrieval practice by generating practice questions automatically and intelligently for learners according to the cognitive complexity associated with each question. The system is automatic in that the questions appear chapter-wise as the learner/instructor specifies the number of questions to be offered at each complexity level.

3.1 Question classification using bi-directional GRU and attention mechanism

The problem of question classification in terms of cognitive levels is framed as a text classification task. Suppose each question $Q = (\tilde{w}_1, \tilde{w}_2, \dots, \tilde{w}_{|Q|})$, where $\tilde{w}_j$ is defined as individual words within the question sequence and $|Q|$ denotes the number words in a given question, the objective is to maximize the probability of the class label $P(\hat{y} | Q, \theta)$ by estimating a function $f(\cdot)$ that is parameterized by θ . The following description of the model and its associated variables are as shown in Fig. 3.1.

As noted in previous literature, word features that discriminate between question classes should be effectively captured by a machine learning model to achieve good performance. However, these words may not be expressed in their exact forms. To capture the soft relationships between them without using a hard-coded dictionary, we utilize word embeddings that numerically computes closeness between

FIGURE 3.1: Flowchart of variables in training a bi-directional GRU classifier.

words. Each word in the question is converted into continuous numerical representations in the form of embedding vectors. The set of embeddings are initialized from GloVe [68], pre-trained from generic corpora. Words are embedded in a space where spatial relationships grant them lexical semantic meanings. Initializing embeddings this way has been shown to achieve excellent performance in tasks such as sentiment analysis, document classification and automated question-answering due to a regularization effect [69] that minimizes variance and introducing a bias towards generalizable semantics extracted from external documents. In this embedding layer, each word is mapped to its corresponding vector via a lookup table, thus giving $Q = (\mathbf{w}_1, \dots, \mathbf{w}_{|Q|})$.

To adapt the individual word features to the question's context, a bi-directional GRU (BiGRU) layer is employed. The encoder enhances representation of each word by incorporating contextual information using neighboring words from both sides. Defining j as the time-step index, the hidden representation of a given word is obtained by concatenating hidden states $\vec{\mathbf{h}}_j$ and $\overleftarrow{\mathbf{h}}_j$ of both forward- and backward-direction GRUs, i.e.,

$$\vec{\mathbf{h}}_j = \overrightarrow{GRU}(\mathbf{w}_j, \vec{\mathbf{h}}_{j-1}),$$

$$\overleftarrow{\mathbf{h}}_j = \overleftarrow{GRU}(\mathbf{w}_j, \overleftarrow{\mathbf{h}}_{j+1}),$$

$$\mathbf{h}_j = [\vec{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j].$$

The complexity of a question, according to Bloom’s taxonomy, highly hinges on the use of verbs and the associated concepts mentioned in the question. Although these verbs generally appear at the beginning of the question as shown in Table 2.1, they may appear at any location of the question. Therefore the neural model is required to dynamically select segments of words where these indicators appear. To achieve this, a data-driven neural attention [70, 71] layer is applied on top of the encoded vectors to select important segments of the question that discriminates between question complexities. A non-linear transformation is first applied on the encoded vectors

$$\mathbf{u}_j = \tanh(\mathbf{W} \cdot \mathbf{h}_j + \mathbf{b}), \quad (3.1)$$

with $\mathbf{W}$ and $\mathbf{b}$ being the transformation weights and bias, respectively. Each encoded vector then interacts with a parameterized attention vector $\mathbf{u}_w$ giving an attention coefficient

$$a_j = \mathbf{u}_j^\top \mathbf{u}_w \quad (3.2)$$

which is then normalized via softmax. Finally, the vector representation for a question $\mathbf{q}$ is obtained via a weighted average of the word hidden representations given by

$$\mathbf{q} = \sum_j \mathbf{h}_j \text{softmax}(a_j).$$

Defining the class labels as $\{\textit{Knowledge (K)}, \textit{Applied (A)}, \textit{Transfer (T)}\}$, the question representation then undergoes a linear transformation followed by softmax to obtain probabilities of the question being of a particular class $\hat{y} = \{K, A, T\}$, i.e.,

$$P(\hat{y}) = \text{softmax}(\mathbf{W} \cdot \mathbf{q} + \mathbf{b}).$$

Defining N as the total number of training samples in a mini-batch, the model is trained by minimizing the cross-entropy loss computed between true and predicted

Question class	Exemplar of common keywords	#instances
Knowledge	What is, Is the	190
Applied	Determine, Find	82
Transfer	Why, Describe how	77

TABLE 3.1: Dataset statistics of each complexity class.

labels across all N training samples, given by

$$L = - \sum_{n=1}^N \mathbf{y}_n \cdot \log P(\hat{\mathbf{y}}_n).$$

3.2 Dataset

The dataset comprises of 349 questions obtained from an undergraduate DSP course at the Nanyang Technological University. These questions are extracted from resources that are frequently used for creating learner assessments in the form of assignment, homework, quiz, test, examination and online practice questions. Based on the instructor’s experience, these are effective questions that assess learners’ mastery and are originated from well-known textbooks [72]. Some of these questions are self-generated by the instructor who has also generously labeled the questions for training the algorithm. These labels follow the complexity of test items in accordance with the achievement of learning outcomes in line with Bloom’s taxonomy [17, 20] described in Section 2.1.1. The six taxonomy levels were compressed into three ordinal categories of increasing complexity, i.e., *Knowledge* (K), *Applied* (A), *Transfer* (T) to make labeling more tractable, as shown in Table 3.1. These questions encompass topics such as discrete-time signals, discrete-time Fourier transform (DTFT), discrete Fourier transform (DFT), and the z-transform. The type of question prompts (open-ended, multiple-choice, short-structured, essay) and solution process for each question are also taken into account by the instructor during labeling as they trigger different levels of cognitive

functions from the learners.

Many of these questions contain mathematical equations and diagrams to better illustrate the problem contexts. However, the proposed neural model can only perform textual analysis on English words. Therefore, in the pre-processing stage, non-text information are removed, followed by the removal of all symbols and numbers. The remaining words are then lowercased before given as input to the model.

3.3 Experiment setup

To evaluate the effectiveness of this approach, the model is compared against several other baselines:

- **tf-idf**. Bag-of-words features are extracted for each question, followed by computing their relative importance via tf-idf statistical feature selection. These features are then fed to a linear SVM classifier.
- **LDA** [20]. An LDA [46] topic model is trained on the questions, treating each as a document. A linear SVM is then employed to classify each question using topics probabilities as features that represent the semantic composition of the question.
- **CNN** [61]. A convolutional neural network that employs convolutional filters to extract n -gram features, followed by a max-pooling and a fully-connected layer.
- **BiGRU+Max** [71]. In this network architecture, a simpler max-pooling operation is employed in place of the proposed attention mechanism to extract salient hidden features without being dynamically-driven by the attention parameters in (3.1). This serves as a benchmark to determine the effectiveness of the proposed attention mechanism architecture.

To evaluate the performance of the model in classifying the K, A, T classes, precision

$$Precision = \frac{TP}{TP + FP}$$

and recall

$$Recall = \frac{TP}{TP + FN}$$

are being applied. Here, TP, FP, TN , and FN denote true positive, false positives, true negatives, false negatives, respectively. The F1 score is defined as

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall},$$

which describes a harmonic mean between recall and precision. Maximizing this measure ensures that all questions from a particular class are identified (high recall), while ensuring the identified are indeed the actual ones (high precision). Defining $F1_c$ as the F1 score for a particular class c , macro-average F1 is computed by taking a simple arithmetic mean across the classes $\{K, A, T\}$ without accounting for sample sizes, i.e.,

$$Macro-F1 = \frac{1}{|classes|} \sum_{c \in classes} F1_c. \quad (3.3)$$

This measure computes the overall performance of the algorithm and will give fair consideration to smaller-sized classes which typically perform worse than the dominant class.

Method	K	A	T	Macro-F1
LDA [46]	25.00	74.47	54.55	51.34
tf-idf [35]	54.55	82.05	89.66	75.42
CNN [61]	58.82	82.05	85.71	75.53
BiGRU+Max [71]	62.86	81.58	82.76	75.73
BiGRU+Attn [71]	65.00	81.69	82.76	76.48

TABLE 3.2: Classification performance on DSP question dataset. Scores are expressed in percentages (%).

3.4 Results

3.4.1 Comparison analysis

For all methods under comparison, the F1 scores for cognitive class labels ‘K’, ‘A’ and ‘T’ described in Section 3.2 are shown along with the Macro-F1 in Table 3.2. Neural models shown in the bottom group are observed to achieve higher performance, with the proposed BiGRU+Attn algorithm in particular, achieving the highest Macro-F1 of 76.48%. It also outperforms BiGRU+Max with a simpler max-pooling selection layer in all but the ‘T’ class. This is attributed to the BiGRU encoder’s ability to model semantic context of all constituent words, and the attention mechanism that dynamically exploit segments along the question to search for discriminative features. While other algorithms only achieved slightly reduced macro-F1 scores, the LDA features in combination with linear SVM achieved the lowest score at 51.34%. It is worth noting that BiGRU+Attn outperforms all others with greatly improved F1 for ‘K’ labeled questions, with only modest reduction in ‘A’ and ‘T’. This is crucial for real-world applications where fundamental ‘K’ questions form the majority in any assessments, which is also observed in Table 3.1 where the 190 questions make up over 50% of the data.

In addition, results show that the F1 scores of ‘A’ and ‘T’ questions are generally higher than ‘K’, with nearly all models achieving over 80%. This is because many of these questions involve consistent word patterns that could be exploited

Question	Label
consider an lti discrete time system with an impulse response $\langle eq \rangle$ determine the frequency response of the system	Knowledge
a white random sequence with zero mean and unit variance is processed with an lti system that satisfies the following difference equation $\langle eq \rangle$ determine the impulse response and the transfer function of the lti system	Knowledge
the discrete time fourier transform is important in our everyday life describe how the dtft is used in one of the following applications	Transfer

FIGURE 3.2: Attention visualizations of three exemplar DSP questions. The color depth is proportional to the attention neuron activation a_j during inference of the question’s label $\hat{y}$. Contiguous dark spots indicate important segments for the class label.

by the models. For example, many ‘T’ questions contain the expression of ‘*describe the role of...*’, ‘*describe the relationship between...*’, ‘*describe how ... can be used...*’, which constitute instructing the learner to relate between concepts. Additionally, many of the ‘A’ questions assess the learners’ capability in transforming signals between frequency- and time-domain, therefore causing these models to exploit keywords such as *z-transform*, *dft* and *impulse response* in combination with other words. ‘K’ questions, on the other hand, are composed of a wider range of words, as they may involve description of a technical scenario before prompting the learner with the conceptual question. Therefore, a contextual analysis of the full question text is required to classify these questions accurately, whereas relying on the keywords will result in errors. This explains why tf-idf feature selection is highly successful at the keyword-based categories ‘A’ and ‘T’, resulting in the competitive scores. On the other hand, LDA features assume global topical relationships between these words and less likely to make use of the localized keywords, hence performing the worst overall.

