Compiling Questions into Balanced Quizzes about Documents

Cristina Menghini*
Sapienza University of Rome
menghini@dis.uniroma1.it

Jessica Dehler Zufferey
Coorpacademy
jessica.dehler@coorpacademy.com

Robert West
EPFL
robert.west@epfl.ch

ABSTRACT

In the educational framework, knowledge assessment is a critical component, and quizzes (sets of questions with concise answers) are a popular tool for this purpose. This paper focuses on the generation of balanced quizzes, i.e., quizzes that relate to a given set of documents, and to the central concepts described by the documents, in an evenly distributed manner. Our approach leverages a graph representing the relationships between questions, documents, and concepts, and phrases quiz construction as a node selection problem in this graph. We provide algorithms for constructing the graph and for selecting a good set of quiz questions. In our concrete implementation, we build quizzes for a collection of Wikipedia articles and evaluate them both with simulated students and with real human quiz takers, finding that our balanced quizzes are better suited at determining which articles the user has not read (corresponding to their knowledge gaps) than reasonable baselines.

ACM Reference Format:
Cristina Menghini, Jessica Dehler Zufferey, and Robert West. 2018. Compiling Questions into Balanced Quizzes about Documents. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3269206.3269298

1 INTRODUCTION

Quizzes have played an important role in education for a long time. As opposed to more extensive forms of knowledge assessment, such as essays, quizzes have the advantage of testing knowledge about specific questions in a concise way. For this reason, with the rise of massive open online courses (MOOCs), which require scalable yet effective ways of assessing student knowledge, quizzes have recently been gaining further importance.

In this paper, we define a quiz as a set of questions requiring students to provide answers either by selecting from a set of several candidates (multiple choice) or in the form of a short free-form response. Building quizzes implies two tasks: (1) generating candidate questions from a set of documents (e.g., lectures or encyclopedic articles); (2) selecting a small subset of the candidate questions.

Previous work has mainly focused on generating candidate questions (task 1) from various sources, such as Wikipedia articles [1, 10, 11], text books [4], TED talks [2], or knowledge graphs [9], mostly addressing task 2 by simply selecting a random subset of questions [1, 4, 8, 10]. This paper, on the contrary, focuses on task 2, striving to carefully combine questions into balanced quizzes.

Envisioned use case. We envision the following practical use case for our method. A teacher wants to instruct her class about a given topic. She starts by selecting a small set of topic-related documents (e.g., Wikipedia articles) that she wants her students to read as homework. She would like students to *read all documents* and to *retain the most important facts*. To achieve this, she gives them a quiz together with the assignment. In line with her goals, being able to answer all quiz questions correctly should require reading all documents (i.e., laziness should not be an option); the questions should cover the most central aspects of the documents; and they should at the same time touch upon as many aspects of the documents as possible in as balanced a way as possible. Under these conditions, the quiz gives students a natural incentive to complete the reading assignment as desired by the teacher.

Contributions. The following are the main contributions of this paper: first, we devise a graph-based way of formalizing the properties of a good, balanced quiz (Sec. 2); second, we provide solutions for building the input graph and solving the ensuing optimization problem (Sec. 3); and finally, we evaluate our method in a simulation study as well as with real human quiz takers (Sec. 4).

2 PROBLEM FORMULATION

This section describes how we formalize our setting in terms of a graph (Sec. 2.1) and how we frame quiz construction as a node selection problem in this graph (Sec. 2.2).

2.1 Representation as a graph

QuDoCo graph. Our central data structure is a tripartite graph $G = (V, E)$ whose nodes V represent candidate questions Q, documents D, and concepts C (i.e., $V = Q \cup D \cup C$, hence the name "QuDoCo graph"; Fig. 1).

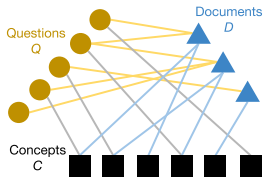


Figure 1: QuDoCo graph.

