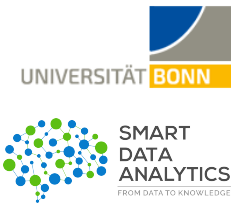
German University in Cairo
Faculty of Media Engineering and Technology



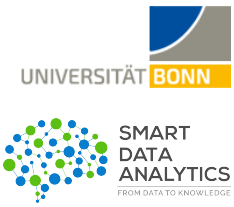
Universitat Bonn
Smart Data Analytics

A Benchmark Platform for Fact Validation Algorithms

Bachelor Thesis

Author:	Omar Ashraf Hussein Sallam
Supervisors:	Prof. Dr. Jens Lehmann
	M.Sc. Diego Esteves
Submission Date:	28 August, 2018

German University in Cairo
Faculty of Media Engineering and Technology



Universitat Bonn
Smart Data Analytics

A Benchmark Platform for Fact Validation Algorithms

Bachelor Thesis

Author:	Omar Ashraf Hussein Sallam
Supervisors:	Prof. Dr. Jens Lehmann
	M.Sc. Diego Esteves
Submission Date:	28 August, 2018

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

---

Omar Ashraf Hussein Sallam
28 August, 2018

Acknowledgments

First of all, I'd like to start by thanking my family for their continuous support, love and guidance in whatever I pursue.

This work would not have been possible without the supervision of M.Sc. Diego Esteves. He provided me with encouragement and patience throughout the duration of this thesis. Also, I would like to thank him for the huge amount of knowledge and experience he provided which helped me a lot. I wish him all the best in his future.

Finally, I would like to thank my friends for supporting me throughout the whole duration of thesis.

Abstract

With the web evolution in the last decade, the amount of misleading and invalid information on web is increasing rapidly. Getting to know which of the data on the web is accurate and which is misleading is becoming a harder hard not only for humans but for machines too. This can affect many businesses costing them billions of dollars annually. Research committees that uses information extraction methods from RDF graphs and web sources to investigate different research fields are having problems in their studies because of the low veracity of the data. This can lead to insufficient investigation and low potential for these research fields.

This thesis provides an overview of the related work in the field of fact-checking and web trustworthiness including top conference submitted work and challenges. It also presents a benchmark platform proposal for the fact-checking algorithms in the form of a poster. It provides a proposal for a fact-checking pipeline along with implementation details for some of its phases. Finally, it includes the results for a suggested model in the fact-checking pipeline. This model classifies whether for a claim c and a proof p, if p supports c or not.

Contents

Acknowledgments **V**

Abstract **VII**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Aim of The Thesis . . . 1
1.3 Thesis Structure . . . 2

2 Background **3**
2.1 Machine learning . . . 3
2.1.1 Supervised Learning . . . 3
2.1.2 Unsupervised Learning . . . 4
2.1.3 Scikit Learn . . . 4
2.2 Classification . . . 4
2.2.1 Support Vector Machines . . . 4
2.2.2 Naive Bayes . . . 5
2.2.3 Decision Trees . . . 5
2.3 Natural Language Processing . . . 5
2.3.1 Named Entity Recognition . . . 6
2.3.2 Stanford-NER . . . 6
2.3.3 Bag of words . . . 7
2.3.4 Lemmatization and Stemming . . . 7
2.3.5 Word2Vec . . . 7
2.4 Related Work . . . 7

3 ISWC 17 **11**
3.1 ISWC 17 challenge . . . 11
3.1.1 Dataset . . . 12
3.1.2 Approaches . . . 12
3.1.3 Results . . . 15
3.2 TAA . . . 16
3.2.1 Approach . . . 16
3.2.2 Results . . . 17

4 Methodology **23**
4.1 Fact Validation Dataset Preprocessing . . . 23
4.1.1 FactBench Dataset . . . 24
4.1.2 Preprocessing . . . 24
4.1.3 Searching Web For Evidences . . . 25
4.1.4 Finding Topic Terms . . . 26
4.2 Claims & Proofs Model . . . 26
4.2.1 Google Relation Extraction Dataset . . . 26
4.2.2 Preprocessing . . . 27
4.2.3 Features and Labels Extraction . . . 28
4.2.4 Labels Extraction . . . 29
4.2.5 Training . . . 31
4.3 Fact Checking Benchmark Poster . . . 31

5 Results **33**
5.1 Confusion Matrix . . . 33
5.2 Performance Metrics . . . 34
5.2.1 Accuracy . . . 34
5.2.2 Precision . . . 34
5.2.3 Recall . . . 34
5.2.4 f_1 score . . . 35

6 Conclusion & Future Work **39**
6.1 Conclusion . . . 39
6.2 Future Work . . . 39

Appendix **41**

A Lists **42**
List of Abbreviations . . . 42
List of Figures . . . 43
List of Tables . . . 44

References **47**

Chapter 1

Introduction

1.1 Motivation

Huge amount of user-generated data exists in different forms (e.g., web pages, tweets, KBs, multimedia, etc.). With that huge amount of data generated everyday, the data is becoming more noisy and its quality can be much worse, Its hard to distinguish the facts from the misleading data. Although properties like volume, velocity and variety are important to characterize big data, data quality or in other words data veracity is increasingly recognized [25]. Data veracity refers to the biases, noise and abnormality in data.

The Data Warehousing Institute (TDWI) estimates that erroneous data costs US businesses 600 billion dollars annually [5]. On the web, 58% of the documents available on the Internet are in XML format, among these there is only 25% that refers to a DTD/XSD, of which just a third which is valid [5, 11]. If we pick a random XML document, there is a 14.6% chance of it being not well-formed. Also, just 10% of the well-formed documents are valid [11].

Validity is rare on web which creates a huge problem. For that, this problem is studied in different research communities around and work is being done under the name of fact-checking, truth-finding and trustworthiness to solve this problem. In fact-checking a claim is given as input to some algorithm that evaluates the trustworthiness of that claim based on some information sources and its common sense rules [22]. Fact-checking algorithms have a wide applications domain from validating different types of news to validating facts in KBs and providing sources for them [8, 12].

1.2 Aim of The Thesis

The aim of the thesis is to explore related work in the field of fact-checking and go through an exploratory analysis of the work submitted in International Semantic Web

Conference (ISWC) 2017. Moreover, we create a poster about a benchmark platform for fact-checking algorithms. Finally, a proposal of a fact-checking pipeline is suggested. Then, we start implementing some of its components and including evaluation results.

1.3 Thesis Structure

The thesis is organized as follows. Chapter 2 gives an overview of techniques and technologies existing in the research field. Also, It reviews some of the related work in the field of fact-checking and web trustworthiness. Chapter 3 includes overview of the ISWC challenge and some of the papers submitted in the field of fact-checking in this conference too. Chapter 4 describes a proposal of fact-checking pipeline and the implementation for a part of that pipeline. Also, it includes datasets, methods and libraries used in the pipeline. Finally, it includes a proposal for a benchmark platform for the fact-checking algorithms in the form of a poster. Chapter 5 includes results for proof and claims model presented in chapter 4. Finally, Chapter 6 is for the conclusion of the thesis work and future work.

Chapter 2

Background

This chapter presents an overview of the techniques and technologies existing in the research field nowadays. This chapter also reviews related work in the field of fact-checking.

2.1 Machine learning

Machine Learning (ML) is the field of study that gives the computer the ability to learn without being explicitly programmed. A more engineering-oriented definition: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. Tom Mitchell, 1997. Machine Learning can be useful in many tasks: problems for which existing solutions require a long list of complex rules, complex problems for which there isn't a good solution that is known at all and also, for observing and getting more insights about the problem by looking into what the ML algorithms learned. ML systems can be classified into two main categories depending on whether or not they are trained with human supervision. These two main categories are Supervised Learning and Unsupervised Learning.

2.1.1 Supervised Learning

In supervised learning, the training data for the models includes labels which represent the desired solution that should be predicted. The model tries to find a relation between labels and features fed to it. The two typical tasks for supervised learning are:

Regression: The task of regression is used for predicting quantities such as amounts and sizes. For example, given the prices of houses in a certain district, it can predict the price for a new house given its location compared to other houses and more other features.

Classification: The task of classification is used to categorize an input x into a class from a set of classes s. For example, the task of categorizing an email if it's a spam or not is a classification problem.

2.1.2 Unsupervised Learning

Unlike supervised learning, in the unsupervised learning the dataset fed to the model doesn't include labels(solutions) for each instance. The system tries to learn by itself without guidance from the programmer. The most famous tasks in unsupervised learning are:

Clustering is the task of grouping similar entities together. The goal of this is to find similarities in the data. This could be useful for a task of grouping website visitors into small groups, so the owner of the website could learn more about these small groups and enhance the user experience for each of these groups.

Visualization can output a 2D/3D representation of complex and unlabeled data input. These algorithms try to preserve as much structure as they can trying to keep separate clusters from each other.

2.1.3 Scikit Learn

Scikit Learn[1] [23] is Python module for machine learning built on top of SciPy. The project was started in 2007 by David Cournapeau as a Google Summer of Code project. We have used this module for using machine learning models to solve some of the subtasks in the fact-checking pipeline.