3.4.2 Qualitative analysis

Discussion in the prior section highlights the importance of exploiting certain keywords for the ‘A’ and ‘T’ classes, but this is still limited for the ‘K’ class, which

demands a contextual interpretation of the question for accurate classification. This can be achieved by the proposed model’s usage of semantic embeddings with BiGRU encoder that incorporates surrounding contextual information to better direct the attention mechanism towards certain segments of the question text. To gain further insight into the above, a qualitative analysis can be performed by inspecting attention weights extracted from (3.2) during model inference.

Three questions from the test dataset are presented in Fig. 3.2 which shows how attention is allocated along the given text. The color depth indicates importance of the word in determining the question’s feature, where a darker shade implies higher importance. For the top two ‘K’ questions, an elaborated scenario is provided before the actual question to provide better context. This is a common pattern for many questions in this category. In spite of this, the attention parameters have learned to place emphasis on the directive portion, therefore choosing ‘determine’, ‘frequency/impulse response’ and ‘transfer function’ as the features for the labels. Although these questions do require computation from the learner, which also relates to the skill of ‘A’, the questions formulated in this format do not require extensive computation required for ‘A’, but a simple recollection of the learned concept to obtain the answer. This highlights the significance in a model’s capability to dynamically select only the latter parts of the text for labeling after interpreting it fully. It can also be observed that the attended features also correspond to the $\langle VERB \rangle \langle CONCEPT \rangle$ templates for determining intended learning outcomes under Bloom’s Taxonomy in Table 2.1. In the last question belonging to ‘T’ class, the model was able to exploit the keywords ‘following applications’ commonly found in ‘T’ questions, in which students are instructed to relate to other use cases in their daily lives. This shows that the attention can also achieve keyword selection comparable to tf-idf, while also selectively choosing parts of the question to determine the question’s required cognitive complexity from the learner.

3.5 Quiz generation system (QGS)

Although formative and summative assessments are relatively common in on-line courses, current e-learning platforms are not designed to encourage retrieval practice and, by extension, meaningful long-term learning. Using technology to scaffold and encourage self-regulated learning skills among distant learners is critical. Nevertheless, prior works suggest that learners with high self-regulated learning skills often engage in retrieval practice of their own volition [73]. As such, facilitating means for learners to perform structured retrieval practice is imperative for massive open online courses (MOOC) platforms. This is so that learners can methodologically plan their learning sessions to fulfill their intended learning goals.

Such systems formulate practice sessions with a mixture of questions at different complexity levels. As such, categorizing the questions accurately is crucial. For instance, if a learner’s performance is quantified based on high-complexity questions being falsely categorized as low-complexity questions, this could hurt their confidence and may result in gaining an incorrect skill-set—both situations will have undesirable implications during retrieval practice. However, providing learners with questions according to their complexity level is not a trivial task and demands significant man-hours from subject matter expert (or course coordinator). To automatically label each question in the repository with the above-mentioned complexity levels accurately at scale, the classifier described in Section 3.1 is employed as shown in Fig. 3.3 (a). The reusable question bank repository contains all DSP questions from the dataset.

In addition, a web interface for the QGS is developed to facilitate the learner in generating a set of questions according to his/her learning needs, as depicted in Fig. 3.3 (b)-(d). The web interface is developed using HTML and Javascript

3.5. Quiz generation system (QGS)

FIGURE 3.3: Schematic diagram of the proposed quiz generation system (QGS).

for the display of visual elements and form fields to collect user input. The question classifier is developed with Python Flask to handle the generation of labeled questions in a desired proportion.

A few options are available to the learner in the input section to create a customized set of practice questions.

- Chapter selection - Learners/instructors can select the topic based on which the questions will be retrieved from.
- Customized complexity ratio - Learners/instructors can set the number of questions in each category according to self-evaluation of practice requirements.

Submission of the above details will trigger the retrieval of questions from the question bank that has already been automatically pre-classified into the three classes by the proposed question classifier. The retrieval system automatically

selects questions from each category (K, A, and T) per chapter to fulfill the number of questions in preferred proportions specified by the learner. These questions are then displayed under the respective sections of ‘K’, ‘A’, and ‘T’, which can then be utilized for self-practice and to obtain immediate feedback (Fig. 3.3 (c)). The interface also includes a module to allow the learner to provide feedback to the instructor (Fig. 3.3 (d)) for future improvements to the system and learning resources.

3.6 Chapter summary

In this chapter, a neural model is proposed to classify questions generated by subject matter experts into complexity levels. The network is capable of producing a contextual representation for the question with the attention mechanism, which, as a result, generalizes well in identifying question types of the complexities in Bloom’s taxonomy. In addition, a web-based system that produces practice sessions is also developed. This system is supported by the question classifier to accurately label questions, then generates practice sessions at preferred mixture of complex questions to facilitate learners’ own retrieval practice.

One of the limitations with the BiGRU+Attn neural model is its requirement of ample data to train up its parameters. This may be of concern for applications to categorize shallow question banks. In such cases, classifiers such as SVM or decision trees coupled with simpler bag-of-words features should be considered. It is also worthwhile to note that the word embeddings are currently sufficient for classifying question complexity based on word surface patterns. For sophisticated questions demanding deep associations between concepts, the performance may be limited as this knowledge has not been captured within the embeddings.

Chapter 4

Classification of Question Quality in Learner Questions

General availability of the internet has given rise to community question-answering sites and virtual learning spaces, where communities engage in knowledge building. Users or learners initiate discussions with a question, from which it can evolve over time. As opposed to classifying questions based on cognitive levels in Chapter 3, classifying quality in such cases serve to organize information for future users and preserve site content quality. Higher quality questions are those that pose useful problem statements and are well-written. These questions should be ranked higher and recommended to searchers, whereas badly-authored questions should be routed back for amendments or, at times, even deleted.

In the previous chapter, the model operates on instructor-authored assessment questions which have undergone rigorous checks and revision before given to students. These questions are, therefore, logically sound and contain few linguistic errors. However, in communal knowledge-building settings that will be discussed

Part of this chapter has been published as Mun K. Ho, S. Tatinati, Andy W. H. Khong, “A Hierarchical Architecture for Question Quality in Community Question Answering Sites,” in *Proc. Int. Joint Conf. Neural Networks, IJCNN, 2020*.

<p>Block scope in Python</p> <p>When you code in other languages, you will sometimes create a block scope, like this: [code]</p> <p>One purpose (of many) is to improve code readability: to show that certain statements form a logical unit or that certain local variables are used only in that block.</p> <p>Is there an idiomatic way of doing the same thing in Python?</p>			<p>name not defined errors</p> <p>I continue to try nd run this program but keep getting this error: [code]</p> <p>Here is my code: [code]</p> <p>I am new to this and very confused.</p> <p>Please help</p>		
Human	Proposed model	True	Human	Proposed model	True
good	very good	very good	bad	bad	bad

FIGURE 4.1: Sentences identified as highly-discriminative by the proposed model and how its predicted label compares against human and true labels. (left) Example of a *very good* question, and (right) Example of a *bad* question.

in this chapter, elaborated descriptive information is provided by the asker to establish common ground with the answerer. Moreover, casual authorship by non-experts without auditing may also produce many error-prone texts characterized by spelling errors, abbreviations and redundant information.

To address the above-mentioned issues in classifying question quality for longer, noisy learner-generated questions, a hierarchically-arranged neural model is proposed to interpret disjointed sentences progressively and select discriminative features based on relative importance. This is supported by a new topic-based attention mechanism.

4.1 The proposed tHAN architecture

Consider each question Q being composed of a set of $|Q|$ sentences given by $Q = \{S_1, S_2, \dots, S_{|Q|}\}$. Each sentence S_i consists of a set of $|S_i|$ words, given by $\{\tilde{w}_{i,1}, \tilde{w}_{i,2}, \dots, \tilde{w}_{i,|S_i|}\}$, where i denotes the sentence index. Typical human interpretation of QQ involves identifying essential words from sentences and subsequently ordering the sentences in terms of contextual importance. The proposed two-stage hierarchical attention architecture, as shown in Fig. 4.2, takes this into account by learning the weighting schemes at these two levels using parameters $\mathbf{u}_w$ and $\mathbf{u}_s$. Here, the subscripts w and s denote for the words and sentences, respectively.

4.1.1 The proposed two-stage hierarchical attention network (HAN) with topic-weighted attention (TwAtt)

The question words are first mapped into vector representations using a pre-trained embedding layer described in Section 3.1. A sentence encoder in the form of a bi-directional gated recurrent unit (BiGRU) is then employed to incorporate contextual information from surrounding words by learning hidden representations of the sequences. The final hidden representation of each word $\mathbf{h}_{i,j}^{(w)}$ is obtained by concatenating the forward and backward hidden states as

$$\mathbf{h}_{i,j}^{(w)} = [\overrightarrow{GRU}(\mathbf{w}_{i,j}, \vec{\mathbf{h}}_{i,j-1}^{(w)}); \overleftarrow{GRU}(\mathbf{w}_{i,j}, \vec{\mathbf{h}}_{i,j+1}^{(w)})]. \quad (4.1)$$

To eliminate noisy text elements that do not contribute significantly towards the sentence semantics, attention mechanisms are applied to the sentence encoder.

Attention mechanisms have gained popularity for enabling neural networks to focus only on the important features. While a few common variants have been introduced [74], [70], the neural architecture typically share identical structural components of key-value pairs and queries. An attention mask is first computed by matching a query against all keys to find compatibility scores. The mask scores then determine the corresponding values at the output. This reduces subsequent computations to only the most relevant feature, therefore improving model learning. Two variants of sentence-level attention for QQ are proposed. The conventional attention mechanism (vanilla) that is identical to [75] is first described. The second proposed topic-weighted attention (TwAtt) technique is subsequently introduced to regularize the attention mechanism and hence achieve better representation of each sentence.

