Fake News Detection

Albertus Andito

Apr 14, 2021

CONTENTS:

CHAPTER

ONE

KNOWLEDGE-BASED FAKE NEWS DETECTION

This project is the implementation part of my undergraduate final year project, titled "Using Dynamic Knowledge Graph for Fake News Detection". The final report will be available to read here once it is done and submitted.

1.1 Prerequisites to Run

In order to run this project locally, you will need the followings:

1. Docker and Docker Compose.

   This is required for running DBpedia. If you do not already have Docker and Docker Compose, install it from https://docs.docker.com/engine/install/ and https://docs.docker.com/compose/install/.

2. A running DBpedia instance.

   Follow the instructions in https://github.com/dbpedia/virtuoso-sparql-endpoint-quickstart to get DBpedia running locally. Note that it is not strictly required to have the full DBpedia loaded for demonstration purpose, as it could take hours to load. Alternatively, use a smaller collection instead, as also instructed in their documentation.

3. MongoDB

   This is required to store the scraped articles and their extracted triples. If you do not already have it, install it from https://www.mongodb.com/try/download/community.

4. Conda

   This project uses Conda environment to manage Python and its packages. If you do not already have it, install Anaconda (including Conda) from https://www.anaconda.com/products/individual.

5. Node.js

   This project uses Node.js for the User Interface. If you want to run the UI, you will need Node.js. If you do not already have it, install it from https://nodejs.org/en/download/

6. Stanford CoreNLP

   This is used for the triple extraction process. Download it from https://stanfordnlp.github.io/CoreNLP/. Java is required to run this.

7. (Optional) IIT OpenIE

   This can also be used for the triple extraction, as an alternative to Stanford OpenIE. Changes are required in the appropriate places in the code. Download it from https://github.com/dair-iitd/OpenIE-standalone.

8. (Recommended) Guardian Open Platform API Key

   In order to smoothly scrape content from The Guardian, the Guardian API is used. Register for the developer key here: https://bonobo.capi.gutools.co.uk/register/developer.

1.2 Install The Project Locally

1. From the terminal, run:

```
conda env create --file environment.yml
```

2. Once the Conda environment is installed, additionally you will need to install NeuralCoref library which needs to be built locally because the Spacy version used in this project is later than version 2.1.0.

   Run the following commands:

```
git clone https://github.com/huggingface/neuralcoref.git
cd neuralcoref
pip install -r requirements.txt
pip install -e .
```

3. Create .env file by copying the content of .env.default file. Fill out all of the necessary values, or replace the default ones.

4. Install the UI. Go to the ui folder (`cd ui`), and run the following commands:

```
npm install
cp .\src\style\my-theme.less .\node_modules\antd\dist\
cd .\node_modules\antd\dist\
lessc --js my-theme.less ..\..\..\src\style\custom-antd.css
```

1.3 Run The Project Locally

This project consists of 4 components that can be run individually.

1.3.1 Run REST API

The REST API is needed to access the main functionalities of this project. It is also needed when running the UI.

1. Make sure that the local DBpedia and MongoDB are running.

2. Run the Stanford CoreNLP by using this command: (make sure you are running it from the directory where you have the Stanford CoreNLP jar file):

```
java -mx4g -cp "*" edu.stanford.nlp.pipeline.StanfordCoreNLPServer \
-preload tokenize,ssplit,pos,lemma,depparse,natlog,openie \
-port 9000 -timeout 15000
```

3. Run the REST API from the project root folder:

```
conda activate fake-news-detection
python -m api.main
```

4. You should now be able to hit the REST API endpoints on port 5000. You can also access the Swagger UI documentation and demo from http://localhost:5000/apidocs/.

1.3.2 Run User Interface

The UI provides an intuitive way for users to interact with the REST API and the project as a whole.