2.2 Classification

This section gives an overview of three classifiers: Support Vector Machines, Naive Bayes and Decision Trees. All of them were used in our fact-checking pipeline.

2.2.1 Support Vector Machines

Support Vector Machines (SVM) are supervised machine learning models for classification and regression problems. The idea behind SVM is simple, the algorithm creates a a line called the decision line that separates data points from different classes of each other. The goal of the algorithm is not just to find this line but also it tries to keep it as far as possible from the closest training instances. This is shown in figure 2.1. SVM can be very useful in the case of classification of complex problem but with small/medium size datasets. Also, we should avoid using SVM in the case of having number of features much greater than the number of samples.

[1] http://scikit-learn.org/

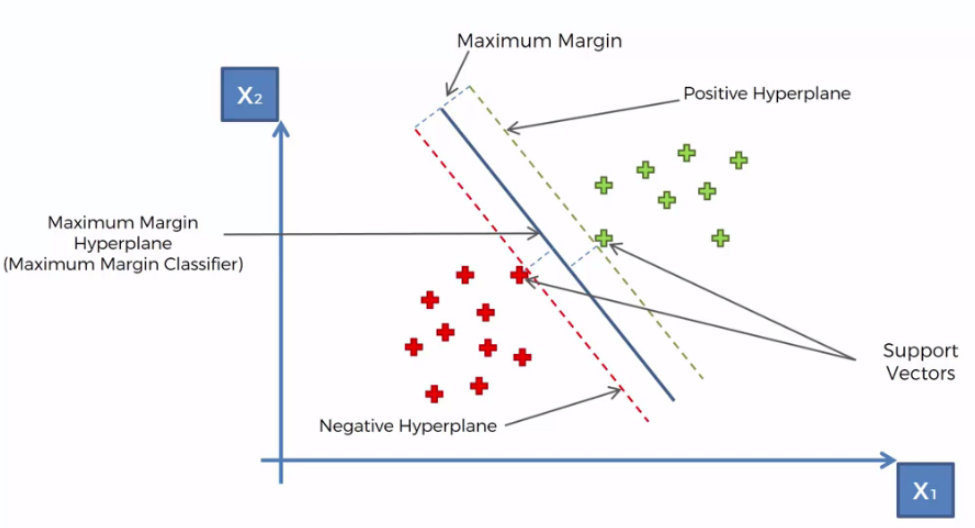


Figure 2.1: Support Vector Machines [1]

2.2.2 Naive Bayes

Naive Bayes classifier calculates the probabilities for every factor (class), Then it selects the outcome with the highest probability. This can be represented mathematically as follows. if we have a certain event E and test actors t1,t2,t3, etc. We first calculate P(t1|E), P(t2|E) ..., where P(A|B) is the probability of A given that B happens. Then, the algorithm selects the test actor t with the highest probability value. The Naive Bayes algorithm can be used for real time prediction, spam filtering, recommendation systems and many more applications.

2.2.3 Decision Trees

The idea behind Decision Trees is that the algorithm tries to divide the training dataset into small sets based on their features until it reaches a small set that contains data which falls under one label. Each feature becomes a root(parent) node and the leaf(child) nodes that represent the outcomes. The decision on which feature to split on is made based on resultant entropy reduction or information gain from the split. The tree representation is shown in figure 2.2.

2.3 Natural Language Processing

Natural Language Processing (NLP) is the field of study that enables machines to understand and process human languages. NLP nowadays is used in a lot of applications like: search engines, fact-checking, question answering, text auto correction and much more applications. The next subsections present some of the NLP techniques that was used in our work or was mentioned in related work in the fact-checking field.

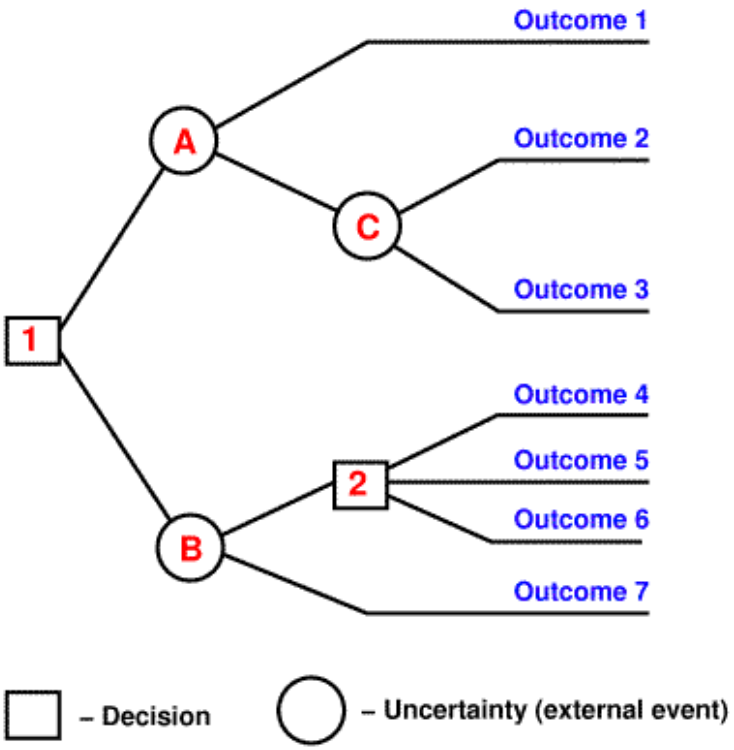


Figure 2.2: Decision Trees [2]

2.3.1 Named Entity Recognition

Named Entity Recognition (NER) is a subtask of Information Extraction from natural language understanding. NER identifies relevant nouns (people, places and organizations) in a text(paragraph/sentence). For example, for a sentence like "Albert Einstein was born in Ulm" if it was given as an input to some NER implementation, it will output that Albert Einstein is a person and Ulm is a place.

2.3.2 Stanford-NER

This section is discussing Stanford implementation of Named Entity Recognition which was used by some approaches in the related field of fact-checking. Stanford-NER[2] [7] is an implementation of linear chain Conditional Random Field (CRF) sequence models. This means by training your own models on labeled data, you can actually use this code to build sequence models for NER or any other task. CRF Models were discussed more here [13, 28, 29].

[2] https://nlp.stanford.edu/software/CRF-NER.shtml

2.3.3 Bag of words

A Bag-of-words model (BoW), is a way of extracting features from text for use in modeling, such as with machine learning algorithms. BoW is a representation of text that describes the occurrences of words in a given document. That representation comes in two data-structures: a vocabulary of the known words and a measure of the presence of known words.

2.3.4 Lemmatization and Stemming

Lemmatization and Stemming is grouping together different forms of the same word to reduce inflectional forms of words. Also, this can be useful to derive all the words into a common base form that we can deal with easier. For example, words like (car, cars, car's, cars') have the same base form which is "car", or a word like "different" can be also derived into the form "differ". To reach this goal, both Lemmatization and Stemming have different approach from each other. Stemming algorithms tends to cut off some letters off the end and the start of the word to reach its base form. The best known algorithm for Stemming is Porter stemming algorithm [24]. However, Lemmatization algorithms tends to derive the base form by the use of a vocabulary and morphological analysis of words by depending on a lexical knowledge base like WordNet[3]. In general, Lemmatization offers better precision than Stemming, but at the expense of recall.

2.3.5 Word2Vec

Vector Representations of Words or Word2Vec model [20] is used for computing vector representations of words, called Word Embeddings. Word Embeddings are vectorized representations of words where words having the same meaning are close to each other in vector space. They also allow analogies like "King" - "Man" + "Queen" == "Woman". These vectorized forms can be useful in computing similarity between words.

2.4 Related Work

Nowadays, the task of automated fact-checking is being discussed and implemented in many research committees. This task is quite complicated due to the complexity inherent in creating and connecting logical arguments [6]. Credibility level of websites and how much of trustworthiness information it presents is an important factor too in the fact checking pipeline [3]. Triple Veracity Assessment can be performed in three ways: validation, plausibility or ranking.

Triple Plausibility refers to the task of verifying whether a triple contains meaningful information or not. For triple ranking, the main task is to order(rank) a set of triples

[3] https://wordnet.princeton.edu/

by their relevance. Finally, the task of triple validation is to classify triples into true or false triples. This section gives an overview over some of the state-of-the-art approaches to fact-checking pipelines and some existing work at the direction of web credibility.

DeFacto [6, 8] is a system that gives a score of validity for RDF triples depending on web evidences. DeFacto determines the trustworthiness of a web page through some steps. First, it sees how much of similarity between the web page content and the RDF input triple. This done by how many topics are in common between the query and the web page. Topics terms are determined through querying wikipedia with the subject and object terms in the RDF triple separately. Nakamura et al. [21] suggested that approach. They combine that trustworthiness with textual evidences as features for the machine learning algorithm. The experiments showed that they have achieved a F1 measure of 84.9% over the fact validation dataset (FactBench mix) on DBpedia as well as Freebase data.

ClaimEval [18] is a fact-checking framework too. It searches for triples on the web extracting evidences from these sources and estimates the source credibility. It uses both source credibility and claim evaluation combined in one single model using Probabilistic Soft Logic. That is a pure machine learning approach. Their experiments showed their system effectiveness over other state-of-the-art baselines.

