Automated Generation of Customizable Recall Questions

Eran Raveh

Seminar für Sprachwissenschaft, Universität Tübingen

November 13, 2015

Extract from the B.A. thesis with the same title, supervised by Prof. Dr. Detmar Meurers, and written as part of the Hauptseminar NLP for Tutoring Systems, SS15
Outline

1 Introduction

2 Background
   - Reading Comprehension
   - Automated Question Generation

3 ARCQC
   - Overview
   - Text Analysis
   - Task Generation
   - Overgeneration and Ranking
   - Generalization and Customization

4 Evaluation
   - Method and Settings
   - Output Examples
   - Results
Outline

1 Introduction

2 Background
   - Reading Comprehension
   - Automated Question Generation

3 ARCQC
   - Overview
   - Text Analysis
   - Task Generation
   - Overgeneration and Ranking
   - Generalization and Customization

4 Evaluation
   - Method and Settings
   - Output Examples
   - Results
Reading Comprehension

**Definition** (Rayner et al., 2001)

The set of abilities that includes understanding of a text, making inferences from it, recall information from it and apply prior knowledge to it, [...]

Eran Raveh

Automated Generation of Customizable Recall Questions
**RC Tests**

- One of the language skills typically tested (e.g. TOEFL, TestDaF)

<table>
<thead>
<tr>
<th></th>
<th>Verbal (time-depended)</th>
<th>Written (time-proof)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Productive</td>
<td>Speaking</td>
<td>Writing</td>
</tr>
<tr>
<td>Static</td>
<td>Listening</td>
<td>Reading</td>
</tr>
</tbody>
</table>

- Learned at early age, important later in life
- one of the most important skills at university (Levine et al., 2000)
- 2% and 12% lower grades than LC in TOEFL and TOEIC respectively (2012 reports)
Task Types

- Free text
- Paraphrasing/Summarization
- Drawing
- Graphical analysis
- Parallelism/analogy
- Marking/Copying
- Open questions
- Fill in the blank
- Multiple-choice
- True/False

Differentiate!

Productive vs. Informative

Inference vs. Recall
Task Types

- Free text
- Paraphrasing/Summarization
- Drawing
- Graphical analysis
- Parallelism/analogy

- Marking/Copying
- Open questions
- Fill in the blank
- Multiple-choices
- True/False

Differentiate!

Productive vs. Informative

Inference vs. Recall
Recall Questions

- Sometimes referred to as "factual questions"
- Deal with recalling information from (short-term) memory
- Check retrieval of relevant information from the correct place
- Target a single piece of information
- Often used in language level tests
Outline

1. Introduction

2. Background
   - Reading Comprehension
   - Automated Question Generation

3. ARCQC
   - Overview
   - Text Analysis
   - Task Generation
   - Overgeneration and Ranking
   - Generalization and Customization

4. Evaluation
   - Method and Settings
   - Output Examples
   - Results
Motivation and Goals

- Student heterogeneity → diff. RC levels (Pearson and Gallagher, 1983)
- 12% of class-time dedicated to RC activities (Durkin, 1978)
- Learning at student’s pace
- Encourage pre-engagement with material to support learning (Marzano, 2007)
- Easy frequent reading-check
## Typical Workflow

1. **Text analysis**

2. **Term extraction** (e.g. \([AN]+N \text{ and } [AN]^*\text{NP}[AN]^*N\) (Mitkov et al., 2003))

3. **Distractor selection** (e.g. ontologies (Al-Yahya, 2014), POS-based (Coniam, 2013))

4. **Instruction formulation**

5. **[optional] Filtering and/or ranking** (e.g. feature-based (Becker et al., 2012), categorical (Smith, 2010))
Usages

Type of an ITS — so-called "technology-enhanced learning environment" or "A computer-aided environment"
Usages

Type of an ITS —
so-called "technology-enhanced learning environment" or "A computer-aided environment"

Generating Questions to support learning, for example:

- Physics exercises (Graesser et al., 2004)
- Grammar training (Chen et al., 2006)
- Reading comprehension (Mitkov et al., 2003 & 2006)
- Academic writing support (Liu et al., 2012)
- ...
Outline

1 Introduction

2 Background
   - Reading Comprehension
   - Automated Question Generation

3 ARCQC
   - Overview
   - Text Analysis
   - Task Generation
   - Overgeneration and Ranking
   - Generalization and Customization

4 Evaluation
   - Method and Settings
   - Output Examples
   - Results
Main Purposes

Automatic Re-Call Questions Creator

- Verification of basic knowledge
- Save regular quizzes creation time
- More expressivity to human teachers’ expertise
- Self RC evaluation
- …
Main Principles