For the vanilla attention mechanism [75], a vectorized parameter $\mathbf{u}_w$ serves as the query that interacts with the transformed hidden vectors to generate an

FIGURE 4.2: Architecture of proposed tHAN network.

attention coefficient. This parameter is analogous to a learned ‘locus of attention’ that guides the attention on certain words during interpretation of a sentence. Defining matrix $\mathbf{W}_w$ as the transformation weights, the attention coefficient is therefore computed using

$$a_{i,j}^{(w)} = \mathbf{u}_w^\top \tanh(\mathbf{W}_w \mathbf{h}_{i,j}^{(w)}). \quad (4.2)$$

The softmax-normalized coefficients

$$\lambda_{i,j} = \frac{\exp(a_{i,j}^{(w)})}{\sum_j \exp(a_{i,j}^{(w)})} \quad (4.3)$$

are then used as weights that determine the importance of a word in forming the overall sentence representation. Finally, the vector representation for sentence i is obtained via a weighted-average of the word hidden representations, i.e.,

$$\mathbf{s}_i = \sum_j \lambda_{i,j} \mathbf{h}_{i,j}^{(w)}. \quad (4.4)$$

Humans prioritize certain textual clues according its context while comprehending a problem. Likewise, a single attention scheme learned by a single vector

FIGURE 4.3: Topic-weighted attention (TwAtt) mechanism.

$\mathbf{u}_w$ may underfit the highly diverse range of question topics. Therefore, an augmentation of the vanilla version with a context-dependent attention is proposed. The proposed context-dependent attention mechanism computes the attention coefficient based on topical words, which, as a consequence, allows the algorithm to focus on features learned to within a local topic space. To achieve this topic-weighted attention (*TwAtt*), topic words are obtained from two sources, either tags assigned by the questioner, or words generated from topic models. The topic model is trained using latent Dirichlet allocation (LDA) [46] on all questions at document level. As shown in Fig. 4.2, word embeddings of the K most representative words $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_K\}$ for the most relevant topic are then used to enhance information passed to the attention layer.

The key operation of this attention mechanism is illustrated in Fig. 4.3. Inspired from other attention module designs [65], the single parameter vector in (4.2) is replaced by a variable query vector guided by topic words $\mathbf{v}_k$. These topic words are first transformed into query vectors via a matrix $\mathbf{W}_v$ and non-linear function $\tanh$. Similarly, the word hidden representations undergo an identical process to form the key vectors. A dot-product attention then computes an interaction score

between the transformed topic word representations and the hidden representation of the question words, i.e.,

$$score_i(j, k) = \tanh(\mathbf{W}_v \mathbf{v}_k)^\top \tanh(\mathbf{W}_w \mathbf{h}_{i,j}^{(w)}).$$

Salient latent features from the transformed topic word representations are obtained via max-pooling, which is then used to derive the attention coefficient

$$a_{i,j}^{(w)} = \max_k score_i(j, k).$$

This is subsequently followed by a weighted-average to generate the sentence representation $\mathbf{s}_i$ described in (4.3) and (4.4).

4.1.2 Sentence importance selection

Different discourse function of each sentence indicates that not every sentence is equally important in determining the quality of the question. Hence, the identification of highly-discriminative sentences is proposed at this layer. Similar to word representation described in (4.1), each sentence representation $\mathbf{s}_i$ first undergoes an encoding process to obtain its hidden representation

$$\mathbf{h}_i^{(s)} = [\overrightarrow{GRU}(\mathbf{s}_i, \vec{\mathbf{h}}_{i-1}^{(s)}); \overleftarrow{GRU}(\mathbf{s}_i, \vec{\mathbf{h}}_{i+1}^{(s)})].$$

This module consists of a vectorized parameter $\mathbf{u}_s$ that interacts with hidden vectors to generate a sentence attention coefficient $a_i^{(s)}$, which is subsequently normalized by the softmax function, i.e.,

$$a_i^{(s)} = \mathbf{u}_s^\top \tanh(\mathbf{W}_s \mathbf{h}_i^{(s)});$$

$$\mu_i = \frac{\exp(a_i^{(s)})}{\sum_i \exp(a_i^{(s)})}.$$

Finally, the vector representation for the question $\mathbf{q}$ is obtained via a weighted average of the sentence hidden representations

$$\mathbf{q} = \sum_i \mu_i \mathbf{h}_i^{(s)}.$$

To determine the objective function, $\mathbf{q}$ then undergoes a linear transformation followed by softmax to obtain probabilities of each predicted label $\hat{y}$ given by

$$\mathbf{p}_{\hat{y}} = \text{softmax}(\mathbf{W}_q \mathbf{q} + \mathbf{b}),$$

where $\mathbf{W}_q$, and $\mathbf{b}$ denote transformation weights and bias, respectively. Defining y as the true label, the model is subsequently trained by minimizing the cross-entropy loss

$$L = - \sum_n^N y_n \log p_{\hat{y}_n}$$

across a mini-batch of N samples. This enables the algorithm to determine an optimal set of attention parameters $\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*$ to generate the question representation for each Q

$$\mathbf{q} = f(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^* | Q).$$

Hereafter, following the naming convention in [75], the proposed hierarchical approach with vanilla attention mechanism is named as hierarchical attention network (HAN), whereas the proposed HAN with TwAtt is referred as tHAN.

4.2 Dataset

Experiments are conducted on a subset of community-generated questions available in the Stack Overflow data dump. While a subset of questions between

[Accessed: March 2019] <https://archive.org/details/stackexchange>

2011-2012 tagged with “Python” have been chosen, the proposed algorithm can be extended to other question tags.

Leveraging on the collective wisdom of the Stack Overflow community, quality labels are computed mainly using *votes* that are available in the dataset. These votes are awarded by other users of the website. To ensure that these questions have adequately been peer-reviewed, only questions with more than 1000 views are retained. Improvising on quality classes in [25], questions with a score of less than or equal to 0 are considered as *bad*, whereas questions having scores above the 3rd quantile are considered *very good*. The remaining questions are considered readable but do not possess exceptional properties that call for either recommendation or deletion; these are therefore simply labeled as *good*. To further enhance the quality of the dataset, questions marked by moderators as duplicates were removed from the dataset, whereas questions closed by moderators for reasons including off-topic, subjective and argumentative, not a real question, too localized are also considered as *bad* in the dataset. The above selection criteria results in a total of 55,380 questions, comprising 12,710 (23%) *very good*, 34,461 (62%) *good* and 8,209 (15%) *bad* questions. To train the model, the questions are split into training and testing datasets using a 80:20 ratio via stratified sampling.

Data cleaning and pre-processing procedures similar to that of [76] have been applied to minimize out-of-vocabulary words. This includes the removal of programming language snippets, HTML tags and escape characters, URLs and numbers. However, short code snippets and camel-cased words are preserved and normalized because these may contain useful entities that contribute towards the question semantics. This dataset presents challenges in text analysis since it includes noisy text inherent in all user-generated texts, including spelling errors, abbreviations, and low-frequency technical words. Therefore, only the top 3,000 occurring words are kept to allow the model to focus on the statistically-significant features.

Additional experiments using more vocabulary words resulted in no significant difference in performance.

4.3 Experiment setup

Both question title and body are sentence-tokenized and concatenated as input to the model. These words are provided to the proposed hierarchical approaches and existing methods to predict QQ. Existing baseline methods include:

- A linear ridge classifier [24] that employs topic model features at three different granularities;
- A CNN model consisting of two sets of 5×5 convolutional kernels and 2×2 max-pooling layers, followed by a fully-connected layer [27];
- A CNN with 3,4,5-width convolutional kernels is employed to extract contiguous n -gram features followed by a max-pooling layer and a fully-connected layer [61];
- A bi-directional LSTM (BiLSTM) max-pooling network [71] that extracts the most representative features in both forward and backward directions;
- Transformer (a state-of-the-art encoder neural network model) that employs multiple layers of self attention to generate contextual representations for each word. Similar to the implementation of BERT_{BASE}-classifier in [77], the encoded representation from the first timestep is used as the question representation, which is subsequently passed to a fully-connected layer for classification.

Word embeddings of all neural models are initialized with pre-trained GloVe embeddings glove.6B.300d [68] before fine-tuning during the training process.

The sizes of the hidden unit of all neural encoder layers (except transformer) and the attention vector parameters u_w and u_s are tuned amongst $\{50, 100, 150, 200\}$. The transformer maintains 300 hidden units at every layer and utilizes five attention heads for feature extraction. For the TwAtt layer, an LDA topic model is trained with Dirichlet priors with parameters $\alpha = 0.01, \beta = 0.01$ for 100 passes over the dataset to obtain twenty question topics. Only the set of $K = 10$ words for the most representative topic in each question is used. These topic word embeddings v_k are also initialized similarly with GloVe but fine-tuned separately. Optimization is performed using Adam [78] with its initial learning rate tuned amongst $\{1 \times 10^{-6}, 3 \times 10^{-5}, 1 \times 10^{-5}\}$ and a weight decay of 1×10^{-5} . A grid search over each set of hyper-parameters was performed, combined with five-fold cross-validation for each set. The hyper-parameters and model parameters with the most stable losses and highest F1 score as observed across the five runs is considered the best-performing model and used for evaluation on the test dataset.

4.4 Evaluation

To quantify the label classification performance, precision, recall and the F1 score defined in Section 3.3 were employed. Recall is crucial for ensuring that most of the real positives of both *very good* and *bad* are correctly identified by the model. On the other hand, precision is important for the *good* class as it verifies the model against overconfidence in its predictions for this dominant class. A balance is sought between these two metrics via the F1 score (harmonic mean between recall and precision) to quantify classification performance for all three classes.