1. Make sure the REST API is running.
2. Go to the ui folder (`cd ui`), and run the following command: `npm run start`.
3. The UI can be accessed through http://localhost:3000

1.3.3 Run Article Scraper in background

If you want to run the article scraper that runs periodically all the time, you need the followings:

1. Make sure MongoDB is running.
2. From the project's root directory, run `python -m articlescraper.main`

1.3.4 Run Knowledge Graph Updater in background

If you want to have the triples extracted from the recently scraped articles all the time, you need the followings:

1. Make sure the Stanford CoreNLP server is running. See step 2 of Run REST API.
2. From the project's root directory, run `python -m knowledgegraphupdater.kgupdaterrunner`

CHAPTER

TWO

FAKE-NEWS-DETECTION

2.1 articlescraper package

2.1.1 articlescraper.main module

articlescraper.main.**main**()
Driver class for Article Scraper. If run, this will periodically poll the RSS endpoints and scrape articles.

2.1.2 articlescraper.poller module

class articlescraper.poller.**NewsPoller**
Bases: object

NewsPoller polls the RSS endpoints periodically (currently every minute) and uses scrapers to scrape articles.

BBC_RSS_URL = 'http://feeds.bbci.co.uk/news/rss.xml'

GUARDIAN_RSS_URL = 'https://www.theguardian.com/uk/rss'

INDEPENDENT_RSS_URL = 'https://www.independent.co.uk/news/rss'

logger = <RootLogger root (WARNING)>

poll_news_feed(*rss_url*, *scraper*)
Polls the news feed RSS and uses the scraper to scrape articles.

Parameters

- **rss_url** (*str*) – News feed RSS URL
- **scraper** (*articlescraper.ArticleScraper*) – Scraper for the corresponding news website

start()
Starts the periodical polling process. Currently set to every minute.

2.1.3 articlescraper.scrapers module

class articlescraper.scrapers.**ArticleScraper**
Bases: abc.ABC

Base class for article scrapers. It scrapes articles and saves them to DB.

execute(*url*)
Scrapes article and saves them to DB if there is an article.

Parameters **url** (*str*) – Article url

logger = <RootLogger root (WARNING)>

save_to_db(*article*)
Saves article to DB.

Parameters **article** (*dict*) – News article text

abstract scrape(*url*)
Scrapes article.

Parameters **url** (*str*) – Article url

Returns Dictionary of article in the format of {'headlines':..., 'date':..., 'text':..., 'source':...}

Return type dict

class articlescraper.scrapers.**BbcScraper**
Bases: *articlescraper.scrapers.ArticleScraper*

BBC articles scraper.

scrape(*url*)
Scrapes BBC article.

Parameters **url** (*str*) – Article url

Returns Dictionary of article in the format of {'headlines':..., 'date':..., 'text':..., 'source':...}

Return type dict

class articlescraper.scrapers.**GenericScraper**
Bases: *articlescraper.scrapers.ArticleScraper*

Scraper for a generic website.

scrape(*url*)
Scrapes text from <p> tags of the webpage.

Parameters **url** (*str*) – url

Returns Dictionary of article in the format of {'headlines':..., 'date':..., 'text':..., 'source':...}

Return type dict

class articlescraper.scrapers.**GuardianScraper**
Bases: *articlescraper.scrapers.ArticleScraper*

Guardian articles scraper. It requires the GUARDIAN_API_KEY to be set in the .env

scrape(*url*)
Scrapes Guardian article using Guardian Open Platform API (https://open-platform.theguardian.com/.

Parameters **url** (*str*) – Article url

Returns Dictionary of article in the format of {'headlines':..., 'date':..., 'text':..., 'source':...}

Return type dict

scrape_fallback(*url*)
Fallback scraping method if the Guardian API is having issues.

Parameters **url** (*str*) – Article url

Returns Dictionary of article in the format of {'headlines':..., 'date':..., 'text':..., 'source':...}

Return type dict

class articlescraper.scrapers.**IndependentScraper**
Bases: *articlescraper.scrapers.ArticleScraper*

Independent articles scraper.

scrape(*url*)
Scrapes Independent article.

Parameters **url** (*str*) – Article url

Returns Dictionary of article in the format of {'headlines':..., 'date':..., 'text':..., 'source':...}

Return type dict

class articlescraper.scrapers.**Scrapers**
Bases: object

Collection of scrapers.

scrape_text_from_url(*url*, *save_to_db=False*)
Scrapes text from the url given. It uses the generic scraper if the url is not for BBC, Guardian, or Independent.

Parameters

- **url** (*str*) – url
- **save_to_db** (*bool*) – whether to save the scraped text to the database or not. This is only applicable for BBC, Independent, and Guardian URLs.