1. Customizability
   - Capture different text styles, change focus
   - Feasible and accessible to end-user
   - Realized by external XML-based patterns
Main Principles

1 Customizability
   ▶ Capture different text styles, change focus
   ▶ Feasible and accessible to end-user
   ▶ Realized by external XML-based patterns

2 Independency
   ▶ Based only on given text
   ▶ Consider syntactic/semantical context
   ▶ Realized (mainly) by the distractor selection module
Main Principles

1 Customizability
   - Capture different text styles, change focus
   - Feasible and accessible to end-user
   - Realized by external XML-based patterns

2 Independency
   - Based only on given text
   - Consider syntactic/semantical context
   - Realized (mainly) by the distractor selection module

_confer_

Mitkov and Ha (2003) — hardcoded templates; external resources
Architecture
Outline

1 Introduction

2 Background
- Reading Comprehension
- Automated Question Generation

3 ARCQC
- Overview
- Text Analysis
- Task Generation
- Overgeneration and Ranking
- Generalization and Customization

4 Evaluation
- Method and Settings
- Output Examples
- Results
Analyses Performed

- Basic normalization
- Sentence detection
- Chunking
- NE Recognition
- Co-referencing
- …
Analyses NOT Performed

- Sentence splitting (e.g. Heilman and Smith, 2010)
- Apposition elision (Kalady, 2010)
- Acronym expansion
- Direct speech constructions merging/re-formulation
- Grammatical functions (Culy and Raveh, 2014 (internal))
- ...
Outline

1 Introduction

2 Background
   - Reading Comprehension
   - Automated Question Generation

3 ARCQC
   - Overview
   - Text Analysis
   - Task Generation
   - Overgeneration and Ranking
   - Generalization and Customization

4 Evaluation
   - Method and Settings
   - Output Examples
   - Results
**Target Detection**

- Sentence level
Target Detection

- Sentence level
- Compared against each pattern

Algorithm 1 Target Detection

1: procedure FIND SUITABLE TARGETS(Text T, XMLPatterns M)
2:   for sentence $s \in T$ do
3:     for pattern $p \in M$ do
4:       if $s$ matches $p$ then
5:         Generate task for $s$ based on $p$
6:       end if
7:     end for
8:   end for
9: end procedure
**Target Detection**

- Sentence level *(whole sentences)*
- Compared against **each** pattern

```
Algorithm 1 Target Detection
1: procedure FIND SUITABLE TARGETS(Text T, XMLPatterns M)
2:    for sentence s ∈ T do
3:       for pattern p ∈ M do
4:         if s matches p then
5:             Generate task for s based on p
6:       end if
7:    end for
8:  end for
9: end procedure
```
**Target Detection**

- Sentence level (*whole sentences*)
- Compared against each pattern

```plaintext
Algorithm 1 Target Detection
1: procedure FIND SUITABLE TARGETS(Text T, XMLPatterns M)
2:    for sentence s ∈ T do
3:       for pattern p ∈ M do
4:          if s matches p then
5:             Generate task for s based on p
6:          end if
7:       end for
8:    end for
9: end procedure
```

- Single task per sentence per pattern
Term Extraction

- Get term (correct answer) according to pattern
- Term can be anything defined in pattern
- All terms are detected, only *important* ones are used
  - NE are automatically considered important
  - "Used" does not necessarily mean shown to the user
Term Extraction

- Get term (correct answer) according to pattern
- Term can be anything defined in pattern
- All terms are detected, only important ones are used
  - NE are automatically considered important
  - "Used" does not necessarily mean shown to the user

Done using a 3-phased TF-IDF measurement

1. Static IDF (corpus-based)
2. Dynamic TF (document-based)
3. TF-IDF calculation (variation on Rajaraman et al., 2012)
Term Extraction (II)

1 Static IDF

- Based on PukWaC corpus (Baroni et al., 2009; Johansson, 2008))
- Threshold-ed \(\log(IDF) < 13 \Rightarrow\) using only 10.46% of types
- Tokens as-is (not lemmatized, not stemmed, not normalized)

\[
IDF(t, D) = \log \left( \frac{N}{1 + \left( \sum_{n=1}^{N} [t \in d_n] \right)} \right)
\]
Term Extraction (II)

