AMERICAN UNIVERSITY OF BEIRUT

LIFELONG CHATBOTS FOR CUSTOMER SUPPORT

by
NATALY ADEL DALAL

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Electrical and Computer Engineering
of the Maroun Semaan Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
January 2021

AMERICAN UNIVERSITY OF BEIRUT

LIFELONG CHATBOTS FOR CUSTOMER SUPPORT

by

NATALY ADEL DALAL

Approved by:

Hazem

Prof. Hazem Hajj, Associate Professor Electrical and Computer Engineering	Advisor

[illegible]

Prof. Zaher Dawy, Associate Professor Electrical and Computer Engineering	Member of Committee

Shady

Prof. Shady Elbassuoni, Associate Professor Computer Science	Member of Committee

Date of thesis defense: January 22, 2021

AMERICAN UNIVERSITY OF BEIRUT

THESIS RELEASE FORM

Student Name: _____Dalal______________Nataly______________Adel___________

Student Name:	Dalal	Nataly	Adel
	Last	First	Middle

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of my thesis; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes:

☐ As of the date of submission

☐ One year from the date of submission of my thesis.

☐ Two years from the date of submission of my thesis.

☒ Three years from the date of submission of my thesis.

[illegible]

______________________________February 8, 2021_________________

Signature   Date

ACKNOWLEDGEMENTS

I would like to first express my sincere appreciation and gratitude to my advisor Prof. Hazem Hajj for his continuous support and guidance along the way. Moreover, I would like to thank the committee members Prof. Zaher Dawy and Prof. Shady Elbassuoni for their valuable feedback and contribution to shaping this thesis. Additionally, I would like to give a special thanks to Mr. Samer Salam, for his tremendous help in proposing the idea behind the application and supporting me throughout the implementation. Secondly, I owe a huge thanks to my colleagues in the AUB MIND lab for being an amazing support group (technically and emotionally) since the first day. Thirdly, I would like to sincerely thank Kaizo for their invaluable support in allowing me to pursue this thesis while working with them. And finally, a big thank you to my family and friends in giving me the strength, confidence, and motivation to keep pushing through and achieving more.

ABSTRACT
OF THE THESIS OF

Nataly Adel Dalal for Master of Engineering
Major: Electrical and Computer Engineering

Title: Lifelong Chatbots for Customer Support

With advances in machine learning, chatbots are gaining increasing popularity in different domains, specifically bots for customer support. The benefit of these bots is that they provide customers instant responses anytime and are able to save cost and time while enhancing customer service. However, one of the limitations of existing work on customer support bots (CSB) is that models developed for CSBs are either static or updated infrequently as more training data becomes available. Additionally, updating the learned models often needs human intervention.

To address the limitations of discontinuous CSB learning and the required manual intervention, we propose the design of an automated Lifelong Learning CSB (LL-CSB) that can continuously learn and adapt its answers based on new knowledge acquired from different sources like support forums and discussions on the web. The proposed design of the LL-CSB addresses several challenges for lifelong learning (LL) including continuous tasks: extraction of new knowledge, update to existing knowledge, integration, and updates to LL-CSB response model. We propose to setup the CSB as an Information Retrieval (IR) system where the user asks for the solution of a technical problem and the response is a potential solution from the support knowledge base. To facilitate continuous knowledge updates, we design knowledge base of the LL-CSB as a knowledge graph (KG) consisting of <problem, solution> pairs. For continuous knowledge update, we propose a lifelong learning algorithm capturing rules for extraction of new knowledge from the web and checking whether a problem already exists in the knowledge base before adding it with the corresponding solution and linking it to similar problems in the KG. The similarity matching is based on a computationally low-cost and fast method using hashing TF-IDF vectorizer. For continuous updates of the LL-CSB response model and instead of retraining with the whole dataset with new knowledge, a TF-IDF hashing solution is used to enable fast additions of vectors representing new problems. Finally, we implemented the LL-CSB with real data from CISCO corporation’s network support. In addition to the LL-CSB design and implementation, we proposed a CSB-specific simulator for evaluation of the LL strategy. The simulator models real-time updates of knowledge over time and evaluates the performance of customer queries over simulated time.

To evaluate the proposed design, we created a validation dataset comprised of real question-answer discussions crawled from CISCO’s online support forum. Using the simulator, our experimental results demonstrated the superiority of LL-CSB with up to 3.18X improvement in F1 score compared to CSB without LL. The simulator also showed how the baseline CSB suffered from a drop in recall and precision with more queries when it does not take advantage of lifelong learning.

TABLE OF CONTENTS

ILLUSTRATIONS

Figure

TABLES

CHAPTER 1

INTRODUCTION

When dealing with customer support, there are common struggles faced by the customers from one end and the businesses from the other, where customers worry about waiting hours or days to get a reply or a fix for their, most of the time, simple issues, while businesses struggle to keep up with the continuous flow of requests. This is where chatbots, or in that context, question-answering (QA) systems, are of great value. They provide instant 24-hour responses to simple questions posed by the clients thus allowing support staff in companies to focus on harder problems and saving staffing costs for these companies [1][2]. The problem however with these traditional QA customer support bots (CSB), which receive as an input a user's problem or question and reply back the top-k answers or solutions to that problem, is that as new support knowledge emerges over time, the only way they can be updated is through manual re-training performed by employees themselves. This re-training process can be arduous as it requires continuous human intervention and critical timely updates. Additionally, the operation causes a lag in a somewhat discontinued learning process that inevitably causes the bot's performance to suffer as new questions or queries are presented by users during that lag. To overcome this limitation, this thesis aims at exploring the integration of automated lifelong learning (LL) with the CSB. The proposed LL-CSB would continuously and automatically learn new knowledge, thus avoiding human intervention and any lag in updates while ensuring improvement in its performance with new user queries [3].

Several approaches were developed for CSBs in answering a user problem or query. These approaches include pattern matching [4]–[6], rule-based methods [7]–[9], Artificial Intelligence Modelling Language (AIML) [10]–[13], Latent Semantic Analysis (LSA) [12], Natural Language Understanding (NLU) [14]–[16], Neural Networks (RNN, LSTM) [17]–[25], and Information or Answer Retrieval (IR-QA) [26]–[28]. Most use hybrid approaches that combine one or more of the mentioned methods. However, their work misses the continual knowledge learning process. On the other hand, research in lifelong learning for chatbots, which mostly used knowledge graphs (KGs), involved extending knowledge-base completion (KBC) that further links more entities with new relations in the KG [29][30] into open knowledge-base completion (OKBC). OKBC allows the bot to accept and deal with unseen entities or relations, which is a key-part of LL [29]–[32]. There has been some work where continuous learning for bots was implemented based on the conversation with the user [35]–[37]. Others have aimed at achieving lifelong information extraction by continuously extracting facts and opinions from the web [38]–[43]. However, none of the mentioned methodologies were applied for customer support.

To tackle the gap in previous work and the problem of discontinuous learning for CSBs, this thesis targets designing lifelong learning for a customer support bot (LL-CSB) that continuously learns new knowledge from the web. The flow of LL-CSB is demonstrated in Figure 1; the bot takes as an input a user question or problem and returns its top-k answers or solutions. The approach is evaluated for network data from CISCO's online forum[1].

[1] https://community.cisco.com/t5/technology-and-support/ct-p/technology-support

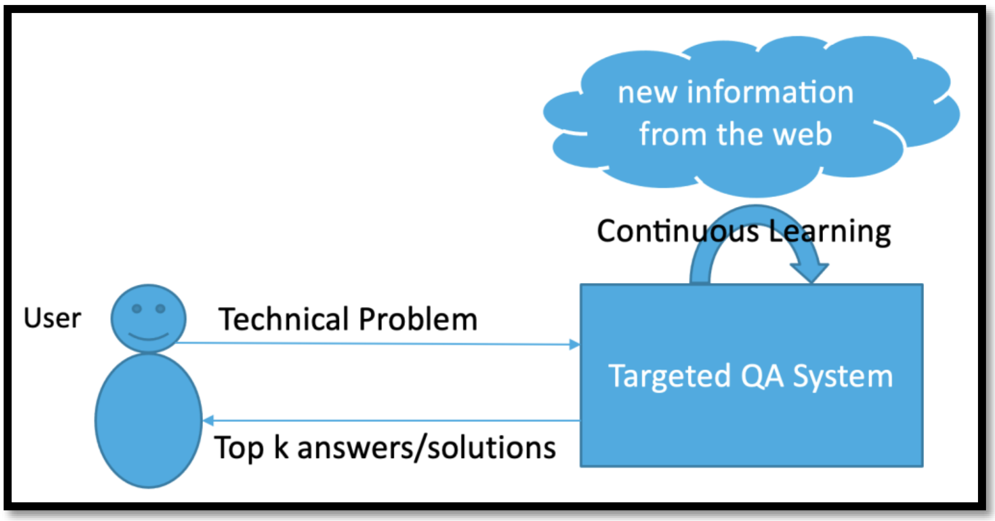


Figure 1: Our targeted LL-CSB system

The lifelong flow in our system consists of continuously extracting new problem-solution pairs from the online forum along with other info related to them, matching the new problem with existing problems in the KG, and finally adding the extracted info into the KG while linking the problem to similar existing problems there. The KG is implemented using the Neo4j graph database, and the matching of similar problems during the lifelong flow and the answering flow is done through a hashing TFIDF vectorizer. This vectorizer saves us the cost of recomputing the vector representations of all problems every time we add a new one. The similarity is then calculated based on the cosine distance between those vectors. Finally, the answering flow involves matching the top similar problem from the KG to the user's problem or query and returning its solution and the solutions of the similar problems linked to this problem based on the degree of this similarity.

To evaluate the impact of the LL-CSB, it's tested in comparison to a static learned system that's missing the LL (baseline CSB). For that, we need a simulator that simulates more knowledge/learning being added over time and more queries being applied over time. The latter is applied for the baseline model to evaluate its performance with more user queries, while the LL-CSB is evaluated for all those queries as more knowledge is being learned. For the data split, we have the KG dataset i.e., the data that will be eventually added to the KG, and an evaluation dataset consisting of user queries, relevant set of problems to those queries, and irrelevant set of problems to those queries. These relevant and irrelevant sets will be added to the KG while the user queries will be used for evaluation. We use common IR metrics for evaluation, namely recall, precision, F1 score, mean average precision (MAP), and Normalized Discounted Cumulative Gain (NDCG). Our results showed the superiority of the LL-CSB over the baseline CSB in the mentioned metrics. Additionally, we compare our LL model to other state of the art LL models in terms of application to verify its uniqueness.

Our thesis contributions consist of:

1. Designing a KG to represent network support discussions where this KG should provide a seamless learning/updating mechanism (ease of updates)
2. Designing a LL flow that permits the CSB to continuously update its KG with new knowledge form the web while simultaneously linking the newly added problems to existing problems in the KG
3. Designing a low-cost method for the continuous update of the answering model with the newly acquired knowledge
4. Designing an approach to validate the effectiveness of the lifelong process applied to support bots

This report is divided as follows: chapter 2 discusses the previous related work in CSBs and LL for chatbots. Chapter 3 presents the methodology followed for the three main units in our system (LL flow, KG, and answering integration/flow) in addition to their implementation and the evaluation methodology used. In chapter 4, we discuss the dataset used, how we formed the evaluation dataset, the experimental setup followed, and the results of the conducted experiments along with their analysis/discussions. The thesis is then concluded in chapter 5 with a list of possible future work.

CHAPTER 2

LITERATURE REVIEW

2.1. Chatbots for Customer Support

Several work has targeted building chatbot systems for customer support considering the value these bots convey to the businesses incorporating them and to their clients.

Most of these older chatbots were based on pattern matching i.e. matching with templates to produce a response like in Eliza [4], Alice [5], and Chat Script [6]. Other chatbots use rule-based techniques where input keywords or features are mapped to their outputs. These chatbots include Cleverbot [7], Chatfuel [8], and Watson [9]. Some of these pattern-matching and rule-based chatbots suffer from generating static responses. Chakrabarti et. al. [44] used Grice's cooperative maxim and goal fulfillment maps to build troubleshooting chatbots with longer and more meaningful conversations. On the other hand, using Natural Language Understanding (NLU) to identify valuable information or keywords that can be matched to certain outputs has proven to be very useful in some systems like LUIS [14], Dialog flow [15], and Amazon Lex [16]. It however limits the chatbot sometimes to certain languages.

Several chatbots, like Mitsuku [10], have been developed based on Artificial Intelligence Modelling Language (AIML) which is an XML based markup language consisting of categories and possible replies within them. Ghose et. al. [11] uses ALICE [5] for advising (in education) where the information source has the form of entities and links connecting them with a similar structure to AIML. Reshmi et. al. [13] integrated AIML with a knowledge base as a database consisting of basic facts that can be used if something is not found directly using the AIML (a modified version). If missing information is encountered, the chatbot asks the user to clarify it. Thomas et. al. [12] developed a chatbot system that answers general questions using AIML and service specific questions or those AIML could not answer using LSA (Latent Semantic Analysis). The latter is trained using a Frequently Asked Questions (FAQs) dataset by dividing the documents and creating a matrix showing word similarities based on their presence in the documents. The right answer is selected by computing the similarity between the question and the FAQs. This strategy is known as information retrieval (IR) or answer retrieval where the user's question or problem is matched to similar problems in a knowledge base subsequently returning their respective solutions (a ranking method). Another IR-based chatbot that uses FAQ pages from the web was developed in [26] where question/answer Q/A pairs were automatically extracted from the collected pages, and the users' questions were answered by detecting similar questions in the Q/A pairs using a syntactic tree matching approach. Online discussion forums were also used in [27] to create a chatbot's knowledge as <thread-title, reply> pairs that are extracted and ranked using SVM.

Some papers have used neural networks for chatbots. The authors in [17] introduced a sequential matching framework (SMF) where RNNs were used on context-response matching vectors to model the relationships between statements in a context. RNNs were also used in [18] for question-answering and in [19] where a seq2seq model was used with GloVe word vectors and trained using cross-entropy error. The latter proved to work well with yes/no questions, but attention mechanisms were preferred in other cases to treat sentences independently. Sutskever et. al. [20] used attention mechanisms to capture several parts of the input and applied neural machine translation with multilayered Long Short-Term Memory (LSTM). Vinyals et. al. [21] made use of a seq2seq framework to map sequences and extract knowledge from a domain specific dataset for IT troubleshooting (computer issues) and from a larger and general noisy domain dataset of movie subtitles. LSTM was used to develop both models to avoid the vanishing gradient problem in RNN. The IT model was able to find solutions to technical issues through conversations. Boyanov et. al. [24] used online forums to train a seq2seq model for answering. The relevant sentences were extracted by matching similar ones from the questions and answers using the cosine similarity of their word2vec embeddings. The idea was to train the conversational agent from scratch on the forum data alone. To deal with the seq2seq models' limitations of generating long and incoherent responses due to the fixed length of the decoder's hidden state vector, Shao et. al. [22] introduced a seq2seq model with stochastic beam-search decoding method. Attention-based seq2seq models were used in [23] to improve input-output alignment and understand queries, user interactions, and ad recommendations. Bidirectional Encoder Representations from Transformers (BERT) was used in [25] where given a conversation history and a passage that contains the answer, the bot extracts the relevant answer to a new question from that passage. The relevant passage is chosen based on IR techniques.

All the mentioned chatbots require human intervention in their training and evaluation procedures and are missing the continual learning that allows them to quickly adapt to new information needed to improve their performance.

2.2. Lifelong Chatbot and Information Extraction Systems

Ample recent work has been aiming at allowing a certain machine learning system to continuously learn new knowledge on its own. Our focus is on lifelong chatbots attempting to learn new knowledge continuously from a certain source. Most of the mentioned chatbots rely on knowledge graphs (KGs) which have the form of (entity, relation, entity).

In [29], Shi et al. was able to handle encountering unknown entities by inference using external text corpus, but was still incapable of dealing with unknown relations. Some knowledge base population (KBP) techniques have been discussed in [30]. They target finding new facts about entities in the KB and augmenting them with that KB. Hence, they cannot handle unknown entities, but can only find new knowledge based on the existing entities in the KB.

Several recent papers are trying to build chatbots that can learn from their interactions with the users. Mazumder et al. [31][33] implemented a continuous learning chatbot by developing the knowledge base completion (KBC) system that infers new facts (knowledge) represented as triples (s: source entity, r: relation, t: target entity) from a closed-world into an open world assumption (open-world knowledge base completion OKBC [29]) that accepts new facts with unknown entities or relations. The system tries to infer the new facts from these unknowns by asking the user to fill the missing links, thus forming a lifelong interactive learning and inference (LiLi) system. Another OKBC approach is introduced in [34] which aims at discovering facts for unknown entities and adding them to the KB or KG based on KG embeddings (embedding-based knowledge graph completion). Abujabal et. al. [32] tried to handle and learn new and different syntaxes for a question by getting semantically close questions from the templates each time the system encounters a question asked in a different syntax, then allowing the user to vote for the right answer among them to be added to the templates while periodically retraining the models. A self-feeding chatbot in a dialogue setting was developed in [35].

This chatbot can detect the satisfaction in the user's response: if the user is satisfied with the replies, their responses are used as new training data, otherwise if not, a feedback is requested from the user so that the machine can take the correct scenario instead and re-train with it. In a movie QA domain, the bot in [37] improved its performance through user interaction by learning to ask questions in offline supervised settings and online reinforcement learning settings. Moreover, an ongoing project [36] is also aiming at developing a chatbot that can learn from its conversation with the user. However, our main focus is a chatbot that continuously learns from the web, not from its conversation with the user.

