Outline

1. Introduction
   - Question Answering
     - Challenges
     - Our approach

2. Question Classification

3. Passage Retrieval

4. Answer Extraction

5. Evaluation

6. Contributions
What is Question Answering?

Definition

*Question answering (QA) is the task of automatically answering a question posed in natural language.*

Example

- What is the capital of Portugal? *Lisbon.*
- Where is the Louvre Museum? *Paris.*
What is Question Answering?

**Definition**

*Question answering (QA) is the task of automatically answering a question posed in natural language.*

**Example**

- What is the capital of Portugal? *Lisbon.*
- Where is the Louvre Museum? *Paris.*
Outline

1. Introduction
   - Question Answering
   - Challenges
   - Our approach

2. Question Classification

3. Passage Retrieval

4. Answer Extraction

5. Evaluation

6. Contributions
Natural Language Challenges

- Coping with the vast variety of questions that can be asked
  - Many different formulations for the same question

Example

- What is Portugal’s capital city?
- What’s the name of the capital of Portugal?

- Passages that contain the answer to a given question can also be phrased in many different ways

Example

- Faro, 310 km from the capital of Portugal – **Lisbon**.
- **Lisbon** became the capital city of Portugal in 1255.
Natural Language Challenges

- Coping with the vast variety of questions that can be asked
- Many different formulations for the same question

Example

- What is Portugal’s capital city?
- What’s the name of the capital of Portugal?

- Passages that contain the answer to a given question can also be phrased in many different ways

Example

- Faro, 310 km from the capital of Portugal – Lisbon.
- Lisbon became the capital city of Portugal in 1255.
Natural Language Challenges

- Coping with the vast variety of questions that can be asked
- Many different formulations for the same question

Example

- What is Portugal’s capital city?
- What’s the name of the capital of Portugal?

- Passages that contain the answer to a given question can also be phrased in many different ways

Example

- Faro, 310 km from the capital of Portugal – Lisbon.
- Lisbon became the capital city of Portugal in 1255.
Natural Language Challenges

- Coping with the vast variety of questions that can be asked
- Many different formulations for the same question

Example

- What is Portugal’s capital city?
- What’s the name of the capital of Portugal?

- Passages that contain the answer to a given question can also be phrased in many different ways

Example

- Faro, 310 km from the capital of Portugal – Lisbon.
- Lisbon became the capital city of Portugal in 1255.
Natural Language Challenges

- Coping with the vast variety of questions that can be asked
- Many different formulations for the same question

Example

- What is Portugal’s capital city?
- What’s the name of the capital of Portugal?

- Passages that contain the answer to a given question can also be phrased in many different ways

Example

- Faro, 310 km from the capital of Portugal – Lisbon.
- Lisbon became the capital city of Portugal in 1255.
Web Challenges

- Separating the *wheat from the chaff*

**Example**

- Who is the current prime minister of Portugal?
  - *Salazar* became finance minister in 1928 and Prime Minister of Portugal in 1932.
  - *José Sócrates* has been the prime minister of Portugal since February 24th 2005.
  - I am glad to be able to welcome the Prime Minister of Portugal, Mr. *Sócrates*, to Moscow.
Web Challenges

- Separating the *wheat from the chaff*

**Example**

- Who is the current prime minister of Portugal?
  - *Salazar* became finance minister in 1928 and Prime Minister of Portugal in 1932.
  - *José Sócrates* has been the prime minister of Portugal since February 24th 2005.
  - I am glad to be able to welcome the Prime Minister of Portugal, Mr. Sócrates, to Moscow.
Web Challenges

- Separating the *wheat from the chaff*

**Example**

- Who is the current prime minister of Portugal?
  - **Salazar** became finance minister in 1928 and Prime Minister of Portugal in 1932.
  - On the night of November 10, 2004, President **Hu Jintao** met with Portuguese Prime Minister **Pedro Santana Lopes** in the northern city of Porto.
  - **José Sócrates** has been the prime minister of Portugal since February 24th 2005.
  - I am glad to be able to welcome the Prime Minister of Portugal, Mr. Sócrates, to Moscow.
Web Challenges

- Separating the *wheat from the chaff*

**Example**

- Who is the current prime minister of Portugal?
  - Salazar became finance minister in 1928 and Prime Minister of Portugal in 1932.
  - José Sócrates has been the prime minister of Portugal since February 24th 2005.
  - I am glad to be able to welcome the Prime Minister of Portugal, Mr. Sócrates, to Moscow.
Web Challenges

- Separating the *wheat from the chaff*

**Example**

- **Who is the current prime minister of Portugal?**
  - *Salazar* became finance minister in 1928 and Prime Minister of Portugal in 1932.
  - *José Sócrates* has been the prime minister of Portugal since February 24th 2005.
  - I am glad to be able to welcome the Prime Minister of Portugal, Mr. Sócrates, to Moscow.
Web Challenges

- Separating the *wheat from the chaff*

**Example**

- **Who is the current prime minister of Portugal?**
  - **Salazar** became finance minister in 1928 and Prime Minister of Portugal in 1932.
  - On the night of November 10, 2004, President **Hu Jintao** met with Portuguese Prime Minister **Pedro Santana Lopes** in the northern city of Porto.
  - **José Sócrates** has been the prime minister of Portugal since February 24th 2005.
  - I am glad to be able to welcome the Prime Minister of Portugal, Mr. **Sócrates**, to Moscow.
Outline

1. Introduction
   - Question Answering
   - Challenges
   - Our approach

2. Question Classification

3. Passage Retrieval

4. Answer Extraction

5. Evaluation

6. Contributions
Our approach

A multi-pronged approach to web question answering

- Strong focus on machine learning techniques that allow the system to **learn** rules instead of having a human expert handcrafting them
- Leverages the redundancy of the Web to accurately answer questions
Our approach

A multi-pronged approach to web question answering

- Strong focus on machine learning techniques that allow the system to learn rules instead of having a human expert handcrafting them
- Leverages the redundancy of the Web to accurately answer questions
Our approach

A multi-pronged approach to web question answering

- Strong focus on machine learning techniques that allow the system to *learn* rules instead of having a human expert handcrafting them
- Leverages the redundancy of the Web to accurately answer questions
Outline

1. Introduction

2. Question Classification
   - Introduction
   - Question Type Taxonomy
   - Question Feature Set
   - Experimental Setup
   - Experimental Results

3. Passage Retrieval

4. Answer Extraction

5. Evaluation
The goal of question classification is to assign a semantic category to a question, which represents the type of answer sought after.

**Example**

- What country is home to Heineken beer? ⇒ **LOCATION:COUNTRY**
- Who invented the telephone? ⇒ **HUMAN:INDIVIDUAL**
- When was Mozart born? ⇒ **NUMERIC:DATE**

The set of question categories into which the questions are to be assigned is referred to as the *question type taxonomy*. 
The goal of question classification is to assign a semantic category to a question, which represents the type of answer sought after

Example

- What country is home to Heineken beer? ⇒ LOCATION:COUNTRY
- Who invented the telephone? ⇒ HUMAN:INDIVIDUAL
- When was Mozart born? ⇒ NUMERIC:DATE

The set of question categories into which the questions are to be assigned is referred to as the question type taxonomy.
The goal of question classification is to assign a semantic category to a question, which represents the type of answer sought after.

Example

- What country is home to Heineken beer? ⇒ LOCATION: COUNTRY
- Who invented the telephone? ⇒ HUMAN: INDIVIDUAL
- When was Mozart born? ⇒ NUMERIC: DATE

The set of question categories into which the questions are to be assigned is referred to as the question type taxonomy.
The goal of question classification is to assign a semantic category to a question, which represents the type of answer sought after.

**Example**

- What country is home to Heineken beer? ⇒ **LOCATION:COUNTRY**
- Who invented the telephone? ⇒ **HUMAN:INDIVIDUAL**
- When was Mozart born? ⇒ **NUMERIC:DATE**

The set of question categories into which the questions are to be assigned is referred to as the *question type taxonomy*. 
The goal of question classification is to assign a semantic category to a question, which represents the type of answer sought after.

**Example**

- What country is home to Heineken beer? ⇒ **Location:Country**
- Who invented the telephone? ⇒ **Human:Individual**
- When was Mozart born? ⇒ **Numeric:Date**

The set of question categories into which the questions are to be assigned is referred to as the *question type taxonomy*. 
The goal of question classification is to assign a semantic category to a question, which represents the type of answer sought after.

**Example**

- What country is home to Heineken beer? ⇒ **LOCATION:COUNTRY**
- Who invented the telephone? ⇒ **HUMAN:INDIVIDUAL**
- When was Mozart born? ⇒ **NUMERIC:DATE**

The set of question categories into which the questions are to be assigned is referred to as the *question type taxonomy*.
The goal of question classification is to assign a semantic category to a question, which represents the type of answer sought after.

**Example**

- What country is home to Heineken beer? \(\Rightarrow\) **LOCATION: COUNTRY**
- Who invented the telephone? \(\Rightarrow\) **HUMAN: INDIVIDUAL**
- When was Mozart born? \(\Rightarrow\) **NUMERIC: DATE**

The set of question categories into which the questions are to be assigned is referred to as the *question type taxonomy*. 
The goal of question classification is to assign a semantic category to a question, which represents the type of answer sought after.

**Example**
- What country is home to Heineken beer? ⇒ **LOCATION:COUNTRY**
- Who invented the telephone? ⇒ **HUMAN:INDIVIDUAL**
- When was Mozart born? ⇒ **NUMERIC:DATE**

The set of question categories into which the questions are to be assigned is referred to as the *question type taxonomy*. 
Why is question classification important?

The question category can be used for two different purposes

1. Depending on the question category, different processing strategies can be chosen to find an answer

Example

For **DESCRIPTION:DEFINITION** questions, the system can search for possible answers in encyclopaedic sources – e.g., Wikipedia –, which are more likely to contain an accurate answer.