1 Static IDF
- Based on PukWaC corpus (Baroni et al., 2009; Johansson, 2008))
- Threshold-ed ($\log(IDF) < 13 \Rightarrow$ using only 10.46% of types)
- Tokens as-is (not lemmatized, not stemmed, not normalized)

$$IDF(t, D) = \log \left( \frac{N}{1 + \left( \sum_{n=1}^{N} [t \in d_n] \right)} \right)$$

2 Dynamic TF
- Based on given text
- Rewarding terms without IDF score

$$TF(t, d) = \frac{\sum_{n=1}^{W} [w_n \in d = t]}{W} + B [IDF(t, D) = \log N]$$
**Term Extraction (II)**

1. **Static IDF**
   - Based on PukWaC corpus (Baroni et al., 2009; Johansson, 2008))
   - Threshold-ed ($\log(IDF) < 13 \Rightarrow$ using only 10.46% of types)
   - Tokens as-is (not lemmatized, not stemmed, not normalized)

   $$IDF(t, D) = \log \left( \frac{N}{1 + \left( \sum_{n=1}^{N} [t \in d_n] \right)} \right)$$

2. **Dynamic TF**
   - Based on given text
   - Rewarding terms without IDF score

   $$TF(t, d) = \frac{\sum_{n=1}^{W} [w_n \in d = t]}{W} + B \left[ IDF(t, D) = \log N \right]$$

3. **TF-IDF Score**
   - Multiple-word terms are averaged

   $$TFIDF = (1 + TF(term)) \cdot IDF(term)$$
Term Extraction (III)

Testing with Wikipedia entry for "Reading Comprehension"

<table>
<thead>
<tr>
<th>Manzo</th>
<th>self-correcting</th>
<th>people</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>sentence-level</td>
<td>Craik</td>
<td>work</td>
<td>development</td>
</tr>
<tr>
<td>text</td>
<td>pragmatics</td>
<td>years</td>
<td>level</td>
</tr>
<tr>
<td>read-test</td>
<td>Comprehension</td>
<td>number</td>
<td>process</td>
</tr>
<tr>
<td>narrative-level</td>
<td>Questioning</td>
<td>area</td>
<td>research</td>
</tr>
</tbody>
</table>
**Term Extraction (IV)**

*Exhausting targets using co-references*

1. *Up until recently, I was living in Tübingen.*
2. *It is a very beautiful city.*

⇒ *too low TF-IDF score* We might lose a task for sentence (2), just because "It" would not have a high-enough TF-IDF score.
Term Extraction (IV)

Exhausting targets using co-references

(1) *Up until recently, I was living in Tübingen.*

(2) *It is a very beautiful city.*

⇒ too low TF-IDF score We might lose a task for sentence (2), just because "It" would not have a high-enough TF-IDF score.

(3) With co-referencing:

*What is Tübingen? (a beautiful city)*
**Term Extraction (IV)**

*Exhausting targets using co-references*

(1) *Up until recently, I was living in Tübingen.*

(2) *It is a very beautiful city.*

⇒ too low TF-IDF score We might lose a task for sentence (2), just because "It" would not have a high-enough TF-IDF score.

(3) With co-referencing:

*What is Tübingen? (a beautiful city)*

Uses *max TF-IDF score*(mention, representation)
### Term Extraction (V)

**Algorithm 2 Term Extraction**

```plaintext
1: procedure EXTRACT CORRECT ANSWER(Pattern P, Sentence S, Text T)
2:    answer ← extractTermTag(P)
3:    if not answer ∈ NET then
4:       answerCoref ← getCoref(answer) ▷ if no coref, assign term itself
5:       answer ← termOfMax(TFIDFanswer, TFIDFanswerCoref)
6:    end if
7:    if not TFIDFanswer is significant then
8:       return NULL ▷ don’t create task for this term
9:    end if
10:   return answer ▷ use answer as term of task
11: end procedure
```
Instruction Formulation

- Defined in *Produce* tag wrt to *Target* tags
  - Not only the term is a target
  - Not all targets must be used
  - Targets are processed *sequentially*
Instruction Formulation

- Defined in *Produce* tag wrt to *Target* tags
  - Not only the term is a target
  - Not all targets must be used
  - Targets are processed **sequentially**
- Starting point — either all or nothing
Instruction Formulation

- Defined in *Produce* tag wrt to *Target* tags
  - Not only the term is a target
  - Not all targets must be used
  - Targets are processed **sequentially**
- Starting point — either all or nothing
- Possible modifications — add-before, add-after and replace
Instruction Formulation

- Defined in *Produce* tag wrt to *Target* tags
  - Not only the term is a target
  - Not all targets must be used
  - Targets are processed **sequentially**