Continuously extracting knowledge from the web is also known as lifelong information extraction (IE). The NELL system introduced in [39] was able to add the continuous learning part with facts extracted from the web. It consists of several models including classifiers, models that execute inference based on embeddings, and an ontology expander. These models work together with coupling constraints to widen a KB with facts extracted from the web. The updated knowledge is then used to retrain the models to guide the following extraction cycles. Similarly, ALICE [38] performs lifelong information extraction from the web by utilizing the entity-relation-entity triples extracted and KBs like Wordnet to find concepts (general categories of entities) and relations between them. Lifelong IE has been used for aspect extraction and opinion mining as well, where [40] used lifelong learning and similarity measures to try to find correlated words like phone and battery. Shu et. al. [41] also implemented L-CRF (lifelong conditional random fields) as a supervised aspect extraction process to improve new domains by using knowledge about aspects from previous domains. NELL has been used within other systems to apply its lifelong IE concept for morphologically rich languages like Russian [42], where some components were modified, and to train a neural tensor network (NTN) [43] from its KG to evaluate the NTN behavior in a more practical scenario. NTNs work on inferring new knowledge within the KG (KBC) based on word and KG embeddings. Although these mentioned systems apply continuous learning from the web, which is our target, they are focused on basic factual knowledge only. They have not been applied for discussions that handle support and troubleshooting, or for chatbots in customer service/support. Handling such scenarios requires different approaches and techniques, hence introducing new challenges.

CHAPTER 3

METHODOLOGY

3.1. High-level Design of the Targeted Lifelong System

Figure 2 below illustrates the components involved in our targeted QA system from Figure 1. These three components are:

- **Knowledge Graph Representation:** It represents the extracted data as a KG composed of entities or nodes and relations connecting them.
- **Lifelong Model/Process:** It performs the following steps:
  1. Continuously extracts newly posted and solved problems with their solutions and information
  2. Transforms the extracted problems and information into nodes/relations (sub-graph) according to the KG architecture
  3. Matches the extracted problems to problems in the KG
  4. Inserts the new problems with their information/connections and links them to their similar problems in the KG
- **Chatbot Model:** It receives the user's question or problem, matches it to the top-k similar problems in the KG (method discussed later), and returns their respective top-k solutions to the user.

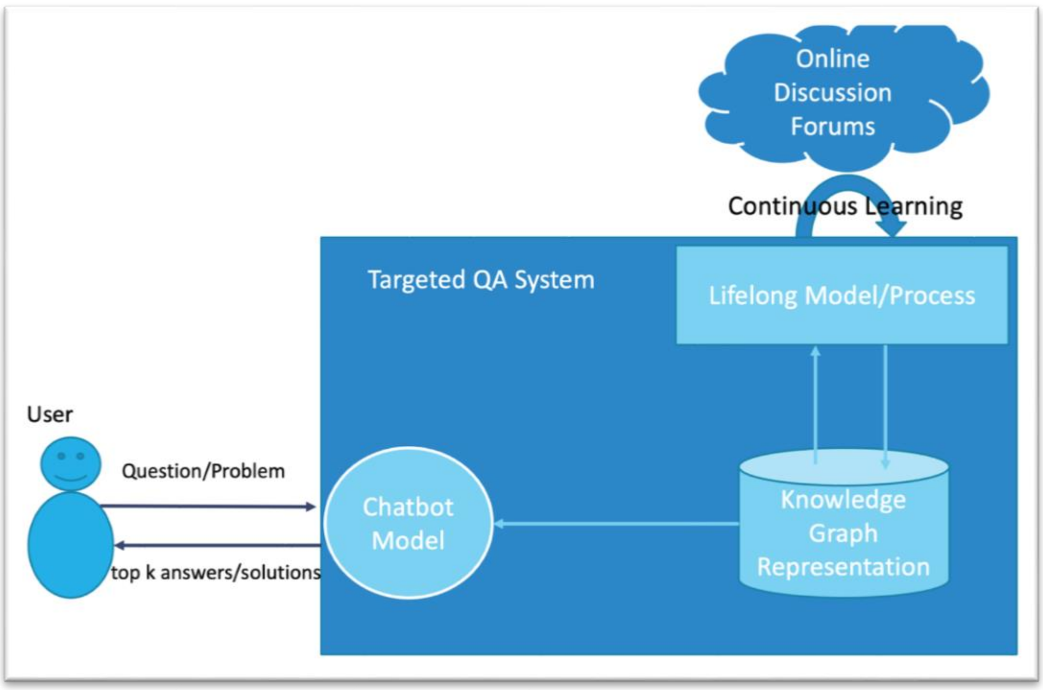


Figure 2: High level design of our system

Each of these components and their implementations are discussed next. First, the online forum's hierarchy and data structure is demonstrated in section 3.2. Section 3.3 depicts the LL process adopted and the implementation details of the similarity measure used for the matching. Section 3.4 presents the KG's final design and execution. The chatbot model (the third component from above) and its integration with the LL flow are described in section 3.5. Finally, an overview of the evaluation method followed for the system is presented in section 3.6.

3.2. Description of Customer Support Data and Hierarchy Structure

Our focus in this thesis is on online discussion forums for CISCO. This section demonstrates how these forums look like and what hierarchy they follow, in addition to the characteristics involved for the problems and solutions. These aspects are important and need to be captured later in the KG design.

Figure 3 below shows the overall hierarchy of the forum starting with domains at the top. Within each domain exists several topics within which several labels and tags can be found. At the last level of the hierarchy are the problems and solutions where each pair is related to one or more labels, but the tags are optional.

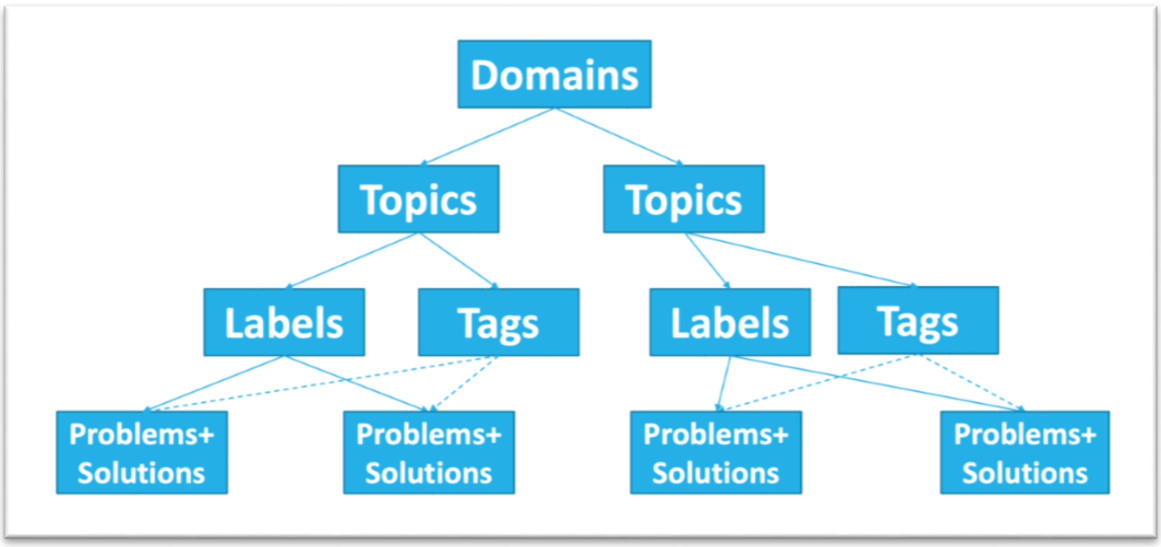


Figure 3: Overall hierarchy of the online forum data

Examples of this hierarchy are shown in **Figure 4**. Some of the domains that appear are Networking, Security, Wireless – Mobility and others[2]. Inside the Wireless – Mobility domain, one topic can be found: "Wireless"[3]. This topic consists of a group of

[2] https://community.cisco.com/t5/technology-and-support/ct-p/technology-support

[3] https://community.cisco.com/t5/wireless-mobility/ct-p/4931-wireless-mobility

labels[4] and tags[5] (shown at the third level in the figure) with problems listed for each of those labels/tags. At the end, one problem within the "5508" tag[6] is selected to further explore the last level of the hierarchy.

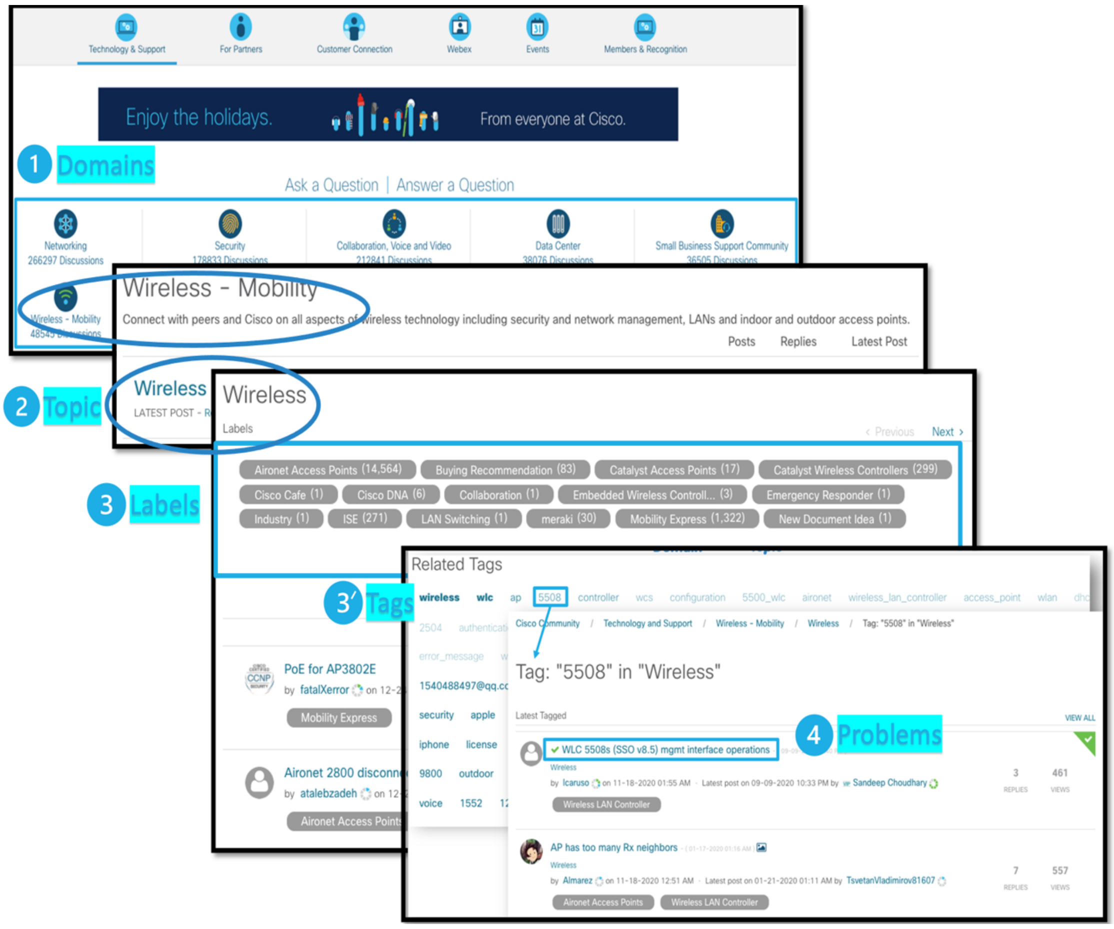


Figure 4: Example of CISCO's online forum hierarchy and data structure

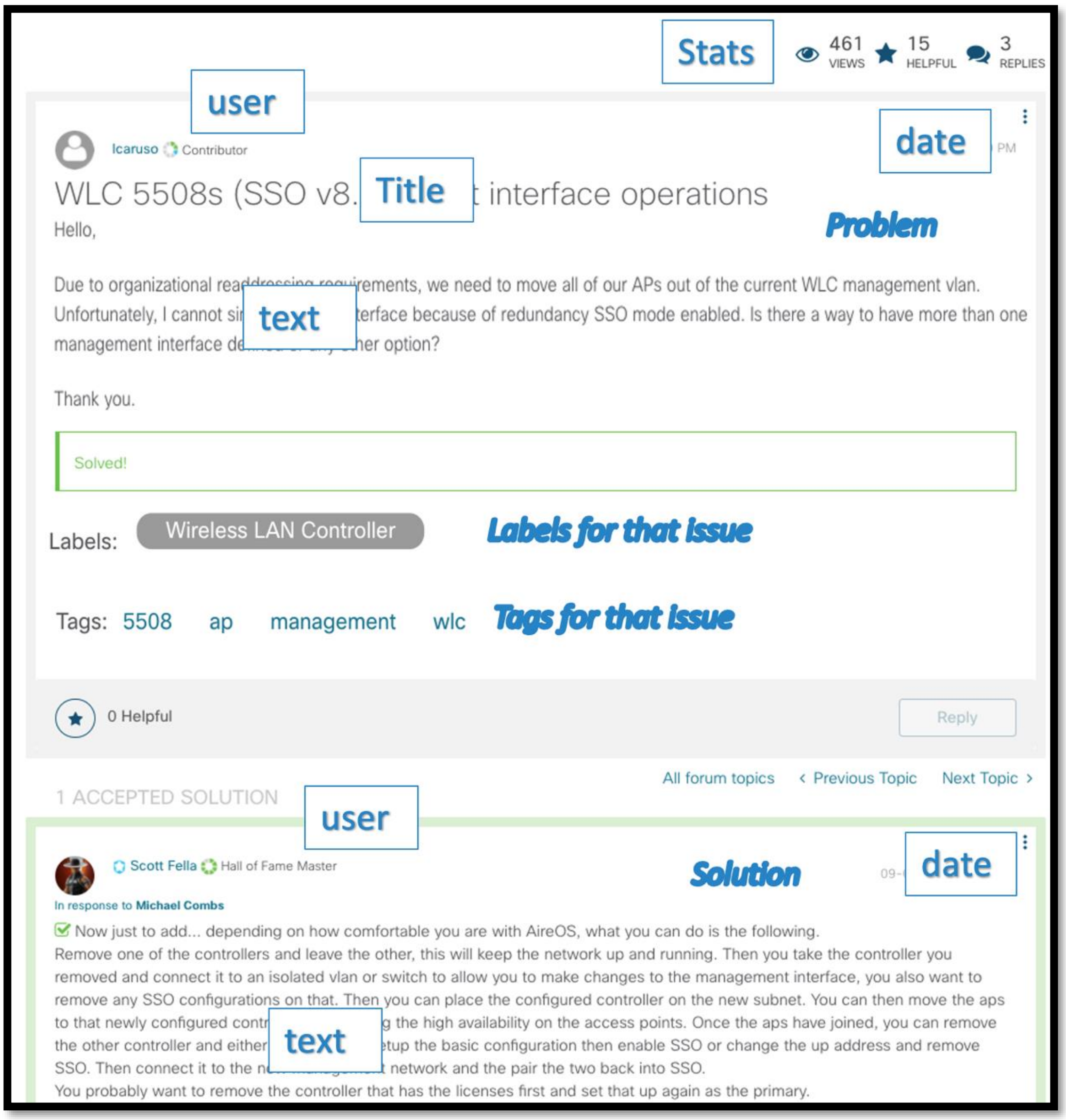
Figure 5 shows that last level which comprises of the problem text and its accepted solution text, in addition to features like title, user (with username and level),

[4] https://community.cisco.com/t5/wireless/bd-p/discussions-wireless

[5] https://community.cisco.com/t5/forums/tagdetailpage/tag-cloud-grouping/tag/tag-cloud-style/related/message-scope/core-node/board-id/discussions-wireless/user-scope/all/tag-scope/all/timerange/all/tag-visibility-scope/public

[6] https://community.cisco.com/t5/tag/5508/tg-p/board-id/discussions-wireless

date, and stats like the number of views, the number of users who found it helpful, and the number of replies.



461 VIEWS 15 HELPFUL 3 REPLIES

Icaruso Contributor

PM

WLC 5508s (SSO v8. t interface operations

Hello,

Due to organizational rea ements, we need to move all of our APs out of the current WLC management vlan. Unfortunately, I cannot si terface because of redundancy SSO mode enabled. Is there a way to have more than one management interface d her option?

Thank you.

Solved!

Labels: Wireless LAN Controller

Tags: 5508 ap management wlc

0 Helpful

Reply

All forum topics ‹ Previous Topic Next Topic ›

1 ACCEPTED SOLUTION

Scott Fella Hall of Fame Master

09-

In response to Michael Combs

Now just to add... depending on how comfortable you are with AireOS, what you can do is the following.
Remove one of the controllers and leave the other, this will keep the network up and running. Then you take the controller you removed and connect it to an isolated vlan or switch to allow you to make changes to the management interface, you also want to remove any SSO configurations on that. Then you can place the configured controller on the new subnet. You can then move the aps to that newly configured contr g the high availability on the access points. Once the aps have joined, you can remove the other controller and either tup the basic configuration then enable SSO or change the up address and remove SSO. Then connect it to the n network and the pair the two back into SSO.
You probably want to remove the controller that has the licenses first and set that up again as the primary.

Figure 5: A sample of a forum problem, its solution, and characteristics[7]

[7] https://community.cisco.com/t5/wireless/wlc-5508s-sso-v8-5-mgmt-interface-operations/m-p/4148629

3.3. Design of Lifelong Learning Logic Flow for Continuous Knowledge Update

Figure 6 below shows the lifelong flow. This flow starts by extracting a problem-solution pair from CISCO's online forum. The extracted problem is first compared to all the existing problems with three possible scenarios based on the level of similarity of the extracted problem with any of the existing ones:

1. Identical problems are substituted with the more recent one.
2. Similar but non-identical problems are kept and linked together with the link expressing the degree of similarity as well.
3. Unsimilar problems are just added to the KG without being linked to other problems.