Returns text scraped from the url

Return type str

2.2 common package

2.2.1 common.entitycorefresolver module

class common.entitycorefresolver.**EntityCorefResolver**
Bases: object

Entity Coreference Resolver (using Spacy and Neuralcoref)

BLACKLIST = ['i', 'me', 'my', 'mine', 'you', 'your', 'yours', 'he', 'him', 'his', 'she

get_coref_clusters(*doc*)
Gets coreference clusters in DBpedia format. It returns a dictionary, where each key is the most representative mention for the cluster, and each value is a set of the other mentions for the cluster Standard pronouns (listed in BLACKLIST) are excluded.

Parameters **doc** (*str*) – a text

Returns dictionary of coreference clusters, as described above

Return type dict

2.2.2 common.kgwrapper module

class common.kgwrapper.**KnowledgeGraphWrapper**
Bases: object

A wrapper for RDF Triple Store (Knowledge Graph) operations.

add_sameAs_relation(*entity_a*, *entity_b*)
Add a sameAs relation between two entities.

Parameters

- **entity_a** (*str*) – a DBpedia resource/entity (must be prepended by "http://dbpedia.org/resource/")
- **entity_b** (*str*) – a DBpedia resource/entity (must be prepended by "http://dbpedia.org/resource/")

check_resource_existence(*resource*)
Checks if a resource exists in the Knowledge Graph, either as a Subject or Object.

Parameters **resource** (*str*) – resource name in DBpedia format (must be prepended by "http://dbpedia.org/resource/")

Returns True if resource exists, False otherwise

Return type bool

check_sameAs_relation(*entity_a*, *entity_b*)
Check the sameAs relation existence between two entities.

Parameters

- **entity_a** (*str*) – a DBpedia resource/entity (must be prepended by "http://dbpedia.org/resource/")
- **entity_b** (*str*) – a DBpedia resource/entity (must be prepended by "http://dbpedia.org/resource/")

check_triple_existence(*subject*, *relation*, *obj*, *transitive=False*)
Checks if a triple exists or not in the Knowledge Graph.

Parameters

- **subject** (*str*) – triple's Subject (must be prepended by "http://dbpedia.org/resource/")
- **relation** (*str*) – triple's Relation/predicate/property (must be prepended by "http://dbpedia.org/ontology/")
- **obj** (*str*) – triple's Object
- **transitive** (*bool*) – whether a check should also be done for entities that are in the sameAs relation with the subject

Returns True if triple exists, False otherwise

Return type bool

check_triple_object_existence(*triple*, *transitive=False*)
Checks if a triple exists or not in the Knowledge Graph.

Parameters

- **triple** (triple.Triple) – a triple of type triple.Triple
- **transitive** (*bool*) – whether a check should also be done for entities that are in the sameAs Relation with the Subject

Returns True if triple exists, False otherwise

Return type bool

check_triple_object_opposite_relation_existence(*triple*, *transitive=False*)
Checks if a triple with the opposite relation (Objects-Relation-Subject) exists or not in the Knowledge Graph. To be able to do this, the original Object (which becomes the Subject in this case) needs to be a DBpedia resource/ entity. If it is a literal String, then the check is not performed (because Subject always needs to be a resource).

Parameters

- **triple** (triple.Triple) – a triple of type triple.Triple
- **transitive** (*bool*) – whether a check should also be done for entities that are in the sameAs relation with the subject

Returns True if the opposite relation triple exists, False otherwise

Return type bool

delete_triple(*subject*, *relation*, *obj*)
Deletes triple from the knowledge graph.

Parameters

- **subject** (*str*) – triple's Subject (must be prepended by "http://dbpedia.org/resource/")
- **relation** (*str*) – triple's Relation/predicate/property (must be prepended by "http://dbpedia.org/ontology/")
- **obj** (*str*) – triple's Object

delete_triple_object(*triple*, *transitive=False*)
Deletes triple from the knowledge graph.

Parameters

- **triple** (triple.Triple) – a triple of type triple.Triple
- **transitive** (*bool*) – whether the delete should also be done for entities that are in the sameAs relation with the Subject

get_entity(*subject*, *transitive=False*)
Get all triples that the Subject or entity has in the Knowledge Graph.