2. Narrow down the number of possible answer candidates that the system needs to consider

Example

Knowing that a question belongs to the category **LOCATION:CITY**, allows the system to only consider cities as possible answers.
Why is question classification important?

The question category can be used for two different purposes:

1. Depending on the question category, different processing strategies can be chosen to find an answer.

Example:

For **DESCRIPTION:DEFINITION** questions, the system can search for possible answers in encyclopaedic sources – e.g., Wikipedia –, which are more likely to contain an accurate answer.

2. Narrow down the number of possible answer candidates that the system needs to consider.

Example:

Knowing that a question belongs to the category **LOCATION:_CITY**, allows the system to only consider cities as possible answers.
Why is question classification important?

The question category can be used for two different purposes

1. Depending on the question category, different processing strategies can be chosen to find an answer

Example

For Description:Definition questions, the system can search for possible answers in encyclopaedic sources – e.g., Wikipedia –, which are more likely to contain an accurate answer.

2. Narrow down the number of possible answer candidates that the system needs to consider

Example

Knowing that a question belongs to the category Location:City, allows the system to only consider cities as possible answers.
Why is question classification important?

The question category can be used for two different purposes

1. Depending on the question category, different processing strategies can be chosen to find an answer

Example

For **Description:Definition** questions, the system can search for possible answers in encyclopaedic sources – e.g., Wikipedia –, which are more likely to contain an accurate answer.

2. Narrow down the number of possible answer candidates that the system needs to consider

Example

Knowing that a question belongs to the category **Location:City**, allows the system to only consider cities as possible answers.
Why is question classification important?

The question category can be used for two different purposes

1. Depending on the question category, different processing strategies can be chosen to find an answer

Example

For Description:Definition questions, the system can search for possible answers in encyclopaedic sources – e.g., Wikipedia –, which are more likely to contain an accurate answer.

2. Narrow down the number of possible answer candidates that the system needs to consider

Example

Knowing that a question belongs to the category Location:City, allows the system to only consider cities as possible answers.
Why is question classification important?

The question category can be used for two different purposes

1. Depending on the question category, different processing strategies can be chosen to find an answer

Example

For **DESCRIPTION:DEFINITION** questions, the system can search for possible answers in encyclopaedic sources – e.g., Wikipedia –, which are more likely to contain an accurate answer.

2. Narrow down the number of possible answer candidates that the system needs to consider

Example

Knowing that a question belongs to the category **LOCATION:CITY**, allows the system to only consider cities as possible answers.
Our approach

- We have adopted a machine learning approach to question classification
- Question classification modeled as a supervised learning classification problem
- Learning models were trained using a rich set of lexical, syntactic, and semantic features
Our approach

- We have adopted a machine learning approach to question classification.
- Question classification modeled as a supervised learning classification problem.
- Learning models were trained using a rich set of lexical, syntactic, and semantic features.
Our approach

- We have adopted a machine learning approach to question classification
- Question classification modeled as a supervised learning classification problem
- Learning models were trained using a rich set of lexical, syntactic, and semantic features
Outline

1. Introduction

2. Question Classification
   - Introduction
   - Question Type Taxonomy
   - Question Feature Set
   - Experimental Setup
   - Experimental Results

3. Passage Retrieval

4. Answer Extraction

5. Evaluation
Table: Li & Roth’s two-layer question type taxonomy.

<table>
<thead>
<tr>
<th>Coarse</th>
<th>Fine</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABBREVIATION</td>
<td>abbreviation, expansion</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>definition, description, manner, reason</td>
</tr>
<tr>
<td>ENTITY</td>
<td>animal, body, color, creative, currency, medical disease, event, food, instrument, language, letter, other, plant, product, religion, sport, substance, symbol, technique, term, vehicle, word</td>
</tr>
<tr>
<td>HUMAN</td>
<td>description, group, individual, title</td>
</tr>
<tr>
<td>LOCATION</td>
<td>city, country, mountain, other, state</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>code, count, date, distance, money, order, other, percent, period, speed, temperature, size, weight</td>
</tr>
</tbody>
</table>
Outline

1 Introduction

2 Question Classification
   - Introduction
   - Question Type Taxonomy
   - Question Feature Set
   - Experimental Setup
   - Experimental Results

3 Passage Retrieval

4 Answer Extraction

5 Evaluation
Word level $n$-grams

- A word level $n$-gram is a sequence of $n$ consecutive words from a given question.
- We have experimented $n$-grams with $n = \{1, 2, 3\}$, which are hereafter referred to as unigrams, bigrams, and trigrams, respectively.

Example

When was Kennedy born?

- **Unigrams**: Where, was, Kennedy, born, ?
- **Bigrams**: Where was, was Kennedy, Kennedy born, born ?
- **Trigrams**: Where was Kennedy, was Kennedy born, Kennedy born ?
Word level $n$-grams

- A word level $n$-gram is a sequence of $n$ consecutive words from a given question.
- We have experimented $n$-grams with $n = \{1, 2, 3\}$, which are hereafter referred to as unigrams, bigrams, and trigrams, respectively.

**Example**

When was Kennedy born?

- **Unigrams**: Where, was, Kennedy, born, ?
- **Bigrams**: Where was, was Kennedy, Kennedy born, born ?
- **Trigrams**: Where was Kennedy, was Kennedy born, Kennedy born ?
Word level $n$-grams

- A word level $n$-gram is a sequence of $n$ consecutive words from a given question.
- We have experimented $n$-grams with $n = \{1, 2, 3\}$, which are hereafter referred to as unigrams, bigrams, and trigrams, respectively.

**Example**

When was Kennedy born?

- **Unigrams**: Where, was, Kennedy, born, ?
- **Bigrams**: Where was, was Kennedy, Kennedy born, born ?
- **Trigrams**: Where was Kennedy, was Kennedy born, Kennedy born ?
A word level \( n \)-gram is a sequence of \( n \) consecutive words from a given question.

We have experimented \( n \)-grams with \( n = \{1, 2, 3\} \), which are hereafter referred to as unigrams, bigrams, and trigrams, respectively.

**Example**

When was Kennedy born?

- **Unigrams**: Where, was, Kennedy, born, ?
- **Bigrams**: Where was, was Kennedy, Kennedy born, born ?
- **Trigrams**: Where was Kennedy, was Kennedy born, Kennedy born ?
A word level \( n \)-gram is a sequence of \( n \) consecutive words from a given question.

We have experimented \( n \)-grams with \( n = \{1, 2, 3\} \), which are hereafter referred to as unigrams, bigrams, and trigrams, respectively.

**Example**

When was Kennedy born?

- **Unigrams**: Where, was, Kennedy, born, ?
- **Bigrams**: Where was, was Kennedy, Kennedy born, born ?
- **Trigrams**: Where was Kennedy, was Kennedy born, Kennedy born ?
A word level \( n \)-gram is a sequence of \( n \) consecutive words from a given question.

We have experimented \( n \)-grams with \( n = \{1, 2, 3\} \), which are hereafter referred to as unigrams, bigrams, and trigrams, respectively.

**Example**

When was Kennedy born?

- **Unigrams**: Where, was, Kennedy, born, ?
- **Bigrams**: Where was, was Kennedy, Kennedy born, born ?
- **Trigrams**: Where was Kennedy, was Kennedy born, Kennedy born ?
The question headword is a word in a given question that represents the information that is being sought after.

Example

- What is Australia’s national flower? ⇒ Entity:Plant
- Name an American made motorcycle. ⇒ Entity:Vehicle
- What Japanese city was once called Edo? ⇒ Location:City
- How tall is Prince Charles? ⇒ Numeric:Distance

As we shall demonstrate, this feature is of prominent importance to unveil a question’s true category.
The question headword is a word in a given question that represents the information that is being sought after.

**Example**

- What is Australia’s national **flower**? ⇒ **ENTITY:PLANT**
- Name an American made **motorcycle**. ⇒ **ENTITY:VEHICLE**
- What Japanese **city** was once called Edo? ⇒ **LOCATION:_CITY**
- How **tall** is Prince Charles? ⇒ **NUMERIC:DISTANCE**

As we shall demonstrate, this feature is of prominent importance to unveil a question’s true category.
Question headword

- The question headword is a word in a given question that represents the information that is being sought after.

Example

- What is Australia’s national **flower**? ⇒ **ENTITY:PLANT**
- Name an American made **motorcycle**. ⇒ **ENTITY:VEHICLE**
- What Japanese **city** was once called Edo? ⇒ **LOCATION:_CITY**
- How **tall** is Prince Charles? ⇒ **NUMERIC:DISTANCE**

As we shall demonstrate, this feature is of prominent importance to unveil a question’s true category.
Question headword

- The question headword is a word in a given question that *represents* the information that is being sought after.

**Example**

- What is Australia’s national **flower**? ⇒ **ENTITY:PLANT**
- Name an American made **motorcycle**. ⇒ **ENTITY:VEHICLE**
- What Japanese **city** was once called Edo? ⇒ **LOCATION:_CITY**
- How **tall** is Prince Charles? ⇒ **NUMERIC:DISTANCE**

- As we shall demonstrate, this feature is of prominent importance to unveil a question’s true category.
Question headword

- The question headword is a word in a given question that represents the information that is being sought after.

Example

- What is Australia’s national flower? ⇒ ENTITY:PLANT
- Name an American made motorcycle. ⇒ ENTITY:VEHICLE
- What Japanese city was once called Edo? ⇒ LOCATION:City
- How tall is Prince Charles? ⇒ NUMERIC:Distance

- As we shall demonstrate, this feature is of prominent importance to unveil a question’s true category.
The question headword is a word in a given question that represents the information that is being sought after.

**Example**

- What is Australia’s national **flower**? ⇒ **ENTITY:PLANT**
- Name an American made **motorcycle**. ⇒ **ENTITY:VEHICLE**
- What Japanese **city** was once called Edo? ⇒ **LOCATION:_CITY**
- How **tall** is Prince Charles? ⇒ **NUMERIC:DISTANCE**

As we shall demonstrate, this feature is of prominent importance to unveil a question’s true category.
The question headword is a word in a given question that represents the information that is being sought after.