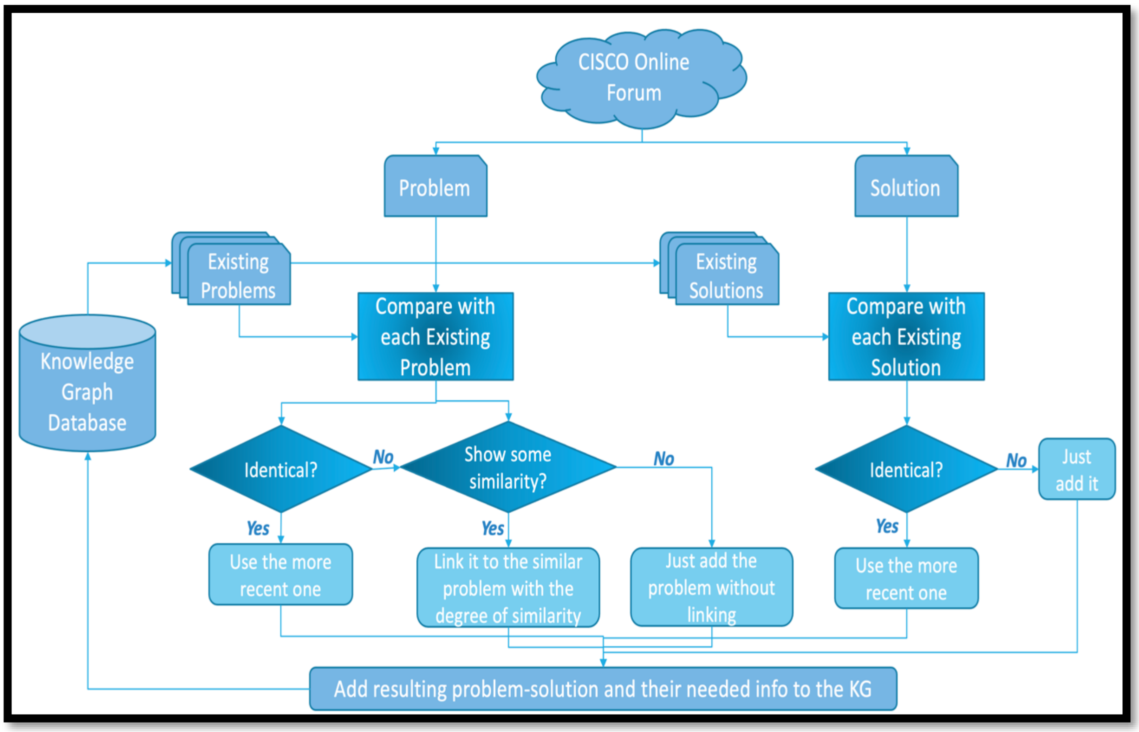

CISCO Online Forum
Problem
Solution
Existing Problems
Existing Solutions
Compare with each Existing Problem
Compare with each Existing Solution
Knowledge Graph Database
Identical?
No
Show some similarity?
No
Identical?
No
Just add it
Yes
Yes
Yes
Use the more recent one
Link it to the similar problem with the degree of similarity
Just add the problem without linking
Use the more recent one
Add resulting problem-solution and their needed info to the KG

Figure 6: Illustration of the lifelong flow as described in Section 3.3.1

For example, we denote our extracted problem as problem x1. If this problem proves to be almost identical (highly similar) to problem x3 from the KG (first scenario), two alternative actions can be practiced based on both problems' dates (which are extracted alongside the problems). If problem x1 is older than problem x3 (possible in case x1 was solved later and so extracted later), problem x1 will not be added into the KG to avoid duplication, but problem x3 will be used as its substitute. This means that when adding problem x1's solution and other information, they are linked to problem x3 as the substitute of problem x1. Otherwise, if problem x3 is older than problem x1, the former is replaced by the latter while keeping all its existing connections.

On the other hand, if no identical problem is found to this extracted problem x1, but one or more problems from the KG show high similarity with it, then x1 is added and further linked to those similar problems, with the link reflecting the degree of similarity as well (second scenario). Finally, for the third scenario, if problem x1 shows no similarity with any of the existing problems, it is simply added to the KG without linking it to any other problem.

For the solution, the system only checks if an identical solution was found to the extracted solution. If so, the more recent one is used similar to what is done for identical problems. This implies that only similar problems are linked to each other (not similar solutions) since looking at the problems alone is sufficient. The steps on the solution's side are only to avoid duplication.

Eventually, the resulting problem-solution pairs are added to the KG (or their identical substitutes are used instead) along with their relative information, then connected to each other. The nodes and relations used are discussed in section 3.4. .

For deciding the similarity in the explained flow (and in the answering later), TFIDF was used. It stands for Term Frequency (TF) - Inverse Dense Frequency (IDF). TFIDF represents documents, or in our case, problem/solution texts as vectors based on the frequency of the words used (TF, eq.1) and the importance of the words used (IDF, eq.2). Less common words are the ones considered important where for example, the word "the" receives a low IDF score for being very common. The equations for computing TF and IDF are shown below [45]. After reaching the vector representations for the documents or problems/solutions, similarity is realized by computing the **cosine distance** between those vectors. Note that after experimenting with different thresholds, we determined that a cosine similarity above 0.9 indicated "identical" problems/solutions, and a similarity between 0.1 and 0.9 indicated "similar" problems.

$$TF = \frac{number\ of\ repetitions\ of\ a\ word\ in\ a\ document}{number\ of\ words\ in\ a\ document}\ (1)$$

$$IDF = \frac{\log(number\ of\ documents)}{number\ of\ documents\ containing\ the\ word}\ (2)$$

TFIDF was invented for document search where it provides the most relevant results to a search query [46]. It matches the results to the query based on word similarity. In our case, specific technical terms are of great importance where the more common these technical terms are between two documents, the more these documents should be similar or tackling similar problems. For that reason, TFIDF can be of great use. Moreover, it is more suitable and less complex for our system since it is:

- Easy to use: TFIDF does not require training but only transforming the documents into their vector representations based on the total words in all documents. This further means that no annotated dataset is needed which is difficult to get in our case.
- Important in evaluation: TFIDF was additionally used later to form the evaluation dataset for the experiments (section 4.1. Dataset).

- Allows optimization: TFIDF facilitates LL by providing room for optimizing the learning process through using the hash vectorizer which will be explained in the next section.

In the future, more advanced solutions can be used for the similarity technique, most important of which is word embeddings that can be extracted from language models and deep learning models. Changing the similarity model used does not affect the LL idea or flow. What matters is that a low-cost solution is found for the automatic update of the chosen model to the continuous knowledge addition.

In the standard TFIDF vectorizer, the size of the vectors is determined by the number of distinct words in all documents. This means that as more documents or in this case, problems are acquired, the vectors need to be recomputed for all the existing problems as well since more words are added. This is costly for our lifelong flow. The Hash Vectorizer solves this issue by fixing the size of the vector through applying a hashing function to the word frequency counts in each document. It is a low-memory scalable solution that is very suitable for streaming. To avoid collisions in hashing, the vector size was chosen to be large enough for our system (2^{18}). The only disadvantage of the Hashing Vectorizer is that it does not apply the IDF part since it cannot hold any state. This is usually compensated for by applying preprocessing to the text, which is demonstrated in section 4.1. Dataset. [47]

Having a pre-defined, fixed vector size means that no vector re-computation is required for any of the existing problems. Whenever a new problem is extracted, its computed vector representation can be simply appended to the matrix of vectors for all problems. This matrix is then saved aside and used for all future comparisons of a new problem to the existing problems in the KG (in the lifelong flow and the answering flow).

This is significant since it means retrieving any problems from the KG is no longer needed in both flows to compute similarity; the cosine distance can be directly applied with the saved matrix. This is demonstrated in **Figure 7** below.

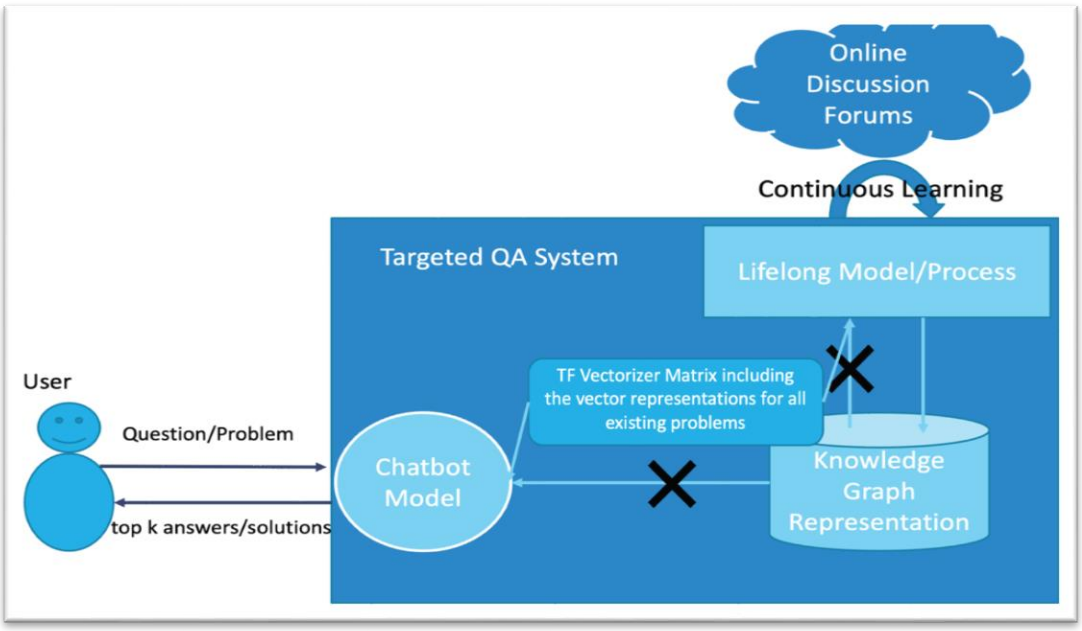


Figure 7: Low-cost solution using the Hash Vectorizer

3.4. Design and Implementation of Knowledge Graph Database

3.4.1. Requirements

KGs are the suitable choice to represent our data for two main reasons. The first reason is that KGs facilitate LL where new instances can be directly and easily added without affecting the previous version of the KG. Therefore, KGs seem to be the most appropriate choice for a system that’s continuously growing over time [48]. The second reason is that due to the connections that can be achieved with KGs, the knowledge of the system can be improved beyond the extracted data (more insights) [3].

The KG design should represent the problems and solutions in a direct way so that it facilitates retrieving the needed ones. This includes being able to easily reach similar problems to a certain problem and retrieve their respective solutions. Moreover, the KG design should allow linking new problems to similar existing ones and capture the hierarchy presented in **Figure 3** and the characteristics shown in **Figure 5**. These characteristics represent important information about each problem-solution pair and can be used to improve the lifelong and answering models now and in the future.

3.4.2. Final Design

Figure 8 below shows the final design of our KG.

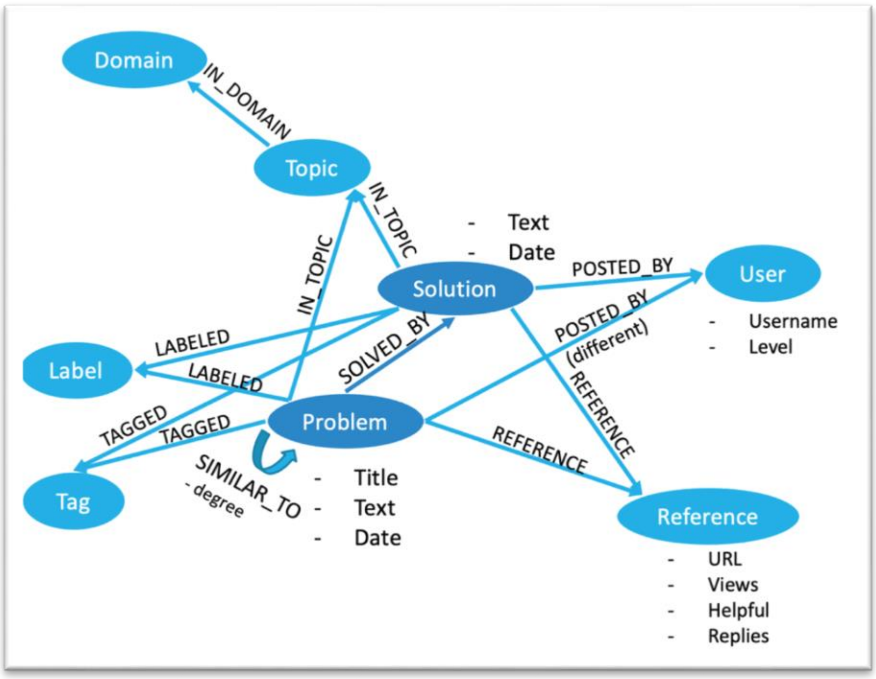
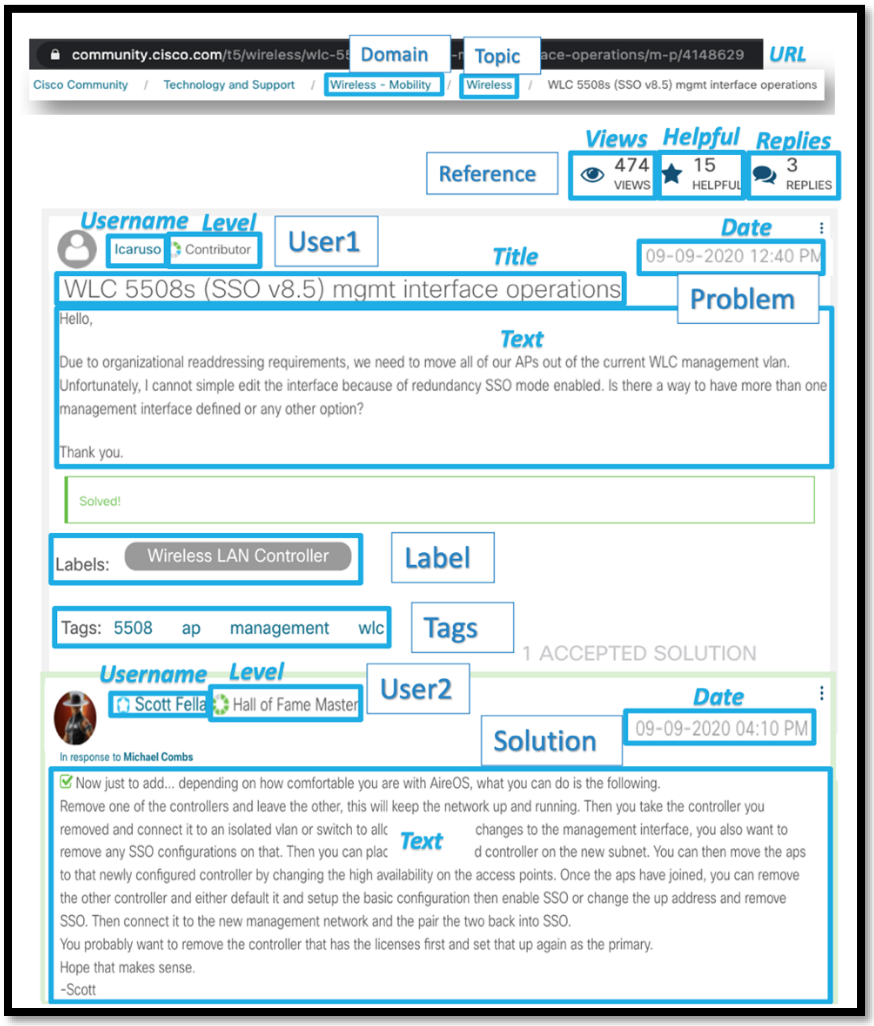


Figure 8: Final knowledge graph design

An example exhibiting the information captured in the nodes is presented in **Figure 9**. It is important to have the problems and solutions as separate nodes (instead of attributes in another node or relation) since they are the main entities in our system and thus, need to be represented and reached easily. The attributes for these two nodes are “Text” and “Date”, in addition to “Title” in problem. They’re connected by the “SOLVED_BY” relation and similar problems can be connected by the “SIMILAR_TO” relation with the degree of similarity stored in the “degree” attribute of that relation. For each of the problem and solution, their respective “Users” are represented as nodes with “Username” and “Level” as their attributes. Problems and solutions are also connected to nodes representing their respective “Labels”, “Tags”, “Topic”, and “Domain”. Finally, the “Reference” of the problem-solution pair is marked in a separate node consisting of the “URL”, number of “Views”, number of users who found this “Helpful”, and the number of “Replies”.



community.cisco.com/t5/wireless/wlc-55 -r ace-operations/m-p/4148629

Cisco Community / Technology and Support / Wireless - Mobility / Wireless / WLC 5508s (SSO v8.5) mgmt interface operations

474 VIEWS 15 HELPFUL 3 REPLIES

lcaruso Contributor

09-09-2020 12:40 PM

WLC 5508s (SSO v8.5) mgmt interface operations

Hello,

Due to organizational readdressing requirements, we need to move all of our APs out of the current WLC management vlan. Unfortunately, I cannot simple edit the interface because of redundancy SSO mode enabled. Is there a way to have more than one management interface defined or any other option?

Thank you.

Solved!

