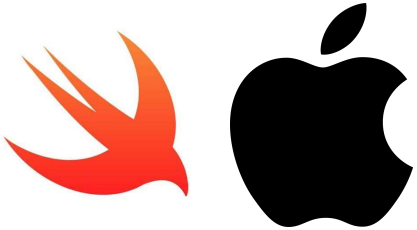
Swift App Backend Team
Spring 2023

Zhen Hong Tan, Avi Kapadia, Devang Ajmera, Sabina Ajjan, Rishi Nopany

Project Description



- The purpose of this quiz app is to aid students in studying the course.
- Offer a fun and engaging way for students to learn and reinforce their knowledge.
- This app is made specifically for iOS devices
- Allowed the members to gained more mobile development experiences

Semester Goals

Backend Team

- Improved from the existing Swift version of Quiz App:
  - Researched and implement a remote database is most suitable for mobile applications
  - Implement more new features on the app.

Algorithms Team

- Develop and research algorithms that utilize artificial intelligence in order to improve learning for the user.
- Utilize natural language processing to allow for more efficient and accurate keyword searching and question grouping.

User Research

- Looked into various databases
  - MySQL
  - Firebase
  - Realm
  - MongoDB
- Looked into other quizzing apps
  - Quizlet
  - Anki

Research Doc: https://docs.google.com/document/d/15algXJnZo9IXTvhdOfe71PjFgjcvsyvNWr_wMwEMzCs/edit

Firebase

- Firebase is a NoSQL database that stores and syncs data in real-time.
- Firebase handles large data set effectively
- Supports a lot of programming languages
- Support multiple read/write at the same time

UserData > YDe6sFkZZC4B..

More in Google Cloud

vip-its-f3c1f

+ Start collection

UserData

UserData

+ Add document

YDe6sFkZZC4BJVYoSXq6

oMwmlsGSdyzojCX8blcC

YDe6sFkZZC4BJVYoSXq6

+ Start collection

QuestionsAndAnswers

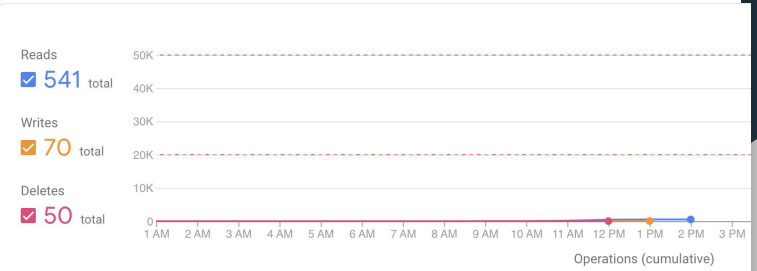
questions_answers

+ Add field

Password: "password1"

SeenQuestions: ""

UserName: "user1"



UserData > YDe6sFkZZC4B... > QuestionsAndA... > 0mQF8i6372Zk...

More in Google Cloud

YDe6sFkZZC4BJVYoSXq6

+ Start collection

QuestionsAndAnswers

questions_answers

+ Add field

Password: "password1"

SeenQuestions: ""

UserName: "user1"

