Exercise Concept and Design

Question –> answer, that is the concept of the game. The game is designed to be fast paced, so to facilitate that only short questions are allowed.

Dictionary

Missing Content Paragraph

Maximum Total Length: 200 characters

Maximum Number of Missing Parts: 6

Maximum Length of Each Part: 20 characters

Missing Content Structure

Paragraphs are added from exercises crated by Content Adder users. We shall use the term Paragraph to describe a single question and its answer in an exercise. The structure of such a paragraph (Q-A) is divided:

* Question Body
* Answer Body

An exercise must contain at least five Paragraphs. We do not allow Paragraphs to exist independently for the following reasons:

* Some Paragraphs hold additional unique guidance regarding how to solve them. For example, in sentence completion, one is required to complete the sentence. But it may additional contextual meaning such as – use the correct modal verb. This additional guidance should not be duplicated with any instance of a Paragraph.
* To define a minimal number of Paragraphs that must be added by any user – larger question database.
* To support future features including exact weights of Paragraphs within exercises.
* To have the ability to present more than one similar Paragraph in a row (similar questions are defined as sharing an exercise).
* To support the option of having users adding Paragraphs to each other’s exercises.
* To support future exercises where: Paragraphs are related to one another; share contextual content (e.g. story); relate to the answers of one another (e.g. matching).

To create an exercise, a user must follow these minimal guidelines:

1. Supply a guiding instruction. No more than 150 UTF8 characters.
2. Supply a minimum of 5 paragraphs, each constrained to maximum length of 200 UTF8 characters.
3. Supply at least one correct answer per paragraph, with a maximum consecutive length of 50 chars.

Notes:

1. The details of the user are stored within the exercise.
2. The paragraphs must all conform to the instruction given by the user. Although this is currently not validated by the system (we assume content verification by humans at the time being), consideration of a more strict input system is in order.
3. In cases of Paragraphs where answers are missing content within the sentence, the standard mark is the following: ‘<!>-’. This withholds information regarding the characters that constitute a part of an answer, and provides the GUI designers the ability to represent some other unique mark.

The Paragraph Tag Schema

We view all possible Exercises to have a single format:

1. Instructions
2. The question content
3. A place to write answers

Sometimes, either instruction or question content is not required. Consider the following:

*Select the odd noun:*

1. *Fish*
2. *Bread*
3. *Water*
4. *Dog*

In here, the instructions are the question.

We may rewrite the previous:

*Select the odd noun:*

*Fish, Bread, Water, Dog.*

*Answer: \_\_\_\_\_\_\_*

Here the second sentence is the question while the previous and following are instructions and answer, respectively.

We view that all paragraphs may be written as the latter case and so we define the following rule for creating any paragraph.

**All paragraphs must contain a placeholder for answers to be inserted (such as with the example above).**

In case where you are required to complete content, the place to write the answer in within the paragraph itself. Otherwise it follows it.

Representing an answer placeholder is the following tag expression:

Where <!> marks the start of input, s is a any sequence of characters and x is any number of unknown characters, which the user must input. The entire length of all concatenated place holders should not exceed the maximum length allowed per answer (50 chars).

Examples using such a template:

*The current president of the United States of America.*

*B\_\_\_\_ \_\_ssein \_b\_ma*

<!><s>B</s><x>4</x><s> </s><x>2</x><s>ssein </s><x>1</x><s>b<s><x>1</x><s>ma</s></!>

20 characters in total.

The structure that is of interest to the user is the input template with which he interacts, i.e. where he inputs the answer. All Paragraphs by default do not have multiple answer options (we shall elaborate on this decision soon).

The second step we must take is concerns the answers to a paragraph. An answer is defined as follows:

**An ordered group of individual character sequences, each mapped to exactly one answer template defined by its order of appearance, and evaluating as single unit to one of two states: True or False.**

A paragraph may have multiple correct answers to it. While such a case demands the addition of individual weights per answer, we shall assume for the time being that all answers are of equal weight.

Functionality

After defining the structure of a Paragraph, i.e. with what the user may interact, we proceed to its functionality – i.e. how the user interacts with an exercise Paragraph.

We mentioned earlier that the default scheme for answer input is free-text as opposed to multiple answer. We have an underlining assumption guiding us through this decision. We vision that every question may be altered to represent itself in various formats. We may introduce a set of answers to a question and query for its validity (yes/no). We may present all possible answers in an exercise at once and ask the user to match a question to one or more answers (matching). Or we may introduce dynamically generated answers close by some characteristics to one or more correct answers, and request the student to select the correct one (multiple answer). We shall examine this assumption later.

