Building Knowledge Graphs

Lukas Blübaum
Nick Düsterhus
Monika Werner

University of Paderborn

https://github.com/LukasBluebaum/BKG

July 19, 2018

Overview

Extraction and cleaning of text from Wikipedia Dump

Foundation

Wikipedia dump consisting of plain text without markup, infoboxes or chapter subdivision

- Using regular expressions to remove unnecessary URLs, parentheses, whitespace, null char, symbols
- WikiCleaner class using Writer and Reader Threads for faster cleaning

Extraction and cleaning of text from Wikipedia Dump

```java
package wikicleaner;

import java.util.concurrent.BlockingQueue;

public class Worker implements Runnable{

    private static final Pattern URLS = Pattern.compile("http.*?\\s");

    private static final Pattern PARENTHESES = Pattern.compile("\\(.*?\\)");

    private static final Pattern SYMBOLS = Pattern.compile("[^a-zA-Z0-9. ]");

    private static final Pattern NULLCHAR = Pattern.compile("\0");

    private static final Pattern WHITESPACE = Pattern.compile("\\s+");

    private BlockingQueue<String> readQueue = null;

    private BlockingQueue<String> writeQueue = null;

    public Worker(BlockingQueue<String> readQueue, BlockingQueue<String> writeQueue){
        this.readQueue = readQueue;
        this.writeQueue = writeQueue;
    }

    @Override
    public void run() {
        try {
            while(true) {
                String article = readQueue.take();
                if(article.equals(WikiCleaner.END)){
                    readQueue.put(WikiCleaner.END);
                    writeQueue.put(WikiCleaner.END);
                    break;
                }
                writeQueue.put(cleanArticle(article));
            }
        } catch(InterruptedException e) {
            e.printStackTrace();
        }
    }

    private String cleanArticle(String article) {
        article = NULLCHAR.matcher(article).replaceAll("");
        article = URLS.matcher(article).replaceAll("");
        article = PARENTHESES.matcher(article).replaceAll("");
        article = SYMBOLS.matcher(article).replaceAll("");
        article = WHITESPACE.matcher(article).replaceAll(" ");
        return article;
    }
}
```

```java
public class WikiCleaner {

    protected final static int WORKERAMOUNT = 4;

    protected final static String END = "END";

    private final static int QUEUESIZE = 10000;

    public void cleanWikiDump(File input, File output) {
        BlockingQueue<String> readQueue = new ArrayBlockingQueue<String>(QUEUESIZE);
        BlockingQueue<String> writeQueue = new ArrayBlockingQueue<String>(QUEUESIZE)

        Reader reader = new Reader(readQueue, input);
        Writer writer = new Writer(writeQueue,output);

        Worker[] workers = new Worker[WORKERAMOUNT];
        for(int i = 0; i<workers.length; i++) {
            workers[i] = new Worker(readQueue,writeQueue);
        }

        new Thread(reader).start();
        for(int i = 0; i<workers.length; i++) {
             new Thread(workers[i]).start();
        }
        new Thread(writer).start();
    }

    public static void main(String[] args) {
        File input = new File("resources/enwiki-20171103-pages.tsv");
        File output = new File("resources/out.txt");

        WikiCleaner cleaner = new WikiCleaner();
        cleaner.cleanWikiDump(input, output);
    }


}
```

Coreference Resolution

- Stanford NLP CoreRef to find the representative mention
- Replace pronouns and possessive pronouns

Example

- John met Judy in 1960. He married her during his college year.

  $\Rightarrow$ John met Judy in 1960. John married Judy during John's college year.

Named Entity Recognition

- Requesting Spotlight Demo and parsing JSON response to Entity class

```
public class Entity {

    private ArrayList<String> types = new ArrayList<String>();

    private String uri;

    private String surfaceForm;

    private int offset;

    public Entity(String types, String uri, String surfaceForm, int offset) {
        this.toList(types);
        this.setUri(uri);
        this.setSurfaceForm(surfaceForm);
        this.setOffset(offset);
    }

    private void toList(String t) {
        String[] comma = t.split(",");
        for(String typeLink: comma) {
            String[] type = typeLink.split(":");
            if(type.length == 2) types.add(type[1]);
        }
    }

    public ArrayList<String> getTypes() {
        return types;
    }

    public String getUri() {
        return uri;
    }

    public void setUri(String uri) {
        this.uri = uri;
    }

    public String toString() {
        return "URI: " + uri + "- Type:" + types + "- Surface Form: " + surfaceForm;
    }

    public String getSurfaceForm() {
        return surfaceForm;
    }

    public void setSurfaceForm(String surfaceForm) {
        this.surfaceForm = surfaceForm;
    }

    public int getOffset() {
        return offset;
    }

    public void setOffset(int offset) {
        this.offset = offset;
    }
}
```