Labels: Wireless LAN Controller

Tags: 5508 ap management wlc

1 ACCEPTED SOLUTION

Scott Fella Hall of Fame Master

09-09-2020 04:10 PM

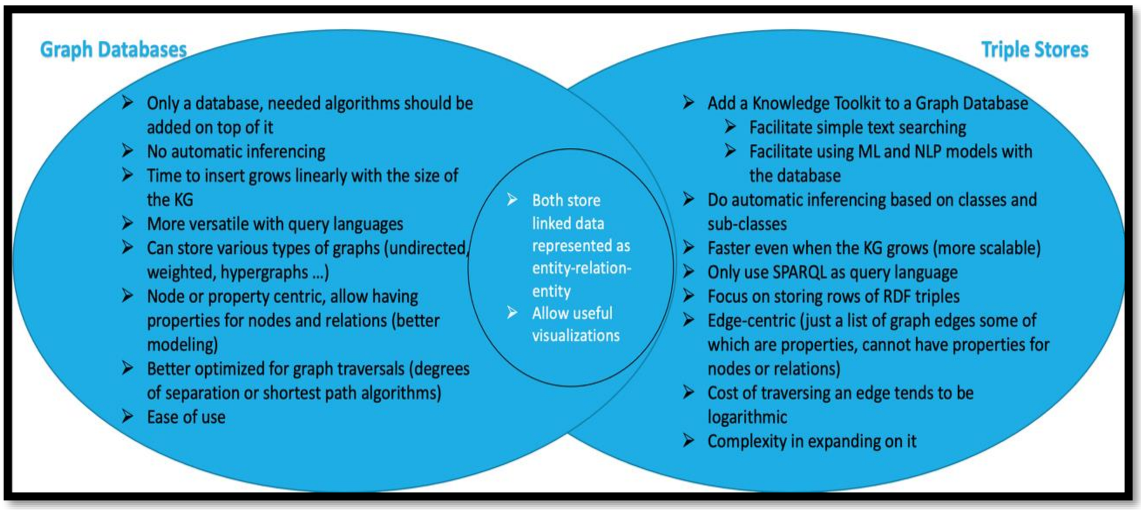
In response to Michael Combs

Now just to add... depending on how comfortable you are with AireOS, what you can do is the following.
Remove one of the controllers and leave the other, this will keep the network up and running. Then you take the controller you removed and connect it to an isolated vlan or switch to allc changes to the management interface, you also want to remove any SSO configurations on that. Then you can plac d controller on the new subnet. You can then move the aps to that newly configured controller by changing the high availability on the access points. Once the aps have joined, you can remove the other controller and either default it and setup the basic configuration then enable SSO or change the up address and remove SSO. Then connect it to the new management network and the pair the two back into SSO.
You probably want to remove the controller that has the licenses first and set that up again as the primary.
Hope that makes sense.
-Scott

Figure 9: Example of the information captured in the graph design[7]

3.4.3. Implementation: Neo4j Graph Database

For the implementation of the KG, two options are graph databases and triplestores. Below (**Figure 10**) is a comparison between the two. Graph databases greatly suit our system implementation since they provide better modeling with the option of having attributes which are deeply needed in our graph as shown in the previous section. Graph databases are also convenient due to their capability of storing various types of graphs, in addition to their easy and straightforward usage. Switching to triplestores in the future might help in the scalability of the system.


Graph Databases

- Only a database, needed algorithms should be added on top of it
- No automatic inferencing
- Time to insert grows linearly with the size of the KG
- More versatile with query languages
- Can store various types of graphs (undirected, weighted, hypergraphs …)
- Node or property centric, allow having properties for nodes and relations (better modeling)
- Better optimized for graph traversals (degrees of separation or shortest path algorithms)
- Ease of use

- Both store linked data represented as entity-relation-entity
- Allow useful visualizations

Triple Stores

- Add a Knowledge Toolkit to a Graph Database
  - Facilitate simple text searching
  - Facilitate using ML and NLP models with the database
- Do automatic inferencing based on classes and sub-classes
- Faster even when the KG grows (more scalable)
- Only use SPARQL as query language
- Focus on storing rows of RDF triples
- Edge-centric (just a list of graph edges some of which are properties, cannot have properties for nodes or relations)
- Cost of traversing an edge tends to be logarithmic
- Complexity in expanding on it

Figure 10: Graph Databases vs. Triplestores [49]–[51]

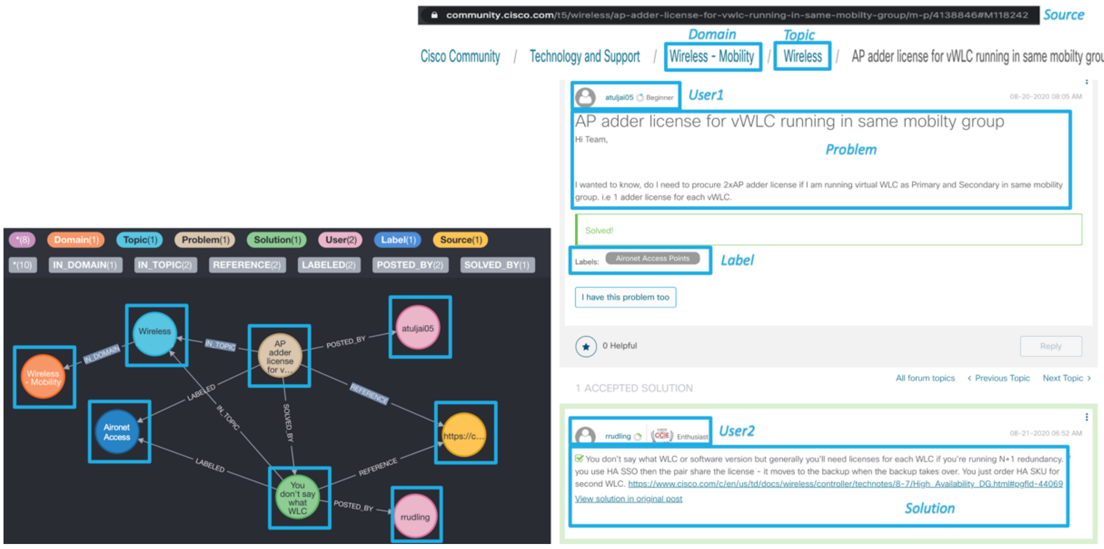
The Neo4j[8] (NoSQL) graph database was selected for our implementation. It's one of the leading graph databases and has several advantages that suit our system including [52]:

- Flexible data model

[8] https://neo4j.com/

- High performance due to Native Graph Storage and Processing, and high scalability
- Easy to learn and implement
- Big active community
- Can be used with python

Figure 11 below shows one example extracted to Neo4j (attributes not shown inside each node in the picture), and **Figure 12** shows how the graph grows as we add more knowledge and connections via the LL flow. The left picture in **Figure 12** highlights one of the "SIMILAR_TO" relations where the degree of similarity can be observed at the bottom of the picture as the attribute "degree".



community.cisco.com/t5/wireless/ap-adder-license-for-vwlc-running-in-same-mobilty-group/m-p/4138846#M118242

Cisco Community / Technology and Support / Wireless - Mobility / Wireless / AP adder license for vWLC running in same mobilty group

atuljai05 Beginner 08-20-2020 08:05 AM

AP adder license for vWLC running in same mobilty group

Hi Team,

I wanted to know, do I need to procure 2xAP adder license if I am running virtual WLC as Primary and Secondary in same mobility group. i.e 1 adder license for each vWLC.

Solved!

Labels: Aironet Access Points

I have this problem too

0 Helpful Reply

1 ACCEPTED SOLUTION All forum topics ‹ Previous Topic Next Topic ›

rrudling CCIE Enthusiast 08-21-2020 06:52 AM

✓ You don't say what WLC or software version but generally you'll need licenses for each WLC if you're running N+1 redundancy. you use HA SSO then the pair share the license - it moves to the backup when the backup takes over. You just order HA SKU for second WLC. https://www.cisco.com/c/en/us/td/docs/wireless/controller/technotes/8-7/High_Availability_DG.html#pgfId-44069

View solution in original post

Figure 11: Example of a problem and its information extracted and represented in Neo4j[9]

[9] https://community.cisco.com/t5/wireless/ap-adder-license-for-vwlc-running-in-same-mobilty-group/m-p/4138846#M118242

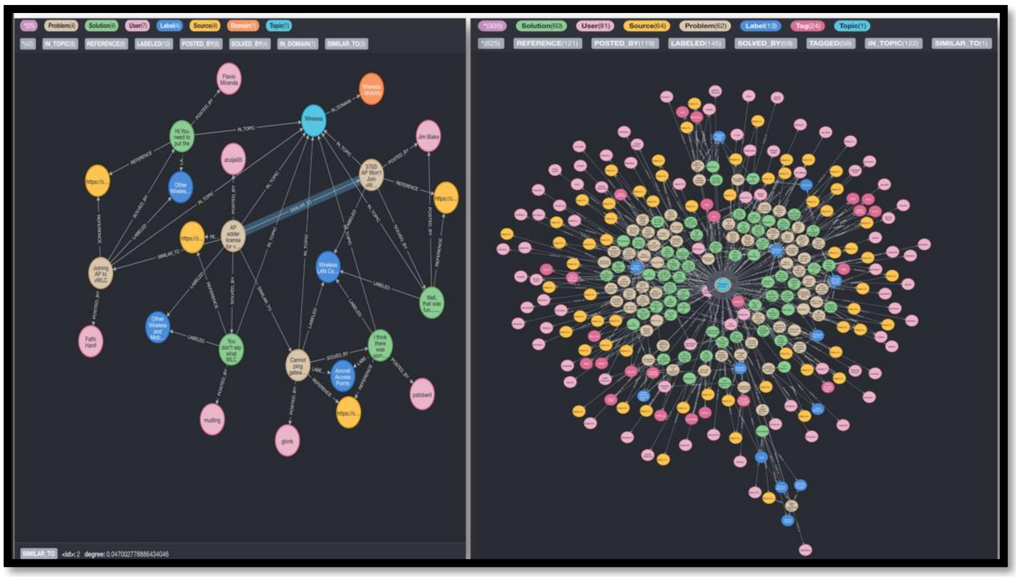


Figure 12: Adding and linking more knowledge with Neo4j (LL Flow)

3.5. Integration of the Answering Flow with the Lifelong Learning Flow

The executed answering flow is shown in **Figure 13** below where the user's question or problem is first compared to the existing problems in the KG. The comparison here is done through TFIDF, specifically the Hashing Vectorizer, similar to how it is done in the LL flow. Likewise, the Hashing Vectorizer allows us here to avoid actually retrieving the existing problems from the KG, and instead, compute the cosine similarity with the saved matrix of these existing problems. If no problems showing a similarity higher than 0.1 with the user's problem are found, no answers are returned, only a "Cannot Answer" reply. Otherwise, the problem showing the highest similarity is chosen. This problem's direct solution or solutions are returned at the first level as the highest ranked solution/s. The number of answers returned (k) can be specified by us or by the user. In addition to the most similar problem's solution, the solutions of the similar problems to this top problem are also returned in order of their similarity with this top problem. If any of the problems has multiple solutions, these solutions are currently returned at the same level (see **Figure 14**). For the future, multiple methods can be followed to rank these solutions like the "Helpful" attribute which can show how many users have found this specific solution helpful.

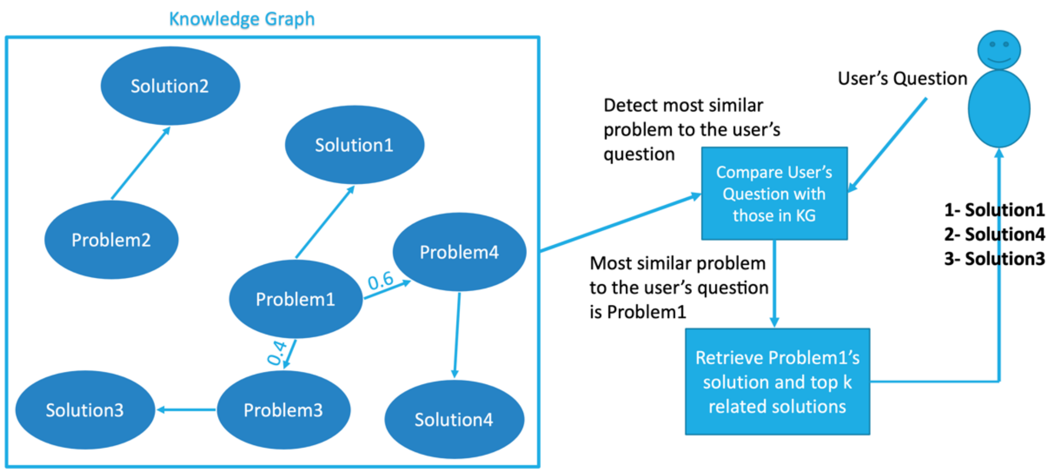
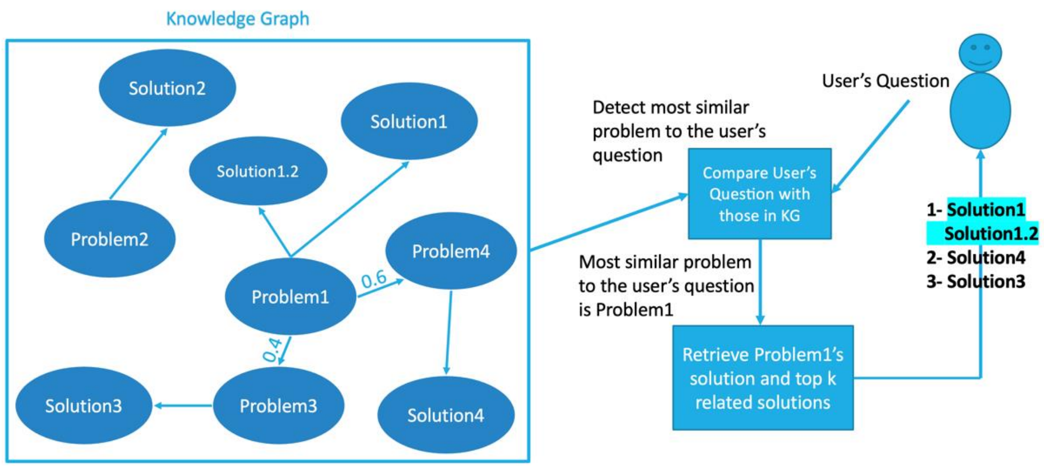
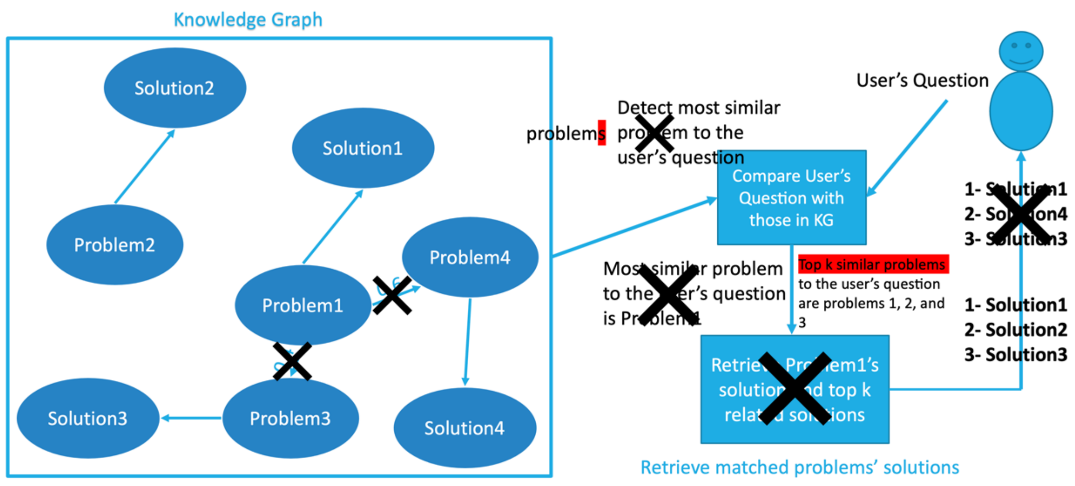

Knowledge Graph
Solution2
Solution1
Problem2
Problem4
Problem1
0.6
0.4
Solution3
Problem3
Solution4
User's Question
Detect most similar problem to the user's question
Compare User's Question with those in KG
1- Solution1
2- Solution4
3- Solution3
Most similar problem to the user's question is Problem1
Retrieve Problem1's solution and top k related solutions

Figure 13: The answering flow


Knowledge Graph
Solution2
Solution1
Solution1.2
Problem2
Problem4
Problem1
0.6
0.4
Solution3
Problem3
Solution4
User's Question
Detect most similar problem to the user's question
Compare User's Question with those in KG
1- Solution1
Solution1.2
2- Solution4
3- Solution3
Most similar problem to the user's question is Problem1
Retrieve Problem1's solution and top k related solutions
Figure 14: Answering with multiple solutions for a problem

If similar problems are not linked in the KG, the answering flow will then be modified to detect the top-k similar problems instead of the top-1 problem. Subsequently, the solutions to these problems are returned in the order of their similarity to the user's problem instead of returning the solutions connected to the top problem. Answering the same example from **Figure 13** without the linked problems is shown in **Figure 15**. This is just to demonstrate how answering differs without linked problems (which is not our case). This is used in the experimental results later (**Error! Reference source not found.**) to prove the value of this linking.



Knowledge Graph
Solution2
Solution1
Problem2
Problem4
Problem1
Solution3
Problem3
Solution4
Detect most similar problems problem to the user's question
User's Question
Compare User's Question with those in KG
1- Solution1
2- Solution4
3- Solution3
Most similar problem to the user's question is Problem1
Top k similar problems to the user's question are problems 1, 2, and 3
1- Solution1
2- Solution2
3- Solution3
Retrieve Problem1's solution and top k related solutions
Retrieve matched problems' solutions

Figure 15: Answering without linked problems

3.6. Design of Evaluation Process for Lifelong Learning for Customer Support

The evaluation consists of three main steps that focus on testing LL in comparison to a static learned system:

1. Compute the performance of the system without LL (baseline)
2. Compute the performance of the system with LL
3. Compare

The idea is that a baseline model without LL is a model that suffers from a lag in its knowledge update process. During that delay, it will receive more and more user questions that it may or may not be able to answer yet, especially the most recent ones. On the other hand, a LL model is a model that avoids this lag by continuously updating itself with the new knowledge emerging from its source of data while answering the same user queries. To mimic this behavior, we utilize a simulator that simulates more queries being applied over time and more knowledge/learning being added over time. This allows us to evaluate the baseline model with more user queries and the LL model on these queries as it continuously learns additional knowledge over time.