Example

- What is Australia’s national flower? ⇒ Entity: Plant
- Name an American made motorcycle. ⇒ Entity: Vehicle
- What Japanese city was once called Edo? ⇒ Location: City
- How tall is Prince Charles? ⇒ Numeric: Distance

As we shall demonstrate, this feature is of prominent importance to unveil a question’s true category.
The question headword is a word in a given question that represents the information that is being sought after.

**Example**

- What is Australia’s national **flower**? ⇒ **Entity:Plant**
- Name an American made **motorcycle**. ⇒ **Entity:Vehicle**
- What Japanese **city** was once called Edo? ⇒ **Location:City**
- How **tall** is Prince Charles? ⇒ **Numeric:Distance**

As we shall demonstrate, this feature is of prominent importance to unveil a question’s true category.
The question headword is a word in a given question that represents the information that is being sought after.

**Example**

- What is Australia’s national **flower**? \(\Rightarrow\) **ENTITY:PLANT**
- Name an American made **motorcycle**. \(\Rightarrow\) **ENTITY:VEHICLE**
- What Japanese **city** was once called Edo? \(\Rightarrow\) **LOCATION:City**
- How **tall** is Prince Charles? \(\Rightarrow\) **NUMERIC:Distance**

As we shall demonstrate, this feature is of prominent importance to unveil a question’s true category.
The question headword is a word in a given question that represents the information that is being sought after.

Example

- What is Australia’s national flower? ⇒ ENTITY:PLANT
- Name an American made motorcycle. ⇒ ENTITY:VEHICLE
- What Japanese city was once called Edo? ⇒ LOCATION:CITY
- How tall is Prince Charles? ⇒ NUMERIC:DISTANCE

As we shall demonstrate, this feature is of prominent importance to unveil a question’s true category.
Question headword (cont.)

Example

```
<table>
<thead>
<tr>
<th>S</th>
<th>B</th>
<th>A</th>
<th>R</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHNP</td>
<td>WHNP</td>
<td>WDT</td>
<td>WHNP</td>
<td>WHNP</td>
</tr>
<tr>
<td>WP</td>
<td>WP</td>
<td>NN</td>
<td>NN</td>
<td>NNP</td>
</tr>
<tr>
<td>What</td>
<td>What</td>
<td>country</td>
<td>country</td>
<td>Godiva</td>
</tr>
<tr>
<td>VBZ</td>
<td>VBP</td>
<td>VBP</td>
<td>are</td>
<td>are</td>
</tr>
<tr>
<td>is</td>
<td>are</td>
<td>are</td>
<td>NNP</td>
<td>NNP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NNS</td>
<td>NNS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>chocolates</td>
<td>chocolates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question headword (cont.)

Example

```
SBARQ

WHNP
WP
What

SQ
VBZ
is

NP
NNP
Australia

POS
national

NP
NN
flower
```

Example

```
SBARQ

WHNP

WDT
What

SQ
VBZ
is

NP
NNP
Godiva

POS
chocolates
```

Example

```
SBARQ

WHNP

WDT
What

SQ
VBZ
is

PP
IN
from

NP
IN
of

NN
Babar
```

Example

```
SBARQ

WHNP

WDT
What

SQ
VBZ
is

NP
NN
capital

POS
's
```

Example

```
SBARQ

WHNP

WDT
What

SQ
VBZ
is

NP
NN
'country'
```

Example

```
SBARQ

WHNP

WDT
What

SQ
VBZ
is

NP
NN
Godiva
```

Example

```
SBARQ

WHNP

WDT
What

SQ
VBZ
is
```

Example

```
SBARQ

WHNP

WDT
What

SQ
VBZ
is

NP
NN
Godiva
```

Example

```
SBARQ

WHNP

WDT
What

SQ
VBZ
is
```
Question headword (cont.)

Example

SBARQ

WHNP

WP

What

VBZ

is

NP

NNP

POS

national

NN

Australia

's

Example

SBARQ

WHNP

NP

VBZ

is

NN

POSS

capital

' s

Example

SBARQ

WHNP

NP

VBZ

are

NP

NNP

NNS

IN

from

Godiva

chocolates

Example

SBARQ

WHNP

NP

VBZ

is

NN

POSS

capital

IN

of

NP

NN

animal

Babar
Question headword (cont.)

Example

```
SBARQ
  WHNP
    WP
    What
  SQ
    VBZ
    is
    NP
      NNP
      Australia
      POS
      's
      JJ
      national
      NN
      flower
```

Example

```
SBARQ
  WHNP
    WDT
    What
    NP
      NNP
      country
      POS
      's
      NN
      capital
      NP
        NNP
        Tirana
```

Example

```
SBARQ
  WHNP
    WDT
    What
    NP
      NNP
      kind
      IN
      of
      NN
      Babar
      NNS
      animal
      PP
      from
```
Part-of-speech tags

- Given the parse tree of a question, the pre-terminal nodes are extracted to use as features.
- Pre-terminal nodes represent the part-of-speech (POS) tags or grammatical classes of the question tokens.

Example

```
  SBARQ
   /
  WHNP
   /  
 WP  What
   /    
 VBZ  NP
    /   / 
  is  DT NN  IN
     /   /   /  
    the capital  of  NNP
         /     /     |
        France
```
Part-of-speech tags

- Given the parse tree of a question, the pre-terminal nodes are extracted to use as features.
- Pre-terminal nodes represent the part-of-speech (POS) tags or grammatical classes of the question tokens.

Example

```
(SBARQ
  (WHNP WP)
  (SQ
    (VBZ is)
    (NP
      (DT the)
      (NN capital)
      (IN of)
      (NNP France)
    ))
)```
Part-of-speech tags

- Given the parse tree of a question, the pre-terminal nodes are extracted to use as features.
- Pre-terminal nodes represent the part-of-speech (POS) tags or grammatical classes of the question tokens.

Example

```
(SBARQ
  (WHNP WP What)
  (SQ VBZ is)
)
```

```
(SQ
  (NP DT the)
  (IN of)
  (NP NNP France)
  (PP IN)
)
```

```
(PP IN)
```

```
(VBZ is)
```

```
(WHNP WP What)
```

```
(SQ VBZ is)
```

```
(NP DT the)
```

```
(NP NNP France)
```

```
(IN of)
```

```
```
Named entities

- Named entity recognition is the task of finding and classifying particular names in a natural language text into predefined categories.

**Example**

When did [**NE_Person** Christopher Columbus] [**NE_Location** Hispaniola]?

- Three types of named entities were extracted: **NE_Person**, **NE_Location**, and **NE_Organization**.
- The recognized named entity types in a question are used as features.
Named entities

- Named entity recognition is the task of finding and classifying particular names in a natural language text into predefined categories.

**Example**

When did [NE_Person Christopher Columbus] [NE_Location Hispaniola]?

- Three types of named entities were extracted: NE_PERSON, NE_LOCATION, and NE_ORGANIZATION.
- The recognized named entity types in a question are used as features.
Named entities

- Named entity recognition is the task of finding and classifying particular names in a natural language text into predefined categories.

**Example**

When did [NE_Person Christopher Columbus] [NE_Location Hispaniola]?

- Three types of named entities were extracted: NE_Person, NE_Location, and NE_Organization.
- The recognized named entity types in a question are used as features.
Named entities

- Named entity recognition is the task of finding and classifying particular names in a natural language text into predefined categories.

Example

When did [NE_Person Christopher Columbus] [NE_Location Hispaniola]?

- Three types of named entities were extracted: NE_Person, NE_Location, and NE_Organization.
- The recognized named entity types in a question are used as features.
Semantic headword

- Further improves the question headword feature, by using WordNet to enrich the question headword with semantics

Example

- What explorer was nicknamed Iberia’s Pilot?
- What actor first portrayed James Bond?
- What dictator has the nickname “El Maximo”?

- All of the above examples fall under the HUMAN:INDIVIDUAL category
- Yet, the question headword is different in all of them
- But, they share a common trait: they are all subordinates (hyponyms) of the word person – the superordinate (hypernym)
- We exploit this observation to devise a new feature that adds semantics to the question headword
Semantic headword

- Further improves the question headword feature, by using WordNet to enrich the question headword with semantics.

Example

- What **explorer** was nicknamed Iberia’s Pilot?
- What **actor** first portrayed James Bond?
- What **dictator** has the nickname “El Maximo”?

- All of the above examples fall under the **HUMAN:INDIVIDUAL** category.
- Yet, the question headword is different in all of them.
- But, they share a common trait: they are all subordinates (hyponyms) of the word **person** – the superordinate (hypernym).
- We exploit this observation to devise a new feature that adds semantics to the question headword.
Semantic headword

- Further improves the question headword feature, by using WordNet to enrich the question headword with semantics.

**Example**

- What explorer was nicknamed Iberia’s Pilot?
- What actor first portrayed James Bond?
- What dictator has the nickname “El Maximo”?

- All of the above examples fall under the Human:Individual category.
  - Yet, the question headword is different in all of them.
  - But, they share a common trait: they are all subordinates (hyponyms) of the word person – the superordinate (hyponym).
  - We exploit this observation to devise a new feature that adds semantics to the question headword.
Semantic headword

- Further improves the question headword feature, by using WordNet to enrich the question headword with semantics

Example

- What explorer was nicknamed Iberia’s Pilot?
- What actor first portrayed James Bond?
- What dictator has the nickname “El Maximo”?

- All of the above examples fall under the HUMAN:INDIVIDUAL category
- Yet, the question headword is different in all of them
- But, they share a common trait: they are all subordinates (hyponyms) of the word person – the superordinate (hyponym)
- We exploit this observation to devise a new feature that adds semantics to the question headword
Semantic headword

- Further improves the question headword feature, by using WordNet to enrich the question headword with semantics

Example

- What explorer was nicknamed Iberia’s Pilot?
- What actor first portrayed James Bond?
- What dictator has the nickname “El Maximo”?