To further verify the performance, human subjects were also consulted to establish a benchmark for the challenge in identifying quality of questions on Stack Overflow. Four experienced Python programmers were involved and each subject

Models		Precision	Recall	F1	Classes
Human subjects		31.52	41.87	33.66 ± 6.75	Very Good
Baselines	Linear [24]	35.46	14.87	21.01	
	CNN [61]	27.42	1.34	2.55	
	BiLSTM [71]	25.58	12.98	17.22	
	Transformer [65]	24.09	18.21	20.74	
Proposed Approach	HAN [75]	42.41	22.31	29.23	
	tHAN (40-topics)	40.99	23.80	30.11	
Human subjects		66.67	25.36	35.06 ± 5.03	Bad
Baselines	Linear	38.46	1.22	2.36	
	CNN	20.00	0.18	0.36	
	BiLSTM	17.21	5.18	7.96	
	Transformer	15.98	6.33	9.07	
Proposed Approach	HAN	30.10	1.89	3.55	
	tHAN (40-topics)	34.33	4.20	7.49	
Human subjects		-	-	37.45 ± 9.52	Macro-Average
Baselines	Linear	-	-	31.11	
	CNN	-	-	26.45	
	BiLSTM	-	-	32.33	
	Transformer	-	-	33.00	
Proposed Approach	HAN	-	-	35.70	
	tHAN (40-topics)	-	-	37.15	

TABLE 4.1: Comparison analysis for all three quality classes. Metrics are expressed in percentages (%).

is given a set of stratified-sampled 100 questions from the test split. These subjects were briefed on the desirable characteristics of questions before starting their annotation processes.

QQ performance obtained from the baseline models and the proposed approaches are provided in Table 4.1. Although the classification results of all three classes are provided, only those for *very good* and *bad* quality questions are of interest since the identification of these questions are important for maintaining the overall quality of site content. These models are categorized into three groups, namely human subjects, linear and sequential baselines, and the proposed hierarchical models. The lower-than-expected scores by human subjects is attributed to a high variation in their perception of question quality in the presence of the

dominant *good* class. The proposed hierarchical modeling of questions strikes a good balance between precision and recall, thus achieving the highest F1 scores of 37.15 in overall *macro-averaged* and of 30.11 in the individual *very good* class. In terms of overall performance quantified by *macro-averaged* F1 score, the linear and sequential neural models performed worse due to the generally lower precision scores of below 40% and 30% in both *very good* and *bad* classes, respectively. This is because many of these questions have been misclassified into the dominant *good* class. This highlights the limitation of CNN and BiLSTM sequential models in terms of their capability of modeling questions for quality, since all segments of the questions are considered equally without attention. Transformer, being one of the best sequential encoders, is able to outperform both CNN and BiLSTM due to the effectiveness of self-attention in modeling context. Although it achieves the highest F1 score of 9.07% for *bad* questions, the difference against the proposed tHAN model was not significant enough to compensate for its lower *macro-averaged* F1. The CNN model described in [27] has also been implemented. However, it is found that the model has been developed for a customized labeling function that discriminates only between two classes, therefore performing modestly lower for this dataset.

By overcoming limitations of sequential modeling, the proposed hierarchical approach customized with topic words achieves the best results with over 4% improvement against the linear baseline method, represented by the transformer. The proposed approaches achieve comparable performance with human annotators in identifying the *very good* questions and overall F1 score. Two examples are presented in Fig. 4.1 to demonstrate the efficacy of the hierarchical model in selecting discriminative features at sentence level, which was effectively learned by the proposed approach. In the first example, the proposed model assigns a higher weightage to the first sentence that contains a detailed description of the technical problem from the asker. While a human annotator may not find this technical

FIGURE 4.4: Effect of number of trained topics on F1 (%) for HAN and tHAN.

information as important for the problem as the community does, the model (being trained on large amount of data) is able to identify this as a crucial piece of information to determine a quality question. The second question, on the other hand, was written based on poor research. The proposed model correctly identifies the sentence as an expression commonly found under such cases. In these two exemplar analyses, the hierarchical approach achieves higher performance compared to sequential ones. Similar trend was observed for most of the noisy text in the questions over the dataset.

4.5 Ablation analysis

Arguably, topic words may be obtained simply from the asker-assigned question tags instead of words as indicated by latent topics trained from the LDA model. The effectiveness of topic words from both these sources is compared using a stacked bar chart corresponding to three categories of F1 scores in Fig. 4.4. In general, higher number of topics trained with LDA improves performance over HAN with none at all. This is due to the model achieving better context that

is learned from the topic-weighted attention mechanism. Modeling better context therefore allows the model to learn a suitable sentence representation, which, as a consequence, relieves the burden of sentence attention module to select the important sentence for classification as in Fig. 4.1.

The use of question tags modestly reduced the performance of tHAN. This is because many questions are tagged with highly-specific technical jargon which do not occur frequently across questions, therefore making them unsuitable for modeling general contexts. The above problem was mitigated by introducing topic words from LDA. It can be observed that as the number of trained topics increases, the classification performance on QQ improves. This trend continues until forty topics, after which performance starts to reduce modestly. This is because an increase in topics learned introduces more diversity to segregate the common types of problems encountered. This can be seen from Table 4.2 in which words from the three most representative topics are presented. It can be observed that these topics form the general type of questions being asked on Stack Overflow; some common types of questions can be inferred as follows—Topic 1: asking about a problem involving an error code; Topic 2: appropriate ways of passing parameters in API documentations; Topic 3: installation and package issues. However, as the number of trained topics continues to increase to fifty, overfitting occurs and some uncommon topics become too noisy to model question contexts. Using words from these topics therefore results in noisy features and, as a result, the overall performance decreases as observed from Fig. 4.4. These results show that using words from topic models at an appropriate level is effective at guiding the model towards learning semantic properties at the sentence and word levels to form an effective representation for QQ.

Overall, the proposed hierarchical approach outperforms the state-of-the-art transformer encoders with significantly fewer parameters. This underpins the fact

Topic 1	error, code, python, get, trying, following, using, problem, tried, help
Topic 2	type, argument, arguments, pass, parameters, parameter, default, documentation, passing, set
Topic 3	python, install, import, module, installed, lib, py, path, packages, version

TABLE 4.2: Top topics learned using the best model.

that extracting features via hierarchical selection is important for QQ in noisy user-generated questions. This model is particularly more effective when coupled with a topic attention module that introduces global information about common topic structures in the corpus. The performance for identifying *bad* quality questions is still modestly poor since many were classified as the *good* questions.

4.6 Chapter summary

A neural network architecture that automatically evaluates QQ on community question answering sites is proposed. User-generated texts on these sites are often noisy and require customized processing methods to extract relevant features from only the salient parts. To address this issue, a hierarchical model is proposed to aggregate relevant information over textual features at word and sentence levels using neural attention. In addition, a new context-aware TwAtt attention mechanism is developed. This mechanism introduces global topical information from the corpus trained via topic models to complement the hierarchical model. Topic words are useful to distinguish between problem contexts that serve as useful information for the model to vary its attention scheme during the processing of a question. Experiments conducted on the Stack Overflow dataset show that the proposed approach is effective at exploiting these additional features that represent a given question, which, as a consequence, outperforms existing QQ prediction approaches without the use of any platform social indicators as features.

Although experiment results with tHAN are encouraging, it should be noted that the model's reliance on topic words makes it susceptible to weaknesses of LDA. Some of those identified areas are corpora with low word-count and skewed topic distributions [79]. These are the cases where the quality of LDA topics suffer and where the model will not reap the intended benefits. tHAN is hence recommended for user-generated questions that tend to be more verbose, whereas other alternatives should be considered for shorter questions.

Chapter 5

Specificity for Classifying Question Quality

In Chapters 3 and 4, a sequential neural encoder and a hierarchical architecture were incorporated with attention mechanisms to allow the models to focus on salient segments of a question text via a data-driven approach. The architecture may select the best sentences to attend to for generating the question representation, but certain arguments present in sub-parts of the sentence can be useful features. On the other hand, questions supported by granular facts have long been appreciated as being of higher-quality on CQA platforms and also by educators. However, these findings have been limited to qualitative analysis of classroom discussions without developing an automated system for this purpose. This chapter introduces the use of entity embeddings from a named-entity recognizer (NER) to aid the attention mechanism in seeking these specificity features.

5.1 Introduction

Specificity was first used to distinguish specific from generic sentences in news. Proper names and price figures enhance factual information with higher granularity. After being utilized effectively for determining the quality of news articles, automated specificity models were employed to determine the quality of summaries and scientific articles [80, 81]. In CQA sites, community guidelines often encourage users to present details in the asked question. Such higher quality questions enable readers and answerers to readily comprehend the problem, thus leading to more fruitful discussions. As opposed to community-generated questions, learners in classrooms were found to produce higher quality questions when they include elements of the subject matter in more granular detail [2], while others found a correlation between argument quality and *Speciteller* scores involving domain-specific n -grams [82]. These works indicate that quality and specificity are closely intertwined. Using specificity as target labels, models were developed [83] using entity approximators and slow dictionary-lookups as features. Lugini and Litman [84] used entity counts identified by the Stanford NER [85]. While it has been applied to understand argument specificity, the works fall short from directly examining the quality of the questions.

In this work, the notion of specificity is applied to tHAN by incorporating entity embeddings to predict quality. This is inspired from works of aspect-level sentiment analysis, where the attention mechanism is trained to focus on sentence segment in response to specific aspects. Experiments show that entity embeddings can work well with the TwAtt mechanism to attend to granular argument details, resulting in more reliable QQ classification. This is the first attempt to apply specificity features in quality prediction. In addition, the use such widely-available semantic extraction tools (NER) for each domain is demonstrated to enhance features for quality classification.

FIGURE 5.1: A CRF infers NE tag at each step with the highest probability (in red) by using features extracted from the input sequence of words. The full list of NE tags is given in Section 5.3.1. Words marked with tags other than ‘O’ indicate mentions of entities, e.g., PyPy (API) and PostgreSQL (Framework).

5.2 Proposed method

Entity mentions are critical for determining specificity of a text expression. Hence for knowledge-based applications, this crucial information is exploited to classify question quality. For domain-specific texts such as medicine and engineering, taggers are widely-available for such applications. As a first step, questions are supplemented with entity tags before being used as input to the QQ classifier.

5.2.1 Software-specific NER (SNER)

Following the implementation of Ye et al. [29], an NER is employed in the form of a linear chain conditional random field (CRF) [86] to label each word with their respective entity tags. A linear chain CRF takes a sequence of observations and classify each token with a set of pre-defined tags as depicted in Fig. 5.1. In classifying the labels, the CRF also takes into account transition probabilities from the previous state t_{j-1} to the next, as computed from parameterized feature functions $\phi(\cdot)$.