Since our baseline model is an information-retrieval (IR) system, we select the most suitable evaluation metrics for such systems [12], [26]–[28], [53]. The chosen evaluation metrics are recall, precision, F1 score, mean average precision (MAP), and Normalized Discounted Cumulative Gain (NDCG) per number of answers returned (k), and their respective equations are presented below (Equations 3, 4, 5, 6 and 7). Relevance here is based on the problem texts for that is what is being compared and matched in our system. In order to be able to compute these metrics, an annotated evaluation dataset is required consisting of user queries, relevant problems to these queries, and irrelevant problems to these queries. The user queries alone are then used to test the system while the relevant and irrelevant problems should be added to the knowledge graph (KG) eventually. These relevant problems are then expected to be returned by our system when their respective user queries are applied, and the evaluation is built upon that.

$$recall@k = \frac{\#\ of\ our\ returned\ answers\ that\ are\ relevant}{\#\ of\ relevant} \quad (3)$$

$$precision@k = \frac{\#\ of\ our\ returned\ answers\ that\ are\ relevant}{\#\ of\ returned\ answers} \quad (4)$$

$$AP@k = \frac{1}{\#\ relevant}\sum_{i=1}^{k} precision@i\ if\ i^{th}\ answer\ is\ relevant\ else\ 0 \quad (5)$$

$$F1@k = \frac{2 \times recall \times precision}{recall + precision} \quad (6)$$

$$NDCG = \frac{DCG}{IDCG} = \frac{\sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)} \; for\ the\ list\ of\ returned\ answers}{\sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)} \; for\ the\ ideal\ order\ of\ that\ list\ according\ to\ relevance\ rel} \quad (7)$$

Regarding NDCG, since the length of the Discounted Cumulative Gain (DCG) varies with the number of returned answers (k), it needs to be normalized by dividing it by the ideal DCG (IDCG) that consists of all possible answers in the KG ordered according to their **actual** relevance (rel) to the query and taken up to position k [53].

On the other hand, the scalability of the LL system should be evaluated as well. This includes measuring the time needed to add and link new problems in the KG and the time taken to answer the user queries with respect to the size of the KG in terms of the number of problems within it so far.

CHAPTER 4

EXPERIMENTS AND EVALUATION

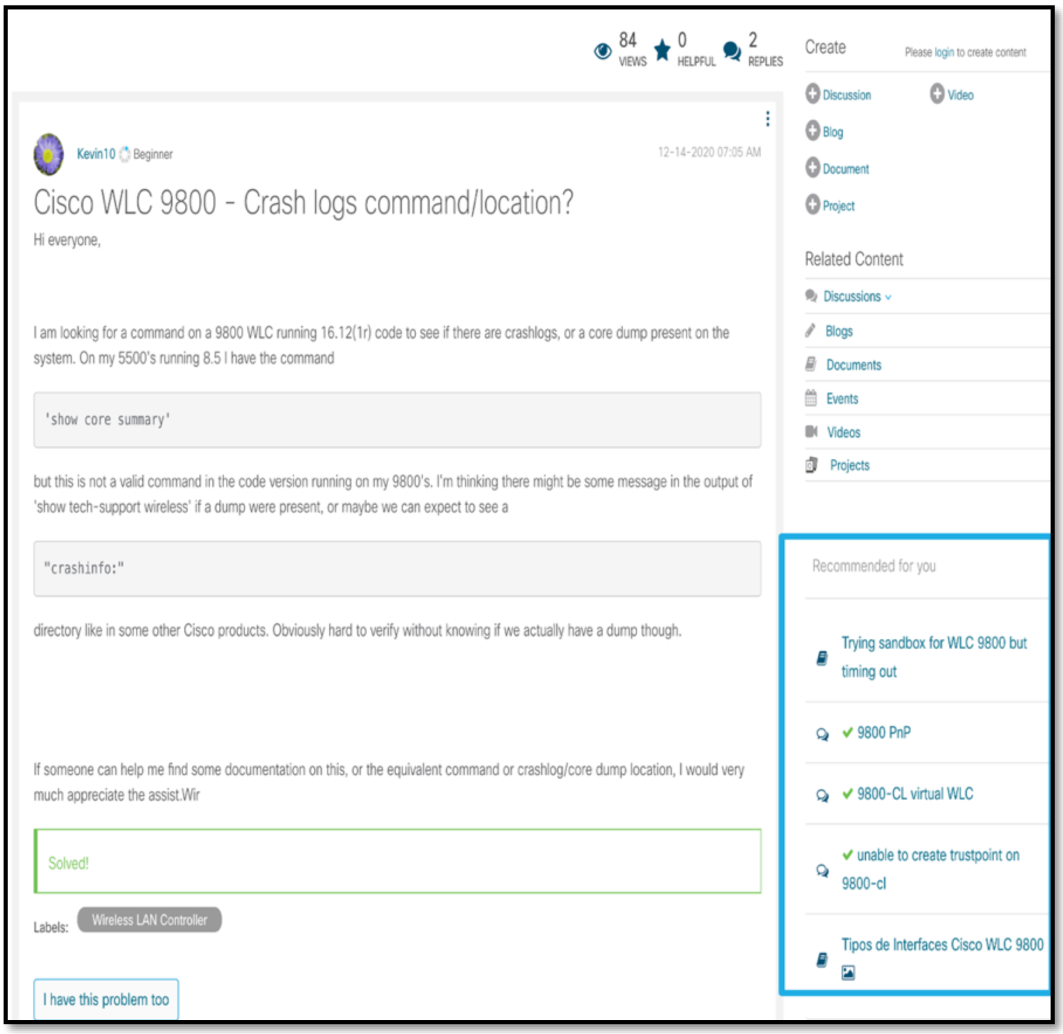
In this section, we describe the dataset used for evaluation and present our experimental setup.

4.1. Dataset

From the wireless topic**Error! Bookmark not defined.**, we crawled all the problem-solution pairs marked as “solved”. This gave us a total of 7945 pairs with their info. In order to enable creating the evaluation dataset, for 1066 out of those 7945 pairs (between 10 and 15%), we crawled their recommended problems from the forum. **Figure 16** shows how each problem receives similar recommended problems on the forum. The amount of crawled recommended problems was 2245, adding up to a total of 10190 crawled problem-solution pairs with their needed information.

The crawling was done using a customizable crawling software called OutWit hub[10], and the results were saved in a csv file with its columns representing the attributes that need to be captured like domain, topic, problem/solution texts and users, title, labels, tags, views, helpful, and replies. Alternatively, in a real-life scenario, this crawling should be scheduled to occur automatically on its own via a running script.

[10] https://www.outwit.com/


84 VIEWS 0 HELPFUL 2 REPLIES

Kevin10 Beginner 12-14-2020 07:05 AM

Cisco WLC 9800 - Crash logs command/location?

Hi everyone,

I am looking for a command on a 9800 WLC running 16.12(1r) code to see if there are crashlogs, or a core dump present on the system. On my 5500's running 8.5 I have the command

```
'show core summary'
```

but this is not a valid command in the code version running on my 9800's. I'm thinking there might be some message in the output of 'show tech-support wireless' if a dump were present, or maybe we can expect to see a

```
"crashinfo:"
```

directory like in some other Cisco products. Obviously hard to verify without knowing if we actually have a dump though.

If someone can help me find some documentation on this, or the equivalent command or crashlog/core dump location, I would very much appreciate the assist.Wir

Solved!

Labels: Wireless LAN Controller

Figure 16: Recommended problems via forum

The preprocessing steps for the problem/solution texts included transforming to lowercase, removing all punctuation, removing one-character words, removing stop-words, and lemmatization. Furthermore, we separated numbers from words; for example, “wlc2709” is transformed into “wlc 2709” since such numbers indicate certain products[11] within CISCO and are thus important to be captured correctly with TFIDF.

[11] https://www.cisco.com/c/en/us/support/wireless/index.html

As previously touched on, the evaluation dataset requires annotated data consisting of queries, relevant problems to these queries, and irrelevant problems to these queries. Since we are missing annotated data, we had to collect this relevant set through two methods. The first is based on the recommended problems via the forum (pre-defined); for that reason, we selected the 1066 problem-solution pairs as the queries since we've crawled their recommended problems which now make-up the relevant set for those queries. The irrelevant set then consists of all the other problems among the total 10190 pairs. Since we want to study the impact of LL alone, we selected those recommended/relevant problems to have somehow similar wording to their respective queries. In other words, in order to make sure that the observed results are only due to the LL factor, the impact of the similarity measure used (TFIDF) should be extracted and removed because it's not our focus in this thesis. However, to further check our results without such TFIDF impact extraction, we used another method to select the relevant problems for these 1066 queries. This method is based on title-answer similarity where TFIDF was applied to the titles and solutions of the 1066 queries on one hand and all the other 9214 pairs on the other hand, and the pairs showing a high cosine similarity (above 0.4) to those queries were selected. We also made sure that no more than 20 relevant problems are included for each query. This second method is based on the idea that if two problems have very similar titles and solutions, these problems should be similar too. A close strategy was followed in [54] to evaluate their IR system. We additionally took into consideration common labels and tags and the number of helpful in this second technique, but the focus was on title-answer similarity.

4.2. Experiment Setup

The formed evaluation dataset should be divided as follows: the user queries should be used alone to evaluate the system, while the relevant and irrelevant sets to these queries should be the ones added to the KG. The data was ordered by the date of the problem to closely mimic the real-life situation. **Figure 17** below shows the process followed in evaluation. The simulator here presents the batches of user queries and the batches of new knowledge to be learned over time. For the baseline model, we add the first 1000 problems alone into the KG and consider we reached the gap here. We then evaluate the model for every 200 new user queries out of the 1066 queries. The lifelong model is evaluated for all those 1066 queries with every 200 new learnt pairs added to its KG starting the 1000 of the baseline model.

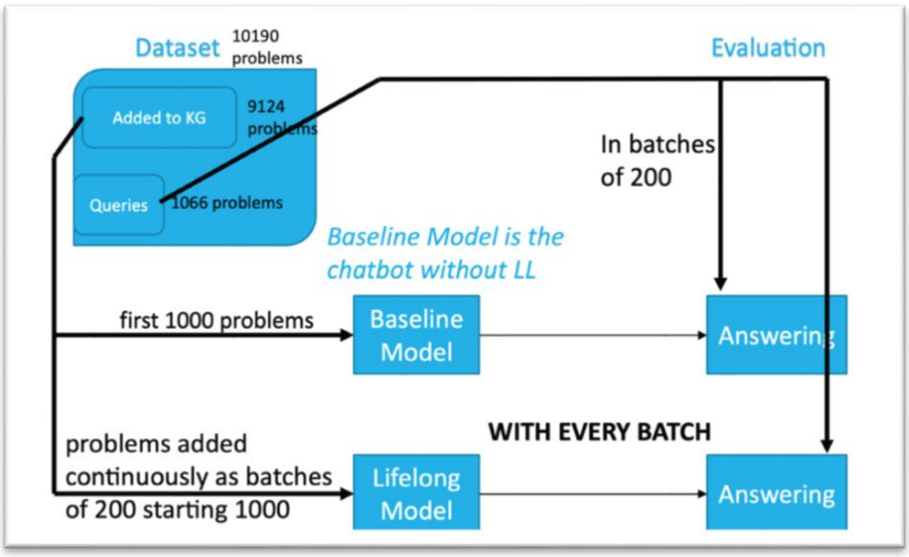


Figure 17: The followed evaluation procedure

As previously mentioned, the KG is created using Neo4j. We implemented all of the LL, answering, and evaluation procedures in python. The neomodel[12] library was employed to connect to and use Neo4j with python, the nltk[13] library for data preprocessing, and sklearn[14] library for applying the TFIDF Vectorizer and the cosine similarity. The evaluation metrics used are recall, precision, F1, MAP, and NDCG as discussed earlier. For NDCG, the relevance rel is taken as 1 for relevant answers and 0 for the irrelevant ones.

4.3. Quantitative Evaluation

4.3.1. Evaluation of the Baseline Model

We present the results of the evaluation for the baseline model in the figures below. It should be pointed out that in this chapter, we're showing results for both evaluation methods used (mentioned earlier) which differ by how relevance was decided for the user queries (based on the recommended problems via the forum or based on title-answer similarity). From **Figure 18** (based on forum-defined relevance) and **Figure 19** (based on title/answer relevance), it's evident that the baseline model performs poorly in all metrics at the beginning, and its performance keeps declining with more user queries for all values of k (k being the number of answers returned and varying between 1, 5, 10, 15, 20, 25, and 30). This proves that the baseline's performance suffers during the lag in update before it gets updated with the new knowledge. **Table 1** specifies the % decrease with every new 200-queries batch for all the metrics.

[12] https://neomodel.readthedocs.io/en/latest/

[13] https://www.nltk.org/

[14] https://scikit-learn.org/stable/

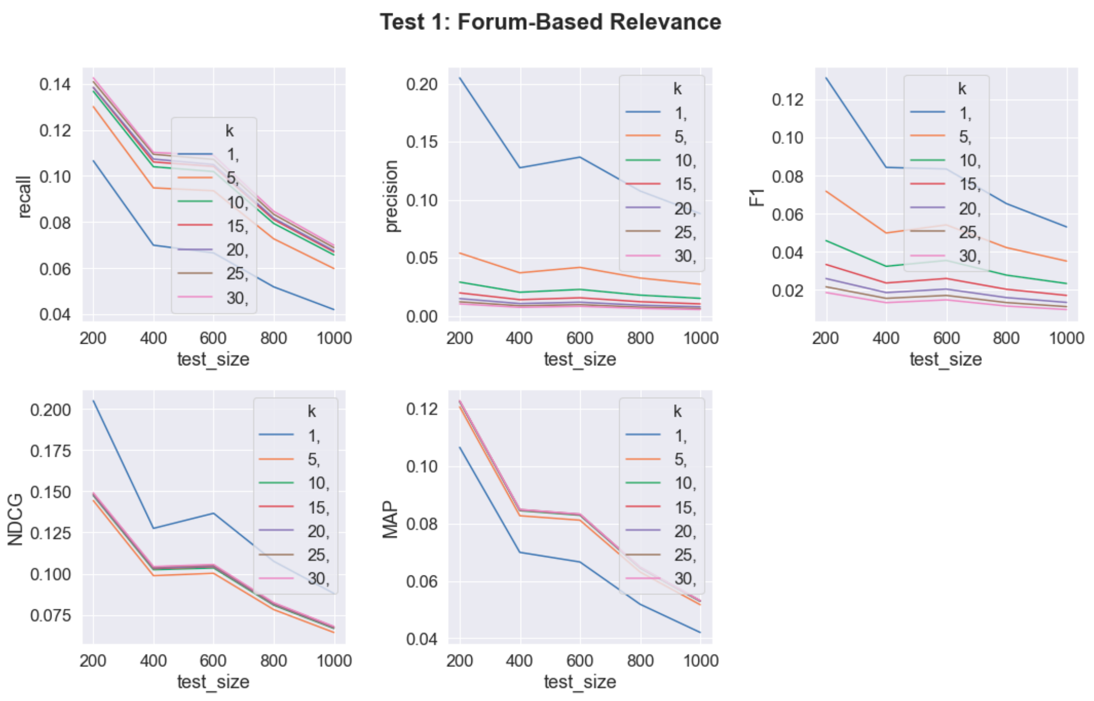


Figure 18: Recall (upper left), precision (upper middle), F1 (upper right), NDCG (lower left), and MAP (lower middle) for baseline model with more user queries based on the first evaluation method and over a range of values for k

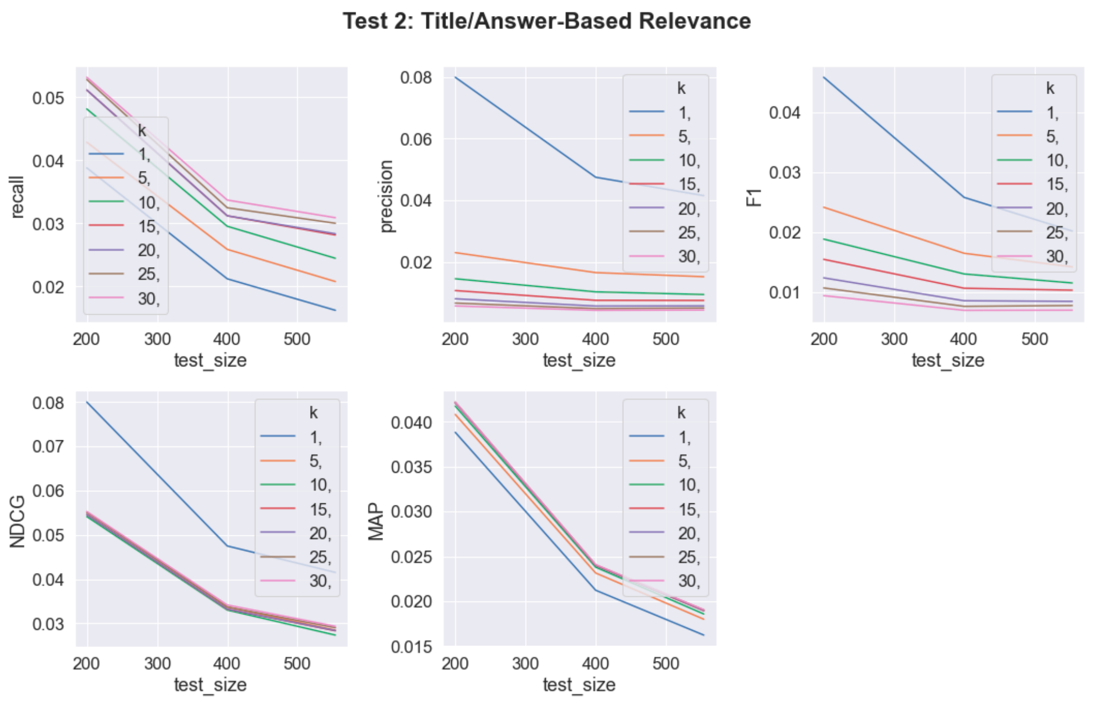


Figure 19: Recall (upper left), precision (middle left), F1 (lower middle), NDCG (upper right), and MAP (middle right) for baseline model with more user queries based on the second evaluation method and over a range of values for k

Table 1: Average (over all values of k) percent decrease per 200 new user queries in the baseline model

	Recall	**Precision**	**F1 Score**	**MAP**	**NDCG**
Test 1	17%	14%	15%	19%	17%
Test 2	27%	17%	20%	33%	27%

Figure 20 below shows the portion of the missed answers that were actually present in the KG and hence, missed by the model itself. The small values of this portion verify that the absence of the answers in the KG due to the learning lag is the main contributor to the model's poor performance.