- Starting point — either all or nothing

- Possible modifications — add-before, add-after and replace

- Conditioning — e.g. for distinguishing between *What* and *Who* questions
**Distractor Selection**

- For MCQs, \( n - 1 \) distractors are required
- Only one (most) correct answer
- Should measure **preciseness** of knowledge (unlike OQ)
- Usually refers to a **single aspect** of the correct answer
- An answer is only correct wrt a **specific text**
Distractor Selection

- For MCQs, \( n - 1 \) distractors are required
- Only one (most) correct answer
- Should measure preciseness of knowledge (unlike OQ)
- Usually refers to a single aspect of the correct answer
- An answer is only correct wrt a specific text

Therefore, distractors should be similar(?) but distinguishable from the answer
Distractor Selection (II)

In ARCQC

- Targets are context-depended (pattern-based)
Introduction
Background
ARCQC
Distractor Selection (II)
Evaluation
Summary

Task Generation

**Distractor Selection (II)**

In ARCQC

- Targets are context-dependent (pattern-based)
- Distractors are taken only from the text itself (self-containment)
  - prevent exclusion of unseen choices
  - verify distinction between terms (e.g. people)
  - learner is not confused by external contents
**Distractor Selection (II)**

In ARCQC

- Targets are context-dependent (pattern-based)
- Distractors are taken only from the text itself (self-containment)
  - prevent exclusion of a unseen choices
  - verify distinction between terms (e.g. people)
  - learner is not confused by external contents

- 2-Phase selection
  1. Setting criteria
     - Define the criteria (properties) of the answer
     - ! Prefer specific criteria over general ones (e.g. in "New York" — NE > NP)
     - Terms with similar Syntactic environment are more likely to be semantically similar
Distractor Selection (II)

In ARCQC

- Targets are context-depended (pattern-based)
- Distractors are taken only from the text itself (self-containment)
  - prevent exclusion of a unseen choices
  - verify distinction between terms (e.g. people)
  - learner is not confused by external contents

2-Phase selection

1. Setting criteria
   - Define the criteria (properties) of the answer
   - ! Prefer specific criteria over general ones (e.g. in "New York" — NE > NP)
   - Terms with similar Syntactic environment are more likely to be semantically similar

2. Collecting distractors
   - Query the preprocessing module for terms with the same criteria
   - Filter terms through TF-IDF threshold
   - ! Check corefs of terms as well
   - Choose random $n-1$ from remaining terms
Outline

1 Introduction
2 Background
   - Reading Comprehension
   - Automated Question Generation
3 ARCQC
   - Overview
   - Text Analysis
   - Task Generation
   - Overgeneration and Ranking
   - Generalization and Customization
4 Evaluation
   - Method and Settings
   - Output Examples
   - Results
Overgeneration — Only x out of all generated tasks are eventually shown to the user (e.g. Heilman, 2011))

Ranking — Better (?) tasks are preferred (?)

Computers’ robustness preciseness ⇔ humans’ understanding

Allows humans use their abilities without being afraid of time waste.

Let each party do what it’s good at...
Ranking Methods in AQG

- Categorical (Smith, 2010)
- Linguistic-based (Becker et al., 2012)
- Length-based (Heilman and Smith, 2010 (Microsoft corpus))
- ...
**Ranking in ARCQC**

- Only **valid** tasks are ranked (not categorical)
- Doesn’t **filter**, only **prefers** (scaled, except very long tasks)
- Tries to refer to most **important terms** (using TF-IDF scores)
- Assumes that **shorter** tasks are **easier to read** (Vajjala and Meurers, 2012)
- Number of tasks in quiz is configurable by user
- May be be ignored (configurable)
Overgeneration and Ranking

## Ranking in ARCQC

- Only **valid** tasks are ranked (not categorical)
- Doesn’t **filter**, only **prefers** (scaled, except very long tasks)
- Tries to refer to most **important terms** (using TF-IDF scores)
- Assumes that **shorter** tasks are **easier to read** (Vajjala and Meurers, 2012)
- Number of tasks in quiz is configurable by user
- May be be ignored (configurable)

\[
Rank(task) = \log \left( \frac{1}{c} \sum_{n=1}^{c} TFIDF(choice_n) \cdot length \right)
\]
Ranking in ARCQC