- All of the above examples fall under the Human:Individual category
- Yet, the question headword is different in all of them
- But, they share a common trait: they are all subordinates (hyponyms) of the word person – the superordinate (hyponym)
- We exploit this observation to devise a new feature that adds semantics to the question headword
Semantic headword

- Further improves the question headword feature, by using WordNet to enrich the question headword with semantics

Example

- What **explorer** was nicknamed Iberia’s Pilot?
- What **actor** first portrayed James Bond?
- What **dictator** has the nickname “El Maximo”?

All of the above examples fall under the **Human: Individual** category

Yet, the question headword is different in all of them

But, they share a common trait: they are all subordinates (hyponyms) of the word **person** – the superordinate (hyponym)

We exploit this observation to devise a new feature that adds semantics to the question headword
Sets of related WordNet synsets were manually grouped together into fifty clusters, with each cluster representing a fine-grained question category.

Given a question headword, we translate it into a WordNet synset using a set of heuristics.

A breadth-first search is applied on the translated synset’s hypernym tree, in order to find a synset that pertains to any of the pre-defined clusters.
### Table: Examples of clusters that aggregate similar synsets together.

<table>
<thead>
<tr>
<th>Cluster name</th>
<th>Synsets</th>
<th>Example hyponyms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity: Animal</strong></td>
<td>animal, animate_being, beast, brute, creature, fauna animal_group</td>
<td>mammal, fish, cat, flock, herd, breed</td>
</tr>
<tr>
<td><strong>Entity: Creative</strong></td>
<td>show, music, writing, written material, piece of writing</td>
<td>movie, film, tv show, song, tune, hymn book, poem, novel</td>
</tr>
<tr>
<td><strong>Entity: Plant</strong></td>
<td>vegetation, flora, botany plant, flora, plant life</td>
<td>forest, garden, flower, tree, shrub</td>
</tr>
<tr>
<td><strong>Human: Individual</strong></td>
<td>person, individual, someone, somebody, mortal spiritual being, supernatural being homo, man, human being, human</td>
<td>actor, leader, god, angel, spirit Homo sapiens</td>
</tr>
<tr>
<td><strong>Human: Group</strong></td>
<td>organization, organisation group, grouping</td>
<td>enterprise, company, team, tribe, dynasty</td>
</tr>
<tr>
<td><strong>Numeric: Date</strong></td>
<td>date, day of the month</td>
<td>birthday, natal day</td>
</tr>
<tr>
<td><strong>Numeric: Distance</strong></td>
<td>distance, dimension</td>
<td>altitude, elevation width, length, height</td>
</tr>
</tbody>
</table>
Semantic headword (cont.)

Example

What **actor** first portrayed James Bond?

1. Sense 1

   actor, histrion, player, thespian, role player

   ⇒ performer, performing artist

   ⇒ entertainer

   ⇒ **person, individual, someone, somebody, mortal, soul**

2. Intersected synset belongs to the cluster **HUMAN:INDIVIDUAL**, whose name is then used as a feature
Example

What **actor** first portrayed James Bond?

1. Sense 1
   - actor, histrion, player, thespian, role player
     - ⇒ performer, performing artist
       - ⇒ entertainer
       - ⇒ **person, individual, someone, somebody, mortal, soul**

2. Intersected synset belongs to the cluster **HUMAN:INDIVIDUAL**, whose name is then used as a feature
Semantic headword (cont.)

Example

What actor first portrayed James Bond?

1 Sense 1
   actor, histrion, player, thespian, role player
   ⇒ performer, performing artist
   ⇒ entertainer
   ⇒ person, individual, someone, somebody, mortal, soul

2 Intersected synset belongs to the cluster HUMAN:INDIVIDUAL, whose name is then used as a feature
Example

What actor first portrayed James Bond?

1 Sense 1
actor, histrion, player, thespian, role player
⇒ performer, performing artist
⇒ entertainer
⇒ person, individual, someone, somebody, mortal, soul

2 Intersected synset belongs to the cluster HUMAN:INDIVIDUAL, whose name is then used as a feature
Example

What **actor** first portrayed James Bond?

1. Sense 1
   - actor, histrion, player, thespian, role player
   - ⇒ performer, performing artist
   - ⇒ entertainer
   - ⇒ **person, individual, someone, somebody, mortal, soul**

2. Intersected synset belongs to the cluster **HUMAN:INDIVIDUAL**, whose name is then used as a feature
Example

What **actor** first portrayed James Bond?

1. Sense 1
   
   actor, histrion, player, thespian, role player

   ⇒ performer, performing artist

   ⇒ entertainer

   ⇒ **person, individual, someone, somebody, mortal, soul**

2. Intersected synset belongs to the cluster **HUMAN:INDIVIDUAL**, whose name is then used as a feature
Semantic headword (cont.)

- Semantic headword effectively enriches the headword with semantic knowledge
- As we shall demonstrate, it is the most discriminative feature of all features described so far

Example

- What actor first portrayed James Bond? \textit{actor IS-A person} $\Rightarrow$ \texttt{HUMAN:INDIVIDUAL}
- What is the \textit{wingspan} of a condor? \textit{wingspan IS-A distance} $\Rightarrow$ \texttt{NUMERIC:DISTANCE}
Semantic headword (cont.)

- Semantic headword effectively enriches the headword with semantic knowledge
- As we shall demonstrate, it is the most discriminative feature of all features described so far

Example

- What *actor* first portrayed James Bond? *actor* IS-A *person* ⇒ *Human:Individual*
- What is the *wingspan* of a condor? *wingspan* IS-A *distance* ⇒ *Numeric:Distance*
Semantic headword (cont.)

- Semantic headword effectively enriches the headword with semantic knowledge.
- As we shall demonstrate, it is the most discriminative feature of all features described so far.

**Example**

- What **actor** first portrayed James Bond?  
  actor IS-A person ⇒ HUMAN:INDIVIDUAL

- What is the **wingspan** of a condor?  
  wingspan IS-A distance ⇒ NUMERIC:DISTANCE
Semantic headword (cont.)

- Semantic headword effectively enriches the headword with semantic knowledge
- As we shall demonstrate, it is the most discriminative feature of all features described so far

Example

- What **actor** first portrayed James Bond? *actor IS-A person* ⇒ *HUMAN:INDIVIDUAL*
- What is the **wingspan** of a condor? *wingspan IS-A distance* ⇒ *NUMERIC:DISTANCE*
Semantic headword (cont.)

- Semantic headword effectively enriches the headword with semantic knowledge
- As we shall demonstrate, it is the most discriminative feature of all features described so far

Example

- What actor first portrayed James Bond? *actor IS-A person* ⇒ *HUMAN:INDIVIDUAL*
- What is the wingspan of a condor? *wingspan IS-A distance* ⇒ *NUMERIC:DISTANCE*
Semantic headword (cont.)

- Semantic headword effectively enriches the headword with semantic knowledge
- As we shall demonstrate, it is the most discriminative feature of all features described so far

Example

- What actor first portrayed James Bond? \textit{actor IS-A person} ⇒ \texttt{HUMAN:INDIVIDUAL}
- What is the \textit{wingspan} of a condor? \textit{wingspan IS-A distance} ⇒ \texttt{NUMERIC:DISTANCE}
Semantic headword (cont.)

- Semantic headword effectively enriches the headword with semantic knowledge
- As we shall demonstrate, it is the most discriminative feature of all features described so far

Example

- What **actor** first portrayed James Bond? *actor IS-A person ⇒ HUMAN:INDIVIDUAL*
- What is the **wingspan** of a condor? *wingspan IS-A distance ⇒ NUMERIC:DISTANCE*
Semantic headword effectively enriches the headword with semantic knowledge.

As we shall demonstrate, it is the most discriminative feature of all features described so far.

**Example**

- What **actor** first portrayed James Bond? *actor IS-A person ⇒ HUMAN:INDIVIDUAL*
- What is the **wingspan** of a condor? *wingspan IS-A distance ⇒ NUMERIC:DISTANCE*
Outline

1. Introduction

2. Question Classification
   - Introduction
   - Question Type Taxonomy
   - Question Feature Set
   - Experimental Setup
   - Experimental Results

3. Passage Retrieval

4. Answer Extraction

5. Evaluation
Algorithms, Data sets, and Evaluation Measures

- **Algorithms**
  - Suport Vector Machine (SVM)
  - k-Nearest Neighbors (kNN)
  - Naïve Bayes (NB)

- **Data sets**
  - Publicly available data set from UIUC, which consists of a training set of 5500 questions, and a test set of 500 questions.
  - The annotated question categories follow the question type taxonomy of Li&Roth
  - Used by many published works in the literature

- **Evaluation measures**
  - Accuracy = \( \frac{\text{# of correctly classified questions}}{\text{# of total questions in the test set}} \)
Algorithms, Data sets, and Evaluation Measures

- **Algorithms**
  - Support Vector Machine (SVM)
  - $k$-Nearest Neighbors ($k$NN)
  - Naïve Bayes (NB)

- **Data sets**
  - Publicly available data set from UIUC, which consists of a training set of 5500 questions, and a test set of 500 questions.
  - The annotated question categories follow the question type taxonomy of Li&Roth
  - Used by many published works in the literature

- **Evaluation measures**
  - $Accuracy = \frac{\# \text{ of correctly classified questions}}{\# \text{ of total questions in the test set}}$
Algorithms, Data sets, and Evaluation Measures

- **Algorithms**
  - Support Vector Machine (SVM)
  - $k$-Nearest Neighbors ($k$NN)
  - Naïve Bayes (NB)

- **Data sets**
  - Publicly available data set from UIUC, which consists of a training set of 5500 questions, and a test set of 500 questions.
  - The annotated question categories follow the question type taxonomy of Li&Roth
  - Used by many published works in the literature

- **Evaluation measures**
  - $\text{Accuracy} = \frac{\# \text{ of correctly classified questions}}{\# \text{ of total questions in the test set}}$
Outline

1 Introduction

2 Question Classification
   - Introduction
   - Question Type Taxonomy
   - Question Feature Set
   - Experimental Setup
   - Experimental Results

3 Passage Retrieval

4 Answer Extraction

5 Evaluation
Comparison of classification algorithms

Table: Question classification accuracy using different machine learning algorithms and different training set sizes, under the coarse grained category.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of questions in training set</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
<th>4000</th>
<th>5500</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>78.4% 83.6% 85.0% 86.2% 88.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>70.4% 72.4% 73.0% 75.6% 78.6%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kNN</td>
<td>62.6% 68.6% 72.4% 73.0% 75.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lexico-syntactic features contribution

Lexico-syntactic features for fine classification

Accuracy vs. Features

Unigrams | Bigrams | Trigrams | POS | Headword

João Silva (IST-UTL)
QA+ML@Wikipedia&Google
November 9, 2009
Semantic features contribution

Semantic features for fine classification

Accuracy vs. Features

- Unigrams
- Headword
- WordNet
- Named entities

Accuracy values:
- 82.2%
- 85.4%
- 85.6%
- 88%
- 90.4%
- 90.6%
- 92%
Comparison with other works

**Table:** Comparison of question classification results attained by this work, against other results reported in the literature.