FIGURE 5.2: Architecture of the proposed s-tHAN, of which tHAN from Section 4.1 is highlighted in gray.

Formally, the CRF is expressed as the conditional probability of corresponding sequence of tags $\mathbf{t}$ given a sequence of words $\mathbf{w}$, i.e.,

$$P(\mathbf{t}|\mathbf{w}) = \frac{1}{Z} \exp \left(\sum_{j=1}^{|\mathbf{S}|} \sum_{m=1}^M \theta_m \phi_m(t_j, t_{j-1}, \mathbf{w}) \right)$$

where j denotes the time position of the input and output sequence $\mathbf{w}$ and $\mathbf{t}$. The variable ϕ_m denotes the m -th feature function designed for the computation of transition and label probabilities whereas its corresponding parameter is denoted as λ_m . In addition, M is defined as the total number of feature functions while Z is the normalization denominator.

5.2.2 s-tHAN

Using the representation from Section 4.1, suppose each question consists of a sequence of sentences given by $Q = (S_1, S_2, \dots, S_{|Q|})$, and each sentence, in turn,

FIGURE 5.3: Post-processing is applied after NER tagging to mimic the tokenizer in tHAN.

comprises multiple words $S_i = (\tilde{w}_1, \tilde{w}_2, \dots, \tilde{w}_{|S_i|})$. From this information, the s-tHAN model as shown in Fig. 5.2 computes a predicted label $\hat{y}$ from the question representation $\mathbf{q}$.

As a first step, a pre-trained SNER is used to produce entity tag for each token. These tags provide higher-level semantic information pertaining the category of each word, which will subsequently be combined with word embedding features. The SNER produces token-tag pairs for each sentence, giving

$$S_i = ((\tilde{w}_1, t_1), (\tilde{w}_2, t_2), \dots, (\tilde{w}_{|S_i|}, t_{|S_i|}))$$

where t denotes entity tag corresponding to each token.

The SNER employs symbolic features on top of words in its feature functions to predict the entity tags. However, words with symbols are not present in the embedding vocabulary. Therefore, a post-processing step is applied after NER tagging as shown in Fig. 5.3 to adapt towards the embeddings required in input format. This process ensures that the word is being covered by embedding's vocabulary to achieve accurate mapping in the QQ classification network. This post-processing unit performs separation and merging of words that mimics the lowercased, characters-only output from the original tHAN tokenizer, while also standardizing entity tags by removing the BIO stems.

The topic-weighted attention (TwAtt) described in Section 4.1.1 is maintained in this architecture for its context-dependent attention that narrows down features

learned to within the question local topic space. As input to this attention mechanism, topic word features $\{v_k\}$ are also obtained from a pre-trained LDA model reused from Section 4.3. Altogether, each question is processed into

$$Q'_n = ((\tilde{w}_{1,1}, t_{1,1}), \dots, (\tilde{w}_{|Q_n|, |S_{|Q_n|}|}), t_{|Q_n|, |S_{|Q_n|}|}), \{v_k\}),$$

which is formally described in Algorithm 1.

The entity tags, when embedded into a feature space, provide additional dimensions to the sentence BiGRU encoder and topic-weighted attention mechanism for computation of the sentence representation. In another words, these tags serve as location markers of high specificity where knowledge discussions occur, which is a key characteristic of *very good* questions. The question words are first mapped into numerical vectors $\mathbf{w}_{i,j}$ using a pre-trained embedding layer. Similarly, a separate embedding lookup table is randomly initialized for the entity tags. Both the tag $\mathbf{t}_{i,j}$ and word $\mathbf{w}_{i,j}$ embeddings are concatenated to create an enhanced feature vector, given by

$$\mathbf{w}'_{i,j} = [\mathbf{w}_{i,j} ; \mathbf{t}_{i,j}].$$

thus replacing $\mathbf{w}_{i,j}$ in (4.1). This set of augmented features which will then undergo computation in the tHAN network similar to that described in Section 4.1, i.e.,

$$\mathbf{q}_n = tHAN(v_{n,k}, \tilde{w}_{n,i,j}, t_{n,i,j}).$$

Overall, the full model is referred to as the s-tHAN network, with ‘s’ denoting for *specificity*.

In an unbalanced dataset, the performance of smaller classes generally suffer because more samples from dominant class are shown to the model during training. To address this problem, the s-tHAN model is trained with weighted cross-entropy

Algorithm 1: Feature augmentation processing for s-tHAN

Input: Dataset $D = \{(Q, y)_n\}_{n=1}^N$,
pre-trained topic model $\text{LDA}(\cdot)$,
pre-trained $\text{NER}(\cdot)$ tagger

Output: Processed dataset D'

Pre-processing:

```

foreach  $Q_n = (S_1, S_2, \dots, S_{|Q_n|})$  do
     $\{v_{n,1}, v_{n,2}, \dots, v_{n,K}\} \leftarrow \text{LDA}(\{\tilde{w}, \tilde{w} \in Q_n\})$ 
    foreach  $S_i \in Q_n$  do
         $t_1, t_2, \dots, t_{|S_i|} \leftarrow \text{NER}(\tilde{w}_1, \tilde{w}_2, \dots, \tilde{w}_{|S_i|})$ 
         $S'_i \leftarrow ((\tilde{w}_{i,1}, t_{i,1}), (\tilde{w}_{i,2}, t_{i,2}), \dots, (\tilde{w}_{i,|S_i|}, t_{i,|S_i|}))$ 
    end
     $Q'_n \leftarrow ((S'_1, S'_2, \dots, S'_{|Q_n|}), \{v_k\}_{k=1}^K)$ 
end
 $D' \leftarrow \{Q'_n\}_{n=1}^N$ 

```

given by

$$L = - \sum_c^C \eta_c \cdot y_c \cdot \log p(\hat{y}_c)$$

where η_c denotes the amplification for each class $c \in C$. A higher η_c is assigned to classes with smaller sample sizes to increase sensitivity of parameter updates to these classes, thus compensating for the lack of samples.

5.3 Dataset and pre-processing

5.3.1 Dataset

For training the SNER tagger, the annotated dataset from [29] is used. In the original dataset, annotators were given sentences from Stack Overflow questions to label the locations of the entities. To maintain the integrity of Python context, only the subset of Python questions were retained. In the case of Stack Overflow, these are categorized under five most common software entities, namely API, Framework (Fram), Programming Language (PL), Platform (Plat) and software

tag	B-API	I-API	B-Fram	I-Fram	B-Plat	I-Plat
counts	130	28	96	26	6	6
tag	B-PL	I-PL	B-Stan	I-Stan	O	
counts	180	20	18	4	10922	

TABLE 5.1: Statistics of tags

standard (Stan). Adopting from annotation conventions for NER data segmentation, the BIO convention is used, representing begin (B), inside (I) and outside (O). A B-tag is therefore used to mark the start of each entity mention. If the entity is a multi-word expression, words starting from the second word is marked by the I-tag. Words that do not belong to any entity expressions, typically English words, will be marked with the O-tag. A Cartesian product between these two sets $\{B, I\} \times \{API, Fram, PL, Plat, Stan\}$ yields eleven unique software NER tags as shown in Table 5.1. Training and testing sets are obtained by splitting the sentences into a 80:20 ratio.

A total of 764 Python-related sentences were extracted from annotations. Cross-overs entities from other languages may also be present, but will also be tagged under ‘Fram’ or ‘API’, hence improving robustness for the tagger. Exploring the dataset yields the total counts of each tag, as tabulated in Table 5.1. It can be observed that entities do not occur frequently in questions, forming only approximately 4.5% of overall token counts, while the remaining are O-tags (95.5%). In addition, ‘API’, ‘Fram’ and ‘PL’ are disproportionately mentioned, over ‘Plat’ and ‘Stan’. It is also useful to note that entities generally exist as a single word, rarely followed by I-tags. Neural networks sequence taggers using LSTM or GRU generally suffer under such imbalanced scenario, whereas CRF fortunately performs well with the right engineered features [29].

For the subsequent QQ classification task, identical set of Python questions from Section 4.2 is used. A new set of embeddings from [56] is employed. These vectors have been trained from software-related texts from Stack Overflow, where

similar software entities are closer to each other in the semantic space. This alleviates the out-of-vocabulary and out-of-domain problem resulted from previous GloVe embeddings. Due to this increased coverage, a greater vocabulary size of top-occurring 10,000 words is used for the experiments.

5.3.2 Pre-processing

The CRF utilizes a set of handcrafted feature functions to determine state transitions between tags at every time step. By exploiting common styles in Stack Overflow questions and Python syntax, the creation of features follows that of [29], including

Brown clusters bitstrings. Originally proposed as a way of dealing with lexical sparsity, Brown clusters [87] are compact representations of word classes that tend to appear in adjacent positions in the training set. Trained on unlabeled Stack Overflow corpora, semantically similar words will obtain identical bitstrings.

Character n-grams extracted from the front and rear. This is motivated by the observation that some API entities may contain mentions of their parent library in the substring.

Boolean features that detects existence of certain patterns, such as alphanumeric, digits, underscores, dots, rear parentheses, camel cases etc.

These feature functions require raw tokens as input in order to produce the boolean features. Hence the cases and programming syntax-related symbols are preserved prior to NER tagging. These symbols will then be removed after the tagging procedure before being used as input to the neural model.

5.4 Experiment setup

The SNER is implemented in Python with the package ‘sklearn-crfsuite’. The CRF is optimized with the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm, and during training, the penalty is set at 0.1 for both L1- and L2-regularization for a maximum of 100 epochs.

As benchmarks to compare against the proposed s-tHAN model, previously-reported models originally intended for classifying specificity are used. These include:

- ***Speciteller***. This model consists of a logistic classifier, employing a dictionary of features from the General Inquirer [88], MPQA [89] lexicon, MRC Psycholinguistic Database [90] to indicate specificity. In practice, the neural embedding version achieves higher performance than the shallow features version, and henceforth used for comparison.
- **BiLSTM + sp. feats.** model [84] that comprises a bi-directional LSTM neural encoder to encode semantics from the sentences. This represents the utility of including the question semantics for determining the question’s quality, alongside handcrafted specificity features.
- **tHAN**. Proposed in Chapter 4, this architecture utilizes the new TwAtt mechanism to generate salient sentence representations to better represent noisy user-generated texts.