The edges E are defined as follows: connect a document $d \in D$ to a question $q \in Q$ if d contains the answer to q; connect d to a concept $c \in C$ if d mentions c; and similarly, connect q to c if q mentions c. Constructing G for a given document collection D and pool of candidate questions Q requires deciding if a given document answers a given question and extracting the concepts C occurring across documents and questions. Sec. 3.1 gives details regarding these steps.

2.2 Optimization problem

Given a QuDoCo graph, constructing a k-question quiz corresponds to selecting a subset $S \subseteq Q$ of size $|S| = k$. The desirable quiz properties given in the introduction can be operationalized as follows:

*Work performed partly at EPFL, with support from ERC Advanced Grant 788893 AMDROMA "Algorithmic and Mechanism Design Research in Online Markets".

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3269298

C1. Laziness avoidance: students should not be able to answer all quiz questions S without reading all documents.
O1. Concept centrality: quiz questions should be about the concepts most central to the documents.
O2. Concept balance: quiz questions should relate to concepts in a balanced fashion (rather than focus on few concepts).
O3. Document balance: similarly, quiz questions should relate to documents in a balanced fashion.

C1 is a hard constraint, whereas O1–O3 are objectives that must be traded off against one another; e.g., choosing questions about the most central concepts (high O1) may yield an unbalanced set of questions relating to only a few top concepts (low O2).

Dedicated questions. It is easy to see that constraint C1 is equal to requiring that, for every document d, the quiz S contain at least one question that is answered by d alone, and by no other document. We call such a question a "dedicated question" for d.

Our **optimization problem** is thus to select $S \subseteq Q$ so we

$$\begin{aligned} &\text{maximize} && f(S) := w_1 f_1(S) + w_2 f_2(S) + w_3 f_3(S) \\ &\text{subject to} && |S| \leq k \\ & && \text{and } S \text{ has a dedicated question for each } d \in D, \end{aligned} \tag{1}$$

where $f_i(S)$ captures how well the quiz S meets objective Oi, and where the objectives Oi are traded off with weights w_i, which can be set manually by the teacher, depending on her requirements.

To conclude, we define the f_i's that formalize objectives O1–O3:

O1. For each concept $c \in C$, we compute a *global centrality* $\Gamma(c)$ capturing how prominent a role c plays across the entire document collection D, as well as a *document-local centrality* $\Gamma_d(c)$ capturing how prominent a role c plays in document d (Sec. 3.1 explains how global and document-local centrality is defined and computed). Based thereupon, we compute the *question-local centrality* $\Gamma_q(c)$ of c with respect to question q by summing $\Gamma_d(c)$ over all documents d that answer q. Finally, $f_1(S)$ sums up the global and question-local centralities of all concepts appearing across all questions $q \in S$:

$$f_1(S) = \sum_{q \in S} \sum_{c \in C : \{c,q\} \in E} \Gamma(c) + \Gamma_q(c). \tag{2}$$

O2. To quantify concept balance, we define $f_2(S)$ as the fraction of concepts covered by the selected quiz questions S:

$$f_2(S) = \frac{1}{|C|} \left| \bigcup_{q \in S} \{c \in C : \{c,q\} \in E\} \right|. \tag{3}$$

O3. A priori, it would be a good heuristic to pick questions that have many connections to the documents D, or, in other words, to maximize the number of edges between S and D, i.e., maximize $g(S) := \sum_{d \in D} n_d(S)$, where $n_d(S)$ is the number of edges between S and document d. While this would result in questions that weave together many documents, it might focus most questions on a small subset of documents and neglect others. To encourage an even spreading of questions over all documents, we may award a smaller amount for every additional time a document is touched by the selected questions S: instead of paying a constant amount of 1 for every time a document is touched (as in the above definition of $g(S)$), we may instead pay an amount of $1/i$ for the i-th time a given document is touched, resulting in

$$f_3(S) = \sum_{d \in D} \sum_{i=1}^{n_d(S)} \frac{1}{i}. \tag{4}$$

3 PROPOSED SOLUTION

In our solution, we need to tackle two challenges: first, we must specify how to build the QuDoCo graph that serves as input to the above optimization problem; and second, we are to devise an algorithm for solving the optimization problem. We address these issues in Sec. 3.1 and Sec. 3.2, respectively.