<table>
<thead>
<tr>
<th>Author</th>
<th>Category granularity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coarse</td>
</tr>
<tr>
<td>This work</td>
<td>95.2%</td>
</tr>
<tr>
<td>(X. Li &amp; Roth, 2002)</td>
<td>91.0%</td>
</tr>
<tr>
<td>(Zhang &amp; Lee, 2003)</td>
<td>90.0%</td>
</tr>
<tr>
<td>(Krishnan et al., 2005)</td>
<td>93.4%</td>
</tr>
<tr>
<td>(Blunsom et al., 2006)</td>
<td>91.8%</td>
</tr>
<tr>
<td>(Pan et al., 2008)</td>
<td>94.0%</td>
</tr>
<tr>
<td>(Huang et al., 2008)</td>
<td>93.6%</td>
</tr>
</tbody>
</table>
Outline

1. Introduction

2. Question Classification

3. Passage Retrieval
   - Introduction
   - Information Sources
   - Query Formulation

4. Answer Extraction

5. Evaluation

6. Contributions
The goal of passage retrieval is to find relevant passages where the answer to a given question might be found. Can be further divided into two stages:

1. **Query formulation**, in which a question is translated into a suitable representation that can be used by an information source to retrieve relevant passages.

2. **Query submission**, wherein the generated queries from the previous step are fed into the endpoint of an information source, and relevant passages are retrieved.

The retrieved passages are used by the answer extraction component to extract plausible answers.
The goal of passage retrieval is to find relevant passages where the answer to a given question might be found.

Can be further divided into two stages:

1. **Query formulation**, in which a question is translated into a suitable representation that can be used by an information source to retrieve relevant passages.

2. **Query submission**, wherein the generated queries from the previous step are fed into the endpoint of an information source, and relevant passages are retrieved.

The retrieved passages are used by the answer extraction component to extract plausible answers.
The goal of passage retrieval is to find relevant passages where the answer to a given question might be found. Can be further divided into two stages:

1. **Query formulation**, in which a question is translated into a suitable representation that can be used by an information source to retrieve relevant passages.
2. **Query submission**, wherein the generated queries from the previous step are fed into the endpoint of an information source, and relevant passages are retrieved.

The retrieved passages are used by the answer extraction component to extract plausible answers.
Introduction

The goal of passage retrieval is to find relevant passages where the answer to a given question might be found.

Can be further divided into two stages:

1. **Query formulation**, in which a question is translated into a suitable representation that can be used by an information source to retrieve relevant passages.
2. **Query submission**, wherein the generated queries from the previous step are fed into the endpoint of an information source, and relevant passages are retrieved.

The retrieved passages are used by the answer extraction component to extract plausible answers.
The goal of passage retrieval is to find relevant passages where the answer to a given question might be found.

Can be further divided into two stages:

1. **Query formulation**, in which a question is translated into a suitable representation that can be used by an information source to retrieve relevant passages.

2. **Query submission**, wherein the generated queries from the previous step are fed into the endpoint of an information source, and relevant passages are retrieved.

The retrieved passages are used by the answer extraction component to extract plausible answers.
Our approach

- A multi-strategy approach to passage retrieval, with each strategy tailored to a specific question category or groups of question categories
- We propose several query formulation strategies, in order to improve the results returned by the information sources
Our approach

- A multi-strategy approach to passage retrieval, with each strategy tailored to a specific question category or groups of question categories
- We propose several query formulation strategies, in order to improve the results returned by the information sources
Our approach

- A multi-strategy approach to passage retrieval, with each strategy tailored to a specific question category or groups of question categories
- We propose several query formulation strategies, in order to improve the results returned by the information sources
Information sources

- Google search is used to retrieve passages – summaries or *snippets* – for factoid-type questions
- Due to Wikipedia’s encyclopaedic nature, we employ it to answer non-factoid questions that require longer answers; particularly, *Description:Definition* and *Human:Definition* questions
- DBpedia can be seen as a machine-readable version of Wikipedia, and is used in conjunction with Wikipedia to answer non-factoid questions
Information sources

- Google search is used to retrieve passages – summaries or *snippets* – for factoid-type questions.
- Due to Wikipedia’s encyclopaedic nature, we employ it to answer non-factoid questions that require longer answers; particularly, Description:Definition and Human:Definition questions.
- DBpedia can be seen as a machine-readable version of Wikipedia, and is used in conjunction with Wikipedia to answer non-factoid questions.
Information sources

- Google search is used to retrieve passages – summaries or *snippets* – for factoid-type questions.
- Due to Wikipedia’s encyclopaedic nature, we employ it to answer non-factoid questions that require longer answers; particularly, Description:Definition and Human:Definition questions.
- DBpedia can be seen as a machine-readable version of Wikipedia, and is used in conjunction with Wikipedia to answer non-factoid questions.
Outline

1. Introduction
2. Question Classification
3. Passage Retrieval
   - Introduction
   - Information Sources
   - Query Formulation
4. Answer Extraction
5. Evaluation
6. Contributions
Keyword-based

- Most simple query formulation strategy
  - Generates a query that comprises all the words in a given question, except stop words, such as articles and adverbs

Example

When was Beethoven born?

- **Query**: Beethoven born

- Used to generate queries for use in Google search
- Applies to factoid-type questions
Keyword-based

- Most simple query formulation strategy
- Generates a query that comprises all the words in a given question, except stop words, such as articles and adverbs

Example

When was Beethoven born?

- **Query**: *Beethoven born*

- Used to generate queries for use in Google search
- Applies to factoid-type questions
Keyword-based

- Most simple query formulation strategy
- Generates a query that comprises all the words in a given question, except stop words, such as articles and adverbs

Example

When was Beethoven born?

- Query: *Beethoven born*

- Used to generate queries for use in Google search
- Applies to factoid-type questions
Keyword-based

- Most simple query formulation strategy
- Generates a query that comprises all the words in a given question, except stop words, such as articles and adverbs

**Example**

When was Beethoven born?

- **Query:** *Beethoven born*

- Used to generate queries for use in Google search
- Applies to factoid-type questions
Keyword-based

- Most simple query formulation strategy
- Generates a query that comprises all the words in a given question, except stop words, such as articles and adverbs

**Example**

When was Beethoven born?

- **Query**: *Beethoven born*

- Used to generate queries for use in Google search
- Applies to factoid-type questions
Wikipedia’s search facilities are used to determine the title of the article where the answer to a given question might be found, using a keyword-based strategy.

DBpedia is then used to retrieve the abstract of the article, which is returned as the answer.

Example

Who was Afonso Henriques?

1. A query with the term *Afonso Henriques* is generated and sent to Wikipedia’s API.
2. The first result returned – *Afonso I of Portugal* – is transformed into a DBpedia resource – http://dbpedia.org/resource/Afonso_I_of_Portugal.
3. A SPARQL query is issued to retrieve the abstract of the article from DBpedia:

   ```sparql
   PREFIX db: <http://dbpedia.org/property/>
   SELECT ?abstract WHERE {
     FILTER (lang(?abstract)="en")
   }
   ```

4. Abstract: *Afonso I (…) was the first king of Portugal.*
Wikipedia & DBpedia

- Wikipedia’s search facilities are used to determine the title of the article where the answer to a given question might be found, using a keyword-based strategy.
- DBpedia is then used to retrieve the abstract of the article, which is returned as the answer.

Example

Who was Afonso Henriques?

1. A query with the term *Afonso Henriques* is generated and sent to Wikipedia’s API.
2. The first result returned – *Afonso I of Portugal* – is transformed into a DBpedia resource – http://dbpedia.org/resource/Afonso_I_of_Portugal
3. A SPARQL query is issued to retrieve the abstract of the article from DBpedia:

   ```sparql
   PREFIX db: <http://dbpedia.org/property/>
   SELECT ?abstract
   WHERE {
     FILTER (lang(?abstract)="en")
   }
   ```
4. Abstract: *Afonso I (…) was the first king of Portugal.*
Wikipedia & DBpedia

- Wikipedia’s search facilities are used to determine the title of the article where the answer to a given question might be found, using a keyword-based strategy.
- DBpedia is then used to retrieve the *abstract* of the article, which is returned as the answer.

**Example**

**Who was Afonso Henriques?**

1. A query with the term *Afonso Henriques* is generated and sent to Wikipedia’s API.
3. A **SPARQL** query is issued to retrieve the abstract of the article from DBpedia:
   ```sparql
   PREFIX db: <http://dbpedia.org/property/>
   SELECT ?abstract WHERE {
     FILTER (lang(?abstract)="en")
   }
   ```
4. Abstract: *Afonso I (...) was the first king of Portugal.*
Wikipedia & DBpedia

- Wikipedia’s search facilities are used to determine the title of the article where the answer to a given question might be found, using a keyword-based strategy.
- DBpedia is then used to retrieve the abstract of the article, which is returned as the answer.