It was reported that the absence or presence of named-entities is sufficient for identifying specificity in classroom discussions. Noting this fact, two variants of the network is constructed. In the first variant, entity embeddings $\mathbf{t}_{i,j}$ serve as binary indicators of entities, whereas the second variant employs all six classes (including ‘O’). These are indicated in the subscripts, for instance, s-tHAN₂ and s-tHAN₆.

Following the evaluation mechanism in Section 4.3, each Stack Overflow question’s title and body are sentence-tokenized and concatenated as input to the model. Word embeddings of all models (except *Speciteller*) are initialized with software-specific *Word2vec* [56] embeddings with 200-dimensions, whereas NER tag embeddings are randomly initialized with 20-dimensions. For the TwAtt mechanism in tHAN, identical topic model from Section 4.3 is used, and using the $K = 10$ words from the most representative topic. Topic word embeddings $\mathbf{v}_k$ are initialized with [56] and all three embedding sets are fine-tuned during training. The Adam [78] optimizer is used to optimize the loss function and the learning rate is tuned amongst $\{50, 100, 150, 200\}$ with a weight decay of 1×10^{-5} . Early stopping is applied when F1 score of validation step does not improve for 10 consecutive epochs. The variable η_c for the loss functions are set at 0.6, 0.1, 0.3 for the *bad*, *good* and *very good* classes respectively, which are approximately inversely proportional to their sample sizes. A five-fold cross validation is performed, with a grid search over the set of hyperparameters. The model parameters with most stable losses and highest F1 score is selected for evaluation on the test set and reported in the results.

5.5 Results and discussion

5.5.1 NER tagging performance

Fig. 5.4 shows the confusion matrix associated with the number of NE tags classified by SNER with respect to their actual tags on the annotated Python sentences. An accuracy score of 99.98% shows that almost all tags can be classified accurately using SNER. Despite the number of O tags that dominate the population with 4369 (95.77%) tags, the model is still capable of identifying the remaining

FIGURE 5.4: Confusion matrix of entity tags by SNER.

entity tags. This is due to the unique feature functions that indicate discriminative features of programming expressions by exploiting conventions within the Python programming language, especially in APIs, Frameworks expressions. The boolean features that indicate presence of parentheses within the expressions such as `rstrip()`, `Twisted.web`, or camel-cases like `QStringList` or `BaseHTTPServer` allow the SNER to identify these entities. For Frameworks, they typically include some capitalizations or common vocabulary that can be quickly identified by the SNER. The implications of this outstanding performance on the randomly-split test dataset is optimistic, indicating that the trained SNER is sufficiently accurate to identify entities in the tHAN dataset, thus minimizing cascading errors that flow

Model	F1 _{very good}	F1 _{bad}	F1 _{macro avg}
HAN (w/o WL & DSE) [75]	29.23	3.55	35.70
tHAN (w/o WL & DSE) [91]	30.11	7.49	37.15
HAN	28.08	11.14	37.99
tHAN	33.58	13.30	39.69
<i>Speciteller</i> [83]	7.16	0.24	28.00
BiLSTM + sp. feats. [84]	36.54	23.85	41.01
s-tHAN ₆	40.40	27.11	40.03
improvement w.r.t. tHAN	20.31% ↑	104.00% ↑	0.85% ↑

TABLE 5.2: QQ classification results comparing s-tHAN against other baseline models. Metrics are expressed in percentages (%).

downstream into QQ classifier. However, it must be noted that the risk of out-of-domain error will still persist for unseen entities and vocabulary in the question sets.

5.5.2 QQ classification comparison analysis

QQ performance obtained from the baseline models and the proposed s-tHAN model are provided in Table 5.2. Performance of these models are quantified using F1 scores, where the macro-average (F1_{macro avg}) quantifies the overall performance across all three classes, as defined in (3.3). In particular, *very good* and *bad* questions are emphasized due to their relative importance. These models are analyzed in three groups. The first group presents baselines from Chapter 4 which do not employ the weighted cross-entropy loss (WL) and domain-specific word embeddings (DSE). These are compared against the second group to account for the impact of WL on their performance differences. The third group contains models with specificity features from which the impact of specificity features is investigated.

Between the first and second groups, there is between two to three times improvement in F1_{bad}. There is not much effect on *very good* questions where tHAN only benefited a 10% relative increase and HAN exhibits a reduction in the score. WL and DSE made the highest impact on the *bad*-class, to the extent that it

resulted in an overall increase in macro-average F1 score to 39.69%. This is mostly attributed to WL which has increased parameter sensitivity towards the minority *bad* class (15% of questions). Therefore it can be concluded that the previous poor performance reported in Section 4.4 is due to QQ being severely impacted by data imbalance, but this can be mitigated with some adjustments in the loss function.

After addressing the impact of WL and DSE, the effect of specificity features for QQ will be explored. It can be observed that the addition of entity features in s-tHAN₆ vastly improved the performance against tHAN (+20.31% and +104%). *Speciteller* performed the worst in this group. This is explained by the sentiment-polarized words and abstract noun features originally used in news corpora, which now offer limited functionality in the discussion of technical knowledge. This is because sentiment is less prevalent in discussions made on Stack Overflow (which focuses more on software development) rendering *Speciteller*’s embeddings inefficient.

For the BiLSTM + sp. feats. model, it can be observed that when an encoded question representation is being added to handcrafted specificity features, it outperforms *Speciteller* significantly, with the highest macro-F1 score of 41.01%. The BiLSTM + sp. feats. model even exceeds HAN and tHAN from the second group that have specialized attentive architectures to generate the question representation. This also highlights the importance of entities from the Stanford NER in determining question quality. s-tHAN₆ achieves the highest performance amongst all algorithms being considered with the highest F1 scores in both *very good* and *bad* questions. This improved performance is, however, achieved without any deliberately crafted specificity features but relying on the entity tags to approximate them. While specificity features are not deliberately engineered in s-tHAN₆ similar gains are achieved compared to BiLSTM + sp. feats. It appears that the entity tags synergizes with the TwAtt mechanism that ‘highlights’ the mentions of entities, thus creating a structural bias for the attention mechanism

Model	Ablated feature	F1 _{very good}	F1 _{bad}	F1 _{macro avg}
s-tHAN ₆	-	40.40	27.11	40.03
s-tHAN ₂	2 NE tags only	38.35	24.32	40.49
s-HAN ₆	removed TwAtt	38.43	25.58	40.60
s-HAN ₂	2 NE tags only & removed TwAtt	37.99	24.28	40.82

TABLE 5.3: QQ results when a subset of NE tags or when TwAtt mechanism are removed from the s-tHAN₆ model. Metrics are expressed in percentages (%).

to focus specificity-related features at these locations. To gain more insights into this hypothesis, an ablation analysis is conducted in which the specialized layers are removed in succession.

5.5.3 Feature ablation

The impact on QQ classification with the removal of feature extraction layers is tabulated in Table 5.3. When comparing within the same base s-tHAN and s-HAN models, it is observed that using binary indicators of entity does not perform as well as using all six entity tags. The segregation of the embedding space enhances the features on top of semantic word embeddings for the hierarchical encoder for QQ. Regarding the use of TwAtt in s-tHAN, negligible benefit is observed as compared to s-HAN when there are only two tags; s-tHAN₂ achieved F1 scores of (38.35%, 24.32%), while s-HAN₂ achieved F1 scores of (37.99%, 24.28%). As the tag number increases to six in s-tHAN₆, increases in F1 scores for the both *very good* and *bad* are observed at 2% and 1.5%, respectively. This shows that the specificity tag embeddings are being utilized by TwAtt thus making it the highest performing model. Instead of employing handcrafted features, s-HAN and s-tHAN follow earlier works of using named-entities directly. This turns out to be effective, as all four of them perform better than BiLSTM + sp. feats.

Overall, an increase in F1 for *bad* and *very good* classes, in general, will result in lowered macro-F1 scores, as observed in Table 5.3. This is due to lower performance

Using MySQL in Flask

Can someone share example codes in Flask on how to access a MySQL DB?
There have been documents showing how to connect to sqlite but not on MySQL.

FIGURE 5.5: Comparison of attention patterns at the sentence attention and *TwAtt* between 3 models for a *very good* Stack Overflow question. Darker squares indicate higher attention activations.

in the dominant *good* class. However observation shows that s-tHAN₆ suffers from a modest reduction in $F1_{\text{macro avg}}$ compared to BiLSTM + sp. feats., but with vastly improved capability in *very good* and *bad* as shown in Table 5.2.

The difference in performance can be further explained by studying the attention neuron activations in Fig. 5.5 for an exemplar question. This allows us to observe how the addition of named-entities affect the attention module in its extraction of features in conjunction with the encoders. The assigned topic words of TwAtt is located in the horizontal axes. On the vertical axes, tokens of each sentence are given. Some words have been dropped due to stopword removal. On the top, a heatmap of the sentence attention activations is also given, indicating the emphasis on each sentence at creating the question representation.

For this question that discusses a combination of database technologies in conjunction with Flask, the problem is of high interest for the community, and demonstrated that some background research has been done on *sqlite*. In tHAN, the model only focuses on the keyword of *mysql* interacting with the topic, neglecting others as seen from the high attention weights in the vertical of *mysql* topic word. With this limited scope, it misclassifies the question as only *good*. With the addition of named-entity markers, both s-tHAN₂ and s-tHAN₆ focus the feature extraction around entities *mysql* and *flask*. Moreover, both assign higher attention around *example*, *codes*, *flask* in the second sentence, whereas tHAN’s attention fails to converge on this critical information. Consequently, s-tHAN₂ and s-tHAN₆ can classify the question correctly as *very good*. However between these two, s-tHAN₆ achieves lower loss for the classifications of sample like this. This is evident from the sentence attention, as a result of specificity-guided feature extraction that assigns weightage to the last sentence that indicates background research. A higher variation of entity embeddings seems to widen coverage of TwAtt in finding more features, thus outperforming s-tHAN₂.