3.1 Building the QuDoCo graph

Nodes. The QuDoCo graph has three types of nodes: documents D, questions Q, and concepts C. The **documents** D are directly provided by the teacher, and as mentioned in Sec. 1, generating candidate **questions** Q from D is not what this paper is about. Instead, the candidate questions are also assumed to be given. They may have been collected by the teacher over the course of the years, crowdsourced, phrased by students via homework assignments, or generated automatically (cf. Sec. 1; deep learning methods in particular have recently been shown to excel at this task [11]).

Finally, **concepts** C are sourced from the documents via a key-phrase extraction method inspired by Rousseau and Vazirgiannis [7], as follows: First, we build a *word graph* whose nodes are nouns and adjectives, and whose edges connect two words if they ever appear within the same 4-gram (edges are directed, following the order of words in the text). Based on this graph, we compute the PageRank of all words and define the 5% most central nodes as *keywords*. Some important phrases consist of more than a single word, so we repeat the same process with bigrams, rather than single words, as nodes, thus obtaining *key bigrams* (we consider only bigrams that co-occur much more often than by chance, by pre-filtering them based on their pointwise mutual information). The concept set C is then defined as the union of keywords and key bigrams. Their PageRank values also serve as the global and document-local centralities (Sec. 2.2): when the word graph is constructed based on the entire corpus D, we obtain the global centrality $\Gamma(c)$; when based on a single document d, the document-local centrality $\Gamma_d(c)$.

Edges. Edges involving concepts C (Fig. 1) are easily found: simply add an edge between a document d (question q) and a concept c if d (q) mentions c. Edges between questions Q and documents D are more challenging, as we need to decide whether a document answers a question. As mentioned above, questions may be generated via machine learning or crowdsourcing; in these cases, we would know explicitly from which document d a question q was generated, so we should certainly add the edge $\{d, q\}$. But q might also be answered by documents other than d, so we need a way of determining, for *any* document d', if d' answers q.

To achieve this, we start from a large dataset of (d, q) pairs, where d is known to answer q, and train a supervised classifier in order to generalize to all pairs. In particular, we use the Stanford Questions Answering Dataset **(SQuAD)** [6], which contains 100K+ questions (alongside answers) generated from 500+ Wikipedia articles.

As documents turn out to be too coarse-grained for this task, we split a document d at the paragraph level, let the classifier predict,

for each paragraph i, the probability π_i that i answers q, and finally compute the probability π_d that d answers q as the probability that at least one of its paragraphs answers q, i.e., $\pi_d = 1 - \prod_i (1 - \pi_i)$.

To estimate π_i, we train a random forest on features that measure the similarities between i and q; between i and q's answer a; and between q and a. For robustness, the features measure similarities in multiple ways: Jaccard coefficient, cosine of TF-IDF vectors, and cosine of word2vec vectors [3]. Results are given in Sec. 4.1.

3.2 Optimization algorithm

Our objective function f (Eq. 1) is a weighted sum of 3 set functions f_1, f_2, f_3, each of which is easily seen to be monotone ($f_i(S \cup \{q\}) \geq f_i(S)$ for all S, q) and submodular ($f_i(S \cup \{q\}) - f_i(S) \geq f_i(T \cup \{q\}) - f_i(T)$ for $S \subseteq T$). As sums of submodular functions are submodular, f is also submodular. This is desirable, as submodularity formalizes an intuitive diminishing-returns property: adding the same question to the solution S later on has less payoff than adding it earlier.

