Good Automatic Authentication Question Generation

Simon S. Woo
Univ. of Southern California
Information Sciences Institute
Marina del Rey, CA
simonwoo@usc.edu

Zuyao Li
Univ. of Southern California
Los Angeles, CA
zuyaoli@usc.edu

Jelena Mirkovic
Univ. of Southern California
Information Sciences Institute
Marina del Rey, CA
mirkovic@isi.edu

Abstract

We explore a novel application of Question Generation (QG) for authentication use, where questions are widely used to verify user identity for online accounts. In our approach, we prompt users to provide a few sentences about their personal life events. We transform user-provided input sentences into a set of simple fact-based authentication questions. We compared our approach with previous QG systems, and evaluation results show that our approach yielded better performance and the promise of future personalized authentication question generation.

1 Introduction

An authentication question (also known as a security question), such as "What is your mother's maiden name?" is widely used for verifying user identity for many online accounts — such as email, banking, e-commerce and social networking. However, past numerous breaches on security questions identify the weakness of the current fixed set of authentication questions. Answers to some of those authentication questions are easy to guess based on simple common sense, with little or no prior knowledge about the individual. Since current security questions are not personalized, users can choose from a finite set of questions whose answers are easily guessed. Also, not all questions are applicable to all users.

Motivated by the research of Woo et al. (2014), in our study we automatically generate security questions from user-provided short texts from personal life events. Given user-provided text such as, "*I visited Beijing in 2001 with John*," we generate more meaningful authentication questions, such as: "*What city did you visit?*" "*What year did you visit?*" "*Who were you with?*" These are more difficult to guess than the maiden name of a user's mother. The contribution of this work is to automatically generate rule-based, concise, simple, fact-based shallow *WH** questions, where we explore 1) dependency parsing based, and 2) semantic role labeling (SRL) based approaches to generating questions.

2 Related Work

Previous Question Generation (QG) research (Heilman and Smith, 2010a; Yao et al., 2012; Heilman and Smith, 2010b; Heilman, 2011) focused on syntactic transformation to construct questions at the sentence level. Also, recent research by Mazidi and Nielsen (2014) improved the QG performance over that of Heilman and Smith (2010a) using semantic role labeling at the paragraph level to construct deeper questions. However, most QG research, including the results presented in the *2010 Question Generation Shared Task Evaluation Challenge*, has primarily focused on generating grammatical, deep, and complete questions for educational purposes. No prior QG research has considered an application for generating personalized authentication questions, which require different Q&A usability characteristics than those needed for education applications.

3 Approach

In our QG system we prompt users to provide a few sentences in a free-form format regarding personal life events (as shown in Woo et al. (2014)). Research has demonstrated that compared to current security questions, the answers to questions which are generated from unique personal memories/events are less likely to be guessed by others, but are far easier for users to remember. While past QG research focused on generating long and grammatically fluent questions, authentication questions impose unique challenges due to security and usability concerns:

- **One concrete fact per question**: If a question is vague, deep, or ambiguous, it can potentially lead to multiple answers, making it difficult to validate user responses. If multiple or similar answers are accepted, then security can be drastically impacted. Hence, it is crucial to ask a *specific* question to reduce the variability in user response and maintain security.

- **Simplicity and brevity**: It is important for a question to be *simple*, *short* and *concise* so that users can interact and enter their authentication responses quickly in real time.

- **Difficult to guess answers:** Answers cannot be easily inferred from the given contexts or questions.

With these design goals, we automatically generate authentication questions from user-provided texts. We take a rule-based, two-phase approach to generate questions: 1) sentence simplification and 2) question generation.

3.1 Phase1: Sentence Simplifications

We break a complex source sentence into shorter sentences. Although other research (Heilman and Smith, 2010a) considered sentence simplification before question generation, we focused on each derived short sentence having one concrete fact. In order to generate a simple one-fact based question, it is crucial to simplify a source sentence as much as possible. To identify a subject, we use clause-, phrase-, and word-level POS tags to break a sentence iteratively, as well as dependency parsing (Collobert et al., 2011) and semantic role labeling (Björkelund et al., 2010) to identify a subject. For example, if the input sentence is *"Caitlin was our flower girl, and got tips, and danced at the dinner,"* then we produce the following three shorter sentences *"Caitlin was our flower girl." "Caitlin got tips."* and *"Caitlin danced at the dinner."* These are the input sentences to the next QG phase. Our iterative sentence break approach works as follows: we first process each word from left to right sequentially for a potential sentence breakpoint, and identify subjects and main verbs in an input source sentence. Then, we take the following steps to break a sentence:

Step 1. Iteratively read word tokens from left to right, and break a sentence before the next subject (Subj), or verb (VB*), or modal (MD) or coordinating conjunction (CC), or subordinating conjunction (IN) occurs; these are potential breakpoints.