- Only **valid** tasks are ranked (not categorical)
- Doesn’t **filter**, only **prefers** (scaled, except very long tasks)
- Tries to refer to most **important terms** (using TF-IDF scores)
- Assumes that **shorter** tasks are **easier to read** (Vajjala and Meurers, 2012)
- Number of tasks in quiz is configurable by user
- May be be ignored (configurable)

\[
\text{Rank}(\text{task}) = \log \left( \frac{1}{c} \sum_{n=1}^{c} \text{TFIDF}(\text{choice}_n) \cdot \text{length} \right)
\]

Re-Useability — each run will generate a different quiz!
Ranking in ARCQC II

Algorithm 3 Overgeneration and Ranking

1: procedure CHOOSE TASKS(Quiz Q, TargetScale S)
2:     if probabilistic then
3:         for task \( t \in Q \) do
4:             calculate(RawRankingScore\(_t\))
5:         end for
6:     r ← Calculate(RawScale(q))
7:     Rescale(r, S) ▷ S is by default 0-100
8:     for task \( t \in Q \) do
9:         calculate(ScaledRankingScore\(_t\))
10:    end for
11:   while targetnumberoftasksnotreached do
12:       \( n \leftarrow Random(S_{\text{min}}, S_{\text{max}}) \)
13:       \( t' \leftarrow GetRandomTask(Q) \)
14:       if rank\(_t \geq n \) then
15:           Choose(t')
16:       end if
17:   end while
18: else
19:   while targetnumberoftasksnotreached do
20:     \( t' \leftarrow GetRandomTask(Q) \)
21:     Choose(t')
22: end while
23: end if
24: end procedure
Outline

1 Introduction

2 Background
   - Reading Comprehension
   - Automated Question Generation

3 ARCQC
   - Overview
   - Text Analysis
   - Task Generation
   - Overgeneration and Ranking
   - Generalization and Customization

4 Evaluation
   - Method and Settings
   - Output Examples
   - Results
Using patterns

- Capturing semantics by syntactic means
Using patterns

- Capturing semantics by syntactic means
- Focus for a specific field
- Function as an interface
- Accessible and easy to modify
- Language independent (tested, not implemented)
Using patterns

- Capturing semantics by syntactic means
- Focus for a specific field
- Function as an interface
- Accessible and easy to modify
- Language independent (tested, not implemented)
Supported Tags

Patterns
Metadata
Pattern [name]
Properties
Description
Types
Schema
Detect

Target [number]
Entity [type]
Chunk [phrase, position, text]
Produce [base]
Answer [target]
Modification [action, condition]
Opening
Closure
A Complete Pattern

1. `<Pattern name="What is something?">`
2. `<Properties>`
3. `<Description>Captures sentences of the form "X is Y"</Description>`
4. `<Types>MCQ</Types>`
5. `</Properties>`
6. `<Schema>`
7. `<Detect>`
8. `<Chunks>`
9. `<Chunk phrase="NP+V+NP[NP|PP]*)"/>
10. `<Chunk phrase="VP position="2" text="is|are|was|were"/>
11. `<Target number="1">`
12. `<Chunk phrase="NP position="1"/>
13. `</Target>`
14. `</Chunks>`
15. `</Detect>`
16. `<Produce base="all">`
17. `<Answer target="1"/>
18. `<Modification action="replace" target="1">What</Modification>`
19. `<Modification condition="person" action="replace" target="1">Who</Modification>`
20. `<Opening>(Choose the correct answer): n</Opening>`
21. `<Closure>?</Closure>`
22. `</Schema>`
23. `</Pattern>`
A Complete Pattern

1 <Pattern name="What something is"> 
2  <Properties> 
3   <Description>Captures sentences of the form "X is Y"</Description> 
4  <Types>MCQ</Types> 
5 </Properties> 
6 <Schema> 
7   <Detect> 
8      <Chunks> 
9         <Chunk phrase="NP\(\text{NP}\)\(\text{NP}\)\|PP\)*"> 
10        <Chunk phrase="VP position="2" text="is\|are\|was\|were" /> 
11   </Target number="1"> 
12      <Chunk phrase="NP" position="1"/> 
13 </Target> 
14 </Chunks> 
15 </Detect> 
16 <Produce base="all"> 
17   <Answer target="1"/> 
18   <Modification action="replace" target="1">What</Modification> 
19   <Modification condition="person" action="replace" target="1">Who</Modification> 
20   <Opening>(Choose the correct answer)\n</Opening> 
21   <Closure>?</Closure> 
22 </Schema> 
23 </Pattern>

S: New York is a state in the north of the united states. 
⇒ Q: What is a state in the north of the united states? (New York)