Dong et al. [17] addressed the problem of evaluating source credibility too. They proposed Knowledge-Based Trust (KBT) a system that estimates the source trustworthiness. They extract a number of facts from many pages using Information Extraction (IE) techniques. For that they rely on Knowledge Vault (KV) project [4] which uses 16 different IE techniques for the extraction of RDF triples. Then, they estimate the correctness of these facts and the source trustworthiness using inference in a probabilistic model. That came from the fact that they believe if a source is accurate the facts will be correct and the other way around. Their experiments showed promising results in evaluating web source quality.

Pasternack et al. [22] introduced a new approach for source credibility problem, Latent Credibility Analysis (LCA). It models the joint probability of the sources making claims and the unseen truth of these claims. Finding a probability for a claim to be true is done using probabilistic graphical model. They introduced 4 different models of their approach to LCA: SimpleLCA, GuessLCA, MistakeLCA, LieLCA. Their experiments showed that Latent Credibility Analysis is a powerful approach to help solving the source credibility problem. GuessLCA was the model which was more promising than the others in performance and tractability.

Nakamura et al. [21] developed a prototype system to help users to determine the trustworthiness of an information source. They came with the features to implement in that prototype system after doing a survey on 1000 Internet users from 4 different age categories using an online questionnaire of 26 questions. The prototype system included four specific features as follows:

1. **Topic majority** represents the number of similar pages to the search result. That's measured by how many topic terms the search result and he web pages have in common.

2. **Topic coverage** represents the number of topics the search results share with the query.

3. **Locality of link sources** represents how far the page is linked by other pages in different area. Users tend to trust pages that are linked from pages distributed over a large area than the ones that are more limited.

4. **Other information**: Meta information about the search results is important for the users. Users tend more to trust pages that has topic details, publisher information, number of views for this page or even the last modified date. So all of this meta information were included too in the prototype system.

Chapter 3

ISWC 17

International Semantic Web Conference (ISWC) is a premiere conference for the Semantic Web / Linked Data Community. It brings together the researchers, practitioners and industry specialists to discuss, advance, and shape the future of semantic technologies. This chapter discuss this conference challenge as it was related directly to the field of fact checking. Also, it includes a summary of an accepted paper in this conference that is about a platform for fact checking.

3.1 ISWC 17 challenge

ISWC 2017 had a challenge that focused on knowledge graphs as knowledge graphs are currently among the most important representations of Semantic Web technologies. The challenge had two important tasks:

- **Knowledge graph population**: Given the name and type of a subject entity and a relation. Participants are expected to provide the value(s) for the relation.
- **Knowledge graph validation**: Given a statement about an entity (e.g., a company), and a relation (e.g., CEO). Participants are expected to provide an estimation about the correctness of the statement.

Participants can use a portion of the knowledge graph for training. Also, different sources (e.g., external datasets, web pages, etc.) can be used as an input for participants' approaches. We will go more in depth into the second task since it's in this thesis interest. In the next subsections, an overview of the dataset for this challenge, some of the top approaches implemented by participants and also the overall results all will be presented.

3.1.1 Dataset

The dataset for the challenge will be the open data exposed by Thomson Reuters[1] at permid.org. This dataset consists of an authoritative graph of entities of interest. The entity types will be organizations and persons. Triples predicates will be one of three: Is Domiciled In, has Latest Organization Founded Date or has Headquarters Phone Number.

3.1.2 Approaches

In this subsection, detailed description for some of the top performing approaches implementations is mentioned.

DeFacto-Wikipedia

This approach[2] got the 2nd place in the tests held for this task. It's implemented in Java. Explanation for some of the technologies/terms used in their approach will be discussed first.

BOA framework[3] [9, 10] is a tool which is developed by AKSW group from Leipzig university. It allows extracting structured data (e.g., RDF triples) from unstructured data (e.g., natural language text,web pages,etc.). Also, it provides a library of natural language patterns for formal relations that allows bridging between structured and unstructured data.

Surface forms [19] are different representations for the same resource. For example, NYC is a surface form for New York City.

Their approach solves the challenge by following these steps:

1. **Generate predicate patterns**: First, BOA framework is used for generating all the possible natural language presentations for the predicate in the input sentence to generate different triples where the predicate is the same for the one in the input sentence.

2. **Retrieving web pages**: In this phase, multiple search queries are issued one for each of the triples generated from phase 1. The results must have in them an exact match for each of the subject, predicate and object from the triple. The first n results are taken where n is an input parameter and then the HTML markup is removed.

[1] https://www.thomsonreuters.com/en.html
[2] https://github.com/dice-group/FactCheck
[3] http://aksw.org/Projects/BOA.html

3. **Evaluating web pages and extracting proofs**: Web pages generated from phase 2 is evaluated to see whether they have some proofs supporting the input triple (s,p,o). First, it generates all the surface forms for the subject and the object in the triple. Then, it searches for all of the occurrences of each combination of those labels in a web page w. This occurrence will be considered as a proof, if the distance between those two labels is within given threshold.

4. **Proofs scoring**: A logistic regression classifier[14] is used for scoring each proof. This classifier is trained with some meta-data features related to the web page itself and the features of similarity between proof phrase and the BOA patterns generated from phase 1.

5. **Web page trustworthiness**: In this phase, web pages from phase 4 are evaluated if they are trustworthy or not. Pages that are on the same topic(s) related to the input triple are more trustworthy and valuable to consider. To determine the topics of the input triple and the web page they extend the approach introduced in [21] by first querying **Wikipedia** with the subject and object in the input triple. The frequency for all the terms in this wikipedia documents returned is counted and all terms above given threshold is considered as topic terms for the input triple. Then, it calculates the topic majority in web, topic majority in search results and top coverage according to formulas represented here [21].

6. **Final Score**: In this phase, the system outputs a confidence value for the input triple that ranges between 0 and 1 is generated by the model they implemented. The higher the value the more likely the input triple is true.

Semicolon Missing: SNLP-Fact-Checker

This approach[4] got the 3rd place in the tests held for this task. It's implemented in Java. After a study of their implementation, their approach solves the challenge in the following phases:

1. **Input processing**: In this phase, the system reads a sentence input which is the claim and tries to find the relation triple (s,p,o) in it. First, It finds the predicate which is in their approach one of these: award, birth place, death place, spouse, leader, team, foundation, stars, author or subsidiary. One of these predicates can be found by finding keywords in the sentence itself that means the same to this predicate. For example, "foundation place is" a keyword for the predicate "foundation". Then, subject and object is extracted from the sentence depending on the location of the keyword as the subject is the word(s) before the keyword and the object is the word(s) after the keyword.

[4] https://github.com/semicolonMissing/SNLP-Fact-Checker

2. **Subject wikipedia info box**: In this phase, wikipedia page for the subject resource extracted from the sentence in phase 1 is loaded. Then, the info box text in the wikipedia page is extracted.

3. **Extract triples**: In this phase, triples are extracted from each sentence in the wikipedia info box from phase 2 in the following steps:

   (a) Predicate will be extracted from the first word in each sentence. The predicate will be identified into one of the ten predicates mentioned in the first phase by following the same way which is assigning some keywords to each predicate. For example, "born" keyword is a keyword for the predicate "birth place".

   (b) Object will be extracted from the sentence depends on the predicate from the previous step. In case of (award, spouse, leader, star, author, team, subsidiary) predicates object will be extracted by string parsing the sentence itself to extract the information. But in the case of (birth place, death place, foundation) predicates, CRF classifier for NER from Stanford is used to identify the entities in the sentence. They use the 3-class model in order to identify the entities with the tag "Location" and these will be the object. In case of multiple objects identified, multiple triples will be generated. By the end of this step, triple(s) will be generated from the subject, predicate from the last step and object from this step.

4. **Find triples with the same relation**: In this phase, a list of triples will be selected from the ones generated from phase 3 if they have the same predicate as the one from phase 1.

5. **Giving a score for the input**: In this phase, an output score is given for the input from 3 values

   (a) **1.0**: If a triple exists in the list generated from step 4 where the object of this triple is a substring of the object from the input string or vice versa.

   (b) **0.0**: If the list generated from step 4 is empty or the subject wikipedia article doesn't exist.

   (c) **0.0**: Otherwise.

Confirmatio Ex Machina

This approach[5] got the 6th place in the tests held for this task. It's implemented in Clojure[6]. First, Explanation for some of the technologies/terms used in their approach will be discussed first.

[5] https://github.com/ConfirmatioExMachina/fact-checker
[6] https://clojure.org/

CoreNLP[7] is a natural language software developed by Stanford. It can give the base forms of words, parts of speech tagging, extract NER, normalize dates, times, and numeric quantities, mark up the structure of sentences in terms of phrases and syntactic dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract particular or open-class relations between entity mentions.

Neo4j[8] is a graph data management system developed by Neo4j, Inc.

Elastic Search[9] is a search engine based on Lucene. It can be used to real time search all kinds of data(documents) in a fast time.

Corpus is Latin word which means body. In NLP it means a collection of texts. It can be searched by word, phrase, part of speech and synonyms.

Concept Graph is a graph representation for logic based on the semantic networks of artificial intelligence and the existential graphs of Charles Sanders Peirce [27].