Parameters

- **subject** (*str*) – triple's Subject
- **transitive** (*bool*) – whether a check should also be done for entities that are in the sameAs relation with the Subject

Returns list of all triples that the Subject has in the knowledge graph, or None if the Subject doesn't exist

Return type list or None

get_relation_triples(*subject*, *obj*, *transitive=False*)
Get triples from Knowledge Graph that have the given Subject and Relation.

Parameters

- **subject** (*str*) – triple's Subject (must be prepended by "http://dbpedia.org/resource/")
- **obj** (*str*) – triple's Object
- **transitive** (*bool*) – whether a check should also be done for entities that are in the sameAs Relation with the Subject

Return type list or None

get_same_entities(*entity*)
Return DBpedia entities that have the owl:sameAs relation with the input. They should be considered as the same entity.

Parameters **entity** (*str*) – a DBpedia resource/entity (must be prepended by "http://dbpedia.org/resource/")

Returns a list of DBpedia entities that have the owl:sameAs relation with the input

Return type list

get_triples(*subject*, *relation*, *transitive=False*)
Get triples from Knowledge Graph that have the given Subject and Relation.

Parameters

- **subject** (*str*) – triple's Subject (must be prepended by "http://dbpedia.org/resource/")
- **relation** – triple's Relation (must be prepended by "http://dbpedia.org/ontology/")
- **transitive** (*bool*) – whether a check should also be done for entities that are in the sameAs Relation with the Subject

Returns list of triples (triple.Triple) that exist in the knowledge graph, or None if such triple doesn't exist

Return type list or None

insert_triple(*subject*, *relation*, *obj*)
Inserts triple to the knowledge graph

Parameters

- **subject** (*str*) – triple's Subject (must be prepended by "http://dbpedia.org/resource/")
- **relation** (*str*) – triple's Relation/predicate/property (must be prepended by "http://dbpedia.org/ontology/")
- **obj** (*str*) – triple's Object

insert_triple_object(*triple*)
Inserts triple to the Knowledge Graph.

Parameters **triple** (triple.Triple) – a triple of type triple.Triple

remove_sameAs_relation(*entity_a*, *entity_b*)
Remove the sameAs relation between two entities.

Parameters

- **entity_a** (*str*) – a DBpedia resource/entity (must be prepended by "http://dbpedia.org/resource/")
- **entity_b** (*str*) – a DBpedia resource/entity (must be prepended by "http://dbpedia.org/resource/")

2.2.3 common.triple module

class common.triple.**Triple**(*subject=None*, *relation=None*, *objects=None*)

Bases: object

Class representation of a Triple, consisting of Subject, Relation, and Objects.

Parameters

- **subject** (*str*) – Subject of the triple
- **relation** (*str*) – Relation or predicate of the triple
- **objects** (*list*) – list of Objects of the triple

static from_dict(*dic*)

Creates a Triple object from the given dictionary.

Parameters **dic** (*dict*) – dictionary representation of a triple {"subject":..., "relation":..., "objects": [...]}

Returns the Triple object

Return type *triple.Triple*

static from_json(*json_data*)

Creates a Triple object from the given JSON string.

Parameters **json_data** (*str*) – JSON string representation of a triple '{"subject":..., "relation":..., "objects": [...]}'

Returns the Triple object

Return type *triple.Triple*

to_dict()

Returns a dictionary representation of the Triple.

Returns dictionary representation of the Triple

Return type dict

to_json()

Returns a JSON representation of the Triple.

Returns JSON representation of the Triple

Return type str

2.2.4 common.tripleextractors module

class common.tripleextractors.**IITExtractor**
Bases: *common.tripleextractors.TripleExtractor*

Triple extraction using IIT OpenIE (https://github.com/vaibhavad/python-wrapper-OpenIE5).

extract(*document*)
Extract SPO triples from document.