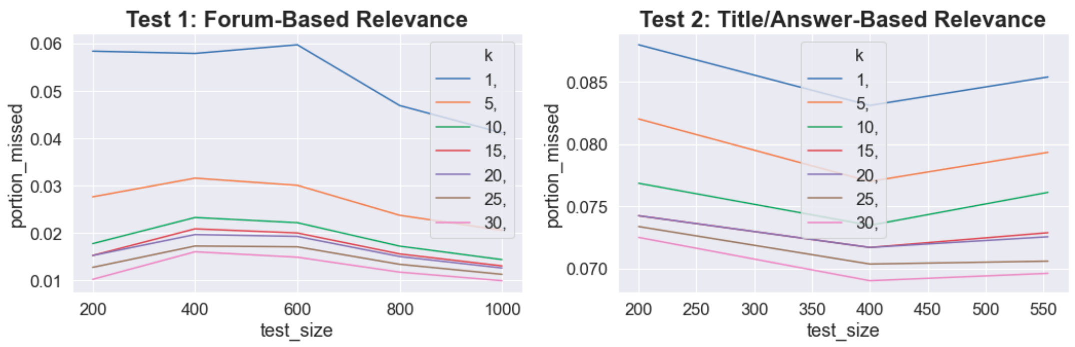


Figure 20: Portion of the missed queries whose answers were actually in the KG

4.3.2. Evaluation of the Lifelong Model

The lifelong evaluation was done for two systems: a system without linked problems in its KG and a system with linked problems (our target system). It was previously demonstrated in section 3.5 how the answering flow occurs without linked problems in the KG and how it differs from the linked case.

4.3.2.1. LL Evaluation for System with Unlinked Problems in its KG

The graphs below show how all five metrics (recall, precision, F1, MAP, and NDCG) of the LL model improve as it learns more knowledge for all values of k (k being the number of answers returned and varying between 1, 5, 10, 15, 20, 25, and 30). **Figure 21** consists of the resulting graphs for the first evaluation method, and **Figure 23** includes those for the second evaluation method. The evaluation was done on all 1066 queries. The x-axis of the graphs expresses the number of problem-solution pairs learnt so far by the model. Since the baseline stopped learning at 1000 pairs, this point can be spotted on the graph and the factor of improvement of the LL model from that baseline with every batch of 200 new learnt pairs can then be computed. This improvement, averaged over the values of k, is shown in **Figure 22** for the first evaluation method and **Figure 24** for the second. The average improvement per batch for all metrics and both tests is presented in **Table 2**.

With respect to k, as expected, we can see that recall increases as we increase k while precision decreases since it’s inversely proportional to k. MAP depends on the ranking of the relevant returned solutions as well, so the observed increasing MAP values indicate that the system is improving at returning the relevant solutions higher in the returned k solutions. The same thing applies to NDCG since it also expresses the ranking of the returned solutions. One difference is that NDCG has higher scores for k=1 since it’s computed with respect to the ideal ordering for k = 1 not the number of relevant answers in general as in MAP.

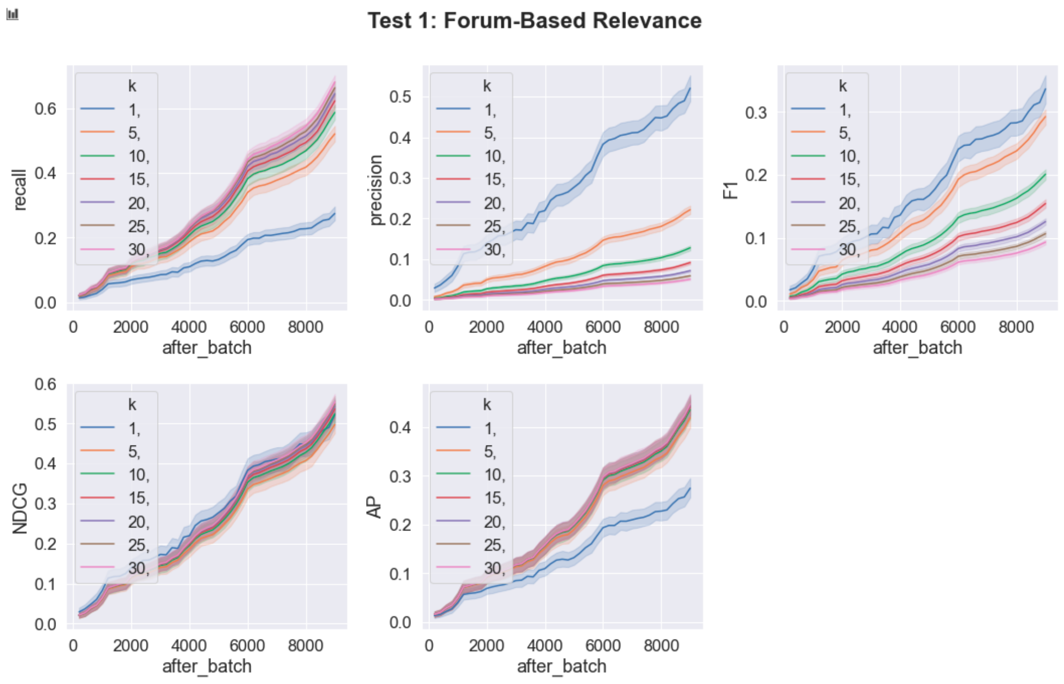


Figure 21: Recall (upper left), precision (upper middle), F1 (upper right), NDCG (lower left), and MAP (lower middle) of unlinked LL model as it learns more knowledge, based on the first evaluation method and over a range of values for k

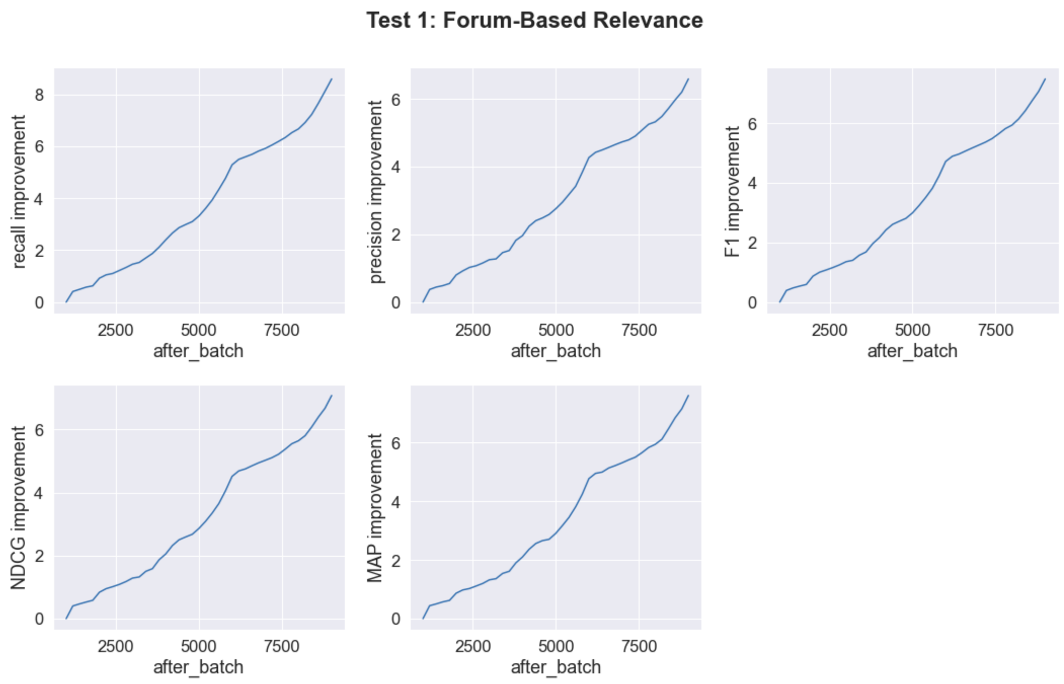


Figure 22: Factor of improvement (averaged over all values of k and based on the first evaluation method) of the unlinked LL model's recall (upper left), precision (upper middle), F1 (upper right), NDCG (lower left), and MAP (lower middle) from the baseline model (the model at batch 1000) with more learnt batches

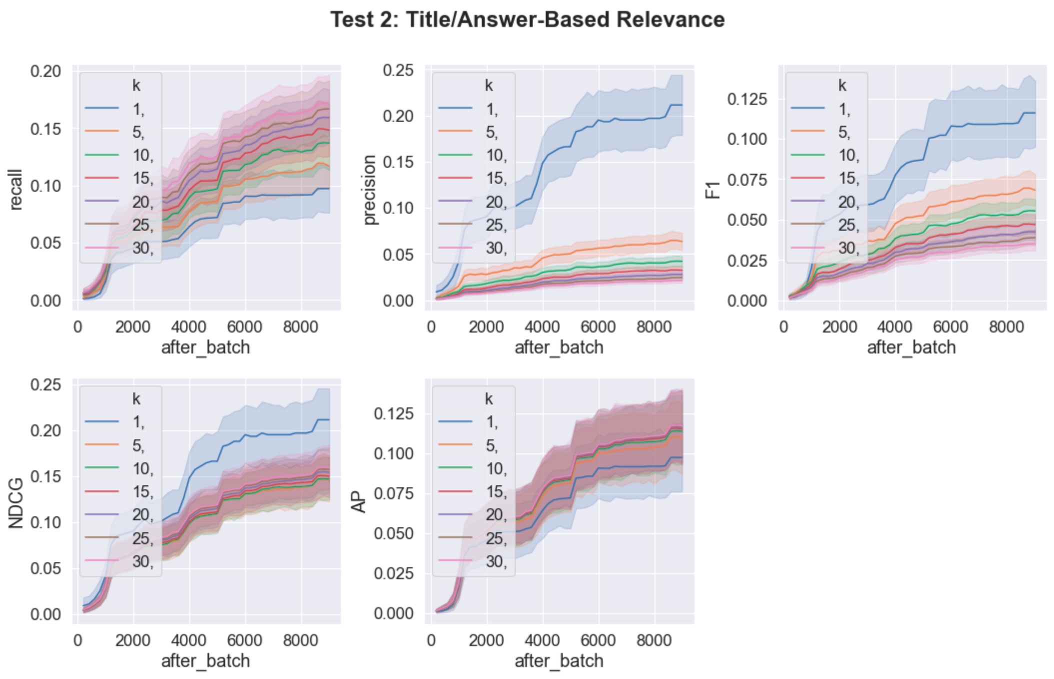


Figure 23: Recall (upper left), precision (upper middle), F1 (upper right), NDCG (lower left), and MAP (lower middle) of unlinked LL model as it learns more knowledge, based on the second evaluation method and over a range of values for k

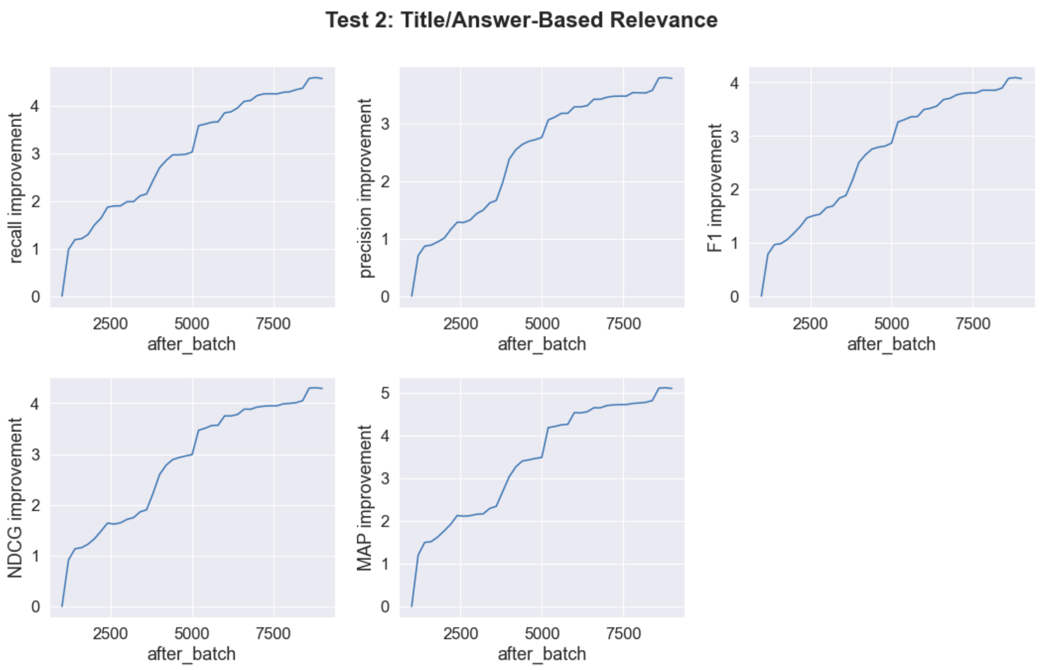


Figure 24: Factor of improvement (averaged over all values of k and based on the second evaluation method) of the unlinked LL model's recall (upper left), precision (upper middle), F1 (upper right), NDCG (lower left), and MAP (lower middle) from the baseline model (the model at batch 1000) with more learnt batches

Table 2: Average (over all values of k) factor of improvement of the unlinked LL model from the baseline model per 200 new learnt pairs

	Recall	**Precision**	**F1 Score**	**MAP**	**NDCG**
Test 1	3.8	3.7	3.7	3.4	3.3
Test 2	3.13	2.5	2.7	3.5	2.9

Figure 25 reveals that as the model learns more knowledge, the missed answers become more due to the model not answering correctly than the answers not actually existing in the KG. However, the number of missed answers in total is still decreasing thanks to the continuous learning of the needed knowledge to answer the questions correctly. This further verifies that LL is achieving its purpose.

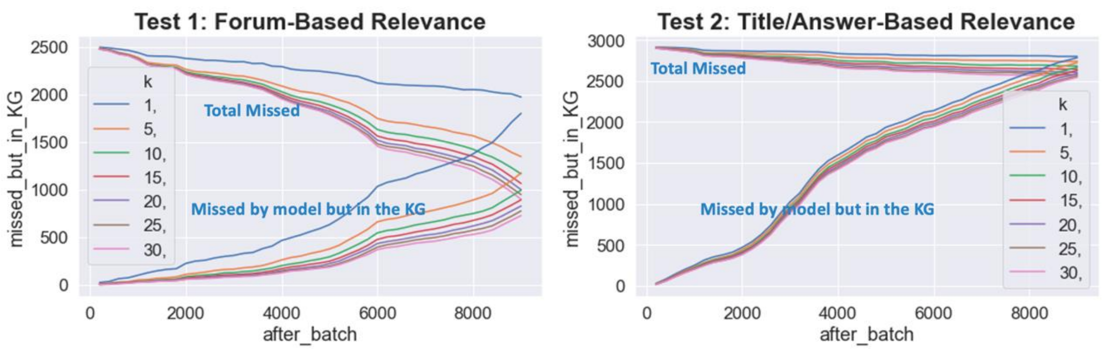


Figure 25: Missed answers that actually existed in the KG at that time out of the total missed answers by the answering model of the unlinked LL system

The overall results discussed so far verify that the LL model performs better than the baseline model whose learning stopped early due to the absence of the automatic LL process in it. The results also show that the LL model keeps improving for the same queries as it learns more knowledge.

4.3.2.2. LL Evaluation for System with Linked Problems in its KG

Similar to the previous section, the graphs below and **Table 3** show the continuous improvement of the LL model (with linked problems) from the baseline as it learns more knowledge (**Figure 26** and **Figure 27** for first evaluation method and **Figure 28** and **Figure 29** for the second evaluation method). Likewise here, the baseline model is taken at the point of 1000 learnt problems (after_batch = 1000) where we consider it stopped further learning. **Figure 30** also shows how the contribution of the incompleteness of the KG to the missed answers decreases as more knowledge is learnt, in addition to a general observed decrease in total missed answers.