QuestionsAndAnswers

+ Add document

0mQF8i6372ZkmXHICrxB

10HO586ilYzriLdRYcK6

5PSpl6B4CYZkcuW5WUhl

7QNTZifFIQ2fjbAkNzQU

7z5I5nhPhl0nWFsWPWvA

EYdO2b6bxm1oWHdNB1qF

FOiYIklHMAgjs8zE5xqG

QFIkVfPOpPyVSjcewCV3

a9loGcnVSahVF7CwQ3zr

aYCzqmYjSpGVDYQTGmlZ

cVFpKTTz7Y4nyDmIwtb0

dzD60vz5aH594MRCxjMd

iN6cO83iEbtYIhNhTuGr

r9tIzZt3fyWeSaHs3qts

0mQF8i6372ZkmXHICrxB

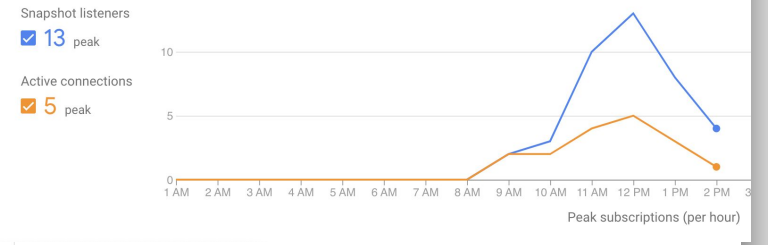
+ Start collection

+ Add field

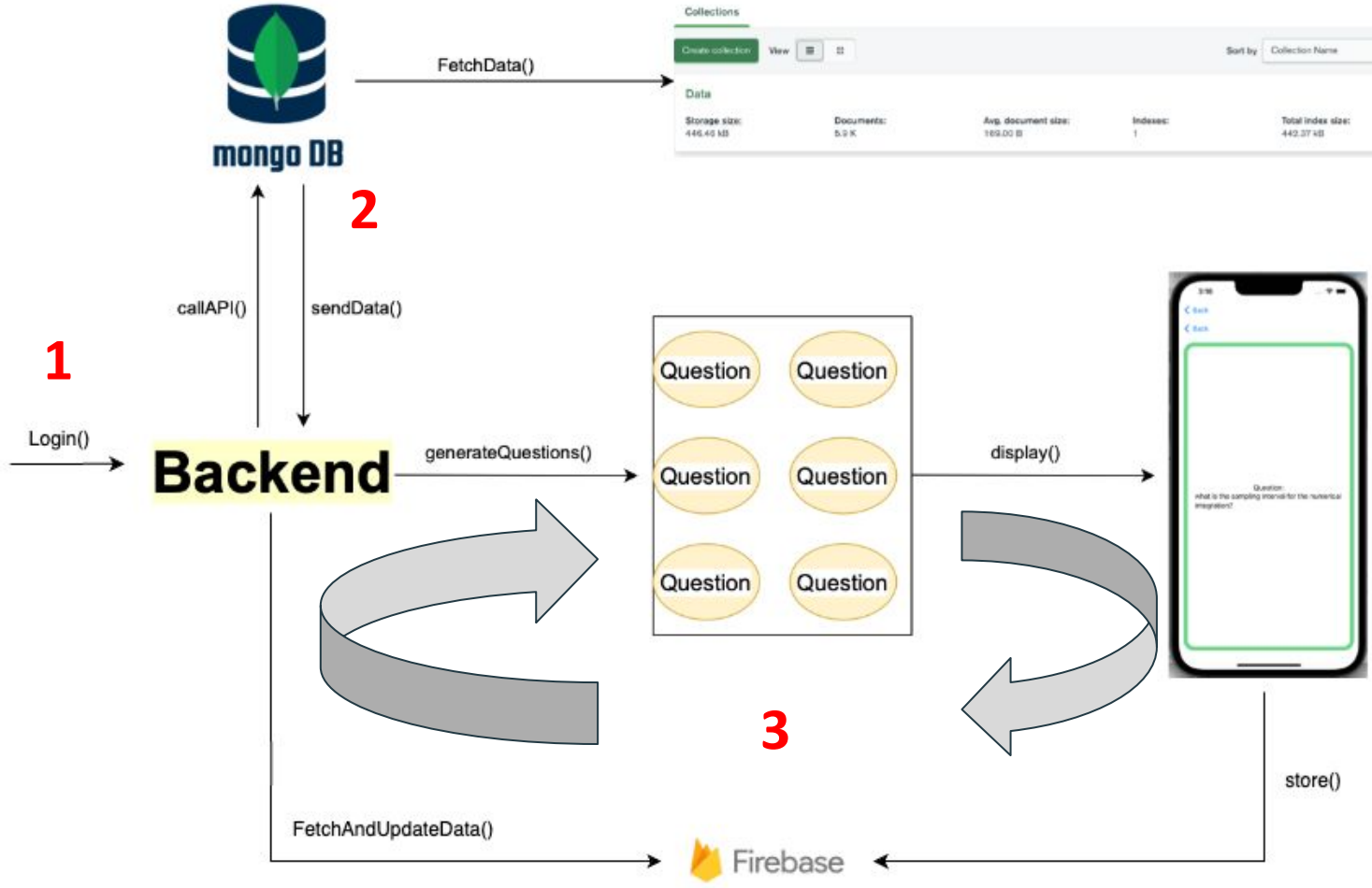
Answer: "The fundamental frequency"

Question: "in mathematical terms, what is the greatest common divisor?"