After a study of their approach, their implementation solves the challenge in the following steps:

1. A corpus for Wikipedia is generated as it would be more powerful this way to search and deal with Wikipedia data more than the standard interface. Then, a concept graph is created from this corpus.
2. Then, a small concept graph is constructed from the input sentence using CoreNLP.
3. Then, the algorithm tries to find a matching starting and ending point between both graphs from step 1 and step 2 which is in the case of graph from step 2 is the subject and the object. This search is done using Elastic Search.
4. Find all the undirected paths between the common starting and ending points using Neo4j and compare the edges and nodes along this paths using Word2Vec.
5. A score is given depends on the max similarity found using Word2Vec between two nodes/edges exist in the common path found in step 4.

3.1.3 Results

Participants submitted their solutions through **Gerbil**[10]. The winner of the challenge was an approach called Fact Extract. It was implemented by team IBM Socrates by Michael Glass, Nandanda Mihindukulasooriya, Oktie Hassanzadeh, and Alfio Gliozzo of IBM Research AI. Socrates won both tasks by using an innovative integration of additional Artificial Intelligence techniques such as NLP and Deep Learning over multiple

[7] https://stanfordnlp.github.io/CoreNLP/
[8] https://neo4j.com/
[9] https://www.elastic.co/products/elasticsearch
[10] http://aksw.org/Projects/GERBIL.html

web sources to find and check facts. Their knowledge graph outperformed the state of the art as set by the baseline, said SWC Chairs Bennett, Dr. Ngonga and Dr. Paulheim. The next tables and figures display the overall results of the training and testing for the fact validation task in the challenge.

The results displayed represents the area under the ROC curve. The area under the curve represents the accuracy measurement. Results are represented in figures 3.1, 3.2, 3.3 and 3.4

3.2 TAA

This section discuss a platform called TAA[11] [16] which stands for Triple Accuracy Assessment. Authors have submitted also a demo [15] in the ISWC 2017 conference. This platform is implemented in Java and its target is to measure triples accuracy in knowledge graph through finding other matched triples from other knowledge graphs. The next subsections present how their platform works and the results they achieved.

3.2.1 Approach

Their approach can be divided into four main components. the first two components find equivalent links for the resources in the source triples and generate triples that match the source triples. Then, the next component tries to find triples that match the source triples. Finally, the last component generates a confidence score that represents the accuracy of each source triple. Before talking about how their approach works, we will discuss some of the technologies they used.

sameAs[12]: This service takes as an input a resource URI and it outputs URIs that may well be co-referent.

owl:sameAs: owl is a namespace prefix referring to http://www.w3.org/2002/07/owl# that also can be used to find the same URIs for a given URI.

Information Content (IC) [26] is a measure of concept specificity. More specific concepts (e.g. dog) have higher values of IC than more general concepts (e.g. animal).

1. **Subject Link Fetching**: In this phase, the system takes both the source triple and the source knowledge graph. Then, it fetches equivalent links for the subject link in the source(input) triple. It fetches these equivalent links using sameAs service. they query it using SameAs4J API [13]. To add more variety and increase the number of the links, the system also tries to query the source knowledge graph using the owl:sameAs property. The set of subject equivalent links are filtered to remove the non-resolvable and duplicate ones.

[11] https://github.com/TriplesAccuracyAssessment
[12] http://sameas.org/
[13] http://99soft.github.io/sameas4j/

2. **Retrieving Predicates and Objects**: In this phase, the system retrieves the predicates and objects related to the subject links obtained from phase 1. It uses many knowledge repositories like DBpedia[14], GeoNames[15], LinkedGeoData[16]. It can also use SPARQL endpoints to retrieve the predicates.

3. **Matching Target Triples**: They combined a predicate semantic similarity and a predicate type and value comparison system which both are introduced in the following subsection to identify target triples generated from phase 2 that have matching predicates with the source triple. First, they filter the target triples by removing the ones with predicate similarity that is lower than a given threshold. Then, the filtered triples are used as an input for the predicate type comparison subtask that filters the triples that have mismatched predicates types. Then, the system continues filtering the triples by applying the predicate value comparison. Finally, The filtered target triples from these subtasks is the set of the triples that can be used to calculate the accuracy of the source triples.

    - **Predicate Semantic Similarity**: This procedure target is to compute the semantic similarity between two words using word-to-word similarity and corpus based information of IC.
    - **Predicate Type and Value Comparison**: This procedure target is to remove the target triples which was assigned a high similarity value from the last procedure but these triples are actually mismatching.

    Both procedure algorithms and mathematical functions used are discussed here [16].

4. **Confidence Score**: After the target triples were filtered in the last phase, these triples can be used to give a confidence score for the source triple that defines its accuracy. For source triples that have objects with numerical property, confidence score will be the weighted average of the matched triples generated. But for source triples that have objects with string property, confidence score will be calculated as the weighted average of the string similarity between each of the objects in the matched triples generated and the object from the source triple.

3.2.2 Results

The approach they suggested for measuring accuracy of RDF triples from different knowledge graphs has proved its correctness. The system can measure the triples accuracy of different property types too whether they are numerical, data or string type properties. The results were an F-measure of 95.2% on a train set containing 750 triples from DBpedia and 96.1% on a test set containing 748 triples from DBpedia.

[14] https://wiki.dbpedia.org/
[15] http://www.geonames.org/
[16] http://linkedgeodata.org/About

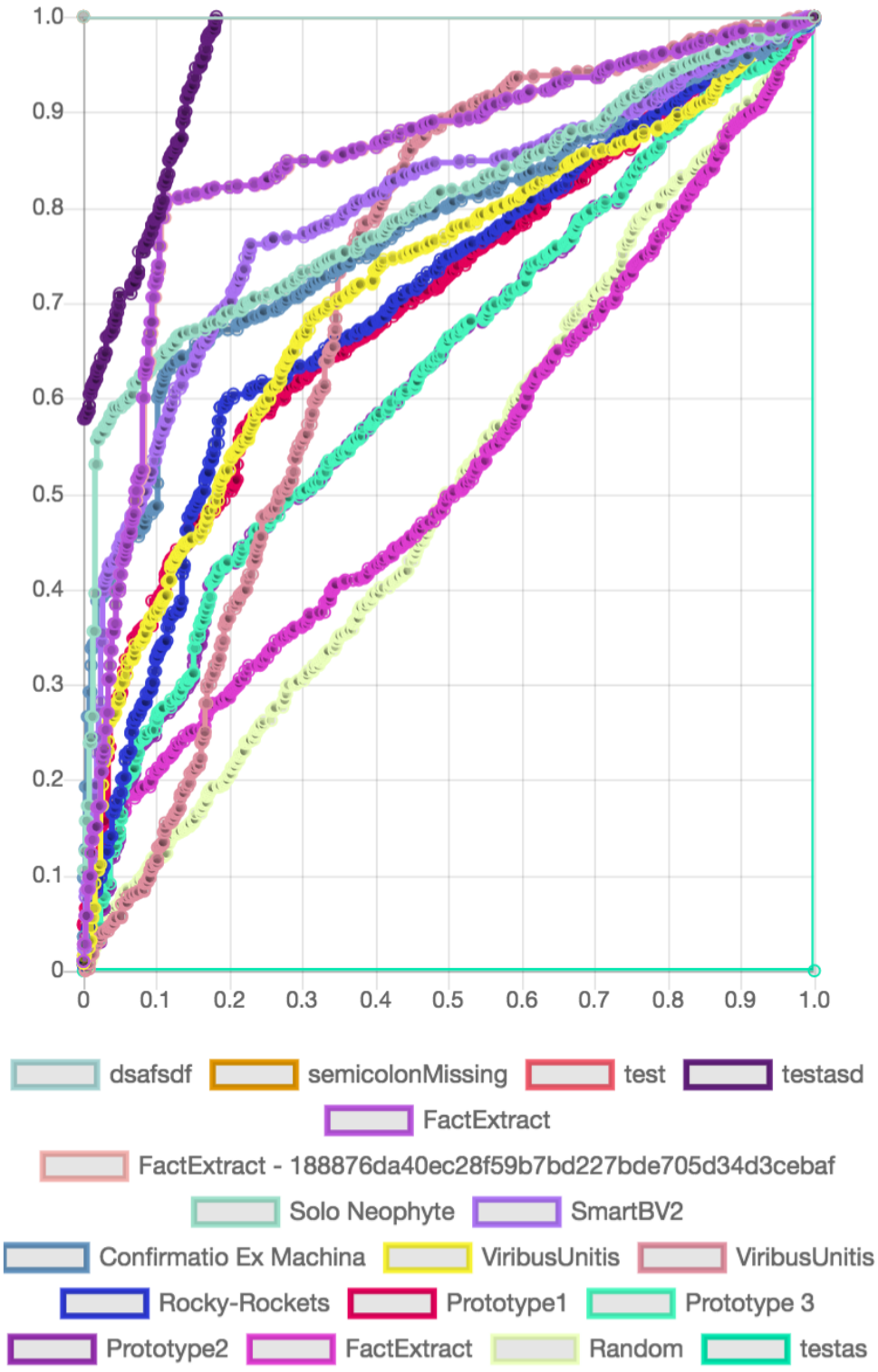


Figure 3.1: Curves for training results

AnnotatorName	Area Under Curve (AUC)
dsafsdf	1.0
semicolonMissing	1.0
test	1.0
testasd	0.9605829150945845
FactExtract	0.8501560874089487
FactExtract - 188876da40ec28f59b7bd227bde705d34d3cebaf	0.8494741635814919
Solo Neophyte	0.8009112767548917
SmartBV2	0.7956234175945787
Confirmatio Ex Machina	0.774625354824552
ViribusUnitis	0.7150077512794918
ViribusUnitis	0.7064612872551033
Rocky-Rockets	0.7024971625833821
Prototype1	0.6992078829450618
Prototype 3	0.6261452898884146
Prototype2	0.6256073015401099
FactExtract	0.526315416905576
Random	0.5044372240745066
testas	0.0