Entity Disambiguation

Frameworks

- Spotlight: Integrated disambiguation with two approaches:
  - The information (context) next to a candidate's surface forms is used to find the most likely disambiguation. The best match determines the selection.
  - Weigh words on their ability to disambiguate between the resources

  [Mendes, Jakob, Garca-Silva, Bizer, 2011]
- Fox: AGDISTIS

$\Rightarrow$ Done by given frameworks

Relation Extraction

- Two ways of extracting relations:
  - Fox in 're' mode, performing entity recognition as well as relation extraction
    - Saving triple statements
  - Our RelationExtraction method using OpenIE and Spotlight

Relation Extraction: Own Approach

- Parsing ontology and creating a list of properties
- Write these to a JSON-File
  - Can then be edited manually or automatically
- Using Spotlight for NER, so we can run it concurrent to FOX
- Using Stanford CoreNLP
  - Experimented with different approaches to find relations given two entities (search algorithms on dependency trees, semanticGraph)
  - Decided to use OpenIE

Relation Extraction: OpenIE

- Splitting sentences into shorter fragments, appeal to natural logic to maintain context
- Traverse dependency tree recursively
  [Angeli, Premkumar, Manning, 2015]
- Disadvantage in our case: have to filter for entity to entity relations
- After finding the binary relation (OpenIE Triple) between entities: Map them to DBpedia ontology

Relation Extraction: OpenIE

- Parse DBpedia properties to Java objects
- Search for a valid property (check domain and range)
- Map relation to keywords to find proper property
  - If a keyword String consists of multiple words
    $\Rightarrow$ All words have to be found to map a relation to the property
  - Work with lemmatization
- Also searches for numbers to find entity to literal relations
- Write triple to graph

Relation Extraction 1.1

Example

- *Multiple Sentences (as many as Spotlight can handle):*
  - Obama was born on August 4, 1961, at Kapiolani Medical Center for Women and Children in Honolulu, Hawaii. He graduated from Harvard University.
- Coreference Resolution:
  - Obama was born on August 4 , 1961 , at Kapiolani Medical Center for Women and Children in Honolulu , Hawaii . Obama graduated from Harvard University.
- Binary Relation Extraction:
  - First sentence: Obama - be bear on - August 4 1961. (among others)
  - Second sentence: Obama - graduate from - Harvard University (among others)