Step 2. Clean unnecessary words from the obtained sentences such as CC, IN and ADV.

Step 3. Determine if two consecutive outputs can be combined.

Step 4. Assign a subject using SRL.

However, sentence breaking at Step 1 overbreaks and generates over-simplified output in some cases. Hence, in Step 3, we attempt to combine any two consecutive outputs from Step 1.

The outputs can be combined to produce a better sentence for the following cases by 1) connecting sentences split by "to"; 2) handling gerunds in a subject; and 3) using phrasal verbs (i.e., do, let). After Step 3, the final subject of the combined sentence is assigned to each shorter sentence. The proposed simple sentence breaking-combining approach is capable of handling most of the following input sentence patterns:

(WDT/WRB/WP/WP$)+Subj1+(MD1)+VB1
+(CC1)+(Subj2)+(MD2)+VB2+...,

where POS tags inside parentheses are optional in a sentence. For more complex sentences that include subordinate clauses, in which the left-to-right iterative approach does not apply, we adopt the tree-based transformation in Heilman and Smith (2010a). However, in most cases, their approach does not simplify the process enough for us to directly generate short questions. Hence, we apply the iterative rules in Step 1 to further break generated sentences after applying Heilman and Smith's tree-based transformation (Heilman and Smith, 2010a) to achieve the one-fact rule for a simplified sentence.

3.2 Phase 2: Question Generation

After breaking sentences, we identify possible answers from simplified sentences. Generally, difficult-to-guess answers are related with location, time, and person, as well as subject, object, and semantic roles in a sentence. We use dependency parsing, semantic role labeling, and a named entity recognizer (NER) to identify the answer phrases and construct questions.

3.2.1 Dependency parsing based approach

Since dependency parsing can capture the relationship among verb, subject, and object, we use the dependency parser (Björkelund et al., 2010). Once we identify the subject and object, we construct the *who* and *what* question types (QType). If an input sentence has LOC (location) and TMP (time), we can construct *where* and *when* questions. Next, we replace a question type with an answer and shift the question type to the left in a simplified sentence. Then, verb tense and subject position are adjusted while preserving the rest of the words in the sentence. Finally, we produce a question. For example, given the input sentence, *"Alice lived in Shenyang in 2007,"* we can construct the following questions:

Q1. We extract the subject *Alice* and replace it with a *Who* QType, and then generate a question: *"Who lived in Shenyang in 2007?"*

Q2. We extract the location *"in Shenyang"* and replace it with a *Where* QType. Then, we shift the QType to the left and adjust the verb tense and produce a question: *"Where did Alice live in 2007?"*

Q3. Similarly, we can produce *"When did Alice live in Shenyang?"* after replacing *in 2007* with *When*, shifting QType to the left, and adjusting the verb form.

We remove generated questions with pronoun answers such as *I*, *We*, *She*, *He*, and *They* as those are very easy to guess. Furthermore, we can refine this to more a specific question word such as *What year* instead of *When*, or *What city* instead of *Where*. This can help users provide more specific information.

3.2.2 Semantic Role Labeling (SRL) based approach

In this approach we focus on verb (action) and *semantic roles* (arguments of a predicate) for question generation; *who* did *what* to *whom?* is important information for QG. We employ a SRL to utilize the several semantic parts with respect to the verb. For each verb, we extract its arguments and identify different semantic roles. They are all potential answers. We mainly focus on four semantic roles in Table 1, where these roles can produce more concrete and specific information. A0 is agent or experiencer, and A1 is usually theme or result. Location and time are specified by AM-LOC and AM-TMP.