Figure 3.2: Area under the curve for all the submissions for training ranked by highest first

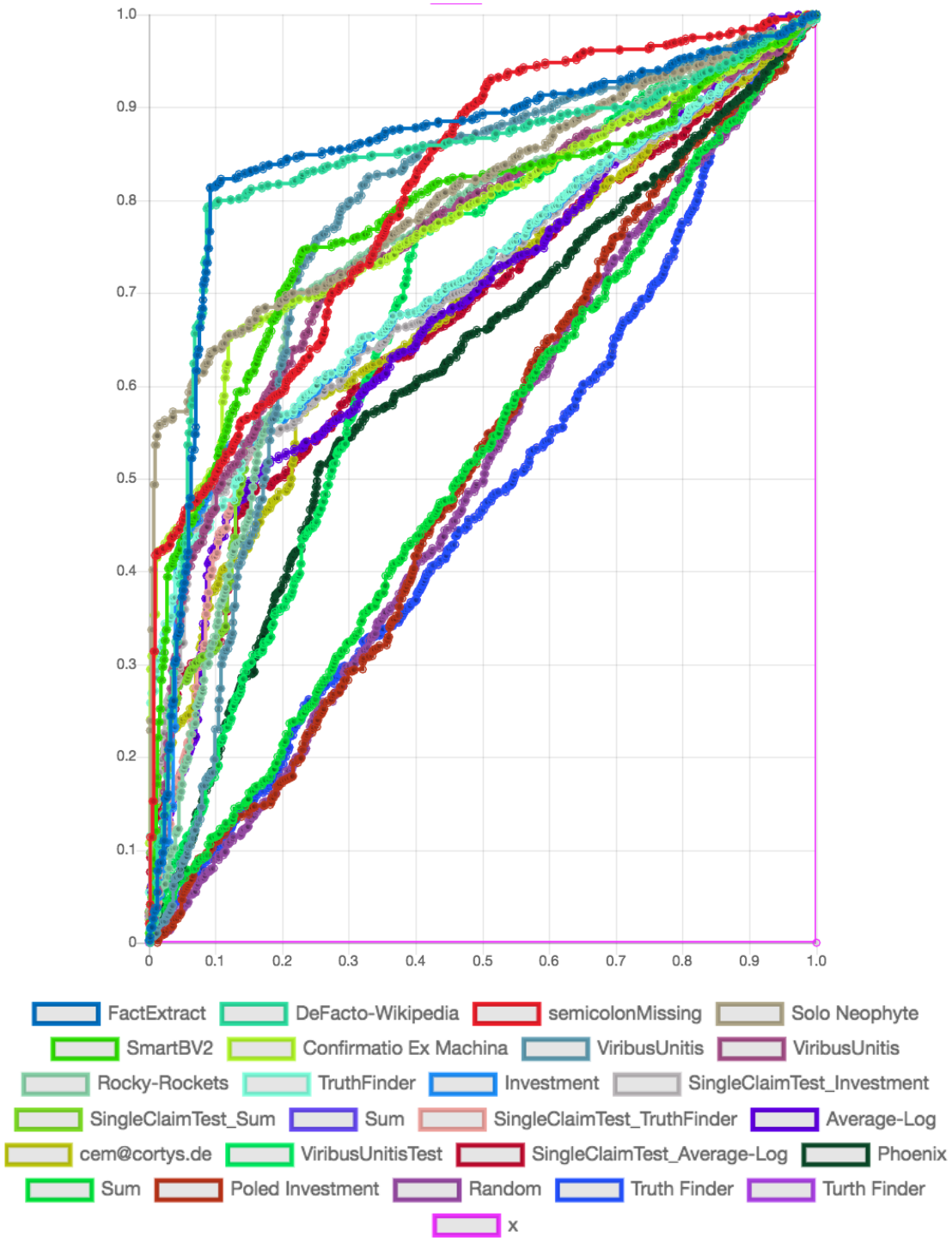


Figure 3.3: Curves for testing results

AnnotatorName	Area Under Curve (AUC)
FactExtract	0.8522891670846036
DeFacto-Wikipedia	0.8361897689132365
semicolonMissing	0.8101659851085993
Solo Neophyte	0.8091315921453647
SmartBV2	0.7796333089001194
Confirmatio Ex Machina	0.7789605917981559
ViribusUnitis	0.7616339068708766
ViribusUnitis	0.7540580031634583
Rocky-Rockets	0.7424699085683427
TruthFinder	0.7211069210292816
Investment	0.7145991666988158
SingleClaimTest_Investment	0.7092680645036843
SingleClaimTest_Sum	0.7031846765171099
Sum	0.7031846765171099
SingleClaimTest_TruthFinder	0.7008506616257092
Average-Log	0.6850767717294857
cem@cortys.de	0.680645036842714
ViribusUnitisTest	0.6794997299486901
SingleClaimTest_Average-Log	0.6773055630569806
Phoenix	0.6178291732572047
Sum	0.5193810038192971
Poled Investment	0.5151301068631614
Random	0.5058784383318544
Truth Finder	0.4829505998996954
Turth Finder	0.4829505998996954
x	0.0

Figure 3.4: Area under the curve for all the submissions for testing ranked by highest first

Chapter 4

Methodology

This chapter presents our proposal for a model that approaches the fact-validation problem. Our approach consists of three main components:

1. **Dataset Preprocessing**: A preprocessing step for our dataset in order for it be ready for training and evaluation of our model. In this component, more features are added to the dataset as part of our proposal. These features will make the overall results better.
2. **Claims & Proofs Model**: This component is a trained model that classifies whether a proof supports a claim or not. A selection for the dataset and the text features for this classification is proposed. Also, a selection of three classifiers is made and an evaluation of their results is conducted to see which classifier is the best for this subtask.
3. **Confidence Score**: In this component, a confidence score is given for triples depending on the givens from the last two components. This confidence score represents the accuracy of this triple.

The next subsections, presents our approach in details. Dataset preprocessing component is mentioned in section 4.1. Also, Implementation for proof & claims model is discussed in section 4.2. For the confidence score component, we have decided that it would be part of our future work. A poster for fact-checking benchmark platform can be found in section 4.3.

4.1 Fact Validation Dataset Preprocessing

In this section, we are discussing preprocessing and preparation of dataset for training and evaluation our fact validation model proposal. The next subsections discuss the dataset used and how it get preprocessed. Also, they discuss further processing to add more features to the dataset which is needed for our model proposal.

4.1.1 FactBench Dataset

FactBench[1] is gold standard multilingual dataset for the fact-checking algorithms evaluations. FactBench is a set of RDF models where each model is a triple and a period in which this triple holds true. This period is either a timespan or a timepoint. In the case of timespan, the period is an interval of two years, e.g. 2016-2018. For timepoint, the period will be a timespan with the same start and end year, e.g. 2018-2018. All facts in this dataset are either in English, German or French. The repository contains the training data, data for testing and the procedure files that were used to generate this dataset. This dataset was obtained from two knowledge graphs DBpedia and Freebase. The dataset has a total of 1500 total RDF models divided into two classes (750 for each class, positive, negative). The negative examples were derived from the positive one by modifying them.The RDF models in each class are equally distributed over 10 predicates. A distribution summary of data is shown in 4.1

Positive examples were generated by issuing SPRQL for DBpedia graph or MQL queries for Freebase for each of the predicates selected. Then, the top 150 results were selected. In case of Freebase, the results were sorted by an internal relevance score and in the case of DBpedia, they were sorted by the number of inbound links of a wikipedia page. 1500 examples were collected, 150 for each predicate (75 for training set and 75 for test set).

On the other hand, Negative examples were generated from the positive ones by modifying the positive example. A positive example is randomly selected and then one of the following properties get modified (subject, object, subject & object, predicate, all three of subject, predicate & object or timespan/timepoint). This modification happens in a way that the resulting set of negative examples is still meaningful.

4.1.2 Preprocessing

The target of these phase is extracting triples(s,p,o) out of the FactBench dataset in order to supply these data for our pipeline. Before talking about how these triples were extracted and saved locally, an overview of some of the technologies used is mentioned.

rdflib[2] is a library implemented in python to read and process different types of RDF files.

Apache Solr[3] is a standalone enterprise search server with a REST-like API. You put documents in it (called "indexing") via JSON, XML, CSV or binary over HTTP. You query it via HTTP GET and receive JSON, XML, CSV or binary results.