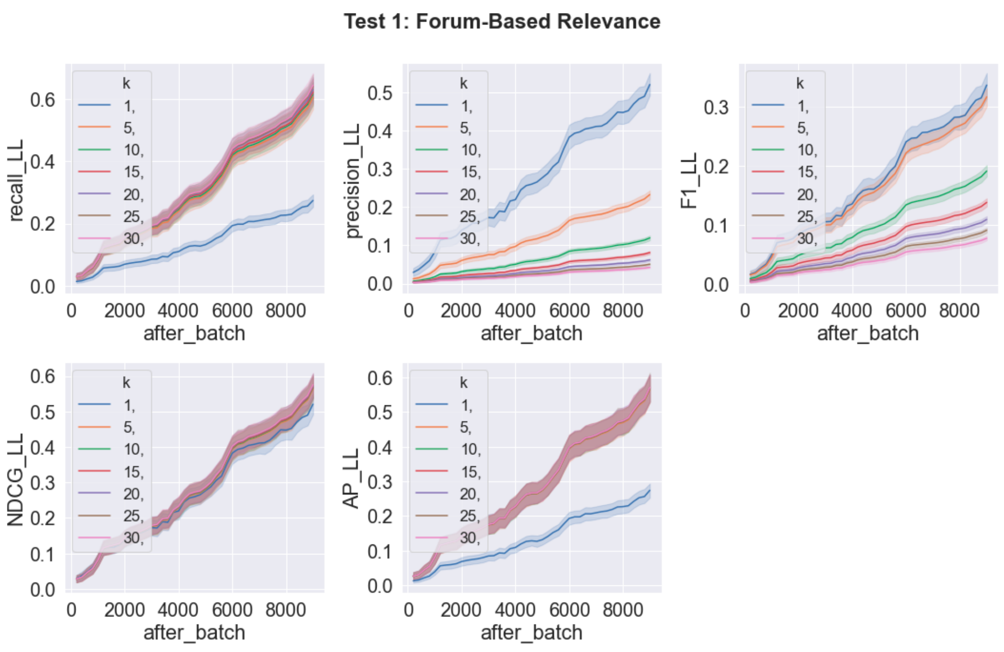


Figure 26: Recall (upper left), precision (upper middle), F1 (upper right), NDCG (lower left), and MAP (lower middle) of linked LL model as it learns more knowledge, based on the first evaluation method and over a range of values for k

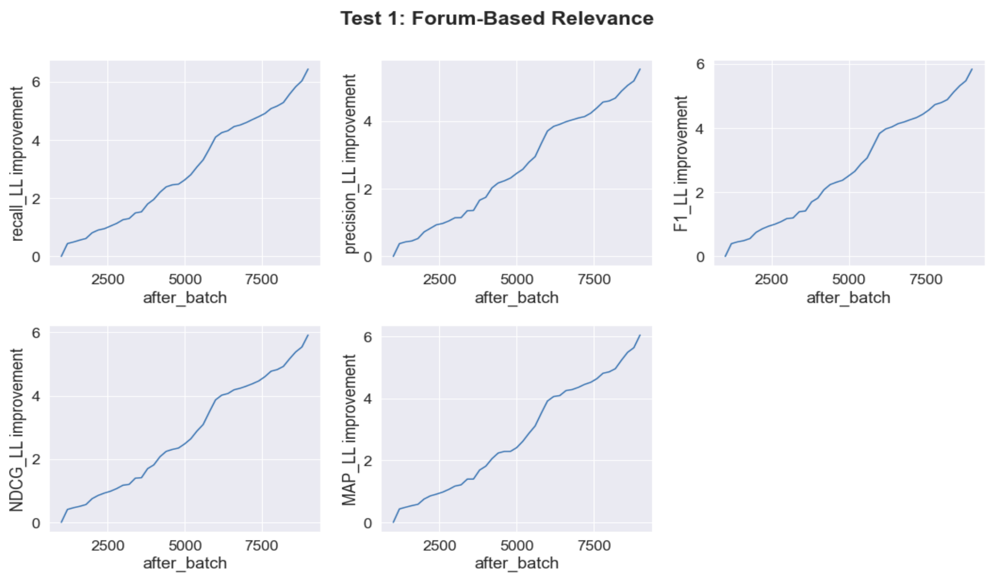


Figure 27: Factor of improvement (averaged over all values of k and based on the first evaluation method) of the linked LL model's recall (upper left), precision (upper middle), F1 (upper right), NDCG (lower left), and MAP (lower middle) from the baseline model (the model at batch 1000) with more learnt batches

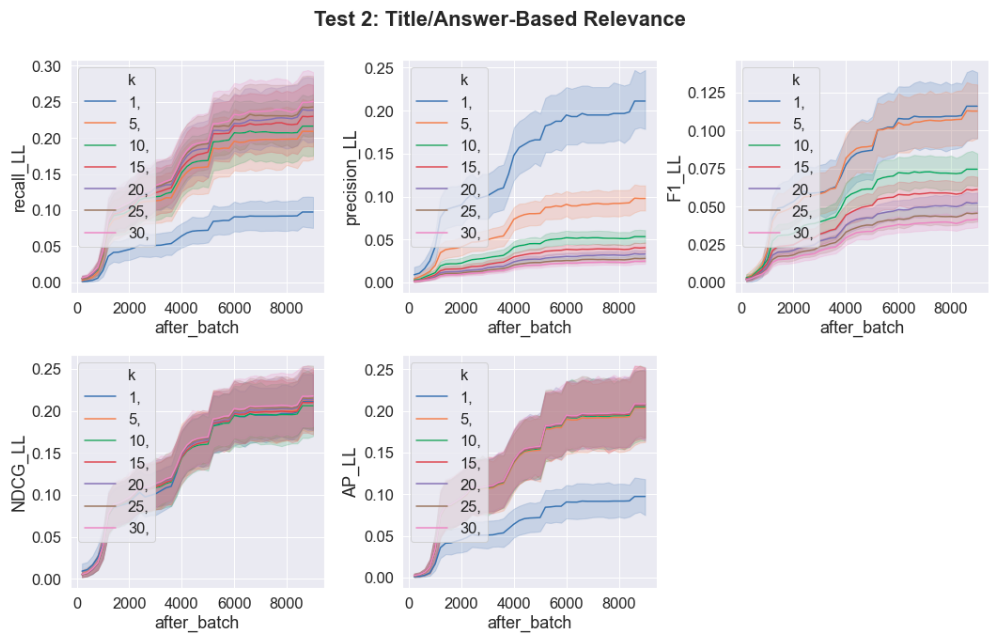


Figure 28: Recall (upper left), precision (upper middle), F1 (upper right), NDCG (lower left), and MAP (lower middle) of linked LL model as it learns more knowledge, based on the second evaluation method and over a range of values for k

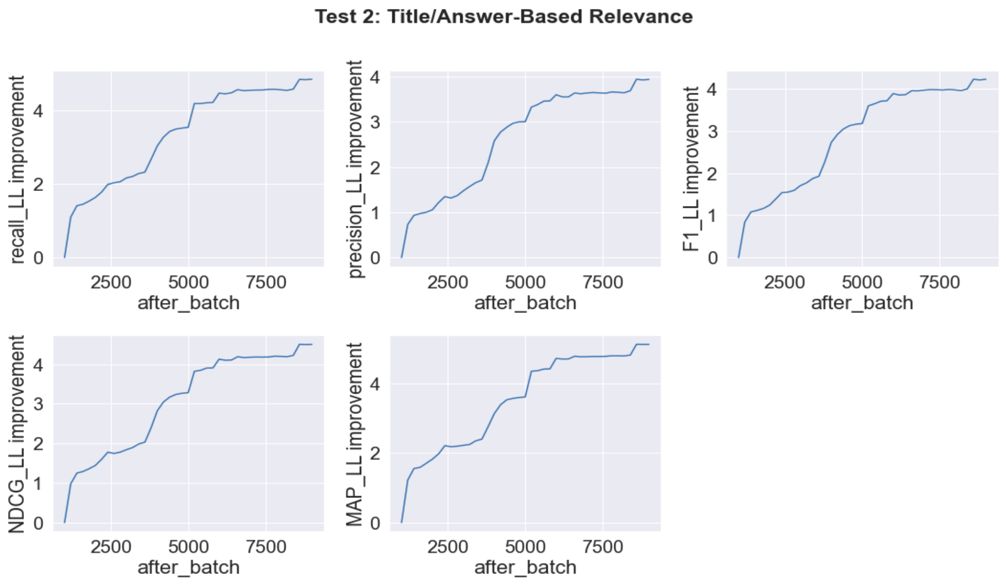


Figure 29: Factor of improvement (averaged over all values of k and based on the second evaluation method) of the linked LL model's recall (upper left), precision (upper middle), F1 (upper right), NDCG (lower left), and MAP (lower middle) from the baseline model (the model at batch 1000) with more learnt batches

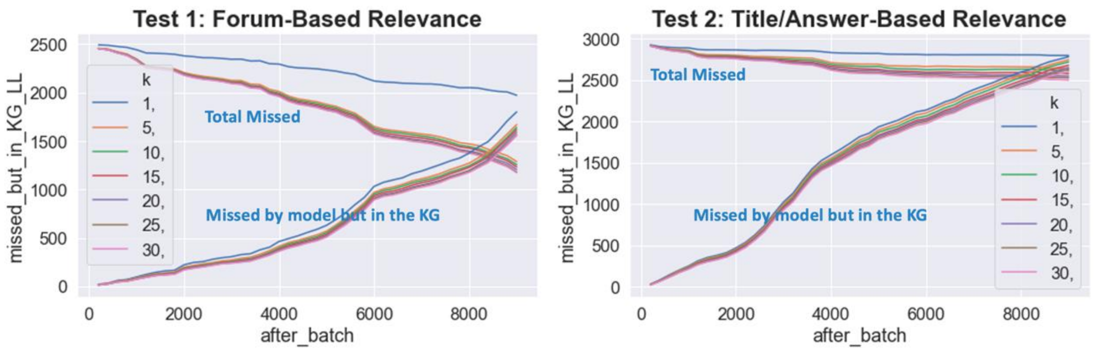


Figure 30: Missed answers that actually existed in the KG at that time out of the total missed answers by the answering model of the linked LL system

Table 3: Average (over all values of k) factor of improvement of the linked LL model from the baseline model per 200 new learnt pairs

	Recall	**Precision**	**F1 Score**	**MAP**	**NDCG**
Test 1	3.0	2.8	2.8	2.9	2.8
Test 2	3.4	2.7	2.9	3.6	3.1

4.3.2.3. LL Linked vs. Unlinked

From the previous graphs and from **Table 4** below, it is demonstrated that the model that uses linked problems in the KG to answer the queries has a much better performance in all metrics than the model with unlinked problems that just retrieves the top-k similar to the user queries. This proves the value of the further linking that we're doing in our system.

Table 4: Average (over all values of k) percent improvement of the linked LL model from the unlinked LL model per 200 newly learnt pairs

	Recall	**Precision**	**F1 Score**	**MAP**	**NDCG**
Test 1	13%	5%	6%	38%	14%
Test 2	48%	23%	28%	68%	34%

4.3.2.4. Using the "helpful" Attribute to Rank Answers

The "helpful" attribute, which indicates how many users have found this answer helpful, was used in this section and experiment to rank the returned results to the user. This was done for the linked LL system by sorting the top-k returned solutions according to how helpful people found them. However, looking at the evaluation metrics that are affected by the answer ranking, that is MAP (**Figure 31**) and NDCG (**Figure 32**), and comparing the linked LL model that uses the "helpful" attribute to that which does not, we concluded that sorting the returned answers according to the "helpful" attribute did not actually show improvement. This is most probably due to the fact that the relevance of an answer to a query still mainly depends on the relevance (i.e. the similarity) of its respective problem to that query, regardless of how many people found that thread useful (especially that we are already using the accepted solution).

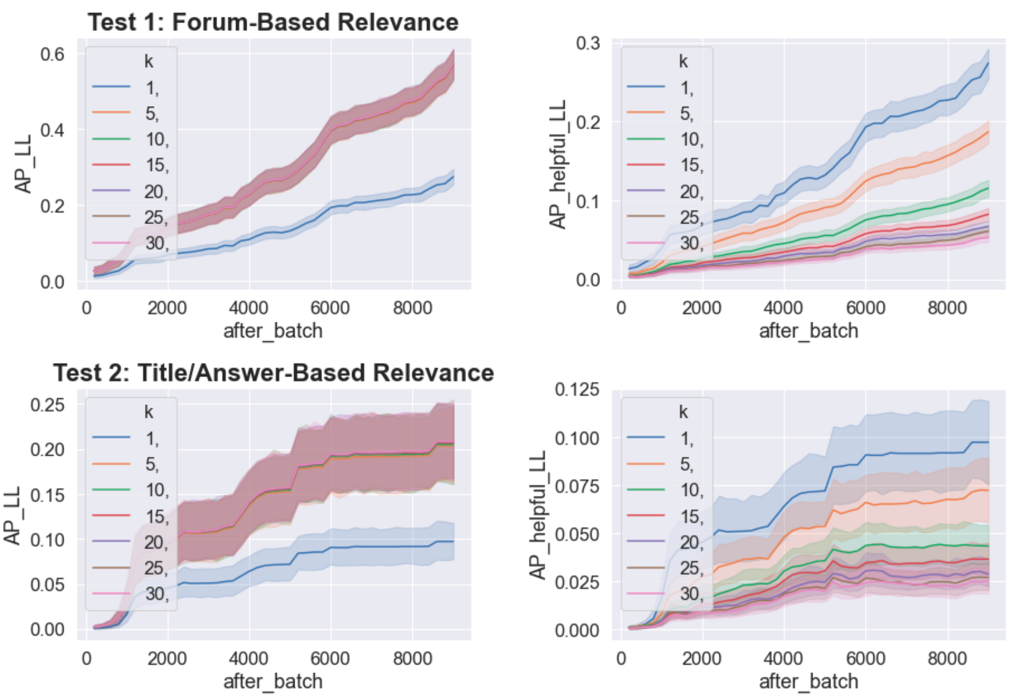


Figure 31: MAP of linked LL system not sorting according to the number of users that found a solution helpful (left) vs. that sorting according to it (right)

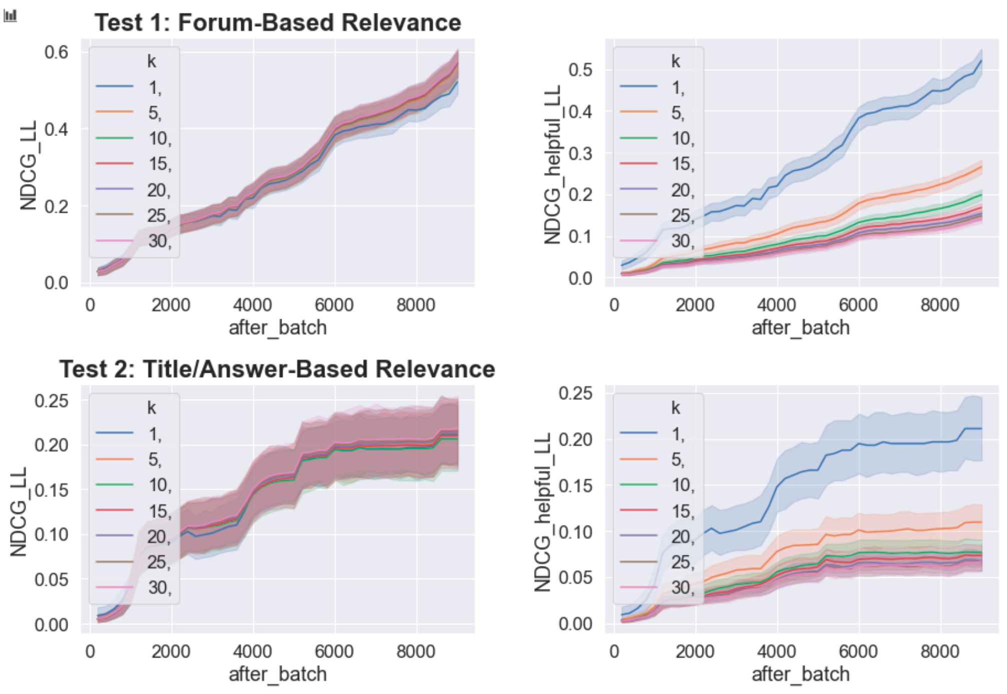


Figure 32: NDCG of linked LL system not sorting according to the number of users that found a solution helpful (left) vs. that sorting according to it (right)

4.3.2.5. Scalability of the LL Model

The figures below show that the time taken to insert and link a problem (**Figure 33**) and the time taken to answer a user query (**Figure 34**) grow somehow linearly with the size of the KG. However, the rate at which they grow is relatively slow especially for answering. In addition, the range of values is still very acceptable for our implementation, where less than a minute per problem is adequate for the insertion process since it does not affect or stop the system, and since the number of problems emerging daily/weekly (to be crawled and added) is not immense. On the other hand, the answering is taking less than 0.04 seconds per user query (even if you project it for more problems in KG), so it is still very convenient.

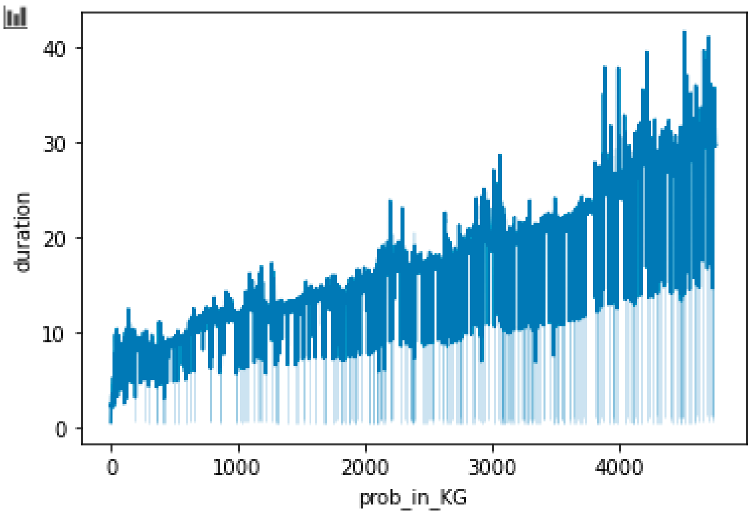


Figure 33: Time taken to insert and link a problem with its info in the KG with respect to the size of the KG (according to the number of problems present there so far)

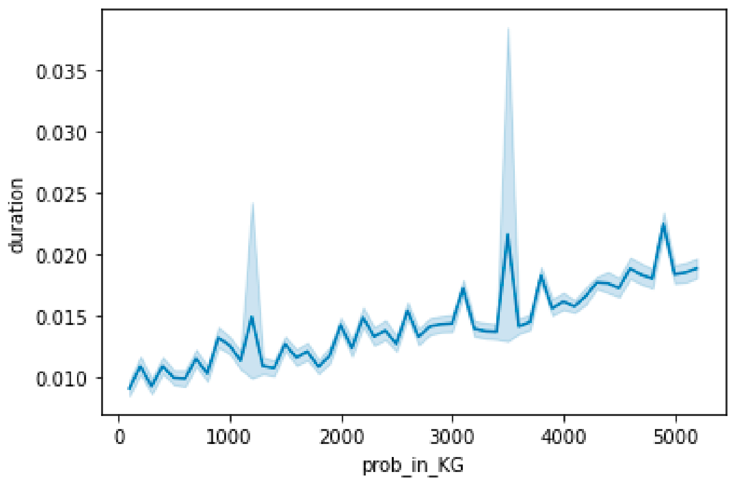


Figure 34: Time taken to answer user queries with respect to the size of the KG (according to the number of problems present there so far)

4.4. Qualitative Comparison of our LL Model with State-of-the-art LL Models

In this section, we discuss why the state-of-the-art LL systems mentioned in the literature review are not applicable to our implementation of web-based LL for customer-support chatbots. That said, those SOTA LL systems cannot be directly compared to our approach.