S: Mozart was a famous composer in the 18th century. 
⇒ Q: Who was a famous composer in the 18th century? (Mozart)
More Involved Structure

What is a state in the United States?
More Involved Structure

⇒ What is a state in [...] the United States?
Another example

Using "because" for capturing cause and result
Another example

Using "because" for capturing cause and result

- John was fired because of Jim.
Another example

Using "because" for capturing cause and result

- John was fired because of Jim.
Another example

Using "because" for capturing cause and result

- John was fired because of Jim.
  ⇒ Because of whom was John fired? (Jim)
Another example

Using "because" for capturing cause and result

- John was fired because of Jim.
  ⇒ Because of whom was John fired? (Jim)

- John was fired, because his boss didn’t like his tie.
Another example

Using "because" for capturing cause and result

- John was fired because of Jim.
  ⇒ *Because of whom was John fired?* (Jim)

- John was fired, because his boss didn’t like his tie.
Another example

Using "because" for capturing cause and result

- John was fired because of Jim.
  ⇒ Because of whom was John fired? (Jim)

- John was fired, because his boss didn’t like his tie.
  ⇒ Why was John fired? (because his boss didn’t like his tie)
Another example

Using "because" for capturing cause and result

- John was fired because of Jim.
  ⇒ *Because of whom was John fired?* (Jim)

- John was fired, because his boss didn’t like his tie.
  ⇒ *Why was John fired?* (because his boss didn’t like his tie)

- John’s boss didn’t like his tie. Because of that, he was fired.
Another example

Using "because" for capturing cause and result

- John was fired because of Jim.
  ⇒ *Because of whom was John fired?* (Jim)

- John was fired, because his boss didn’t like his tie.
  ⇒ *Why was John fired?* (because his boss didn’t like his tie)

- John’s boss didn’t like his tie. *Because of that, he* was fired.
Another example

Using "because" for capturing cause and result

- John was fired because of Jim.
  ⇒ *Because of whom was John fired?* (Jim)

- John was fired, because his boss didn’t like his tie.
  ⇒ *Why was John fired?* (because his boss didn’t like his tie)

- John’s boss didn’t like his tie. *Because of that, he* was fired.
  ⇛ *Why was John fired?* (because his boss didn’t like his tie)
Another example

Using "because" for capturing cause and result

- John was fired because of Jim.
  ⇒ *Because of whom was John fired? (Jim)*

- John was fired, because his boss didn’t like his tie.
  ⇒ *Why was John fired? (because his boss didn’t like his tie)*

- John’s boss didn’t like his tie. Because of that, he was fired.
  ⇓ *Why was John fired? (because his boss didn’t like his tie)*
Outline

1. Introduction

2. Background
   - Reading Comprehension
   - Automated Question Generation

3. ARCQC
   - Overview
   - Text Analysis
   - Task Generation
   - Overgeneration and Ranking
   - Generalization and Customization

4. Evaluation
   - Method and Settings
   - Output Examples
   - Results
Criteria and Corpus

- Scoring Criteria
  - Validness (grammaticality + relevancy + semantic/syntactic integrity)
  - Number of tasks per text

Method and Settings

Text Resources (5 texts from each)
- Reuters News (via WERTi)
- Politics
- Sports
- Business
- Technology
- Wikipedia
  - English Wikipedia
  - Simple English Wikipedia
  - 5 · 6 = 30 texts in total
Criteria and Corpus

- Scoring Criteria
  - Validness (grammaticality + relevancy + semantic/syntactic integrity)
  - Number of tasks per text

- Text Resources (5 texts from each)
  - Reuters News (via WERTi)
    - Politics
    - Sports
    - Business
    - Technology
  - Wikipedia
    - English Wikipedia
    - Simple English Wikipedia

5 · 6 = 30 texts in total
# Test Patterns

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Sentence</th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>NV</td>
<td>Chris drank an espresso</td>
<td>Who Drank an espresso?</td>
<td>Chris</td>
</tr>
<tr>
<td>MD</td>
<td>Mozart was a composer</td>
<td>Who was a composer?</td>
<td>Mozart</td>
</tr>
<tr>
<td>HD</td>
<td></td>
<td>What was Mozart?</td>
<td>a composer</td>
</tr>
<tr>
<td>BW</td>
<td>Kennedy was killed by an assassin</td>
<td>By what Kennedy was killed?</td>
<td>an assassin</td>
</tr>
<tr>
<td>Time</td>
<td>Einstein received the Nobel Prize in 1921</td>
<td>Einstein received the Nobel Prize in ___</td>
<td>1921</td>
</tr>
<tr>
<td>Num</td>
<td>There are 50 states in the USA</td>
<td>There are ___ states in the USA</td>
<td>50</td>
</tr>
<tr>
<td>Loc</td>
<td>Sherlock Holmes works in London</td>
<td>Sherlock Holmes works in ___</td>
<td>London</td>
</tr>
</tbody>
</table>
Outline