Turtle Files: Terse RDF Triple Language is a textual representation of an RDF graph. These files extension is .ttl

[1] https://github.com/DeFacto/FactBench
[2] https://github.com/RDFLib/rdflib
[3] http://lucene.apache.org/solr/

Predicate	Description	Duration	KB
award	persons who received a Nobel Prize	timepoint	Freebase
birth	birth place and date of a person	timepoint	DBPedia
death	death place and date of a person	timepoint	DBPedia
foundationPlace	place and date of a company's foundation	timepoint	Freebase
leader	presidents of countries	timespan	DBPedia
nbateam	team associations of NBA players	timespan	DBPedia
publicationDate	author of a book and it's publication date	timepoint	Freebase
spouse	marriage of two persons	timespan	Freebase
starring	actors who starred in films	timepoint	DBPedia
subsidiary	companies and their subsidiaries	timepoint	Freebase

Table 4.1: FaceBench Summary

First, the system starts reading the turtle files in the dataset. using **rdflib** library. Dataset is processed by reading each RDF model in the dataset, get the different resources (subject,predicate,object) labels along with the respective language and duration in which it was true. Then, it generates RDF triples separated by language along with the timespan/timepoint. The triples generated along with data related to it like language and duration in which it was true is saved to solr core using this library[4].

4.1.3 Searching Web For Evidences

In this phase, the system starts searching the web looking for evidences that maybe support the dataset triples. First, the system reads the triples generated from preprocessing phase from solr. Then, for each triple(s,p,o) the system will formulate three queries for the search engine. (1) "s" where s is the subject of the triple (2) "s" + "o" where s, o is the subject and object of the triple. (3) "s" + "p" + "o" where s, p, o is the subject, predicate and object of the triple. Then, the system will use Bing search API to search for these queries and a limit for the number of search results will be set to 10 per query. the system will use solr again to store metadata about each of the search results for each of the triples:

Body: The body text of each of the search results will be stored.

Last Modified: the system will store the time in which the page was last modified or created.

Publisher information: the system will also store information related to the publisher of the web page content.

[4] https://github.com/django-haystack/pysolr

4.1.4 Finding Topic Terms

In this phase, the system starts finding topic terms related to the subject and object of each triples. Once again, the system starts reading each of the triples from solr. For each triple, it uses a wikipedia api library[5] in python to get wikipedia article for both subject and object individually. For each article, bag of words technique discussed in the background, is used and count frequencies of each word in the article. Then all words that have a frequency over some selected threshold (α) will be considered as topic terms for these article. Topic terms for subject/object will be saved into solr. By the end of this phase, in solr the system can find triples generated from FactBench dataset along with search results for queries generated about each triple and also the topic terms related to each subject and object in the dataset.

4.2 Claims & Proofs Model

This subsection presents a model implementation that checks whether for a given claim c and a proof p, if p supports c. The next subsections are discussing the dataset used, features and labels extracted and the training process. All results achieved for this model are discussed in chapter 5.

4.2.1 Google Relation Extraction Dataset

Relation Extraction task is one of the most difficult NLP tasks. It's an example of Information Extraction (IE)[6] which is one of the goals of natural Language understanding. A relation is a semantic connection between (at least) two entities. For example, Charles Mantoux (French physician) institution is University of Paris. The main goal of relation extraction is to learn relations from unstructured data. These relations can be used to answer questions or build a huge database to help researchers and people explore more of world's information.

To help investigating this field more, Google released human-judged dataset[7] of five relations about public figures on wikipedia. It contains nearly 59,000 examples over 5 predicates. Distribution of examples is mentioned in this table 4.2. Each of these relations were judged by at least 5 raters. These judgments can be used to train and evaluate relation extraction systems. Each relation in the predicate represents a triple of subject, predicate and object. Subject and object are represented with Freebase MID's and the predicate is represented as a Freebase property.

Each relation too has one or more evidences, each is a URL and a snippet from the web page that is judged by humans and it supports this relation. For some of the examples,

[5] https://github.com/martin-majlis/Wikipedia-API/

[6] The task of extracting structured information from unstructured/semi-structured data

[7] https://github.com/google-research-datasets/relation-extraction-corpus

Predicate	Count
Institution	42,628
Place of Birth	9,560
Date of Birth	2,490
Place of Death	3,042
Education Degree	1,850

Table 4.2: Google Relation Extraction Dataset Summary

the evidence doesn't support the relation. These negative examples can be used for training better extraction systems. Finally, judgments by humans for each relation along are included along with the rater id. The files are in JSON format. Relation's JSON structure can be found in 4.1.


```
{"pred":"/people/person/education./education/education
  /institution","sub":"/m/056h7f","obj":"/m/0lk0l","evidences"
  :[{"url":"http://en.wikipedia.org/wiki/Charles_Mantoux"
  ,"snippet":"He graduated from the University of Paris where he
  studied under Broca. For health reasons he relocated to Cannes
  but continued to work in Paris"}],"judgments":[{"rater"
  :"16671464637895630418","judgment":"yes"},{"rater"
  :"18153321702395861849","judgment":"yes"},{"rater"
  :"65706877213633247 18","judgment":"yes"},{"rater"
  :"1295059937994543754","judgment":"yes"},{"rater"
  :"12022408018620867151","judgment":"yes"}]}
```


Figure 4.1: Google relation Relation's JSON structure

4.2.2 Preprocessing

Before training the model, a preprocessing step for the dataset was necessary. In order for the dataset to be ready for training the model, the system needed to find what resource name the Freebase MIDs for subject and object refer to. For that, Google Knowledge Graph Search API[8] was used. The system queried every Freebase MID for subjects and objects existing over the five predicate files searching for the name of those resources. The mappings between the MID and the name of the resource were saved locally so they can be used in further steps of the training process. Some of the MIDs which weren't found after querying the API were replaced manually.

[8] https://developers.google.com/knowledge-graph/

4.2.3 Features and Labels Extraction

After preprocessing, the system starts extracting features from dataset instances. It first goes through predicates files one at a time. For each predicate file, It starts by extracting features and labels for training and evaluating our model. Assume one instance from the dataset in any predicate is (s,p,o,e,j) where these symbols represents the following: **s** is the subject name for this instance of the dataset extracted from the mappings file generated before. **p** is the predicate Freebase property for this instance of the dataset. **o** is the object name for this instance of the dataset extracted from the mappings file generated before. **e** is the set of evidences for this instance. Each evidence is a pair of a URL and a snippet of text. **j** is the set of human judgments of this instance. Each judgment is a pair of rate id and the judgment whether it's a yes or a no. If an instance has multiple evidences, the algorithm treats each of them individually. Then each evidence will be an entry in our training dataset.

Features Extraction

For each instance in the dataset. the system starts extracting the following features for each evidence one at a time:

1. **Has Subject**: This a binary feature (1 if Yes, 0 if No) that represents whether the evidence e contains the subject s. The system checks whether the full name, first name or last name exists in the evidence as substring.

2. **Has Object**: This a binary feature that represents whether the evidence e contains the object o. Though, most of education degrees have different abbreviations that holds the same meaning. So in this case, the algorithm looks for the object and its abbreviations(if any). List of those abbreviations found is found in 4.4. Those abbreviations was found through records of each education degree on wikidata.

3. **Has Predicate**: This is a binary feature that represents whether the evidence e contains the predicate p. For each of the five predicates, some keywords were assigned. Then the system checks if one of the keywords assigned to the predicate p exists in the evidence as a substring or not. List of keywords used for each predicate can be found in 4.3. Some of the verbs included in the list of keywords were just in the past form. For the other verbs, the implementation included both past and present forms but table 4.3 includes just the present form.

4. **Has NER**: This is a binary feature that represents whether the evidence contains named entities or not. For identifying that an NLP library called Spacy[9] is used. The trained model by spacy called "en-core-web-lg" was used for this subtask. This model is an English multi-task CNN trained on OntoNotes, with GloVe vectors

[9] https://spacy.io/

Predicate	Keywords
Institution	graduate, studies, educated, taught, student, joins, attends, enters, enrolls
Place of Birth	born, birth
Date of Birth	born, birth
Place of Death	dies, passed away, death
Education Degree	receives, receiving, graduated, graduating, degree

Table 4.3: List of Keywords Assigned For Each Predicate

trained on Common Crawl. Assigns word vectors, context-specific token vectors, POS tags, dependency parse and named entities. This model is used to identify whether the evidence e has named entities or not.

5. **Has Subject Lemma**: This is a binary feature that represents whether the subject s lemmatization exists in the evidence e. A library called NLTK[10] is used for this subtask. This library has a module called "stem.wordnet" which can be used for finding the lemmatization of the subject. Then the system checks whether the result of that exists in the evidence as a substring or not.

6. **Has Object Lemma**: This is a binary feature that represents whether the object o lemmatization exists in the evidence e. The same library from the last feature is used to find the lemmatization for the object. Then, the system looks whether the result of that exists in the evidence or not.