Example

Who was Afonso Henriques?

1. A query with the term *Afonso Henriques* is generated and sent to Wikipedia’s API.
2. The first result returned – *Afonso I of Portugal* – is transformed into a DBpedia resource – http://dbpedia.org/resource/Afonso_I_of_Portugal.
3. A SPARQL query is issued to retrieve the abstract of the article from DBpedia.
   ```sparql
   PREFIX db: <http://dbpedia.org/property/>
   SELECT ?abstract WHERE {
     FILTER (lang(?abstract)="en")
   }
   ```
4. Abstract: *Afonso I (…) was the first king of Portugal.*
Wikipedia & DBpedia

- Wikipedia’s search facilities are used to determine the title of the article where the answer to a given question might be found, using a keyword-based strategy.
- DBpedia is then used to retrieve the *abstract* of the article, which is returned as the answer.

Example

**Who was Afonso Henriques?**

1. A query with the term *Afonso Henriques* is generated and sent to Wikipedia’s API.
2. The first result returned – *Afonso I of Portugal* – is transformed into a DBpedia resource – http://dbpedia.org/resource/Afonso_I_of_Portugal.
3. A **SPARQL** query is issued to retrieve the abstract of the article from DBpedia:
   ```sparql
   PREFIX db: <http://dbpedia.org/property/>
   SELECT ?abstract WHERE {
     FILTER (lang(?abstract)="en")
   }
   ```
4. Abstract: *Afonso I (...) was the first king of Portugal.*
Wikipedia & DBpedia

- Wikipedia’s search facilities are used to determine the title of the article where the answer to a given question might be found, using a keyword-based strategy
- DBpedia is then used to retrieve the abstract of the article, which is returned as the answer

Example

Who was Afonso Henriques?

1 A query with the term Afonso Henriques is generated and sent to Wikipedia’s API
2 The first result returned – Afonso I of Portugal – is transformed into a DBpedia resource – http://dbpedia.org/resource/Afonso_I_of_Portugal
3 A SPARQL query is issued to retrieve the abstract of the article from DBpedia

```sparql
PREFIX db: <http://dbpedia.org/property/>
SELECT ?abstract WHERE {
  FILTER (lang(?abstract)="en")
}
```
4 Abstract: Afonso I (...) was the first king of Portugal.
Wikipedia & DBpedia

- Wikipedia’s search facilities are used to determine the title of the article where the answer to a given question might be found, using a keyword-based strategy.
- DBpedia is then used to retrieve the *abstract* of the article, which is returned as the answer.

Example

Who was Afonso Henriques?

1. A query with the term *Afonso Henriques* is generated and sent to Wikipedia’s API.
3. A **SPARQL** query is issued to retrieve the abstract of the article from DBpedia:
   ```sparql
   PREFIX db: <http://dbpedia.org/property/>
   SELECT ?abstract WHERE {
     FILTER (lang(?abstract)="en")
   }
   ```
4. Abstract: *Afonso I (...) was the first king of Portugal.*
Wikipedia & DBpedia

- Wikipedia’s search facilities are used to determine the title of the article where the answer to a given question might be found, using a keyword-based strategy.
- DBpedia is then used to retrieve the *abstract* of the article, which is returned as the answer.

**Example**

Who was Afonso Henriques?

1. A query with the term *Afonso Henriques* is generated and sent to Wikipedia’s API.
2. The first result returned – *Afonso I of Portugal* – is transformed into a DBpedia resource – `http://dbpedia.org/resource/Afonso_I_of_Portugal`.
3. A SPARQL query is issued to retrieve the abstract of the article from DBpedia:
   ```sparql
   PREFIX db: <http://dbpedia.org/property/>
   SELECT ?abstract WHERE {
     FILTER (lang(?abstract)="en")
   }
   ```
4. Abstract: *Afonso I (…) was the first king of Portugal.*
Pattern-based

- Given a large collection of documents – e.g., the Web –, the answer to a given question will probably occur in sentences that contain a rewrite of the original question.

**Example**

**What is the capital of Portugal?**

- The capital of Portugal is Lisbon.
- Lisbon is the capital of Portugal.

We developed a semi-supervised algorithm, based on bootstrapping, that learns question rewrite rules:

1. Uses a question-answer seed pair to bootstrap new question rewrite patterns
2. The learnt patterns are validated with a different question-answer pair, to remove incorrect patterns.
Pattern-based

- Given a large collection of documents – e.g., the Web –, the answer to a given question will probably occur in sentences that contain a rewrite of the original question.

**Example**

**What is the capital of Portugal?**

- The capital of Portugal is Lisbon.
- Lisbon is the capital of Portugal.

- We developed a semi-supervised algorithm, based on bootstrapping, that learns question rewrite rules.
  1. Uses a question-answer seed pair to bootstrap new question rewrite patterns.
  2. The learnt patterns are validated with a different question-answer pair, to remove incorrect patterns.
Pattern-based

- Given a large collection of documents – e.g., the Web –, the answer to a given question will probably occur in sentences that contain a rewrite of the original question.

Example

What is the capital of Portugal?

- The capital of Portugal is Lisbon.
- Lisbon is the capital of Portugal.

We developed a semi-supervised algorithm, based on bootstrapping, that learns question rewrite rules.

1. Uses a question-answer seed pair to bootstrap new question rewrite patterns.
2. The learnt patterns are validated with a different question-answer pair, to remove incorrect patterns.
Pattern-based

- Given a large collection of documents – e.g., the Web –, the answer to a given question will probably occur in sentences that contain a rewrite of the original question.

Example

What is the capital of Portugal?

- The capital of Portugal is Lisbon.
- Lisbon is the capital of Portugal.

- We developed a semi-supervised algorithm, based on bootstrapping, that learns question rewrite rules:
  1. Uses a question-answer seed pair to bootstrap new question rewrite patterns.
  2. The learnt patterns are validated with a different question-answer pair, to remove incorrect patterns.
Pattern-based

- Given a large collection of documents – e.g., the Web –, the answer to a given question will probably occur in sentences that contain a rewrite of the original question.

**Example**

**What is the capital of Portugal?**

- The capital of Portugal is Lisbon.
- Lisbon is the capital of Portugal.

- We developed a semi-supervised algorithm, based on bootstrapping, that learns question rewrite rules:
  1. Uses a question-answer seed pair to bootstrap new question rewrite patterns
  2. The learnt patterns are validated with a different question-answer pair, to remove incorrect patterns.
Pattern-based

- Given a large collection of documents – e.g., the Web –, the answer to a given question will probably occur in sentences that contain a rewrite of the original question.

**Example**

What is the capital of Portugal?

- The capital of Portugal is Lisbon.
- Lisbon is the capital of Portugal.

- We developed a semi-supervised algorithm, based on bootstrapping, that learns question rewrite rules:
  1. Uses a question-answer seed pair to bootstrap new question rewrite patterns.
  2. The learnt patterns are validated with a different question-answer pair, to remove incorrect patterns.
Pattern-based (cont.)

Example

**Seed pair:** Where is the Louvre Museum?/Paris

**Validation pair:** Where is the CN Tower?/Toronto

- Extract phrasal nodes from the parse tree of the seed question

```
  SBARQ
  |
  WHADVP
  |
  WRB
  |
  Where

  SQ
  |
  VBZ
  |
  is

  NP
  |
  DT
  |
  the

  NNP
  |
  Louvre

  NN
  |
  Museum
```

- Permute the set \{is, the Louvre Museum, *, Paris\}, resulting in 4! permutations

- Each permutation is transformed into a query, which is issued to Google Search

- For instance, the query "the Louvre Museum is * Paris" returns the following passages
  - the Louvre Museum is located in Paris.
  - the Louvre Museum is six kilometres away from Paris.

- These passages are then transformed into the following lexico-syntactic patterns
  - NP VBZ located in ANSWER.
  - NP VBZ six kilometres away from ANSWER.
Example

**Seed pair:** Where is the Louvre Museum?/Paris

**Validation pair:** Where is the CN Tower?/Toronto

- Extract phrasal nodes from the parse tree of the seed question

- Permute the set \{is, the Louvre Museum, *, Paris\}, resulting in 4! permutations

- Each permutation is transformed into a query, which is issued to Google Search

- For instance, the query "the Louvre Museum is * Paris" returns the following passages
  - the Louvre Museum is located in Paris.
  - the Louvre Museum is six kilometres away from Paris.

- These passages are then transformed into the following lexico-syntactic patterns
  - NP VBZ located in ANSWER.
  - NP VBZ six kilometres away from ANSWER.
Example

Seed pair: Where is the Louvre Museum?/Paris
Validation pair: Where is the CN Tower?/Toronto

- Extract phrasal nodes from the parse tree of the seed question

```
  WHADVP
     WRB
       Where
```
```
  SQ
    VBZ
      NP
        DT
          the
        NNP
          Louvre
        NN
          Museum
```

- Permute the set {is, the Louvre Museum, *, Paris}, resulting in 4! permutations
- Each permutation is transformed into a query, which is issued to Google Search
- For instance, the query “the Louvre Museum is * Paris” returns the following passages
  - the Louvre Museum is located in Paris.
  - the Louvre Museum is six kilometres away from Paris.

- These passages are then transformed into the following lexico-syntactic patterns
  - NP VBZ located in ANSWER.
  - NP VBZ six kilometres away from ANSWER.
### Example

**Seed pair:** Where is the Louvre Museum?/Paris  
**Validation pair:** Where is the CN Tower?/Toronto

1. **Extract phrasal nodes from the parse tree of the seed question**
   - SBARQ  
     - WHADVP  
     - WRB: Where  
     - SQ  
     - VBZ: is  
     - NP: the Louvre Museum

2. **Permute the set** `{is, the Louvre Museum, *, Paris}` **resulting in 4! permutations**

3. **Each permutation is transformed into a query, which is issued to Google Search**

   For instance, the query "the Louvre Museum is * Paris" returns the following passages:
   - the Louvre Museum is located in Paris.  
   - the Louvre Museum is six kilometres away from Paris.