1 Introduction

2 Background
   - Reading Comprehension
   - Automated Question Generation

3 ARCQC
   - Overview
   - Text Analysis
   - Task Generation
   - Overgeneration and Ranking
   - Generalization and Customization

4 Evaluation
   - Method and Settings
   - Output Examples
   - Results
**Good outputs**

Sentence: "He received the Nobel Prize in Physics in 1921."

Who received the Nobel Prize in Physics in 1921? (Albert Einstein)

1. Albert Einstein
2. Elsa Lowenthal
3. Ehud Olmert
4. Mileva
Good outputs

Sentence: "He received the Nobel Prize in Physics in 1921."

1. Who received the Nobel Prize in Physics in 1921? (Albert Einstein)
   - 1. Albert Einstein
   - 2. Elsa Lowenthal
   - 3. Ehud Olmert
   - 4. Mileva

2. Albert Einstein received the Nobel Prize in Physics in _____. (1921)
   - 1. 1986
   - 2. 1919
   - 3. 1921
   - 4. 1940
Good outputs

Sentence: "He received the Nobel Prize in Physics in 1921."

1 Who received the Nobel Prize in Physics in 1921? (Albert Einstein)
   1 Albert Einstein
   2 Elsa Lowenthal
   3 Ehud Olmert
   4 Mileva

2 Albert Einstein received the Nobel Prize in Physics in _____. (1921)
   1 1986
   2 1919
   3 1921
   4 1940

Sentence: "Einstein developed the general theory of relativity?"

3 Who developed the general theory of relativity? (Albert Einstein)
**OK outputs**

1. Conventionally, a computer consists of at least ____ processing element. *(one)*
   - 1. 42
   - 2. one
   - 3. hundred
   - 4. 7
**OK outputs**

1. Conventionally, a computer consists of at least ____ processing element. (one)
   - 1. 42
   - 2. one ← only choice with correct agreement
   - 3. hundred
   - 4. 7
### OK outputs

1. Conventionally, a computer consists of at least ____ processing element.  
   (one)
   - 1. 42
   - 2. one ← only choice with correct agreement
   - 3. hundred
   - 4. 7

2. When Albert Einstein became older, he went to a school in ____.
   (Switzerland)
   - 1. Switzerland
   - 2. Princeton
   - 3. New Jersey
   - 4. Zurich
**OK outputs**

1. Conventionally, a computer consists of at least ____ processing element. (one)
   1. 42
   2. one ← only choice with correct agreement
   3. hundred
   4. 7

2. When Albert Einstein became older, he went to a school in ____.
   (Switzerland)
   1. Switzerland
   2. Princeton
   3. New Jersey
   4. Zurich ← more correct, but not in text
Bad outputs

1 In year ____
## Bad outputs

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In year ____ ← overgeneration</td>
</tr>
</tbody>
</table>
Bad outputs

1. In year ____ ← overgeneration

2. Who think it’s too late, said Steve Elmendorf, a Clinton, whose support for LGBT issues went back years before same-sex marriage was a hot political topic fan and former chair of the Victory Fund, a group that provides financial and political support to gay candidates? (I)

   1. the gay community
   2. Clinton
   3. a primary race
   4. I
Bad outputs

1. In year ____ ← overgeneration

2. Who think it’s too late, said Steve Elmendorf, a Clinton, whose support for LGBT issues went back years before same-sex marriage was a hot political topic fan and former chair of the Victory Fund, a group that provides financial and political support to gay candidates? (I) ← too long

   1. the gay community
   2. Clinton
   3. a primary race
   4. I ← bad coref
Very bad output

Who taught physics at the Institute for Advanced Study at Princeton? (Ehud Olmert)

1. Einstein
2. Maja
3. Ehud Olmert
4. Elsa Lowenthal
Very bad output

1. Who taught physics at the Institute for Advanced Study at Princeton? (Ehud Olmert)
   - 1. Einstein ← here only by chance, should be correct
   - 2. Maja
   - 3. Ehud Olmert ← marked as correct due to bad coref
   - 4. Elsa Lowenthal
Outline