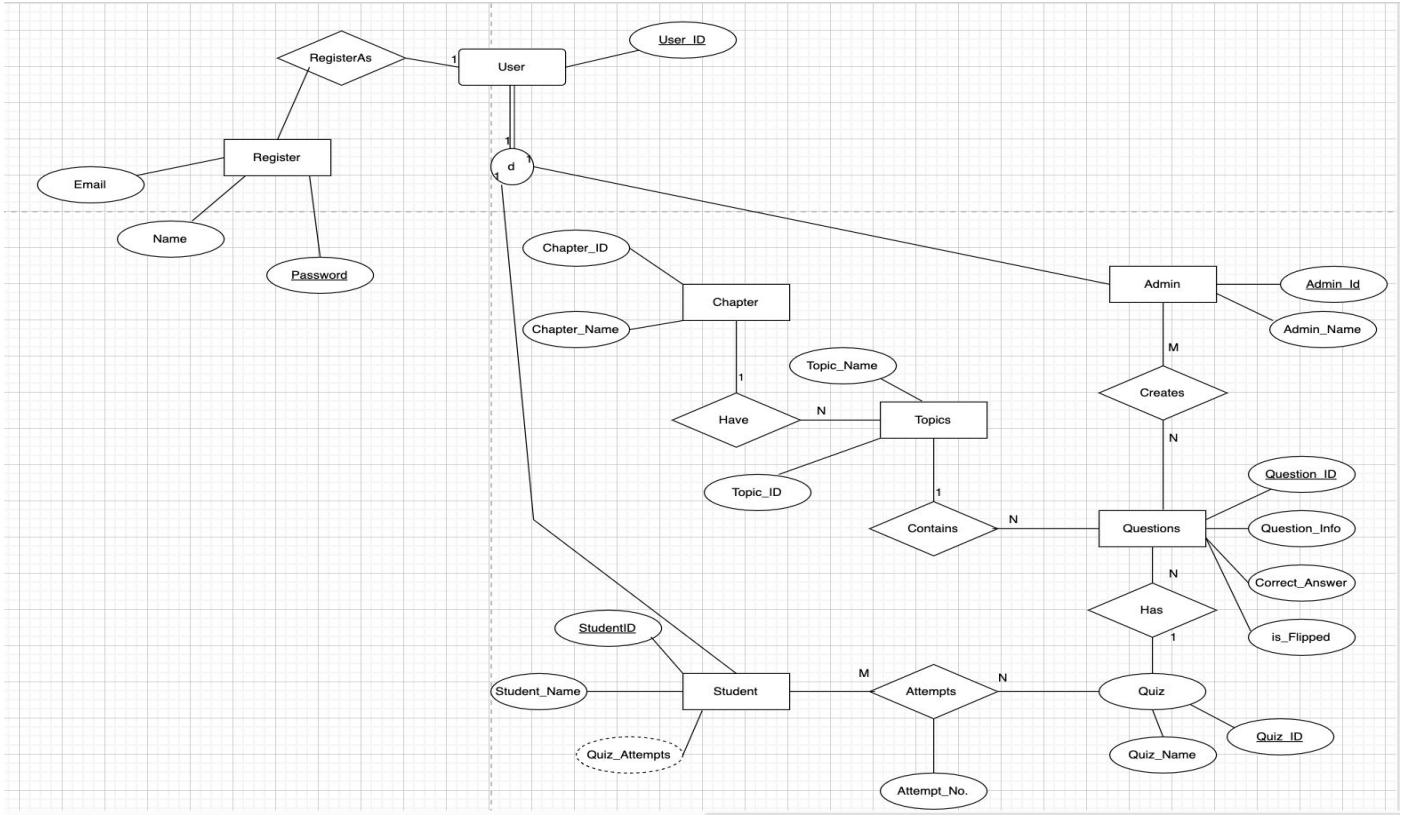
Subscription Metrics



Backend Flow



Entity Relationship Diagram



Implementing Next Button

- @State tag: keeps track of the state of var outside of view
    - Speciality of Swift
- Num and isFlipped are both @State vars
- flipCard() and isFlipped were previously implemented
    - Able to condense code using functions already there

```
@State var num = Int.random(in: 1..<5140)
```

```
ZStack {

    Button("Next") {
        num = Int.random(in:
            1..<5140)
        if (isFlipped == false) {
            flipCard()
            isFlipped.toggle()
        }
```

Analysis

$$\text{Avg pre flipping time} = \frac{\text{Total time spent on questions pre flipping}}{\text{Total number of attempts}}$$

$$\text{Topic accuracy rate} = \frac{\text{Total correct answers for a specific topic}}{\text{Total quiz attempts for that topic}}$$