Relation Extraction 1.2 - Entity to Literal Relation

Example

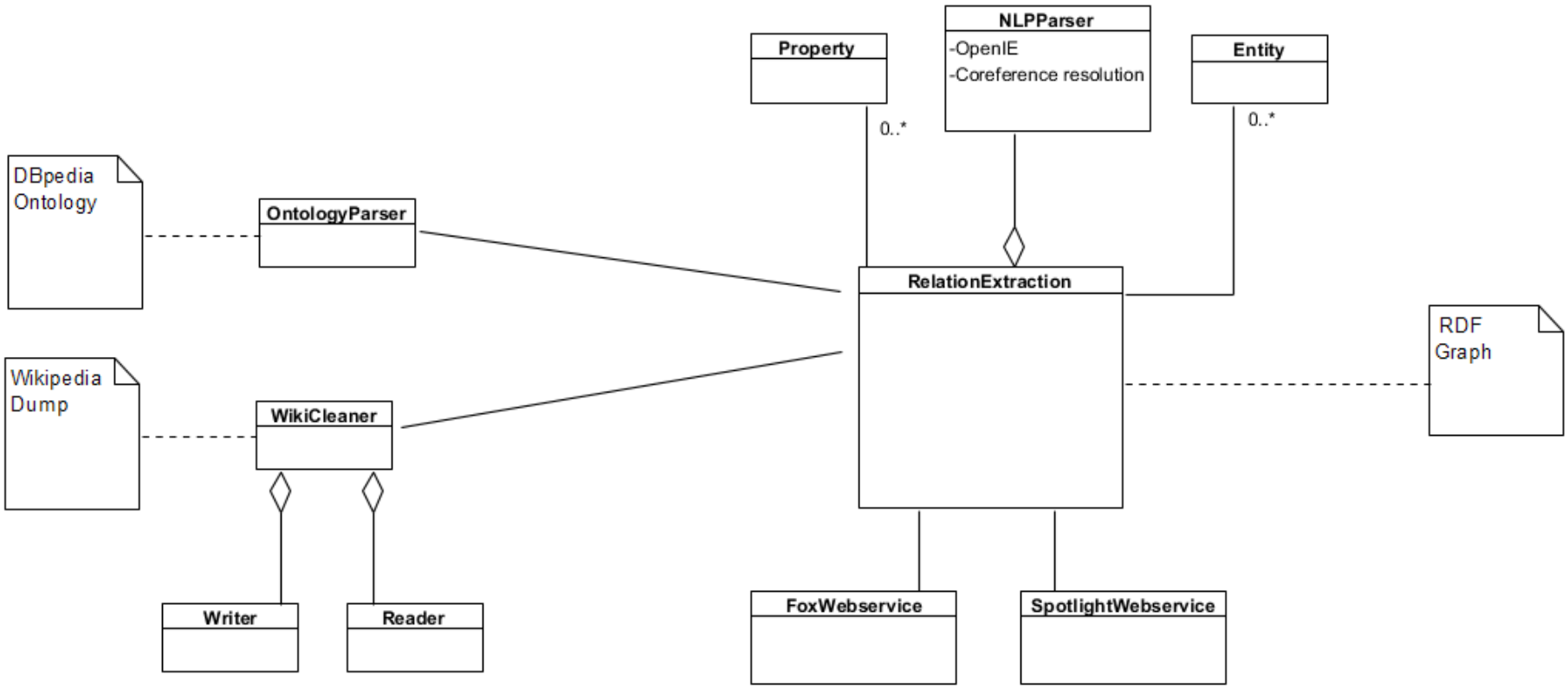
- *Named Entity Recognition and mapping surface form back to sentences*
  - Obama was born on August 4, 1961, at Kapiolani Medical Center for Women and Children in Honolulu, Hawaii. Obama graduated from Harvard University .
- Entity to literal extraction: Obama - be bear on - August 4 1961.
  - Search for entity in subject of the binary relations
    $\Rightarrow$ Obama (Types: Person, ...)
  - Search for literal in object: $\Rightarrow$ detects numbers
  - Search if object contains a date if not extract single literal:
    $\Rightarrow$ detects month + numbers (date found)
  - Convert to date format: $\Rightarrow$ 1961-08-04
  - Map relation to property: keyword: bear $\Rightarrow$ dbo:birthDate
    (Domain: Person - Range: xsd:date)
  - Write dbo:Barack_Obama dbo:birthDate "1961-08-04"^^xsd:date to graph

Relation Extraction 1.3 - Entity to Entity Relation

Example

- *Named Entity Recognition and mapping surface form back to sentences*
  - Obama was born on August 4, 1961, at Kapiolani Medical Center for Women and Children in Honolulu, Hawaii. Obama graduated from Harvard University .
- Entity to entity extraction: Obama - graduate from - Harvard University
  - Search for entity in subject and in object of the binary relations
    $\Rightarrow$ Obama (Types: Person, ...), Harvard University (Type: EducationalInstitution)
  - Iterate over all properties with given domain and range
  - Map relation to property:
    keyword: graduate $\Rightarrow$ dbo:almaMater
  - Write dbo:Barack_Obama dbo:almaMater dbo:Harvard_University to graph

Architecture



Benchmark

- Benchmark class: Given a category and one or more dumps
- Querying for all subjects of the given category
- Can merge multiple n-triple dumps into one file to load it in memory
- Counts and compares all relations with a subject of the given category for the model and dumps

Benchmarking our result

Approach

- Selected category: **Presidents of the United States**
- Compared our result with the given dumps
- Selected 100 triples randomly that were not contained in the dumps and checked them manually
- Only used our relation extraction approach for the benchmarking