Parameters **document** (*str*) – document

Returns list of triples

Return type list

class common.tripleextractors.**StanfordExtractor**
Bases: *common.tripleextractors.TripleExtractor*

Triple extraction using Stanford OpenIE (https://github.com/Lynten/stanford-corenlp).

extract(*document*)
Extract SPO triples from document

Parameters **document** (*str*) – document

Returns list of triples

Return type list

props = {'annotators': 'openie', 'outputFormat': 'json', 'pipelineLanguage': 'en'}

class common.tripleextractors.**TripleExtractor**
Bases: abc.ABC

Abstract class of Triple Extractor

abstract extract(*document*)
Extract SPO triples from document

Parameters **document** (*str*) – document

Returns list of triples

Return type list

2.2.5 common.tripleproducer module

class common.tripleproducer.**TripleProducer**(*extractor_type=None*, *extraction_scope=None*)
Bases: object

TripleProducer produces SPO triples from sentences, where the Subjects and Objects are linked to DBpedia entity resources. and the Predicates (Relations) are linked to DBpedia Ontology, whenever possible.

Parameters

- **extractor_type** (*str*) – SPO extractor tool. 'stanford_openie' or 'iit_openie' can be chosen as the SPO extractor, defaults to 'stanford_openie' for now.
- **extraction_scope** (*str*) – The scope of the extraction, deciding whether it should include only relations between 'named_entities', 'noun_phrases', or 'all', defaults to 'named entities' for now.

FALCON_URL = 'https://labs.tib.eu/falcon/api?mode=long'

SPOTLIGHT_URL = 'https://api.dbpedia-spotlight.org/en/annotate?'

convert_relations(*all_triples*)
Prepend all relations with "http://dbpedia.org/ontology/", even if the relation doesn't exist in DBpedia.

Parameters **all_triples** (*list*) – list of list of triples (top-level list represents sentences)

Returns list of list of triples, where relations have been converted

Return type list

coref_resolution(*spacy_doc*)
Perform coreference resolution on the document using neuralcoref.

Parameters **spacy_doc** (*spacy.tokens.Doc*) – document

Returns Unicode representation of the doc where each corefering mention is replaced by the main mention in the associated cluster.

Return type str

extract_triples(*sentences*)
Extract triples from document using the implementation of TripleExtractor.

Parameters **sentences** (*list*) – list of document sentences

Returns a list of list of raw triples (top-level list represents sentences)

Return type list

filter_in_named_entities(*spacy_doc*, *all_triples*)
Filter in only triples where the Subject and Object are both named entities.

Parameters

- **spacy_doc** (*spacy.tokens.Doc*) – spacy document
- **all_triples** (*list*) – a list of list of triples (top-level list represents sentences)

Returns a list of list of triples in which the Subjects and Objects are all named entities

Return type list

filter_in_noun_phrases(*spacy_doc*, *all_triples*)
Filter in only triples where the Subject and Object are both (in the list of) noun phrases. The list of noun phrases is generated from the spacy document.

Parameters

- **spacy_doc** (*spacy.tokens.Doc*) – spacy document
- **all_triples** (*list*) – a list of list of triples (top-level list represents sentences)

Returns a list of list of triples in which the Subjects and Objects are all noun phrases

Return type list

filter_noun_phrases(*all_triples*)
Filter in only triples where the Subject and Object are both noun phrases. Whether subject or object is a noun phrase or not is determined by making each subject and object into Spacy document, and then check if it is a noun phrase or not, or if it is a noun or not. Due to the Spacy document being created for all subjects and objects, this method is slower than the other method above.

Parameters **all_triples** (*list*) – a list of list of triples (top-level list represents sentences)

Returns a list of list of triples in which the Subjects and Objects are all noun phrases

Return type list

lemmatise_relations(*spacy_doc*, *all_triples*)
Lemmatise relations to their base forms.

Parameters

- **spacy_doc** (*spacy.tokens.Doc*) – spacy document
- **all_triples** (*list*) – list of list of triples (top-level list represents sentences)

Returns list of list of triples where relations have been lemmatised

Return type list

link_relations(*sentences*, *all_triples*)
Link relations to DBpedia Ontology using Falcon (https://labs.tib.eu/falcon/), if available.

Parameters

- **sentences** (*list*) – list of document sentences
- **all_triples** (*list*) – list of list of triples (top-level list represents sentences)

Returns new list of list of triples with dbpedia relations

Return type list

produce_triples(*document*, *extraction_scope=None*)
Produce triples extracted from the document that are processed through the pipeline. The triples produced are in the form of:

[(sentence, [{'subject': subject, 'relation': relation, 'object':[object_1, object2, ...], ...]), ...]