Although maximizing submodular functions under a cardinality constraint is NP-hard, a greedy algorithm is guaranteed to find a 63%-approximation [5]. Our problem has this form, but we also have the additional constraint of having to add a dedicated question for each document (Eq. 1). Hence, we apply a slightly modified greedy algorithm: act greedily at first (starting from an initially empty solution S, add at each step the question with the maximum marginal gain, i.e., $\arg\max_{q \in Q \setminus S} f(S \cup \{q\})$), until the number of documents for which S contains no dedicated question yet equals our remaining budget $k - |S|$, at which point we select the highest-scoring dedicated question for each document that still needs one.

4 EVALUATION

We build quizzes for a collection D of 5 topically coherent Wikipedia articles taken from SQuAD (Sec. 3.1): IMMUNOLOGY, IMMUNE SYSTEM, ANTIBIOTICS, INFECTIONS, BACTERIA. SQuAD contains 849 questions Q about these 5 articles (all SQuAD questions have short text snippets, e.g., entities, as answers), and our goal is to build a balanced quiz by selecting a small subset $S \subseteq Q$ of size $k = 25$. (We set the w_i of Eq. 1 so that the three f_i's lie in a comparable range.)

In SQuAD, each question q is generated from a single document, such that, for each q, SQuAD tells us one document that answers q. But other documents might answer q as well, so we determine the full set of question–document edges by reading all 5 articles in detail and manually deciding for each question by which articles it is answered. Note that this would already suffice for our evaluation on the above 5 articles, but we certainly could not afford such a manual effort in a real deployment scenario, so Sec. 4.1 shows that our classifier for detecting question–document edges (Sec. 3.1) works well. We then proceed to demonstrate that our quizzes are better than baselines at detecting whether a student has read all k articles, and if not, which one they skipped (Sec. 4.2).

4.1 Building the QuDoCo graph

Here we evaluate the random forest (RF) classifier of Sec. 3.1 for determining if a question q is answered by a document d. As we manually labeled all (d, q) pairs consisting of our 5 Wikipedia articles and the 849 question candidates, we may readily evaluate precision and recall for a wide range of thresholds on the RF's probabilistic output. As a result, we find that (d, q) edges included in SQuAD are easier to detect than the extra edges we added by manual inspection; crowd workers who constructed SQuAD seem to have a bias for generating the most obvious questions: e.g., when adding edges for which the RF outputs a probability above 80%, we obtain recall (precision) of 99% (95%) on the SQuAD edges, and 80% (77%) for our manually added edges. Still, we believe these numbers are sufficiently high for a real-world deployment of our classifier.

The resulting QuDoCo graph's connectivity is nontrivial: 78% of questions are connected to a single document (and could thus serve as dedicated questions), 12% to 2, 6% to 3, and 4% to 4 or 5. On average, each question (document) is connected to 3 (10) concepts.

4.2 Detecting unread documents

In the envisioned use case of our quizzes, a teacher asks her students to peruse a document set D and complements this assignment with a quiz that covers all of D, incentivizing them to learn about the most central concepts of D. In this setting, it would be desirable if she could tell from the quiz results if a student indeed read all the documents, and if not, which documents they skipped, as this would allow her to detect likely knowledge gaps on which to focus.

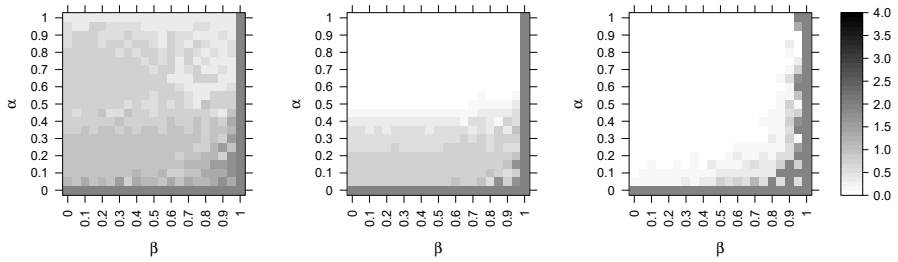
To do so, we compute, for each student s and document d, the fraction p_d of d's questions that s answered correctly, and we then rank all documents in increasing order of p_d. In the ideal case, the documents s did not read will have a lower fraction of correctly answered questions—and thus a lower rank—than those s did read.