5.6 Chapter summary

In knowledge-related discourse, specificity is a crucial factor for holding meaningful discussions and establishing common background knowledge. In this work, s-tHAN is proposed, and the algorithm employs entity embeddings (which serve as specificity markers) to aid the proposed attention networks for prediction of question quality. This is achieved by using a NER model to tag each word with its associated named entity label. Analysis on the attention patterns reveal that the entity tags synergize with the TwAtt mechanism, which results in the creation of a structural bias for the attention mechanism to focus specificity related features at

these locations. Experiment results against other specificity-related baseline models also demonstrate that s-tHAN can achieve improved QQ performance (in terms of F1 score) without any deliberately crafted specificity features, but could benefit from using widely-available shallow semantic extraction tools in the form of NERs.

It should be noted that the performance improvement of s-tHAN is achieved with an NER specifically trained on a single domain of software engineering, which sparsely label the question tokens with entity tags. For interdisciplinary studies and cross-domain forum questions, the model may be extended to stack entity tags from multiple NERs. This approach results in greater number of tag embeddings (variety) and perhaps more importantly, higher numbers of words along the text being tagged as entities (density), whereas less words are tagged as non-entities. In future work it will be interesting to explore the impact of increased variety and density of entity tags on the performance of s-tHAN in the cross-domain applications mentioned.

Chapter 6

Conclusion and Recommendations

6.1 Conclusion

In this thesis, the problem of classifying questions found in knowledge-based interactions is being considered. Differentiated into assessments and learner-initiated questions, the problems are approached and labeled differently due to unique cognitive processes and intents involved in creating the question.

Classification of assessment questions according to cognitive complexity is first explored. A neural network model is proposed with attention mechanism to direct the creation of a question representation for this purpose. The model is evaluated on university-level digital signal processing questions, where it outperforms other keyword feature machine learning models. The network handles the detection of keywords that discriminates between complexities, while dynamically selecting segments of the question for determining the class label. This is supported by attention activation diagrams that show high emphasis around certain predictive keywords and textual templates corresponding to intended learning outcomes. To support learners in their retrieval practice, the neural classifier is also integrated

into the backend of a quiz generation system with a desired mix of questions at different complexity levels.

Next, the problem of classifying the quality of user-generated questions on community question answering sites is considered. These questions are often noisy and require customized processing methods to extract relevant features from only the salient parts. To address this issue, a hierarchical model is proposed to aggregate relevant information over textual features at word and sentence levels using neural attention. Additionally, a context-dependent attention mechanism is developed that introduces global topical information from the corpus via topic models to complement the hierarchical models. Experiments conducted on the Stack Overflow dataset show that the proposed approach is effective at exploiting these features, which as a consequence, outperforms existing QQ prediction approaches without the use of any platform social indicators.

Higher quality student questions typically involve higher degree of specification with mentions of specific entities from the subject matter to lead discussions. Instead of engineering features to capture the notion of specificity, the proposed s-tHAN network makes use of common semantic extraction tools in the form of NER to enhance the word features. The NER model tags each word with its entity class within the subject domain, which enables the training of an embedding space that indicate the degree of specification at each segment of the question. Inspection of attention activations reveal that the embeddings synergize well with the TwAtt attention mechanism to mark these segments, thus outperforming other baseline models that explicitly engineers dictionary-lookup specificity features for this task.

6.2 Recommendations for future research

The following are some possible suggestions for future research:

- **Degree of specificity.** While specificity has been proposed as a feature for predicting QQ, the named-entities are labeled into nominal categories. For some subject areas such as medicine, biology and engineering, ontologies have been constructed to organize the domain knowledge into hierarchical relationships. An entity could belong to a concept at any level within the tree. Louis and Nenkova [92] observed that in scientific journalism, “a sequence of varying degrees of specificity are predictive of writing quality”. The same could be used as features to a machine learning model to predict question quality, where the degree of specificity is measured by the entity’s distance from the root node.
- **Discourse relations between text spans.** Going deeper into the cognitive processes behind question-posing, the connections between arguments within the question can be explored. In communicating the knowledge-seeking question, information presented by the learner do not exist independently but form internal semantic relationships between adjacent sentences. Specification, as explored in this thesis, is related to only ‘Instantiation’ and ‘Restatement’, out of the many discourse relations defined under the Penn Discourse Treebank (PDTB) [93] annotation manual. Extracting patterns from connectivity in information presented will potentially shed more light into a student’s learning mechanism.

List of Author's Awards, Patents, and Publications

Conference Proceedings

Mun Kit Ho, Sivanagaraja Tatinati, Andy W. H. Khong, “A Hierarchical Architecture for Question Quality in Community Question Answering Sites,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2020.

Bibliography

- [1] J. Hintikka, *Socratic epistemology: Explorations of knowledge-seeking by questioning*. Cambridge University Press, 2007.
- [2] A. C. Graesser and N. K. Person, “Question asking during tutoring,” *Amer. Educ. Res. J.*, vol. 31, no. 1, pp. 104–137, 1994.
- [3] M. Watts, G. Gould, and S. Alsop, “Questions of understanding: Categorising pupils’ questions in science,” *School Sci. Rev.*, vol. 79, no. 286, pp. 57–63, 1997.
- [4] C. Chin and J. Osborne, “Students’ questions: A potential resource for teaching and learning science,” *Studies in Sci. Educ.*, vol. 44, no. 1, pp. 1–39, 2008.
- [5] M. Brown, M. McCormack, J. Reeves, D. Brook, S. Grajek, B. Alexander, M. Bali, S. Bulger, S. Dark, N. Engelbert, K. Gannon, A. Gauthier, D. Gibson, R. Gibson, B. Lundin, G. Veletsianos, and N. Weber, “EDUCAUSE Horizon Report, Teaching and Learning Edition,” EDUCAUSE, Tech. Rep., 2020.
- [6] D. Litman, “Natural language processing for enhancing teaching and learning,” in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 4170–4176.
- [7] X. Li and D. Roth, “Learning question classifiers,” in *Proc. 19th Int. Conf. on Comput. Linguistics*, 2002.
- [8] D. Zhang and W. S. Lee, “Question classification using support vector machines,” in *Proc. 26th Annu. Int. ACM SIGIR Conf. on Res. and Develop. in Inf. Retrieval*, 2003, p. 26–32.
- [9] Z. Hui, J. Liu, and L. Ouyang, “Question classification based on an extended class sequential rule model,” in *Proc. Int. Joint Conf. Natural Lang. Process.*, 2011, pp. 938–946.
- [10] J. Rodrigues, C. Saedi, A. Branco, and J. Silva, “Semantic equivalence detection: Are interrogatives harder than declaratives?” in *Proc. 11th Int. Conf. on Lang. Resour. and Eval. (LREC 2018)*, 2018, pp. 3248–3253.
- [11] V. Rus, B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev, and C. Moldovan, “Overview of the first question generation shared task evaluation challenge,” in *Proc. 3rd Workshop Question Gener.*, 2010, pp. 45–57.

- [12] S. Ruseti, M. Dascalu, A. M. Johnson, R. Balyan, K. J. Kopp, D. S. McNamara, S. A. Crossley, and S. Trausan-Matu, “Predicting question quality using recurrent neural networks,” in *Proc. Artif. Intell. in Educ.*, 2018, pp. 491–502.
- [13] R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, and J. Pineau, “Towards an automatic Turing test: Learning to evaluate dialogue responses,” in *Proc. 55th Annu. Meeting Assoc. for Comput. Linguistics*, 2017, pp. 1116–1126.
- [14] E. M. Voorhees, “The TREC-8 question answering track report,” in *Proc. 8th Text Retrieval Conf.*, 1999, pp. 77–82.
- [15] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu, “Falcon: Boosting knowledge for answer engines,” in *Proc. Text REtrieval Conf. (TREC)*, vol. 9, 2000, pp. 479–488.
- [16] J. B. Biggs and K. F. Collis, *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. Academic Press, 1982.
- [17] B. S. Bloom, M. D. Englehart, E. J. Furst, W. H. Hill, and D. R. Krathwohl, *Taxonomy of Educational Objectives*. David McKay Company, 1956.
- [18] K. Jayakodi, M. Bandara, and I. Perera, “An automatic classifier for exam questions in engineering: A process for bloom’s taxonomy,” in *Proc. IEEE Int. Conf. on Teaching, Assessment, and Learn. for Eng. (TALE)*, 2015.
- [19] S. Haris and N. Omar, “Bloom’s taxonomy question categorization using rules and n-gram approach,” *J. of Theoretical and Applied Inf. Technol.*, vol. 76, pp. 401–407, 2015.
- [20] S. Supraja, S. Tatinati, K. Hartman, and A. W. H. Khong, “Automatically linking digital signal processing assessment questions to key engineering learning outcomes,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2018, pp. 6996–7000.
- [21] D. S. McNamara, T. O’Reilly, M. Rowe, C. Boonthum, and I. B. Levinstein, *iSTART: A Web-based tutor that teaches self-explanation and metacognitive reading strategies*. Lawrence Erlbaum Associates Publishers, 2007.
- [22] A. Graesser, V. Rus, and Z. Cai, “Question classification schemes,” 2008.
- [23] K. J. Kopp, A. M. Johnson, S. A. Crossley, and D. S. McNamara, “Assessing question quality using NLP,” in *Proc. Artif. Intell. in Educ.*, 2017, pp. 523–527.
- [24] S. Ravi, B. Pang, V. Rastogi, and R. Kumar, “Great question! Question quality in community Q&A,” in *Proc. Int. AAAI Conf. on Web and Social Media*, 2014, pp. 426–435.