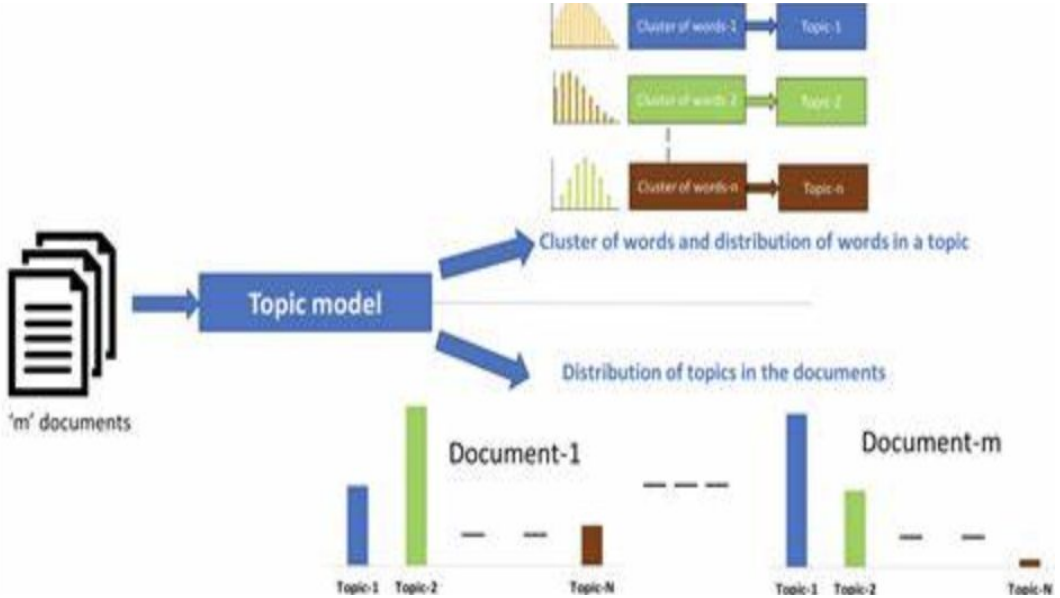
Clustering analysis: Clustering algorithms can be used to group users based on similar behavior, such as usage frequency, performance, and flashcard preference. This can help identify user segments for targeted marketing or personalized study recommendations.

Demo

Latent Dirichlet Allocation Algorithm

How it works:

- LDA assumes that each document is a mixture of topics, and each topic is a mixture of words.
- LDA assigns each word in the documents to a random topic and iteratively updates the topic assignments based on the probability that a given topic generated that word and the probability that a given document contains that topic.
- LDA estimates the topic-word distribution and document-topic distribution that best explain the observed data by maximizing a likelihood function.
- LDA is useful for tasks such as text classification, clustering, and topic modeling. It can be used to automatically find patterns and themes in large collections of text documents.

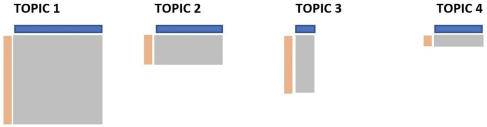


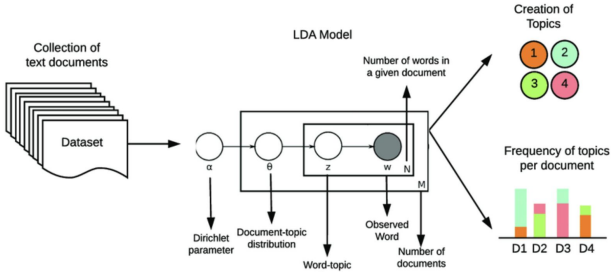
LDA Implementation

- Latent Dirichlet Allocation (LDA) uses Bayesian inference to estimate the posterior distribution of the the topic assignments based off the words in the documents
- LDA uses the Dirichlet distribution to model the topic-word and document-topic distributions, which are both probability distributions over the words and topics
- LDA assigns each word in a document to a random topic based on the document-topic distribution and the topic-word distribution.
- LDA iteratively updates the topic assignments using Gibbs sampling, which involves sampling a new topic for each word based on the current topic assignments for all other words in the corpus.