7. **Subject Similarity**: This feature represents the maximum similarity that can be found between the subject s and a word w, where w is a substring word in e. To find the similarity between the subject and each of the words in the evidence, word2vec vectors embedded in spacy "en-core-web-lg" model are used. Then, the system returns the maximum of these similarities.

4.2.4 Labels Extraction

For each instance in the dataset, a label is generated based on the judgments list j. For our classifier, there are 3 labels(classes) mentioned in 4.5. the system generates label 0, if the count of yes labels in j is at least 80% of the length of j. Label 1 is assigned, if the count of no labels in j is at least 80% of the length of j. Otherwise it's label 2. Also, label 2 is assigned for some instances where the system couldn't find the mapping for the subject or object MID. Instance label generated for one instance of the dataset is assigned to each of the evidences belonging to this instance. Count for each of the classes labels per predicate is displayed in table 4.6.

[10] https://www.nltk.org/index.html

Education Degree	Abbreviations
Bachelor of Science	Bc., B.S., BS, B Sc, B.Sc., BSc, S.B, SB, Sc.B., S.B.
Doctor of Philosophy	Ph.D., PhD, D.Phil., DPhil, Dr. phil.
Bachelor of Arts	B.A., BA, A.B., AB, BA Degree
Bachelor of Engineering	BE, BEng, B.Eng.
Master's Degree	Master Degree, Masters Degree, Master
Bachelor of Fine Arts	BFA
Master of Science	M.S., MSc, MS, M.Sc., M.Sci., MSci, S.M., Sc.M., Sci.M.
Honorary Degree	Honoris Causa, Ad Honorem, H.C.
Master of Arts	MA, M.A., A.M., AM
Bachelor of Business Administration	BBA, B.B.A., BSBA
Juris Doctor	JD, J.D., Doctor of Jurisprudence, D.Jur., DJur
Master of Business Administration	MBA, M.B.A., Masters of Business Administration, Master's of Business Administration
Bachelor of Laws	LL.B., LLB, B.L
Doctor of Letters	Doctor of Literature, D.Litt., Litt.D., D. Lit., Lit. D.
Doctor of Science	Sc.D., Sc. D., D.Sc., D. Sc., S.D., D.S., Dr.Sc., Dr. Sc.
Doctor of Divinity	DD, D.D., D. D., D. Div.
Doctorate of Humane Letters	D. H. L., D.H.L., L. H. D., L.H.D., Litterarum Humanarum Doctor
Master of Fine Arts	MFA, M.F.A.
Doctor of Medicine	Medical doctor, M.D., MD, M. D., Doctorate of Medicine
Master of Divinity	Masters of Divinity, M.Div., MDiv
Doctor of Juridical Science	Doctor of the Science of Law, Doctor of Laws, J.S.D., JSD, S.J.D., SJD
Master of Public Administration	MPA, M.P.A.
Bachelor of Theology	B.Th.
Master of Laws	LL.M, LLM
Master of Philosophy	MPhil, M.Phil., M.Phil

Table 4.4: List of Education Degrees' Abbreviations

Label	Meaning
0	This evidence supports the claim.
1	This evidence doesn't support the claim.
2	Not enough information to decide.

Table 4.5: List of Classes Labels Meanings

Predicate	Class 0	Class 1	Class 2
Institution	24804	12776	5271
Place of Birth	6748	2249	569
Date of Birth	1982	164	344
Place of Death	2207	303	532
Education Degree	1539	134	181

Table 4.6: Count of Each Class Instances Per Predicate

4.2.5 Training

After the generation of features and labels for each predicate instances, the system saves these features and labels locally for future uses. In order to have both dataset for training and one for testing and evaluation of our approach, the systems splits the data into 77% for training and 33% for testing and evaluation. Selection of these two sets are randomly done. For dividing and randomizing data selected, the *train_test_split* method from scikit learn library is used. Three classifiers were used for training. Support Vector Machines, Naive Bayes and Decision Trees. Classifiers used are the ones implemented in Scikit Learn library.

4.3 Fact Checking Benchmark Poster

This subsection includes a fact-checking benchmark poster that was submitted and accepted in CSCUBS 18[11]. The poster is about an idea of how to design a benchmark platform for fact-checking algorithms over multiple datasets. The benchmark evaluates the algorithms by some of the state-of-the-art output measures. Poster can be found in figure 4.2

[11] http://cscubs.cs.uni-bonn.de/2018/

Benchmarking fact-checking algorithms through GERBIL

Fact-checking benchmark

Omar Sallam, Piyush Chawla
Smart Data Analytics Lab
University of Bonn, Germany
German University in Cairo, Egypt

introduction

A benchmark for analysing and comparing different fact-checking algorithms for tasks like: Triple Veracity Assessment or Natural Language claims checking on different datasets by measuring some select features.

Motivation:
Huge amount of data is being generated every data on the web has increased the chances of errors creeping the data itself. This also questions the credibility of facts found on web. Fact-checking is a research field to overcome this problem and it has become a hot research area recently. A benchmark for comparing different algorithms in that field on different datasets will enable the researchers to analyze the problem more and see the differences between the existing approaches.

Examples

RDF Triples

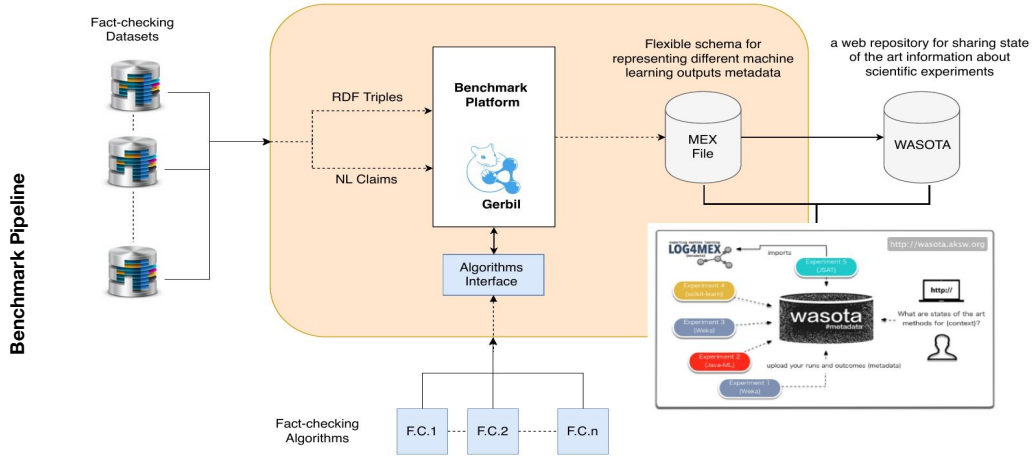
dbr:Albert_Einstein, dbo:birthPlace, dbr:Ulm

NL Claims

A typical taxpayer pays £1,075 less income tax than they did in 2010.

Our benchmark system analyse and run different algorithms on different datasets in the field of fact-checking. We are supporting fact-checking algorithms over RDF triples and NL claims too. That is done through extending **GERBIL KBC Benchmark** to evaluate fact-checking algorithms. We will build an interface for dealing with fact-checking algorithms. We are enhancing the system by using **MEX** which is a standard machine learning format. Then **WASOTA** is used to store the results in the form of the state of art measures existing.

Benchmark Pipeline



Gerbil **WASOTA** **MEX Schema**

SDA BONN GUC German University in Cairo CSCUBS'18

Figure 4.2: Fact-checking Benchmark Poster

Chapter 5

Results

This chapter presents an overview of the results of the Proof & Claims Model discussed in 4.2. All results in this chapter are calculated by Scikit Learn library.

5.1 Confusion Matrix

The confusion matrix is one of the most intuitive and metrics used for finding the correctness and accuracy of the model. It's used for classification problems where the number of classes to be predicted is at least 2. The confusion matrix is a n x n matrix where n is the number of classes for the classification problem and each cell C_{ij} is equal to the number of observations known to be in group i but predicted to be in group j.

TP	FP
FN	TN

Table 5.1: Confusion Matrix 2 x 2

Terms Associated with Confusion Matrix:

1. **True Positives (TP)** are the cases when the actual class of the data point was 1 (True) and the predicted is also 1 (True).
2. **True Negatives (TN)**: are the cases when the actual class of the data point was 0 (False) and the predicted is also 1 (False)
3. **False Positives (FP)** are the cases when the actual class of the data point was 0 (False) but the predicted was 1 (True).
4. **False Negatives (FN)**: are the cases when the actual class of the data point was 1 (True) but the predicted was 0 (False)

Confusion Matrix for all three classifiers trained (SVM, Naive Bayes and Decision Trees) for all five predicates are displayed in tables from 5.5 to 5.18

5.2 Performance Metrics

Tables 5.2, 5.3 and 5.4 include the main classification metrics for each of the five predicates (institution, Place of Birth, Date of Birth, Place of Death and Education Degree). The main performance metrics selected are mentioned in the next subsections.