Here we evaluate how well our quizzes are suited for detecting skipped documents in the above way. Concretely, we have students take quizzes about the above 5 Wikipedia articles, making sure that, for each student s, there is one article d_s they do not read, by excluding d_s from the list of articles we ask s to read. We then compute the rank of d_s with respect to the fraction p_{d_s} of d_s's correctly answered questions, and average this rank over all students s. The lower the average rank of d_s, the better.

We perform two evaluations: one with computationally simulated students, the other, with human quiz takers recruited on the Mechanical Turk crowdsourcing platform. We compare our method of constructing balanced quizzes to two baselines: a **random baseline** that picks k questions from Q uniformly at random, and a **dedicated-question (DQ) baseline** that first picks a random dedicated question for each document $d \in D$ (i.e., a question that can only be answered by d) and then picks $k - |D|$ questions uniformly at random. We run each baseline several times and report the average.

Although random selection might seem like a weak baseline, it is realistic, as it is frequently used in practice (cf. Sec. 1). The DQ baseline is expected to be a strong competitor under the above evaluation: the answer to the dedicated question q_{d_s} of the skipped document d_s is contained only in d_s itself, so it is likely not to be answered correctly by s (unless s already knew the answer before), which in turn decreases the score p_{d_s} and thus the rank of d_s.

Simulated evaluation. For our simulation, we assume a simple model, specified by two parameters. The first parameter, α, defines the probability that a student s discovers the answer to a question q by reading a document containing the answer. The second parameter, β, defines the probability that s knows the answer to q already before taking the quiz, i.e., without reading any document. Both α and β are constant across all s and q; i.e., we assume all questions have the same difficulty for everyone. Also, we assume documents are independent of each other with respect to letting s discover



Method	Mean rank of unread document*	Mean % correct answers
Balanced	1.40 [0.88, 1.96]	49%
Dedic.-qu.	1.60 [1.12, 2.08]	42%
Random	2.00 [1.36, 2.60]	38%

*95% confidence intervals in parentheses.

(a) Random baseline **(b) Dedicated-quest. baseline** **(c) Balanced quiz** **(d) Human evaluation**

Figure 2: (a–c) Simulated evaluation results; colors represent mean rank of unread document (lighter colors are better); definitions of α, β in Sec. 4.2. (d) Human evaluation results. Balanced quizzes are best at detecting documents not read by student.

the answer to q, and that knowing the answer ahead of time (β) is independent of (re)discovering it by reading a document (α).

The probability of answering correctly a question q whose answer appears in n_q documents is thus $p_q = 1 - (1-\alpha)^{n_q}(1-\beta)$, and a document d's expected fraction p_d of questions answered correctly (our ranking criterion) is the average of p_q over the n_d quiz questions that d answers: $p_d = \frac{1}{n_d} \sum_{q \in S:\{d,q\} \in E} p_q$.

The heatmaps in Fig. 2(a–c) plot the mean rank of the skipped document for our quizzes and for the two baselines as a function of α and β. As there are 5 documents, the mean rank can take on values between 0 and 4, with lower values (lighter colors) being better. As expected, the random baseline performs worst. The DQ baseline, on the contrary, reveals the skipped document nearly perfectly when $\alpha \geq 0.5$, but does much more poorly for smaller values of α (i.e., when the simulated student reads sloppily). Our balanced quizzes, finally, show the best performance, with the average rank of the skipped document being 0 for a much wider range of (α, β). Also note that the upper left triangle of the heatmaps is most meaningful, since here the probability of discovering an answer when reading a document is larger than the probability of already knowing the answer before reading any document. In this regime, unlike the two baselines, our quizzes show essentially perfect performance.