Shi et al. [29] and Shah et. al. [34] target connecting new entities, mainly names, to existing ones based on their descriptions and using embeddings. In the study of KBC in [30], the authors aim at adding more information for the existing name entities from external documents by representing this information as new relations connecting those entities together. Hence, those three works maintain a different underlying structure of the KG compared to ours as it's now evident how the entities and relations are connected in both cases, and thus they are handled differently.

Likewise, the models developed in [31][33] only work for triplets in the form of: entity, relation, entity, while also relying on the user to help them find the missing links. This is not applicable to our system because we do not use any user interaction, feedback mechanism, or even an expert intervention to extract the knowledge and form the KG. This essential requirement is what makes our model novel compared to the chatbots that continuously learn from their interaction with the user. One of those chatbots was introduced by Abujabal et. al. [32] where in addition to depending on the user’s feedback and voting to update its knowledge, it only updates the template-query model with new syntaxes for the same questions instead of updating the actual KG. Similarly, the chatbot developed in [36] depends on the user’s feedback, and the one in [37] learns by knowing how to ask questions to the user or teacher. The last mentioned bot to also learn from user interactions is the one in [35] which detects the user’s satisfaction with the response. If the users seem to be unsatisfied with the generated response, the bot directly requests from them the specific answers that they were looking for and automatically assimilates them.

So far, the mentioned systems are not applicable since their KG structure is different and solely depends on user interaction and feedback to learn the new knowledge. Furthermore, most of them are more conversational than QA. However, the implementation in [35] can be an adequate add-on to our system providing it with the capability of learning from conversations. The bot can inquire the solution(s) (among the returned solutions or different new ones) that were most suitable in the user's case and add them to the KG. If no solution works for the user, the bot can direct him/her to the forum where their posted question and its answer will be extracted and learned by our system eventually (in the LL process).

Regarding the lifelong IE systems, one of the main differences between them and our work is that most of them are not actually chatbots or answering systems since their focus is on extracting new knowledge from the web. Additionally, their applications are very different from ours (which ultimately means their KG structure is different too as noticed before). This is observed for system [39] whose only target is to read and learn the whole web (the same applies to the system in [42] but for Russian language). The model in [38] extracts triplets from web texts and tries to find their general concepts (higher categories or classes for those entities) and relations between them using wordnet and KNOWITALL. After that, it guides the next learning cycle (concept-relation finding) based on the previous one. The studies in [40] and [41] also target a different application called opinion mining. They try to expand opinion extraction (aspects, opinions for certain products) by continuously linking similar aspects using recommender systems and opinion texts for other products (shared aspects for many).

To sum up, our work is unique compared to the SOTA LL systems in that its learning process does not require feedback from the user or an expert, it is more QA than conversational, and that its application differs significantly from those systems as well as its KG structure.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This thesis achieved designing lifelong learning (LL) for an information retrieval (IR), question-answering (QA) customer support bot (CSB) for network operations, with CISCO used as the case study. This LL enabled the CSB to continuously learn new knowledge on its own from CISCO's online support forum. Several contributions were tackled in the thesis, the first being the design of the knowledge graph (KG) to represent the online support discussions from CISCO. This design captured the main hierarchy of domains, topics, labels/tags, and problem-solution pairs from that forum and many useful features of the problems/solutions like date, users, views, helpful, and number of replies. The KG was implemented via Neo4j graph database and succeeded in facilitating the continuous update of this KG with new knowledge and linking similar problems to each other.

The second contribution was designing the LL flow which permitted the CSB to continuously learn new knowledge on its own from the online forum. This flow additionally included avoiding problem/solution duplicates and linking new problems to similar existing ones. The matching was done using the hashing TFIDF vectorizer that enabled us to avoid recomputing the vector representations of all problems with every addition of a new problem, hence providing a low-cost solution for the lifelong flow and the answering flow which constitutes our third contribution. This answering flow involved matching the user's question or query to the most similar problem from the KG and returning its solution with the solutions of the similar problems to this problem ranked based on their similarity to that top problem.

Finally, as a third contribution, we introduced an approach to validate the effectiveness of the LL process through a simulator that simulates adding more user queries and adding/learning more knowledge over time. The evaluation dataset consisted of queries, relevant problems to those queries, irrelevant problems to those queries, and we formed the relevant set based on two methods: recommended problems via the online forum (pre-defined) and problems showing similarities with the queries in titles and answers.

Our results showed the superiority of the lifelong model with a 2X to 3X improvement in all of recall, precision, F1 score, MAP, and NDCG compared to answering without LL. The LL model also kept improving in all metrics as it learned more knowledge while the baseline model suffered from a drop in those metrics with more user queries since it does not take advantage of LL. Our results further showed that linking similar problems in the KG during the lifelong flow improved the system's answering performance, whereas, ranking the returned solutions based on how helpful users considered them did not actually help our system. In terms of scalability of the LL model, the time to add new knowledge and the time to answer appeared to grow linearly with the size of the KG but at a slow rate and within a very acceptable range for our implementation. Our system also certified having a unique application after comparing it to state-of-the-art LL systems.

Future work focuses on three main points. The most important one is the similarity model which can be improved by adopting methods that focus more on the context and semantic than the actual words. This part also involves doing advanced text processing and cleaning and even extracting the exact problem and answer from the text. Improving this part will improve both the baseline and LL models’ performances. The second part is expanding the scalability of the system by either further optimizing the current process if possible or switching to using triplestores instead of graph databases. Finally, the answering flow can be deeply enhanced by making more use of the data hierarchy and additional nodes present in the KG, and by looking more into query refinement within a session and creating query logs to keep track of the discussion.

REFERENCES

[1] “Chatbots In Customer Service - Statistics and Trends [Infographic],” 2020. https://www.invespcro.com/blog/chatbots-customer-service/ (accessed Dec. 13, 2020).

[2] “40 AMAZING Chatbot Statistics for 2021,” 2020. https://backlinko.com/chatbot-stats (accessed Dec. 13, 2020).

[3] Z. Chen and B. Liu, *Lifelong Machine Learning, Second Edition*, vol. 12, no. 3. 2018.

[4] J. Weizenbaum, “ELIZA-A computer program for the study of natural language communication between man and machine,” *Commun. ACM*, vol. 9, no. 1, pp. 36–45, Jan. 1966, doi: 10.1145/365153.365168.

[5] R. S. (Robert S. Epstein, G. Roberts, and G. Beber, “The Anatomy of A.L.I.C.E.,” in *Parsing the Turing test : philosophical and methodological issues in the quest for the thinking computer*, Springer, 2009, pp. 181–210.

[6] Bruce Wilcox, “ChatScript,” 2021. https://sourceforge.net/projects/chatscript/ (accessed Dec. 02, 2019).

[7] “Cleverbot.com - a clever bot - speak to an AI with some Actual Intelligence?,” 1997. https://www.cleverbot.com/ (accessed Dec. 02, 2019).

[8] “Chatfuel,” 2015. https://everipedia.org/wiki/lang_en/chatfuel (accessed Dec. 02, 2019).

[9] D. Ferrucci *et al.*, “Building watson: An overview of the deepQA project,” *AI Mag.*, vol. 31, no. 3, pp. 59–79, Sep. 2011, doi: 10.1609/aimag.v31i3.2303.

[10] “Mitsuku,” 2010. https://www.pandorabots.com/mitsuku/ (accessed Dec. 02, 2019).

[11] S. Ghose and J. J. Barua, “Toward the implementation of a topic specific dialogue based natural language chatbot as an undergraduate advisor,” 2013, doi: 10.1109/ICIEV.2013.6572650.

[12] N. T. Thomas, “An e-business chatbot using AIML and LSA,” *2016 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2016*, pp. 2740–2742, Nov. 2016, doi: 10.1109/ICACCI.2016.7732476.

[13] S. Reshmi and K. Balakrishnan, “Implementation of an inquisitive chatbot for database supported knowledge bases,” *Sadhana - Acad. Proc. Eng. Sci.*, vol. 41, no. 10, pp. 1173–1178, Oct. 2016, doi: 10.1007/s12046-016-0544-1.

[14] “LUIS (Language Understanding) – Cognitive Services – Microsoft Azure,” 2015. https://www.luis.ai/home (accessed Dec. 02, 2019).

[15] “Dialogflow,” 2010. https://dialogflow.com/ (accessed Dec. 02, 2019).

[16] “What Is Amazon Lex? - Amazon Lex,” 2017. https://docs.aws.amazon.com/lex/latest/dg/what-is.html (accessed Dec. 02, 2019).

[17] Y. Wu, W. Wu, C. Xing, C. Xu, Z. Li, and M. Zhou, “A Sequential Matching Framework for Multi-Turn Response Selection in Retrieval-Based Chatbots,” *Comput. Linguist.*, vol. 45, no. 1, pp. 163–197, Mar. 2019, doi: 10.1162/coli_a_00345.

[18] E. Stroh and P. Mathur, “Question Answering Using Deep Learning,” 2016.

[19] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global vectors for word representation,” in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014, pp. 1532–

1543, doi: 10.3115/v1/d14-1162.

[20] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in Neural Information Processing Systems*, Sep. 2014, vol. 4, no. January, pp. 3104–3112.

[21] O. Vinyals and Q. Le, “A Neural Conversational Model,” *Proc. 31 st Int. Conf. Mach. Learn.*, Jun. 2015.

[22] L. Shao, S. Gouws, D. Britz, A. Goldie, B. Strope, and R. Kurzweil, “Generating high-quality and informative conversation responses with sequence-to-sequence models,” in *EMNLP 2017 - Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Jan. 2017, pp. 2210–2219, doi: 10.18653/v1/d17-1235.

[23] Z. Yin, K. Chang, and R. Zhang, “DeepProbe: Information Directed Sequence Understanding and Chatbot Design via Recurrent Neural Networks,” in *KDD ’17 Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul. 2017, pp. 2131–2139, doi: 10.1145/3097983.3098148.

[24] M. Boyanov, I. Koychev, P. Nakov, A. Moschitti, and G. Da San Martino, “Building chatbots from forum data: Model selection using question answering metrics,” in *International Conference Recent Advances in Natural Language Processing, RANLP*, Oct. 2017, vol. 2017-September, pp. 121–129, doi: 10.26615/978-954-452-049-6-018.

[25] J. A. Campos, A. Otegi, A. Soroa, J. Deriu, M. Cieliebak, and E. Agirre, “DoQA - Accessing Domain-Specific FAQs via Conversational QA,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, May 2020, pp. 7302–7314, doi: 10.18653/v1/2020.acl-main.652.

[26] V. Jijkoun and M. De Rijke, “Retrieving answers from frequently asked questions pages on the web,” in *ACM International Conference on Information and Knowledge Management*, 2005, pp. 76–83, doi: 10.1145/1099554.1099571.

[27] J. Huang, M. Zhou, and D. Yang, “Extracting chatbot knowledge from online discussion forums,” in *IJCAI International Joint Conference on Artificial Intelligence*, 2007, pp. 423–428.

[28] Y. Watanabe, R. Nishimura, and Y. Okada, “A question answer system based on confirmed knowledge acquired from a mailing list,” *Internet Res.*, vol. 18, no. 2, pp. 165–176, 2008, doi: 10.1108/10662240810862220.

[29] B. Shi and T. Weninger, “Open-world knowledge graph completion,” in *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, Nov. 2018, pp. 1957–1964.

[30] H. Ji, H. Ji, and R. Grishman, “Knowledge Base Population: Successful Approaches and Challenges,” *HLT ’11 Proc. 49th Annu. Meet. Assoc. Comput. Linguist. Hum. Lang. Technol.*, vol. 1, pp. 1148–1158, 2011.

[31] S. Mazumder, N. Ma, and B. Liu, “Towards a Continuous Knowledge Learning Engine for Chatbots,” Feb. 2018.

[32] A. Abujabal, R. Saha Roy, M. Yahya, and G. Weikum, “Never-Ending Learning for Open-Domain Question Answering over Knowledge Bases,” in *Proceedings of the 2018 World Wide Web Conference (WWW’18)*, 2018, pp. 1053–1062, doi: 10.1145/3178876.3186004.

[33] S. Mazumder, B. Liu, S. Wang, and N. Ma, “Lifelong and Interactive Learning of Factual Knowledge in Dialogues,” in *Proceedings of the 20th Annual SIGdial*

Meeting on Discourse and Dialogue, Oct. 2019, pp. 21–31, doi: 10.18653/v1/w19-5903.

[34] H. Shah, J. Villmow, A. Ulges, U. Schwanecke, and F. Shafait, "An Open-World Extension to Knowledge Graph Completion Models," Jun. 2019.

[35] B. Hancock, A. Bordes, P.-E. Mazare, and J. Weston, "Learning from Dialogue after Deployment: Feed Yourself, Chatbot!," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Sep. 2019, pp. 3667–3684, doi: 10.18653/v1/p19-1358.

[36] E. Agirre, A. Otegi, C. Pradel, S. Rosset, A. Peñas, and M. Cieliebak, "LIHLITH: Learning to Interact with Humans by Lifelong Interaction with Humans," 2019.

[37] J. Li, A. H. Miller, S. Chopra, M. Ranzato, and J. Weston, "Learning through Dialogue Interactions by Asking Questions," Dec. 2017, Accessed: Nov. 30, 2020. [Online]. Available: http://arxiv.org/abs/1612.04936.

[38] M. Banko and O. Etzioni, "Strategies for lifelong knowledge extraction from the web," in *K-CAP'07: Proceedings of the Fourth International Conference on Knowledge Capture*, 2007, pp. 95–102, doi: 10.1145/1298406.1298425.

[39] T. Mitchell *et al.*, "Never-ending learning," *Commun. ACM,* vol. 61, no. 5, pp. 103–115, May 2018, doi: 10.1145/3191513.

[40] Q. Liu, B. Liu, Y. Zhang, D. S. Kim, and Z. Gao, "Improving Opinion Aspect Extraction Using Semantic Similarity and Aspect Associations," in *AAAI'16 Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence,* 2016, pp. 2986–2992.

[41] L. Shu, H. Xu, and B. Liu, "Lifelong Learning CRF for Supervised Aspect Extraction," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017, pp. 148–154, doi: 10.18653/v1/P17-2023.

[42] K. Buraya, L. Pivovarova, S. Budkov, and A. Filchenkov, "Towards Never Ending Language Learning for Morphologically Rich Languages," in *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing*, 2017, pp. 108–118, doi: 10.18653/v1/W17-1417.

[43] F. A. O. Santos, F. B. do Nascimento, M. S. Santos, and H. T. Macedo, "Training Neural Tensor Networks with the Never Ending Language Learner," *Adv. Intell. Syst. Comput.*, vol. 738, pp. 19–23, 2018, doi: 10.1007/978-3-319-77028-4_4.

[44] C. Chakrabarti and G. F. Luger, "A semantic architecture for artificial conversations," in *6th International Conference on Soft Computing and Intelligent Systems, and 13th International Symposium on Advanced Intelligence Systems, SCIS/ISIS 2012*, 2012, pp. 21–26, doi: 10.1109/SCIS-ISIS.2012.6505415.

[45] R. Madan, "TF-IDF/Term Frequency Technique: Easiest explanation for Text classification in NLP using Python (Chatbot training on words) | by Rohit Madan | Analytics Vidhya | Medium," 2019. https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3 (accessed Dec. 14, 2020).

[46] B. Stecanella, "What is TF-IDF?," 2019. https://monkeylearn.com/blog/what-is-tf-idf/ (accessed Dec. 14, 2020).

[47] "sklearn.feature_extraction.text.HashingVectorizer — scikit-learn 0.24.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.HashingVecto

rizer.html (accessed Jan. 24, 2021).

[48] M. Veron, A. Peñas, G. Echegoyen, S. Banerjee, S. Ghannay, and S. Rosset, “A cooking knowledge graph and benchmark for question answering evaluation in lifelong learning scenarios,” in *Natural Language Processing and Information Systems NLDB*, Jun. 2020, vol. 12089 LNCS, pp. 94–101, doi: 10.1007/978-3-030-51310-8_9.

[49] D. Alocci, J. Mariethoz, O. Horlacher, J. T. Bolleman, M. P. Campbell, and F. Lisacek, “Property Graph vs RDF Triple Store: A Comparison on Glycan Substructure Search,” *PLoS One*, vol. 10, no. 12, p. e0144578, Dec. 2015, doi: 10.1371/journal.pone.0144578.

[50] “What is a Knowledge Graph | Stardog.” https://www.stardog.com/resources/what-is-a-knowledge-graph/ (accessed Jan. 21, 2021).

[51] R. A. Maturana, “What are the differences between a Graph database and a Triple store? (by Matt Allen in Quora) - Next Web - GNOSS,” 2015. https://nextweb.gnoss.com/en/resource/what-are-the-differences-between-a-graph-database/ced22960-845d-410f-9e3c-5616c603993e (accessed Jan. 20, 2021).

[52] Neo4j, “Top 10 Reasons for Choosing Neo4j for Your Graph Database.” https://neo4j.com/top-ten-reasons/ (accessed Oct. 24, 2020).

[53] “Discounted cumulative gain - Wikipedia.” https://en.wikipedia.org/wiki/Discounted_cumulative_gain (accessed Jan. 31, 2021).

[54] K. Wang, Z. Ming, and T. S. Chua, “A syntactic tree matching approach to finding similar questions in community-based QA services,” in *Proceedings - 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009*, 2009, pp. 187–194, doi: 10.1145/1571941.1571975.