1 Introduction

2 Background
   - Reading Comprehension
   - Automated Question Generation

3 ARCQC
   - Overview
   - Text Analysis
   - Task Generation
   - Overgeneration and Ranking
   - Generalization and Customization

4 Evaluation
   - Method and Settings
   - Output Examples
   - Results
## Results

<table>
<thead>
<tr>
<th>Category</th>
<th>Avg. Word Count</th>
<th>Avg. Task Count</th>
<th>NV</th>
<th>MD</th>
<th>HD</th>
<th>BW</th>
<th>Time</th>
<th>Num</th>
<th>Loc</th>
<th>% Valid Tasks</th>
<th>% Valid Distractors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Politics</td>
<td>474.4</td>
<td>23</td>
<td>8.8</td>
<td>0.2</td>
<td>1.2</td>
<td>0</td>
<td>4.6</td>
<td>2.2</td>
<td>6</td>
<td>71%</td>
<td>80%</td>
</tr>
<tr>
<td>Sports</td>
<td>823.6</td>
<td>36.6</td>
<td>14.4</td>
<td>0.2</td>
<td>0.6</td>
<td>0.2</td>
<td>9.6</td>
<td>4.4</td>
<td>7.4</td>
<td>75%</td>
<td>75%</td>
</tr>
<tr>
<td>Business</td>
<td>400</td>
<td>26.8</td>
<td>10.8</td>
<td>0.2</td>
<td>1.4</td>
<td>0.8</td>
<td>4</td>
<td>4.8</td>
<td>4.8</td>
<td>72%</td>
<td>80%</td>
</tr>
<tr>
<td>Technology</td>
<td>517.2</td>
<td>27.8</td>
<td>10.6</td>
<td>0.4</td>
<td>0.6</td>
<td>0.2</td>
<td>6.8</td>
<td>4.2</td>
<td>5</td>
<td>77%</td>
<td>82%</td>
</tr>
<tr>
<td>Std. Wiki.</td>
<td>593.2</td>
<td>31.6</td>
<td>11.2</td>
<td>1.4</td>
<td>5.6</td>
<td>0</td>
<td>2</td>
<td>4.4</td>
<td>7</td>
<td>75%</td>
<td>76%</td>
</tr>
<tr>
<td>Sim. Wiki.</td>
<td>571.2</td>
<td>24.2</td>
<td>5.8</td>
<td>1</td>
<td>1.8</td>
<td>0</td>
<td>4.6</td>
<td>5</td>
<td>6</td>
<td>76%</td>
<td>71%</td>
</tr>
</tbody>
</table>
Results (II)
Results (III)

<table>
<thead>
<tr>
<th>Category</th>
<th>Tasks</th>
<th>Distractors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ungrammatical</td>
<td>23%</td>
<td>33%</td>
</tr>
<tr>
<td>Non-Sense</td>
<td>56%</td>
<td>41%</td>
</tr>
<tr>
<td>Wrong Target</td>
<td>18%</td>
<td>16%</td>
</tr>
<tr>
<td>Other</td>
<td>3%</td>
<td>10%</td>
</tr>
</tbody>
</table>
Outline

1 Introduction

2 Background
   - Reading Comprehension
   - Automated Question Generation

3 ARCQC
   - Overview
   - Text Analysis
   - Task Generation
   - Overgeneration and Ranking
   - Generalization and Customization

4 Evaluation
   - Method and Settings
   - Output Examples
   - Results
Recap

- AQG
  - Saves time, more frequent exercises
  - Own-pace, own-time self learning
  - Basic-knowledge staring point
  - Reading verification
Recap

- **AQG**
  - Saves time, more frequent exercises
  - Own-pace, own-time self learning
  - Basic-knowledge starting point
  - Reading verification

- **ARCQC — automatic recall questions creator**
  - Works with authentic texts
  - Customizable patterns for dynamic focus
  - Based on given text
  - IR-style term extraction
  - Re-usability
Recap

- AQG
  - Saves time, more frequent exercises
  - Own-pace, own-time self learning
  - Basic-knowledge staring point
  - Reading verification

- ARCQC — automatic recall questions creator
  - Works with authentic texts
  - Customizable patterns for dynamic focus
  - Based on given text
  - IR-style term extraction
  - Re-usability

- Much to improve!
- Other types of questions!
Future Work

- Semantic-relation distractors (self-contained)
- Grammatical functions extraction for common-feature tasks
- Sentence normalization (d. speech, appositions etc.)
- Vector-space vocabulary-based reading-check tasks
- GUI
- Answer-check mechanism
- …
Thank You!

questions? comments? suggestions?