4. **These passages are then transformed into the following lexico-syntactic patterns**
   - NP VBZ located in ANSWER.  
   - NP VBZ six kilometres away from ANSWER.
**Example**

**Seed pair:** Where is the Louvre Museum?/Paris

**Validation pair:** Where is the CN Tower?/Toronto

- Extract phrasal nodes from the parse tree of the seed question

```
  SBARQ
  |    WHADVP
  |     WRB
  |     Where
  |     VBZ
  |     NP

  SQ

  VBZ
  |    NP

  DT
  |    NNP
  |    NN

  the

  Louvre

  Museum
```

- Permute the set \{is, the Louvre Museum, *, Paris\}, resulting in 4! permutations
- Each permutation is transformed into a query, which is issued to Google Search

- For instance, the query "the Louvre Museum is * Paris" returns the following passages
  - the Louvre Museum is located in Paris.
  - the Louvre Museum is six kilometres away from Paris.

- These passages are then transformed into the following lexico-syntactic patterns
  - NP VBZ located in ANSWER.
  - NP VBZ six kilometres away from ANSWER.
Pattern-based (cont.)

Example

**Seed pair:** Where is the Louvre Museum?/Paris  
**Validation pair:** Where is the CN Tower?/Toronto

- Extract phrasal nodes from the parse tree of the seed question

```
  SBARQ
  |    |    |
  | WHADVP | SQ |
  | WRB    |   |
  | Where  | SQ |
  | VBZ    |   |
  | is     | NP |
  | DT     | NNP |
  | the    | NN |
  | Louvre | Museum |
```

- Permute the set \{is, the Louvre Museum, *, Paris\}, resulting in 4! permutations
- Each permutation is transformed into a query, which is issued to Google Search

For instance, the query "the Louvre Museum is * Paris" returns the following passages

- the Louvre Museum is located in Paris.
- the Louvre Museum is six kilometres away from Paris.

These passages are then transformed into the following lexico-syntactic patterns

- NP VBZ located in ANSWER.
- NP VBZ six kilometres away from ANSWER.
Pattern-based (cont.)

**Example**

**Seed pair:** Where is the Louvre Museum?/Paris

**Validation pair:** Where is the CN Tower?/Toronto

- Extract phrasal nodes from the parse tree of the seed question

```
  SBARQ
  |    WHADVP
  |     WRB
  |    Where

  SQ
   |    VBZ
   |     NP
   |      DT
   |      NNP
   |      NN

  is
  the
  Louvre
  Museum
```

- Permute the set \{is, the Louvre Museum, *, Paris\}, resulting in 4! permutations
- Each permutation is transformed into a query, which is issued to Google Search
- For instance, the query "the Louvre Museum is * Paris" returns the following passages
  - the Louvre Museum is located in Paris.
  - the Louvre Museum is six kilometres away from Paris.

- These passages are then transformed into the following lexico-syntactic patterns
  - NP VBZ located in ANSWER.
  - NP VBZ six kilometres away from ANSWER.
Example

**Seed pair:** Where is the Louvre Museum?/Paris

**Validation pair:** Where is the CN Tower?/Toronto

- Extract phrasal nodes from the parse tree of the seed question

```
     SBARQ
    /     \
   /       \
WHADVP   SQ
   /     \
WRB     VBZ
   /     \
Where   is
```

- Permute the set \{is, the Louvre Museum, *, Paris\}, resulting in 4! permutations
- Each permutation is transformed into a query, which is issued to Google Search
- For instance, the query "the Louvre Museum is * Paris" returns the following passages
  - *the Louvre Museum is located in Paris.*
  - *the Louvre Museum is six kilometres away from Paris.*

- These passages are then transformed into the following lexico-syntactic patterns
  - NP VBZ located in ANSWER.
  - NP VBZ six kilometres away from ANSWER.
Pattern-based (cont.)

**Example**

**Seed pair:** Where is the Louvre Museum?/Paris

**Validation pair:** Where is the CN Tower?/Toronto

- Extract phrasal nodes from the parse tree of the seed question
  
  ![Parse Tree](image)

- Permute the set \{is, the Louvre Museum, *, Paris\}, resulting in 4! permutations
- Each permutation is transformed into a query, which is issued to Google Search
- For instance, the query "the Louvre Museum is * Paris" returns the following passages
  - the Louvre Museum is located in Paris.
  - the Louvre Museum is six kilometres away from Paris.
- These passages are then transformed into the following lexico-syntactic patterns
  - NP VBZ located in ANSWER.
  - NP VBZ six kilometres away from ANSWER.
Outline

1. Introduction

2. Question Classification

3. Passage Retrieval

4. Answer Extraction
   - Introduction
   - Candidate Answer Extraction
   - Answer Extraction

5. Evaluation

6. Contributions
Once a question has been classified, and relevant passages have been retrieved, the last step is to extract candidate answers from the passages, and select a final answer.

Example

When was Mozart born? ⇒ NUMERIC:DATE

- Mozart Wolfgang Amadeus Mozart (born in Salzburg, Austria, on January 27, 1756 – died in Vienna, Austria, on December 5, 1791) was a great composer.
- Mozart was born January 27, 1756 in Salzburg, Austria.
Once a question has been classified, and relevant passages have been retrieved, the last step is to extract candidate answers from the passages, and select a final answer.

Example

When was Mozart born? ⇒ \textsc{Numeric:Date}

- Mozart Wolfgang Amadeus Mozart (born in Salzburg, Austria, on January 27, 1756 – died in Vienna, Austria, on December 5, 1791) was a great composer.
- Mozart was born January 27, 1756 in Salzburg, Austria.
Once a question has been classified, and relevant passages have been retrieved, the last step is to extract candidate answers from the passages, and select a final answer.

**Example**

When was Mozart born? ⇒ **NUMERIC**:Date

- **Mozart Wolfgang Amadeus Mozart** *(born in Salzburg, Austria, on January 27, 1756 – died in Vienna, Austria, on December 5, 1791)* was a great composer.

- Mozart was born January 27, 1756 in Salzburg, Austria.
Once a question has been classified, and relevant passages have been retrieved, the last step is to extract candidate answers from the passages, and select a final answer.

Example

When was Mozart born? ⇒ NUMERIC:DATE

- Mozart Wolfgang Amadeus Mozart (born in Salzburg, Austria, on **January 27, 1756** – died in Vienna, Austria, on **December 5, 1791**) was a great composer.
- Mozart was born **January 27, 1756** in Salzburg, Austria.
Our approach

- We divided the answer extraction process into two stages
  1. Candidate answer extraction, where we take advantage of our rich question type taxonomy to devise strategies for each particular question category or groups of question categories
  2. Answer selection, where our strategy is to first filter out unwanted candidate answers, and then apply a clustering algorithm to group together similar answers.
Our approach

- We divided the answer extraction process into two stages
  1. **Candidate answer extraction**, where we take advantage of our rich question type taxonomy to devise strategies for each particular question category or groups of question categories
  2. **Answer selection**, where our strategy is to first filter out unwanted candidate answers, and then apply a clustering algorithm to group together similar answers.
Our approach

- We divided the answer extraction process into two stages
  1. **Candidate answer extraction**, where we take advantage of our rich question type taxonomy to devise strategies for each particular question category or groups of question categories
  2. **Answer selection**, where our strategy is to first filter out unwanted candidate answers, and then apply a clustering algorithm to group together similar answers.
Outline

1. Introduction
2. Question Classification
3. Passage Retrieval
4. Answer Extraction
   - Introduction
   - Candidate Answer Extraction
   - Answer Extraction
5. Evaluation
6. Contributions
Regular expressions

- An extensive set of regular expressions to extract candidate answers for **NUMERIC** type questions
- Regular expressions provide a very concise way to described candidate answers for numerals

**Example**

**NUMERIC:TEMPERATURE**

- `/[0-9]+(K(elvin)\?)?|R(ankine)\?)?|\(^{\circ}C(elsius)\?)?|\(^{\circ}F(ahrenheit)\?)/`
An extensive set of regular expressions to extract candidate answers for **Numeric** type questions

Regular expressions provide a very concise way to described candidate answers for numerals

**Example**

**Numeric: Temperature**

```
/[0-9]+(K(elvin)?|R(ankine)?|°C(elsius)?|°F(ahrenheit)?)\/
```
We employed Stanford’s Conditional Random Field-based named entity recognizer (NER).

Stanford’s machine learning-based NER is able to recognize four entity types: **PERSON**, **LOCATION**, **ORGANIZATION**, and **MISCELLANEOUS**.