Result

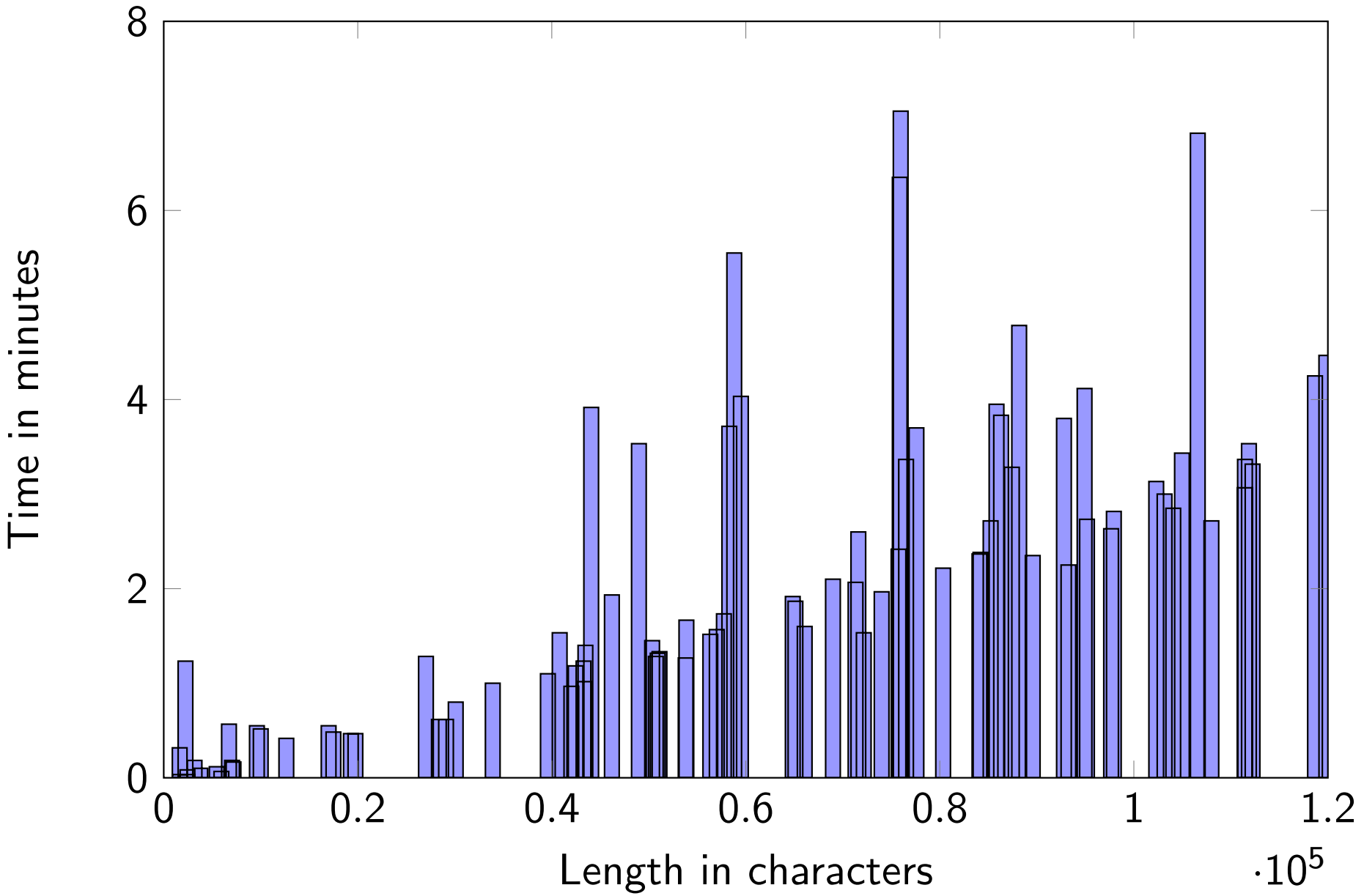
- From the 2810 triples contained in the dumps we also found 556
- Out of the 100 randomly sampled triples 46 were correct.

  $\Rightarrow$ Recall $= 0.197$

  $\Rightarrow$ Precision $= 0.46$

  $\Rightarrow$ F-measure $= 0.27$

Runtime


Time in minutes
0
2
4
6
8
0
0.2
0.4
0.6
0.8
1
1.2
Length in characters
$\cdot 10^5$

Runtime

- Approximately 70-80% of the runtime is the Stanford Coref-Annotator
- Initial runtime due to loading of the models neglected (only once at the start)

Discussion

- Many information can only be found in the infoboxes (Example: height)
- Missing triples due to false disambiguations (Example: History of some party)
- Disambiguations can change depending on the amount of text
- Main difficulty - mapping of properties:
  - We miss context on properties that lack domain and/or range
  - Huge number of carefully selected keywords are needed (Example: appointer as keyword leads to the inverse of the expected result)

References

Pablo N. Mendes, Max Jakob, Andrs Garca-Silva and Christian Bizer (2011)

DBpedia Spotlight: Shedding Light on the Web of Documents

Proceedings of the 7th International Conference on Semantic Systems (I-Semantics) 3.

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning (2015)

Leveraging Linguistic Structure For Open Domain Information Extraction

In Proceedings of the Association of Computational Linguistics (ACL) 2.