The Subjects and Objects are linked to DBpedia entity resources, while the Relations are linked to DBpedia Ontology, whenever possible.

Parameters

- **document** (*str*) – raw texts of document
- **extraction_scope** (*str*) – The scope of the extraction, deciding whether it should include only relations between 'named_entities', 'noun_phrases', or 'all. Defaults to the extraction_scope member variable.

Returns a list of tuples, of sentence and its triples, as explained

Return type list

remove_empty_components(*all_triples*)
Remove triple with empty components (Subject, Relation, or Object).

Parameters **all_triples** (*list*) – list of triples

Returns list of triples without empty components

Return type list

remove_stopwords(*all_triples*)
Remove stopwords from individual Subject and Object. Currently, this is only done when the extraction scope is 'named_entities' or 'noun_phrases'.

Parameters **all_triples** (*list*) – a list of list of triples (top-level list represent sentences)

Returns a list of list of triples in which stopwords have been removed from the Subjects and Objects

Return type list

spot_entities_with_context (*document*, *all_triples*)
Convert Subjects and Objects to DBpedia entity resources (i.e. in the form of 'http://dbpedia.org/resource/...'), if they are spotted using DBpedia Spotlight API.

Parameters

- **document** (*str*) – document
- **all_triples** (*list*) – a list of list of triples (top-level list represents sentences)

Returns a list of list of triples where the Subjects and Objects have been replaced with DBpedia entity resources, if possible

Return type list

spot_local_entities (*all_triples*)
Prepend all subjects with "http://dbpedia.org/resource/" if the subject hasn't been spotted yet as a DBpedia entity. For objects, check first if such entity exists in the local knowledge graph (may not exist in DBpedia Spotlight KG). If yes, convert the object to the DBpedia resource format.

Parameters **all_triples** (*list*) – list of list of triples (top-level list represents sentences)

Returns list of list of triples, where all subjects are dbpedia resources and objects that exist in the local KG also converted to dbpedia resources

Return type list

2.2.6 common.utils module

common.utils.**camelise** (*sentence*)
Util function to convert words into camelCase

Parameters **sentence** (*str*) – sentence

Returns camelCase words

Return type str

common.utils.**convert_to_dbpedia_ontology** (*predicate*)
Converts a relation or predicate string to a DBpedia format (http://dbpedia.org/ontology/).

Parameters **predicate** (*str*) – relation string

Returns DBpedia ontology string

Return type str

common.utils.**convert_to_dbpedia_resource** (*resource*)
Converts a resource string to a DBpedia format (http://dbpedia.org/resource/).

Parameters **resource** (*str*) – resource string

Returns DBpedia resource string

Return type str

2.3 factcheckers package

2.3.1 factcheckers.exactmatchfactchecker module

class factcheckers.exactmatchfactchecker.**ExactMatchFactChecker**
Bases: *factcheckers.factchecker.FactChecker*

An Exact Match Fact Checker, where a truthfulness is decided only by finding the exact match of the triples.

exact_fact_check(*triple*, *transitive=False*)
Checks for the triple existence and conflicts

Parameters

- **triple** (triple.Triple) – triple to be checked
- **transitive** (*bool*) – whether a check should also be done for entities that are in the sameAs relation with the subject

Returns a tuple of its result and list of supporting triples

Return type tuple

fact_check(*article*, *extraction_scope*)
Fact check the given text, by first extracting the triples and then finding the exact match of the triples in the knowledge graph.

Parameters

- **article** (*str*) – article text
- **extraction_scope** (*str*) – The scope of the extraction, deciding whether it should include only relations between 'named_entities', 'noun_phrases', or 'all'.

Returns a list of fact check result (sentence, {triples: their results})

Return type list

fact_check_triples(*triples*, *transitive=False*)
Fact check the given triples, by finding the exact match of the triples in the knowledge graph.

Parameters

- **triples** (*list*) – list of triples of type triple.Triple
- **transitive** (*bool*) – whether a check should also be done for entities that are in the sameAs relation with the subject

Returns a list of fact check result (sentence, {triples: their results})

Return type tuple

2.3.2 factcheckers.factchecker module

class factcheckers.factchecker.**FactChecker**
Bases: abc.ABC

Abstract class of a Fact Checker.

abstract fact_check(*article*, *extraction_scope*)
Abstract method of fact-checking.