Human evaluation. The above simulation study, though useful for initial insights, makes rather strong assumptions. Hence we also conduct an evaluation involving 75 real human quiz takers, recruited via Mechanical Turk. We again use the same 5 immunology-related articles and require $k = 25$ questions per quiz. The 75 quiz takers are split evenly over the three methods. We build one quiz per method, with 25 users taking each quiz. We evaluate each quiz in 5 conditions (5 users per condition): in each condition, a different one of the 5 articles is hidden from the user, the other 4 are shown, thus simulating a scenario where the user skips exactly that document. Users provide their answers as short free-form texts, and we manually determine the correctness of each answer.

Averaging over the 25 instances of each quiz, we obtain the mean rank of the skipped document. Fig. 2(d) shows that it is lowest for our balanced quizzes (1.4), followed by the DQ baseline (1.6), with the random baseline again performing worst (2.0). (Likely due to the small sample size of 25 instances per quiz, the differences are not statistically significant.) The mean ranks are higher than for most (α, β) in our simulations, probably because the quizzes are generally rather hard, with less than half of all questions answered correctly (Fig. 2(d), right column), which induces a lot of noise. Note, however, that our quizzes are easier to answer (49% of questions answered correctly on average) than the baselines (42% and 38%).

5 DISCUSSION AND CONCLUSION

This paper presents a method for combining questions about a document collection into quizzes where questions relate to documents and concepts in a balanced fashion. Our contribution adds to previous work, which has mostly concentrated on generating candidate questions from document collections, whereas our focus is on compiling candidate questions into meaningful quizzes.

Our approach leverages a graph representing the relationships between questions, documents, and concepts, and phrases quiz construction as a node selection problem in this graph. We provide a method for constructing the graph and for selecting a good set of quiz questions using a greedy algorithm. Our results are promising, both in a simulation study and in an evaluation with human quiz takers. Future work should go further both quantitatively and qualitatively, by conducting experiments with more participants as well as with students rather than crowd workers, since the latter might not be representative of real classroom settings.

To further improve our quizzes, future work also should aim to produce sequences, rather than sets, of questions (such that questions build on each other and guide students through the documents in a meaningful order). Finally, it would be useful to extend our algorithm such that it can produce many different quizzes for the same document collection, rather than a single optimal one.

REFERENCES

[1] Q. Guo, C. Kulkarni, A. Kittur, J. Bigham, and E. Brunskill. 2016. Questimator: Generating knowledge assessments for arbitrary topics. In *IJCAI*.
[2] Y. Huang, Y. Tseng, Y. Sun, and M. Chen. 2014. TEDQuiz: Automatic quiz generation for TED talks video clips to assess listening comprehension. In *ICALT*.
[3] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.
[4] R. Mitkov, L. Ha, and N. Karamanis. 2006. A computer-aided environment for generating multiple-choice test items. *Nat. Lang. Eng.* 12, 2 (2006), 177–194.
[5] G. Nemhauser, L. Wolsey, and M. Fisher. 1978. An analysis of approximations for maximizing submodular set functions – I. *Math. Progr.* 14, 1 (1978), 265–294.
[6] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. *arXiv:1606.05250* (2016).
[7] F. Rousseau and M. Vazirgiannis. 2015. Main core retention on graph-of-words for single-document keyword extraction. In *ECIR*.
[8] K. Sakaguchi, Y. Arase, and M. Komachi. 2013. Discriminative approach to fill-in-the-blank quiz generation for language learners. In *ACL*.
[9] D. Seyler, M. Yahya, and K. Berberich. 2015. Generating quiz questions from knowledge graphs. In *WWW*.
[10] A. Singh Bhatia, M. Kirti, and S. Saha. 2013. Automatic generation of multiple choice questions using Wikipedia. In *PReMI*.
[11] T. Wang, X. Yuan, and A. Trischler. 2017. A joint model for question answering and question generation. *arXiv:1706.01450* (2017).