We construct a question by replacing a question type with an argument. Then we move a QType to the left and adjust the verb tense and subject position, and keep the rest of the words in a sentence to generate a question. For instance, given a short input sentence *"Bob liked eating hamburgers and drinking Coke"*:

Role	Question Type (QType)
A0, A1	who (a person), what (not a person)
AM-LOC	where
AM-TMP	when

Table 1: Question type mapping from a semantic role

Q1. We extract *liked* and its argument (A0: Bob), and generate a question, "*Who liked eating hamburgers and drinking Coke?*"

Q2. Similarly, for another argument (A1: eating hamburgers and drinking Coke), we can generate the question, "*What did Bob like?*"

4 Data Collection

We obtained approval from our Institutional Review Board (IRB) to conduct user studies, and collected data from 28 students and 12 Amazon Mechanical Turk workers. We manually created authentication question and answer pairs from user input, extracting factoids about locations, people, time, and activities as baseline results for a comparison. Instructions were given to generate questions similar to current online security question sets that we collected. For consistency, one person from our team generated 519 security question and answer pairs from 358 source sentences from user-provided personal experience over various topics. On average, per each source input sentence, 1.54 security questions were generated. We used these sentences as inputs to the QG system for performance comparison.

5 Evaluation

We compared our systems to two other QG systems developed by Yao et al. (2012) and Heilman and Smith (2010b). Both of those approaches over-generate questions and rank them to provide the best QA pairs. We calculated the average precision, recall, and F1 score based on an exact word match for each question and answer pair. The evaluation results are shown in Table 2, where the dependency parsing system is denoted as *DepPar*, the SRL-based approach is denoted as *SRL*, the system by Yao et al. (2012) is denoted as *OA*, and the system by Heilman and Smith (2010b) is denoted as *H&S*.

The precision is measured by comparing the question type, and the sequence of words between

System	Precision	Recall	F1
SRL	0.407	0.805	0.541
DepPar	0.477	0.927	0.630
OA	0.399	0.807	0.534
H&S	0.325	0.699	0.444

Table 2: Precision, recall, and F1 score for Generated Questions based on an exact word match

System	Precision	Recall	F1
SRL	0.492	0.805	0.611
DepPar	0.524	0.927	0.670
OA	0.439	0.807	0.568
H&S	0.236	0.699	0.352

Table 3: Precision, recall, and F1 score for Generated Answers based on an exact word match

manually generated Q&A pairs and Q&A pairs generated from each approach. From Table 2, we observe that the *DepPar* system performs better than *OA* and *H&S*. The dependency parser approach is better in capturing objects, time, and locations from simplified sentences, constructing better what, when and where questions, covering all the QTypes from manually generated data. The SRL-based system has the second-best performance. On the other hand, *H&S* has the lowest recall and performed poorly since it only generated 70% of the required QA set. The reason that *OA* performed poorly is that it generates the longest questions with an average of 9.8 words per question, while the average number of words in the manual dataset, *H&S*, *DepPar*, and *SRL* is 7.3, 7.2, 7.5, and 8.8 words per question, respectively. Hence, extra words in *OA* are penalized for precision, where the length of generated sentences is critical for the calculation of these evaluation metrics. Also, we evaluated the generated answers from each approach in Table 3, with manually generated answers based on an exact word match. Both dependency and SRL-based approaches were better at capturing the candidate answers for date, location, people, subject, and object. Hence, those approaches constructed better authentication questions. On the other hand, other approaches missed required answers, and their F1 scores were lower as a result.

6 Conclusion

Our research explores the novel applications of Question Generation. Although our approach is simple, we generate more suitable authentications than prior QG systems. In the future, we plan to perform a human evaluation of generated questions and answers, as well as leverage machine learning approaches to improve QG.

7 Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments, as well as Ron Artstein, Elsi Kaiser, and Janice Wheeler for their advice.

References

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 33–36. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Michael Heilman and Noah A Smith. 2010a. Extracting simplified statements for factual question generation. In *Proceedings of QG2010: The Third Workshop on Ques-tion Generation*, page 11.

Michael Heilman and Noah A Smith. 2010b. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617. Association for Computational Linguistics.

Michael Heilman. 2011. *Automatic factual question generation from text.* Ph.D. thesis, Carnegie Mellon University.

Karen Mazidi and Rodney D Nielsen. 2014. Linguistic considerations in automatic question generation. In *ACL (2)*, pages 321–326.

Simon S Woo, Jelena Mirkovic, Ron Artstein, and Elsi Kaiser. 2014. Life-experience passwords (leps). In *Symposium on Usable Privacy and Security (SOUPS)*.

Xuchen Yao, Emma Tosch, Grace Chen, Elnaz Nouri, Ron Artstein, Anton Leuski, Kenji Sagae, and David Traum. 2012. Creating conversational characters using question generation tools. *Dialogue & Discourse*, 3(2):125–146.