Parameters

- **article** (*str*) – article text
- **extraction_scope** (*str*) – The scope of the extraction, deciding whether it should include only relations between 'named_entities', 'noun_phrases', or 'all'.

Returns a list of fact check result (sentence, {triples: their results})

Return type list

2.3.3 factcheckers.nonexactmatchfactchecker module

class factcheckers.nonexactmatchfactchecker.**NonExactMatchFactChecker**
Bases: *factcheckers.factchecker.FactChecker*

A Non Exact Match Fact Checker. It considers the opposite relation (Object-Relation-Triple) of every triple. It also considers the synonyms of the relation, at the moment using WordNet.

check_relation_synonyms(*triple*)
Check the existence of triples, in which the relation is a synonym of the relation of the inputted triple. Once a triple is found, it is returned without checking the other synonyms. Note that it also checks the opposite relation (Object - Relation - Subject).

Parameters **triple** (triple.Triple) – triple of type triple.Triple

Returns the triple, if found in the knowledge graph. None, otherwise.

Return type tuple

fact_check(*article*, *extraction_scope*)
Fact check the given text, by first extracting the triples and then infer the existence of the triples in the knowledge graph. The inference is done by finding the exact match, finding the triples of the opposite relation (Object - Relation - Subject), finding the triples where subject or object are corefering entities, and finding the triples with similar (based on synonymy) relations.

Parameters

- **article** (*str*) – article text
- **extraction_scope** (*str*) – The scope of the extraction, deciding whether it should include only relations between 'named_entities', 'noun_phrases', or 'all'.

Returns a list of fact check result (sentence, {triples: their results})

Return type list

fact_check_triples(*triples*)
Fact check the given triples, by by first extracting the triples and then infer the existence of the triples in the knowledge graph. The inference is done by finding the exact match, finding the triples of the opposite relation (Object - Relation - Subject), and finding the triples with similar (based on synonymy) relations.

Parameters **triples** (*list*) – list of triples of type triple.Triple

Returns a list of fact check result (sentence, {triples: their results})

Return type list

non_exact_fact_check(*original_triple*, *entity_clusters=[]*)
Check whether the triple or the "related triples" exist in the knowledge graph or not. Related triples are:

- triples with the opposite relation (Object - Relation - Subject)
- triples with subject or object replaced with the corefering entity (if entity_clusters is not None)
- triples with relation replaced with its synonyms
- triples with same subject and object, but different relation.

Parameters

- **original_triple** (triple.Triple) – triple extracted from the text or inputted
- **entity_clusters** (*dict*) – dictionary of entity coreference clusters

Returns a tuple of the triple and its existence, if found in the knowledge graph. None, otherwise.

Return type tuple

2.4 knowledgegraphupdater package

2.4.1 knowledgegraphupdater.kgupdater module

class knowledgegraphupdater.kgupdater.**KnowledgeGraphUpdater**(*auto_update=None*)
Bases: object

A Knowledge Graph Updater, which consists of all functionalities related to updating the knowledge graph.

Parameters **auto_update** (*bool*) – whether the knowledge graph will be updated automatically once triples are extracted, or wait for user confirmation

delete_all_knowledge_from_article(*article_url*)
Delete triples that are extracted from an article, from the knowledge graph. Triples from the articles must have been extracted beforehand and stored in DB.

Parameters **article_url** (*str*) – URL of the article source

delete_article_pending_knowledge(*article_url*, *triples*)
Deletes pending article triples, that have been added to the knowledge graph, from the database.

Parameters

- **article_url** (*str*) – URL of the article source
- **triples** (*list*) – list of pending triples to be deleted

delete_knowledge(*triples*)
Remove triples from knowledge graph.

Parameters **triples** (*list*) – list of triples (in the form of dictionaries)

extract_new_article(*url*, *extraction_scope='noun_phrases'*, *kg_auto_update=False*)
Scrape an article given the URL and extract the triples from the article.