- Second let's represent the matrix *n(d,k)* in the following way to show how much a document use each topic

· Third, let's represent *v(k,w)* in the following way to show how many times each topic is assigned to this word





Code

```
        self.idx2word = {i: word for word, i in self.word2idx.items()}

        self.num_docs = len(corpus)
        self.doc_lengths = [len(doc) for doc in corpus]

        self.topic_assignments = [[np.random.randint(self.num_topics) for _ in range(length)] for length in
                                  self.doc_lengths]
        self.topic_counts = np.zeros((self.num_topics, len(self.vocab)))
        self.doc_topic_counts = np.zeros((self.num_docs, self.num_topics))

        for doc_idx, doc in enumerate(corpus):
            for word_idx, word in enumerate(doc):
                z = self.topic_assignments[doc_idx][word_idx]
                self.topic_counts[z, self.word2idx[word]] += 1
                self.doc_topic_counts[doc_idx, z] += 1

        for _ in range(self.max_iter):
            for doc_idx, doc in enumerate(corpus):
                for word_idx, word in enumerate(doc):
                    z = self.topic_assignments[doc_idx][word_idx]
                    self.topic_counts[z, self.word2idx[word]] -= 1
                    self.doc_topic_counts[doc_idx, z] -= 1

                    p_z = (self.doc_topic_counts[doc_idx] + self.alpha) * \
                          (self.topic_counts[:, self.word2idx[word]] + self.beta) / \
                          (self.topic_counts.sum(axis=1) + self.beta * len(self.vocab))
                    z = np.random.multinomial(1, p_z / p_z.sum()).argmax()

                    self.topic_assignments[doc_idx][word_idx] = z
                    self.topic_counts[z, self.word2idx[word]] += 1
                    self.doc_topic_counts[doc_idx, z] += 1
```

```
                    z = np.random.multinomial(1, p_z / p_z.sum()).argmax()

                    self.topic_assignments[doc_idx][word_idx] = z
                    self.topic_counts[z, self.word2idx[word]] += 1
                    self.doc_topic_counts[doc_idx, z] += 1

    def bag_of_words(self, doc):
        bag = np.zeros(len(self.vocab))
        for word in doc:
            if word in self.vocab:
                bag[self.word2idx[word]] += 1
        return bag

    def topic_word_distribution(self):
        return (self.topic_counts + self.beta) / \
               (self.topic_counts.sum(axis=1)[:, np.newaxis] + self.beta * len(self.vocab))

    def document_topic_distribution(self):
        return (self.doc_topic_counts + self.alpha) / \
               (self.doc_topic_counts.sum(axis=1)[:, np.newaxis] + self.alpha * self.num_topics)
```

Demo/ Results:

```
Document 0: Topic 2
Document 1: Topic 3
Document 2: Topic 1
Document 3: Topic 1
Document 4: Topic 1
Document 5: Topic 0
Document 6: Topic 1
Document 7: Topic 2
Document 8: Topic 1
Document 9: Topic 4
Document 10: Topic 1
Document 11: Topic 2
Document 12: Topic 1
Document 13: Topic 2
Document 14: Topic 2
Document 15: Topic 4
Document 16: Topic 4
Document 17: Topic 4
Document 18: Topic 3
Document 19: Topic 2
Document 20: Topic 4
Document 21: Topic 1
Document 22: Topic 2
```

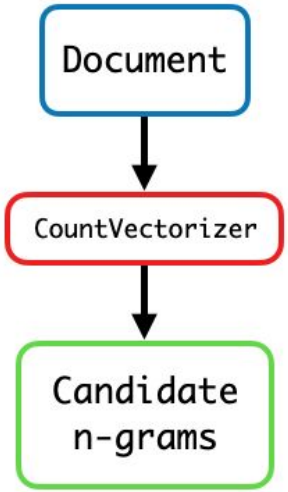
	question
1	
2	What shows numerical examples of complex numbers?
3	What are complex numbers plotted as vectors in the two-dimensional "complex plane"?
4	What are numbers needed to solve for the two roots of a quadratic equation?
5	The point lies in the first quadrant of what?
6	Where is the angle always measured?
7	What is the correct angle for?
8	The laws of exponents apply to what formula?
9	How do you practice computations for complex numbers?