The Solution Tag Schema

A Paragraph is simply a sequence of chars. They may be words, numbers (math problems) or anything else a user may be questioned about.

<SOLUTION>

<PARAGRAPH ID=\_id>

<TEMPLATE></TEMPLATE>

<TRUE>

<ITEM></ITEM>

</TRUE>

This marks represents the answer template of any paragraph. It is defined as follows:

X – represents a part of an answer

We describe the following use case for creating an exercise:

UC1: Create Missing Content Exercise

Actor: Content Adder

User selects missing content type exercise. The details of the user are assigned and the template is presented. He enters an optional name for the exercise. He repeats the process of creating Paragraphs until either the minimal allowed number of paragraphs are created or else.

To create a new Paragraph one follows these steps:

1. Enter a question or instruction per each paragraph.
2. Enter the complete paragraph
3. For each group of words that should be missing from the text.
   1. Mark the starting position with <?>
   2. End with </?>
4. Submit the paragraph for validation.
5. If validation failed, follow correction steps returned.
6. Else the answer is returned separated to parts, where each part was marked as previously mentioned.

A valid paragraph has now been created. A user may select to add additional false/true alternatives to the paragraph. In this case he must follow these steps:

1. Select to add answers to a paragraph
2. Enter content in one or all parts of an answer.
3. Mark the answer as false/true
4. Submit to validation.
5. If validation failed, follow steps for correction.
6. Else, you’re done.

Two basic exercises are available and provide the base of the game. One is Missing Content

Definition of an Answer

An answer is made of at least one part. An answer part is defined as a consecutive sequence of characters. It may contain any character supported. The content creator should be aware of this definition and consider it while deciding on answer, but ultimately it is rather intuitive – If a person is required to supply an answer by parts, the parts are determined by the context of the question. They may be letters in a word, if the assignment regards spelling for example, or

An answer is made of more than one part if it has more than one such sequence of characters, while none overlap. An answer is considered complete is all parts submitted are correct. In case of the above example, “tired” and “like” are parts of an answer. Every answer is defined by the order of its parts. By default, the same parts in different order are not the same. You might notice correctly that ultimately, the ‘correct’ answer is defined by the content creator.

On the other side, while an answer may represent the same set of characters, any non-alpha-ascii-chars are considered invariant and non-interactive. This means a user may not input non language characters to submit an answer to a question. Notice that this rule may be overridden if the input is made by selection rather than insertion.

A special mark is used to give contextual association between an QB and its AB. This mark is optional and is mainly used in cases of sentence completion. Two purposes for establishing a single standard mark:

1. Avoid different marks in different questions that are used for the same purpose.
2. Control the unique appearance of such a mark through the GUI.

Once an answer is created, it is parsed and only non-alpha-ascii-chars are retained.

We take English as the default case. In this case, the body is allowed to represent any ascii character

Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| Body | Answer platform | Answer |  |
| A cat is ? animal. | \_\_\_\_ | An |  |
| ? book ? many pages. | \_\_\_\_, \_\_\_\_ | A, has |  |
| What is the capital of Israel? | \_\_\_\_\_\_ | Jerusalem |  |

To define a template a user first defines the structure of an answer – whether or not the answer

Once a user defines the platform of an answer, or it Template, several variables are controllable:

Templates of exercises are unique and have some unique actions a player may perform in each. Each exercise has an answering scheme – Multiple answer | free text. And exercise may hold both. In every game, a number of questions are played from an exercise, depending on its format and available number of questions. The base types are:

1. Syntax related – were the answer to a question is an alteration of the base syntax of the question, in areas specified by the exercise creator.
2. Answer matching – any other type.

The choice is up to the content creator. If he sees that the exercise tests a student’s knowledge on syntax, choose syntax. Otherwise choose the other. Technically with answer matching, most types of syntax related questions may be modeled.

A player is rewarded a set amount of points upon answering any question correctly. Additional points are rewarded based on the preconditions prior to a user’s answer.

Preconditions:

1. Alteration made by an opponent.
2. Alteration made by team member.
3. State of the game.

Players receive immediate feedback on any question answered. Default amount of times a player may answer the same question in a game is one.

Players’ actions are divided into the following categories:

1. Exercise Template– interactions that are unique to a format of an Exercise
2. Game Mode – interactions that are unique to the type of game played
3. Users – interactions with other users