Parameters

- **url** (*str*) – URL of article whose triples are going to be extracted

- **extraction_scope** (*str*) – The scope of the extraction, deciding whether it should include only relations between 'named_entities', 'noun_phrases', or 'all.
- **kg_auto_update** (*bool*) – whether the non-conflicting triples are added to the knowledge graph or not.

get_all_articles()
Returns all articles' URLs, headlines, and dates, in the form of dictionaries.

Returns list of dictionaries of articles

Return type list

get_all_articles_knowledge()
Returns all triples that have been extracted from all scraped articles.

Returns list of all triples extracted from all articles

Return type list

get_all_extracted_articles()
Returns all articles' URLs, headlines, and dates, whose triples have been extracted, in the form of dictionaries.

Returns list of dictionaries of articles

Return type list

get_all_pending_knowledge()
Returns all pending triples (that are not currently in the knowledge graph) for all scraped articles.

Returns list of all pending triples extracted from all articles

Return type list

get_all_unresolved_corefering_entities()
Returns all unresolved corefering entities extracted from articles.

Returns list of unresolved corefering entities

Return type list

get_article_knowledge(*article_url*)
Returns all triples of that have been extracted from the specified article, regardless of whether the triple has been added to the knowledge graph or not.

Parameters **article_url** (*str*) – URL of the article source

Returns list of triples extracted from the article if exist, or None

Return type list or None

get_article_pending_knowledge(*article_url*)
Returns all pending triples (that are not currently in the knowledge graph) for the specified article.

Parameters **article_url** (*str*) – URL of the article source

Returns list of pending triples extracted from the article if exist, or None

Return type list or None

get_entity(*subject*)
Returns all triples that has the subject parameter as the subject.

Parameters **subject** (*str*) – subject in DBpedia format

Returns list of Triples

Return type list

get_knowledge(*subject*, *relation*, *objects=None*)
Returns triple from the knowledge graph that has the given conditions. If objects are given, it will return back the triple if it exist in the knowledge graph.

Parameters

- **subject** (*str*) – subject of the triple in DBpedia format
- **relation** (*str*) – relation of the triple in DBpedia format
- **objects** (*list or str*) – (list) of objects of the triple in DBpedia format, this parameter is optional

Returns list of triples

Return type list

insert_all_nonconflicting_knowledge(*article_url*)
Insert non-conflicting triples of an article to the knowledge graph.

Parameters **article_url** (*str*) – URL of the article source

insert_articles_knowledge(*articles_triples*)
Insert triples that are related to an article to the knowledge graph. If the triple has conflict, mark the conflict as 'added' in the db. If the triple doesn't exist on db, add the triple to the db.

Parameters **articles_triples** (*dict*) – dictionary of article triples

insert_entities_equality(*entity_a*, *entity_b*)
Resolve two entities as the same.

Parameters

- **entity_a** (*str*) – a DBpedia resource/entity (must be prepended by "http://dbpedia.org/resource/")
- **entity_b** (*str*) – a DBpedia resource/entity (must be prepended by "http://dbpedia.org/resource/")

insert_knowledge(*triple*, *check_conflict*)
Insert triple to the knowledge graph.

Parameters

- **triple** (*dict*) – the triple to be inserted to the knowledge graph
- **check_conflict** (*bool*) – whether it should check for conflicts first or not.

Returns list of conflicts if there are conflicts and check_conflict is True, None otherwise

Return type list or None

update_missed_knowledge(*kg_auto_update=None*, *extraction_scope=None*)
Extract triples from stored articles whose triples has not been extracted yet, and save the triples to the DB. If the auto_update mode is active, the non-conflicting triples are added automatically to the knowledge graph.

Parameters

- **kg_auto_update** (*bool*) – an optional parameter that sets whether the non-conflicting triples are added to the knowledge graph or not. This will only matter if the auto_update field is False. If the auto_update field is already True, then this parameter will not be looked upon.

- **extraction_scope** (*str*) – The scope of the extraction, deciding whether it should include only relations between 'named_entities', 'noun_phrases', or 'all.

2.4.2 knowledgegraphupdater.kgupdaterrunner module

knowledgegraphupdater.kgupdaterrunner.**main**()
A runner for the Knowledge Graph Updater to keep extracting triples from new scraped articles.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

F

G

I

K

L

M

N

P

R

S

T

U