Editing via Table Editor might change the format of the CSV file. Follow the link for more info

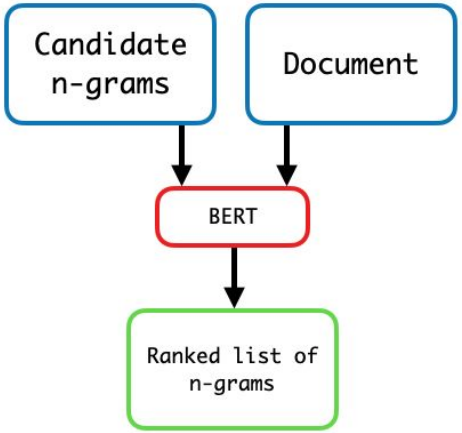
BERT

- **BERT** a.k.a Bi-directional Encoder Representation of Transformers is an encoder-only model which is designed to learn **deep bidirectional representations of text segments from an unlabeled text**.
    - **MLM (Masked Language Modeling):** enables the model to learn the representation of every word/token in the input based on words that occur in its context.
    - **NSP (Next Sentence Prediction):** given two sentences, knowing if the second sentence follows the first one or not.
- We implement BERT in our keyBERT algorithm, which is composed of three different parts:
    - Extracting n-grams from a given text document (question) based on frequency.
    - Using BERT to calculate semantic distance (how relevant the n-grams generated are to the original document (question).
    - Using maximum marginal relevance and cosine similarity to diversify results.

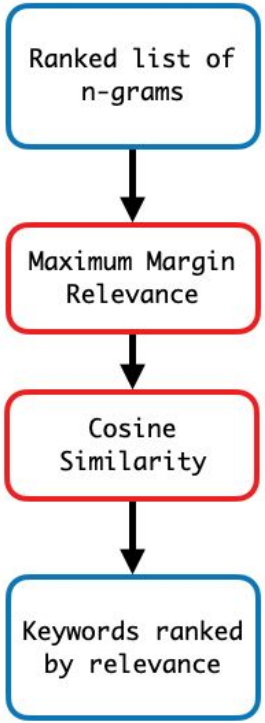
Extracting n-grams



Using BERT to calculate semantic distance



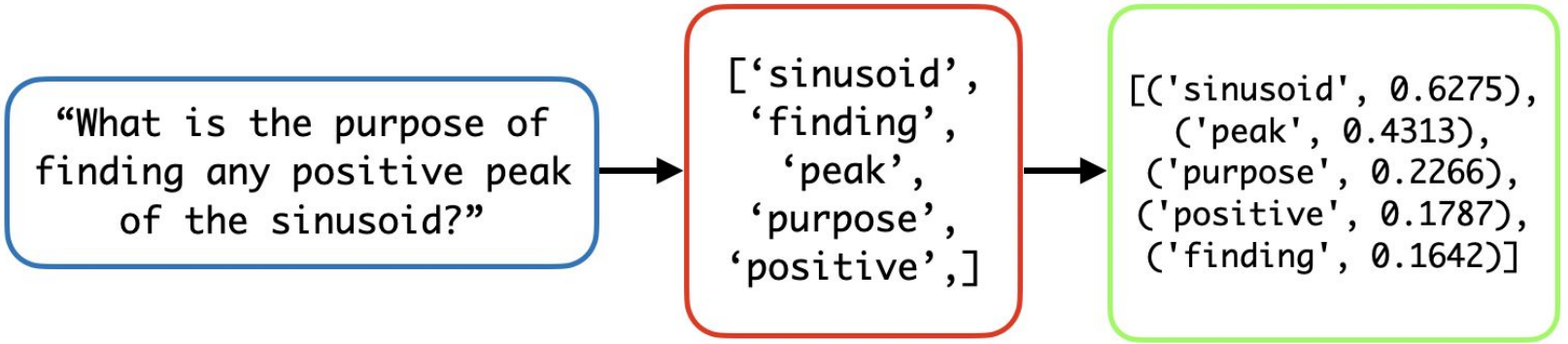
Diversifying results



Results on the first 250 questions:

```
[('matlab', 0.084),
 ('signal', 0.064),
('function', 0.06),
('complex', 0.056),
('signals', 0.048),
('vector', 0.048)…]
```

Sample


“What is the purpose of finding any positive peak of the sinusoid?”
[‘sinusoid’, ‘finding’, ‘peak’, ‘purpose’, ‘positive’,]
[('sinusoid', 0.6275), ('peak', 0.4313), ('purpose', 0.2266), ('positive', 0.1787), ('finding', 0.1642)]

Future goals

Database Team:

- Combine our project with the frontend team
- Profile page implementation
- Handle formulas in latex format
- Add images to import for quiz flashcards and MC questions
- Utilize the database to create a more effective learning environment

Algorithms Team:

- Merge data from keyword and topic extraction algorithms into QuizApp.
- Create question-grouped quizzes based on keywords in QuizApp. Maybe create a new screen after the user selects “quiz” where the user has the option of a random quiz or a quiz with a special topic