**Table:** Mappings between named entity types and question categories.

<table>
<thead>
<tr>
<th>Named entity type</th>
<th>Question category</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSON</td>
<td>HUMAN:INDIVIDUAL</td>
</tr>
<tr>
<td>LOCATION</td>
<td>LOCATION:_CITY</td>
</tr>
<tr>
<td></td>
<td>LOCATION:COUNTRY</td>
</tr>
<tr>
<td></td>
<td>LOCATION:MOUNTAIN</td>
</tr>
<tr>
<td></td>
<td>LOCATION:OTHER</td>
</tr>
<tr>
<td></td>
<td>LOCATION:STATE</td>
</tr>
<tr>
<td>ORGANIZATION</td>
<td>HUMAN:GROUP</td>
</tr>
<tr>
<td>MISCELLANEOUS</td>
<td>ENTITY:CREATIVE</td>
</tr>
<tr>
<td></td>
<td>ENTITY:Event</td>
</tr>
</tbody>
</table>
WordNet-based recognizer

- We developed a novel approach for extracting answers for type of questions, using WordNet’s hyponymy relations
- Candidate answers for these questions are often hyponyms of the question’s headword

Example

Which animal is the fastest?
- animal, animate being, beast, brute, creature, fauna
  - domestic animal
    - house cat
    - dog
  - predatory animal
    - carnivore
      - feline
        - leopard
        - jaguar
        - cheetah
        - lion

- A dictionary is constructed with the entire hyponym tree of the headword
- The dictionary is then used by an exact dictionary matcher algorithm – Aho-Corasick – to extract candidate answers
Gazetteer

- Certain question categories, such as **Location:Country**, have a very limited set of possible answers
- For these situations, a gazetteer can help
- It can be used to assure that only candidate answers of the expected type are extracted
- We used gazetteers for the **Location:Country** and **Location:City** categories
Gazetteer

- Certain question categories, such as Location:Country, have a very limited set of possible answers.
- For these situations, a gazetteer can help.
- It can be used to assure that only candidate answers of the expected type are extracted.
- We used gazetteers for the Location:Country and Location:City categories.
Gazetteer

- Certain question categories, such as **Location:Country**, have a very limited set of possible answers.
- For these situations, a gazetteer can help.
- It can be used to assure that only candidate answers of the expected type are extracted.
- We used gazetteers for the **Location:Country** and **Location:City** categories.
Certain question categories, such as Location:Country, have a very limited set of possible answers. For these situations, a gazetteer can help. It can be used to assure that only candidate answers of the expected type are extracted. We used gazetteers for the Location:Country and Location:City categories.
Outline

1. Introduction

2. Question Classification

3. Passage Retrieval

4. **Answer Extraction**
   - Introduction
   - Candidate Answer Extraction
   - Answer Extraction

5. Evaluation

6. Contributions
Candidate Answer Filtering

- Removes candidate answers which are contained in the original question

Example

For the question "Who is Tom Cruise’s wife?", classified as HUMAN:INDIVIDUAL, this filter prevents the system from extracting Tom Cruise as a plausible answer.
Candidate Answer Filtering

- Removes candidate answers which are contained in the original question

Example

For the question “Who is Tom Cruise’s wife?”, classified as **HUMAN:INDIVIDUAL**, this filter prevents the system from extracting *Tom Cruise* as a plausible answer.
Candidate Answer Clustering

- Similar candidate answers are grouped together, into a set of clusters.
- The following distance measure is used to measure dissimilarity between candidate answers $X$ and $Y$:

\[
\text{overlap}(X, Y) = 1 - \frac{|X \cap Y|}{\min(|X|, |Y|)}
\]

- Overlap distance is used in conjunction with a standard single-link agglomerative clustering algorithm, using a threshold distance of 0.0.

Example

- Candidate answers:
  \{John Kennedy, Kennedy, John F. Kennedy, John McCarthy\}
- Clusters: \{John Kennedy, Kennedy, John F. Kennedy\} and \{John McCarthy\}
Candidate Answer Clustering

- Similar candidate answers are grouped together, into a set of clusters.
- The following distance measure is used to measure dissimilarity between candidate answers $X$ and $Y$

$$\text{overlap}(X, Y) = 1 - \frac{|X \cap Y|}{\min(|X|, |Y|)}$$

- Overlap distance is used in conjunction with a standard single-link agglomerative clustering algorithm, using a threshold distance of 0.0

**Example**

- **Candidate answers:**
  - \{John Kennedy, Kennedy, John F. Kennedy, John McCarthy\}
- **Clusters:** \{John Kennedy, Kennedy, John F. Kennedy\} and \{John McCarthy\}
Candidate Answer Clustering

- Similar candidate answers are grouped together, into a set of clusters.
- The following distance measure is used to measure *dissimilarity* between candidate answers \( X \) and \( Y \):

\[
\text{overlap}(X, Y) = 1 - \frac{|X \cap Y|}{\min(|X|, |Y|)}
\]

- Overlap distance is used in conjunction with a standard single-link agglomerative clustering algorithm, using a threshold distance of 0.0.

**Example**

- **Candidate answers:**
  
  \{John Kennedy, Kennedy, John F. Kennedy, John McCarthy\}

- **Clusters:** \{John Kennedy, Kennedy, John F. Kennedy\} and \{John McCarthy\}
Candidate Answer Clustering

- Similar candidate answers are grouped together, into a set of clusters.
- The following distance measure is used to measure *dissimilarity* between candidate answers $X$ and $Y$

\[
\text{overlap}(X, Y) = 1 - \frac{|X \cap Y|}{\min(|X|, |Y|)}
\]

- Overlap distance is used in conjunction with a standard single-link agglomerative clustering algorithm, using a threshold distance of 0.0.

**Example**

- **Candidate answers:**
  \{John Kennedy, Kennedy, John F. Kennedy, John McCarthy\}

- **Clusters:** \{John Kennedy, Kennedy, John F. Kennedy\} and \{John McCarthy\}
Candidate Answer Clustering

- Similar candidate answers are grouped together, into a set of clusters.
- The following distance measure is used to measure dissimilarity between candidate answers $X$ and $Y$:

\[
\text{overlap}(X, Y) = 1 - \frac{|X \cap Y|}{\min(|X|, |Y|)}
\]

- Overlap distance is used in conjunction with a standard single-link agglomerative clustering algorithm, using a threshold distance of 0.0.

**Example**

- **Candidate answers:**
  
  \{John Kennedy, Kennedy, John F. Kennedy, John McCarthy\}

- **Clusters:** 
  
  \{John Kennedy, Kennedy, John F. Kennedy\} and \{John McCarthy\}
Scoring

- A score is assigned to each cluster, which is simply the sum of the scores of all candidate answers within it.
- Then, the longest answer within the cluster with highest score is returned.
- In case two clusters happen to have the same score, the tie is settled by returning the cluster with highest search rank.
Scoring

- A score is assigned to each cluster, which is simply the sum of the scores of all candidate answers within it.
- Then, the longest answer within the cluster with highest score is returned.
- In case two clusters happen to have the same score, the tie is settled by returning the cluster with highest search rank.
Scoring

- A score is assigned to each cluster, which is simply the sum of the scores of all candidate answers within it.
- Then, the longest answer within the cluster with highest score is returned.
- In case two clusters happen to have the same score, the tie is settled by returning the cluster with highest search rank.
Outline

1 Introduction
2 Question Classification
3 Passage Retrieval
4 Answer Extraction
5 Evaluation
   • Experimental Setup
   • Experimental Results
6 Contributions
Data set and Evaluation measures

• Data sets
  • A test set of 50 questions collected from potential users of a question answering system

• Evaluation measures
  • Mean reciprocal rank (MRR)
  • For a test set of $N$ questions, MRR is defined as:

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank_i},$$

where $rank_i$ is the rank of the returned answer for the $i$-th question in the test set
  • If the system is unable to answer a given question, the reciprocal rank is deemed as zero
Data set and Evaluation measures

- **Data sets**
  - A test set of 50 questions collected from potential users of a question answering system

- **Evaluation measures**
  - Mean reciprocal rank (MRR)
  - For a test set of $N$ questions, MRR is defined as:

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i},$$

  where $\text{rank}_i$ is the rank of the returned answer for the $i$-th question in the test set
  - If the system is unable to answer a given question, the reciprocal rank is deemed as zero
Outline

1. Introduction
2. Question Classification
3. Passage Retrieval
4. Answer Extraction
5. Evaluation
   5.1 Experimental Setup
   5.2 Experimental Results
6. Contributions
The system was evaluated using two different settings

1. Considering only the first returned answer
2. Considering the first three answers returned by the system

For the first setting, we attained a MRR of **0.68**, i.e., our system correctly answered 34 out of the 50 questions of the test set.

For the second setting, we obtained a mean reciprocal rank of **0.70(6)**, and the system was able to correctly answer 37 questions.
The system was evaluated using two different settings

1. Considering only the first returned answer
2. Considering the first three answers returned by the system

For the first setting, we attained a MRR of 0.68, i.e., our system correctly answered 34 out of the 50 questions of the test set.

For the second setting, we obtained a mean reciprocal rank of 0.70(6), and the system was able to correctly answer 37 questions.
The system was evaluated using two different settings
1. Considering only the first returned answer
2. Considering the first three answers returned by the system

For the first setting, we attained a MRR of 0.68, i.e., our system correctly answered 34 out of the 50 questions of the test set.

For the second setting, we obtained a mean reciprocal rank of 0.70(6), and the system was able to correctly answer 37 questions.
The system was evaluated using two different settings

1. Considering only the first returned answer
2. Considering the first three answers returned by the system

For the first setting, we attained a MRR of 0.68, i.e., our system correctly answered 34 out of the 50 questions of the test set.

For the second setting, we obtained a mean reciprocal rank of 0.70(6), and the system was able to correctly answer 37 questions.
Main Contributions

**Contribution 1**
A state-of-the-art, machine learning-based question classifier, that reaches an accuracy of 95.2% and 90.6% for coarse- and fine-grained classification, respectively, using the standard training and test corpus of Li&Roth.

**Contribution 2**
A novel approach for bootstrapping lexico-syntactic query rewrite patterns, that allow the system to retrieve passages which are very likely to contain an answer, thus easing the answer extraction task.

**Contribution 3**
We showed how to leverage the combination of DBpedia and Wikipedia to extract answers for non-factoid questions, such as `Description:Definition` and `Human:Description`.
Main Contributions (cont.)

Contribution 4
A wide range of strategies to extract candidate answers from relevant passages, ranging from handcrafted regular expressions to automatic machine learning-based named entity recognizers, and including a novel approach, which makes use of WordNet’s hyponymy relations to construct an online exact dictionary matcher.

Contribution 5
A three-step answer selection algorithm, which first filters irrelevant candidate answers, then clusters together the remaining, so that similar candidate answers are considered as one (with a combined score), and finally selects the candidate answer with highest score, using a simple heuristic to settle ties.
End.