- [25] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza, and D. Fullerton, “Improving low quality stack overflow post detection,” in *Proc. IEEE Int. Conf. Softw. Maintenance and Evolution*, 2014, pp. 541–544.
- [26] G. W. Hodgins, “Classifying the quality of questions and answers from stack overflow,” 2016.
- [27] Y. Zheng, B. Wei, J. Liu, M. Wang, W. Chen, B. Wu, and Y. Chen, “Quality prediction of newly proposed questions in CQA by leveraging weakly supervised learning,” in *Proc. Adv. Data Mining and Appl.*, 2017, pp. 655–667.
- [28] P. Nakov, L. Màrquez, W. Magdy, A. Moschitti, J. Glass, and B. Randeree, “SemEval-2015 task 3: Answer selection in community question answering,” in *Proc. 9th Int. Workshop Semantic Eval. (SemEval 2015)*, 2015, pp. 269–281.
- [29] D. Ye, Z. Xing, C. Y. Foo, Z. Q. Ang, J. Li, and N. Kapre, “Software-specific named entity recognition in software engineering social content,” in *Proc. IEEE 23rd Int. Conf. Softw. Analysis, Evolution, and Reengineering (SANER)*, vol. 1, 2016, pp. 90–101.
- [30] O. Anuyah, I. M. Azpiazu, and M. S. Pera, “Using structured knowledge and traditional word embeddings to generate concept representations in the educational domain,” in *Companion Proc. World Wide Web Conf.*, 2019, pp. 274–282.
- [31] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, “BioBERT: A pre-trained biomedical language representation model for biomedical text mining,” *Bioinformatics*, 2019.
- [32] V. A. Silva, I. I. Bittencourt, and J. C. Maldonado, “Automatic question classifiers: A systematic review,” *IEEE Trans. Learn. Technol.*, vol. 12, no. 4, pp. 485–502, 2019.
- [33] J. Silva, L. Coheur, A. C. Mendes, and A. Wichert, “From symbolic to sub-symbolic information in question classification,” *Artif. Intell. Rev.*, vol. 35, no. 2, pp. 137–154, 2011.
- [34] G. A. Miller, “Wordnet: a lexical database for english,” *Commun. ACM*, vol. 38, pp. 39–41, 1995.
- [35] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Inf. Process. Manage.*, vol. 24, pp. 513–523, 1988.
- [36] Y. R. Tausczik and J. W. Pennebaker, “The psychological meaning of words: LIWC and computerized text analysis methods,” *J. Lang. and Social Psych.*, vol. 29, no. 1, pp. 24–54, 2010.
- [37] J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, and B. S. Chissom, “Derivation of new readability formulas (automated readability index, Fog count and Flesch reading ease formula) for navy enlisted personnel,” Tech. Rep., 1975.

- [38] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
- [39] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne, “Finding high-quality content in social media,” in *Proc. 2008 Int. Conf. on Web Search and Data Mining*, 2008, pp. 183–194.
- [40] J. M. Kleinberg, “Authoritative sources in a hyperlinked environment,” *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.
- [41] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank citation ranking: Bringing order to the web,” Tech. Rep., 1999.
- [42] B. Li, T. Jin, M. R. Lyu, I. King, and B. Mak, “Analyzing and predicting question quality in community question answering services,” in *Proc. 21st Int. Conf. on World Wide Web*, 2012, pp. 775–782.
- [43] A. Baltadzhieva and G. Chrupała, “Predicting the quality of questions on Stackoverflow,” in *Proc. Recent Adv. in Natural Lang. Process.*, 2015, pp. 32–40.
- [44] T. K. Landauer and S. T. Dumais, “A solution to plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge,” *Psych. Rev.*, vol. 104, no. 2, p. 211, 1997.
- [45] T. Hofmann, “Probabilistic latent semantic indexing,” in *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. and Develop. in Inf. Retrieval*, 1999, pp. 50–57.
- [46] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.
- [47] H. Chen, S. Branavan, R. Barzilay, and D. R. Karger, “Global models of document structure using latent permutations,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics (NAACL)*, 2009, pp. 371–379.
- [48] A. F. Agarap, “Deep learning using rectified linear units (relu),” 2018.
- [49] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *J. Mach. Learn. Res.*, vol. 3, no. Feb, pp. 1137–1155, 2003.
- [50] Z. S. Harris, “Distributional structure,” *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.
- [51] S. Ruder, “Neural transfer learning for natural language processing,” Ph.D. dissertation, National University of Ireland, Galway, 2019.
- [52] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proc. the 25th Int. Conf. on Mach. Learn. (ICML)*, 2008, pp. 160–167.

- [53] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” 2019.
- [54] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol. (NAACL-HLT)*, 2019, p. 4171–4186.
- [55] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” 2016.
- [56] V. Efstathiou, C. Chatzilenas, and D. Spinellis, “Word embeddings for the software engineering domain,” in *Proc. IEEE/ACM 15th Int. Conf. Mining Softw. Repositories (MSR)*, 2018, pp. 38–41.
- [57] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *Int. J. Uncertainty Fuzziness Knowl.-based Syst.*, vol. 6, no. 2, pp. 107–116, 1998.
- [58] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, pp. 1735–1780, 1997.
- [59] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in *Proc. Conf. Empirical Methods in Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.
- [60] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM networks,” in *Proc. IEEE Int. Joint Conf. on Neural Netw.*, vol. 4, 2005, pp. 2047–2052.
- [61] Y. Kim, “Convolutional neural networks for sentence classification,” in *Proc. Conf. Empirical Methods in Natural Lang. Process. (EMNLP)*, 2014, pp. 1746–1751.
- [62] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, “Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling,” in *Proc. 26th Int. Conf. Comput. Linguistics (COLING)*, 2016, pp. 3485–3495.
- [63] A. Komninos and S. Manandhar, “Dependency based embeddings for sentence classification tasks,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol. (NAACL-HLT)*, 2016, pp. 1490–1500.
- [64] L. Mou, H. Peng, G. Li, Y. Xu, L. Zhang, and Z. Jin, “Discriminative neural sentence modeling by tree-based convolution,” in *Proc. Conf. Empirical Methods in Natural Lang. Process. (EMNLP)*, 2015, pp. 2315–2325.

- [65] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proc. Advances Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
- [66] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, and R. Kurzweil, “Universal sentence encoder for English,” in *Proc. 2018 Conf. Empirical Methods in Natural Lang. Process. (EMNLP): Syst. Demonstrations*, 2018.
- [67] D. Davis, R. F. Kizilcec, C. Hauff, and G. Houben, “The half-life of MOOC knowledge: A randomized trial evaluating knowledge retention and retrieval practice in MOOCs,” in *Proc. Int. Conf. on Learn. Analytics and Knowl.*, 2018, pp. 1–10.
- [68] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in *Proc. Conf. Empirical Methods in Natural Lang. Process. (EMNLP)*, 2014, pp. 839–845.
- [69] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, “Why does unsupervised pre-training help deep learning?” in *Proc. 30th Int. Conf. Artif. Intell. and Statistics*, 2010, pp. 201–208.
- [70] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *Proc. Int. Conf. on Learn. Representations (ICLR)*, 2015.
- [71] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, “Supervised learning of universal sentence representations from natural language inference data,” in *Proc. Conf. Empirical Methods in Natural Lang. Process. (EMNLP)*, 2017, pp. 670–680.
- [72] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach*. McGraw-Hill Companies, 2005.
- [73] R. F. Kizilcec, M. Pérez-Sanagustín, and J. J. Maldonado, “Self-regulated learning strategies predict learner behavior and goal attainment in massive open online courses,” *Comput. & Educ.*, vol. 104, pp. 18–33, 2017.
- [74] T. Luong, H. Pham, and C. Manning, “Effective approaches to attention-based neural machine translation,” in *Proc. Conf. Empirical Methods in Natural Lang. Process. (EMNLP)*, 2015, pp. 1412–1421.
- [75] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical attention networks for document classification,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics (NAACL)*, 2016, pp. 1480–1489.
- [76] A. Shirani, B. Xu, D. Lo, T. Solorio, and A. Alipour, “Question relatedness on stack overflow: The task, dataset, and corpus-inspired models,” 2019.

- [77] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, “Huggingface’s transformers: State-of-the-art natural language processing,” 2019.
- [78] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. 3rd Int. Conf. on Learn. Representations (ICLR)*, 2015.
- [79] Y. Zuo, J. Zhao, and K. Xu, “Word network topic model: a simple but general solution for short and imbalanced texts,” *Knowl. Inf. Syst.*, vol. 48, no. 2, pp. 379–398, 2016.
- [80] A. Louis and A. Nenkova, “Text specificity and impact on quality of news summaries,” in *Proc. Workshop Monolingual Text-To-Text Gener.*, 2011, pp. 34–42.
- [81] A. Louis and A. Nenkova, “General versus specific sentences: Automatic identification and application to analysis of news summaries,” University of Pennsylvania, Tech. Rep., 2011.
- [82] R. Swanson, B. Ecker, and M. Walker, “Argument mining: Extracting arguments from online dialogue,” in *Proc. 16th Annu. Meeting of the Special Interest Group on Discourse and Dialogue*, 2015, pp. 217–226.
- [83] J. J. Li and A. Nenkova, “Fast and accurate prediction of sentence specificity,” in *Proc. 29th AAAI Conf. on Artif. Intell.*, 2015, pp. 2281–2287.
- [84] L. Lugini and D. Litman, “Predicting specificity in classroom discussion,” in *Proc. 12th Workshop on Innovative Use of NLP for Building Educ. Appl.*, 2017, pp. 52–61.
- [85] J. R. Finkel, T. Grenager, and C. Manning, “Incorporating non-local information into information extraction systems by Gibbs sampling,” in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2005, pp. 363–370.
- [86] C. Sutton and A. McCallum, “An introduction to conditional random fields for relational learning,” *Introduction to Statistical Relational Learn.*, vol. 2, pp. 93–128, 2006.
- [87] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer, “Class-based n-gram models of natural language,” *Comput. Linguistics*, vol. 18, no. 4, pp. 467–480, 1992.
- [88] P. J. Stone and E. B. Hunt, “A computer approach to content analysis: Studies using the General Inquirer system,” in *Proc. Spring Joint Comput. Conf.*, 1963, pp. 241–256.
- [89] T. Wilson, J. Wiebe, and P. Hoffmann, “Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis,” *Comput. Linguistics*, vol. 35, no. 3, p. 399–433, 2009.

- [90] M. Wilson, “MRC psycholinguistic database: Machine-usable dictionary, version 2.00,” *Behavior Res. Methods, Instrum. & Comput.*, vol. 20, no. 1, 1988.
- [91] M. K. Ho, S. Tatinati, and A. W. H. Khong, “A hierarchical architecture for question quality in community question answering sites,” in *Proc. Int. Joint Conf. on Neural Netw. (IJCNN)*, 2020.
- [92] A. Louis and A. Nenkova, “A corpus of science journalism for analyzing writing quality,” *Dialogue & Discourse*, vol. 4, no. 2, pp. 87–117, 2013.
- [93] R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. Joshi, and B. Weber, “The Penn discourse TreeBank 2.0.” in *Proc. 6th Int. Conf. Lang. Resour. and Eval. (LREC)*, 2008.