5.2.1 Accuracy

Accuracy in classification problems is the number of correct predictions divided by the total number of predictions made by the model. Accuracy is a good measure when the data is balanced over the target classes. However, it can be a bad measure to rely on if the data is a majority of one class. So, for example if 95% of the data is of class A and the rest is of class B and the model predicts all of class A data correctly. Accuracy will be 95% but at the same time the model couldn't predict any of class B items which is bad. Mathematically, Accuracy can be represented using equation 5.1

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{5.1}$$

5.2.2 Precision

Precision is a measure that tells us the percentage of the correct positive predictions. This can be represented mathematically using equation 5.2. Equation 5.3 can be used To calculate the precision of class i using confusion matrix A

$$Precision = \frac{TP}{TP + TF} \tag{5.2}$$

$$P_i = \frac{A_{ii}}{\sum_j A_{ji}} \tag{5.3}$$

5.2.3 Recall

Recall is a measure that tells us the percentage of the true positives that were predicted correctly. This can be represented mathematically using equation 5.4. Equation 5.5 can be used To calculate the recall of class i using confusion matrix A.

$$Recall = \frac{TP}{TP + FN} \tag{5.4}$$

$$R_i = \frac{A_{ii}}{\sum_j A_{ij}} \tag{5.5}$$

Predicate	Accuracy	Precision	Recall	f_1 score
Institution	0.63	0.54	0.63	0.56
Place of Birth	0.70	0.60	0.70	0.61
Date of Birth	0.80	0.63	0.80	0.71
Place of Death	0.74	0.65	0.74	0.66
Education Degree	0.81	0.70	0.81	0.75

Table 5.2: SVM Classifier Performance Metrics for Every Predicate

Predicate	Accuracy	Precision	Recall	f_1 score
Institution	0.60	0.50	0.60	0.51
Place of Birth	0.70	0.49	0.70	0.58
Date of Birth	0.80	0.63	0.80	0.71
Place of Death	0.72	0.51	0.72	0.60
Education Degree	0.82	0.67	0.82	0.74

Table 5.3: Naive Bayes Classifier Performance Metrics for Every Predicate

5.2.4 f_1 score

f_1 score also known as balanced F-score or F-measure, is the weighted average of the precision and recall measures. This can be represented mathematically using equation 5.6. For multi-class classification, the F-score is the weighted average of the F-score for each class.

$$F1_i = 2 \times \frac{precision \times recall}{precision + recall} \tag{5.6}$$

Predicate	Accuracy	Precision	Recall	f_1 score
Institution	0.52	0.51	0.52	0.52
Place of Birth	0.61	0.58	0.61	0.59
Date of Birth	0.75	0.72	0.75	0.73
Place of Death	0.63	0.61	0.63	0.62
Education Degree	0.73	0.73	0.73	0.73

Table 5.4: Decision Tree Classifier Performance Metrics for Every Predicate

7526	679	0
2896	1337	0
1354	349	0

Table 5.5: SVM Confusion Matrix for Institution Predicate

7682	523	0
3459	774	0
1436	267	0

Table 5.6: Naive Bayes Confusion Matrix for Institution Predicate

5587	1887	731
2138	1640	455
1003	505	195

Table 5.7: Decision Trees Confusion Matrix for Institution Predicate

2162	47	0
709	56	0
156	27	0

Table 5.8: SVM Confusion Matrix for Place of Birth Predicate

2209	0	0
765	0	0
183	0	0

Table 5.9: Naive Bayes Confusion Matrix for Place of Birth Predicate

1731	389	89
540	182	43
129	46	8

Table 5.10: Decision Trees Confusion Matrix for Place of Birth Predicate

655	0	0
61	0	0
106	0	0

Table 5.11: SVM & Naive Bayes Confusion Matrix for Date of Birth Predicate

579	19	57
37	17	7
80	7	19

Table 5.12: Decision Trees Confusion Matrix for Date of Birth Predicate

704	0	14
83	0	7
161	0	35

Table 5.13: SVM Confusion Matrix for Death Place Predicate

718	0	0
90	0	0
196	0	0

Table 5.14: Naive Bayes Confusion Matrix for Death Place Predicate

574	60	84
62	8	20
123	22	51

Table 5.15: Decision Trees Confusion Matrix for Death Place Predicate

489	13	0
37	5	0
63	5	0

Table 5.16: SVM Confusion Matrix for Education Degree Predicate

502	0	0
42	0	0
68	0	0

Table 5.17: Naive Bayes Confusion Matrix for Education Degree Predicate

429	43	30
30	10	2
48	10	10

Table 5.18: Decision Trees Confusion Matrix for Education Degree Predicate

Chapter 6

Conclusion & Future Work

6.1 Conclusion

In this thesis, we investigated the related work in the field of fact-checking for RDF Triples. Also, the International Semantic Web Conference (ISWC) 2017 investigated as it included a challenge for fact-validation in Knowledge Graphs and a paper about triple accuracy measurement in knowledge graphs.

Then, a pipeline for RDF triples fact-checking of 3 phases was proposed. We managed to complete the first two phases in which the dataset was processed to be ready for the pipeline and adding web evidences for each instance in the dataset and finding topic terms for resources of each instance. Finally in the second phase, a model for checking whether a proof p supports a claim c was designed. The model was trained on five predicates (Institution, Place of Birth, Date of Birth, Place of Death and Education Degree) from a google dataset. Three classifiers were trained: SVM, Naive Bayes and Decision Trees. SVM had better f_1 score over all the predicates but date of birth predicate. The SVM model reached f_1 score of 0.56, 0.61, 0.66, 0.75 for predicates: institution, place of birth, place of death and education degree, respectively. However, the Decision Tree model had the highest f_1 score of 0.73 for the date of birth predicate. Detailed results are discussed for each of the predicates in chapter 5. The results include other metrics like accuracy, precision and recall.

6.2 Future Work

In future work, we are planning to complete the last phase of the fact-checking pipeline which is giving a confidence score for triples that represents its accuracy. The confidence score is generated by a machine learning model using evidences and topic terms collected from phase 1 of the pipeline and the proof & claims model. Also, different improvements can be made for the proof & claims model to achieve better results:

- Improving the state of google dataset used by finding resources names for the rest of Freebase MIDs that we couldn't find their resources using the Google knowledge graph API.
- Adding more predicates to the model by following the same way the google dataset generated. suggested predicates would be: award, foundation place, leader and starring.
- Extracting more features out of the evidences that can help improving the overall results for all of the predicates.

Appendix

Appendix A

Lists

KB	Knowledge Base
ISWC	International Semantic Web Conference
SVM	Support Vector Machines
ML	Machine Learning
NLP	Natural Language Processing
RDF	Resource Description Framework
NER	Named Entity Recognition
BoW	Bag of Words
IE	Information Extraction
CNN	Convolutional Neural Networks
POS	Part of Speech

List of Figures

2.1 Support Vector Machines [1] 5

2.2 Decision Trees [2] 6

3.1 Curves for training results 18

3.2 Area under the curve for all the submissions for training ranked by highest first 19

3.3 Curves for testing results 20

3.4 Area under the curve for all the submissions for testing ranked by highest first 21

4.1 Google relation Relation's JSON structure 27

4.2 Fact-checking Benchmark Poster 32

List of Tables

4.1 FaceBench Summary 25
4.2 Google Relation Extraction Dataset Summary 27
4.3 List of Keywords Assigned For Each Predicate 29
4.4 List of Education Degrees' Abbreviations 30
4.5 List of Classes Labels Meanings 31
4.6 Count of Each Class Instances Per Predicate 31

5.1 Confusion Matrix 2 x 2 33
5.2 SVM Classifier Performance Metrics for Every Predicate 35
5.3 Naive Bayes Classifier Performance Metrics for Every Predicate 35
5.4 Decision Tree Classifier Performance Metrics for Every Predicate 35
5.5 SVM Confusion Matrix for Institution Predicate 36
5.6 Naive Bayes Confusion Matrix for Institution Predicate 36
5.7 Decision Trees Confusion Matrix for Institution Predicate 36
5.8 SVM Confusion Matrix for Place of Birth Predicate 36
5.9 Naive Bayes Confusion Matrix for Place of Birth Predicate 36
5.10 Decision Trees Confusion Matrix for Place of Birth Predicate 36
5.11 SVM & Naive Bayes Confusion Matrix for Date of Birth Predicate 36
5.12 Decision Trees Confusion Matrix for Date of Birth Predicate 36
5.13 SVM Confusion Matrix for Death Place Predicate 37
5.14 Naive Bayes Confusion Matrix for Death Place Predicate 37
5.15 Decision Trees Confusion Matrix for Death Place Predicate 37
5.16 SVM Confusion Matrix for Education Degree Predicate 37
5.17 Naive Bayes Confusion Matrix for Education Degree Predicate 37
5.18 Decision Trees Confusion Matrix for Education Degree Predicate 37

Bibliography

[1] Support vector machine (svm). http://arun-aiml.blogspot.com/2017/07/support-vector-machine-svm.html, Jul 2017.

[2] What is a decision tree algorithm? https://medium.com/@SeattleDataGuy/what-is-a-decision-tree-algorithm-4531749d2a17, Sep 2017.

[3] Laure Berti-Equille and Javier Holthoefer. *Veracity of Data: From Truth Discovery Computation Algorithms to Models of Misinformation Dynamics*, volume 7. 12 2015.

[4] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 601–610, New York, NY, USA, 2014. ACM.

[5] W. W. Eckerson. Data quality and the bottom line: achieving business success through a commitment to high quality data. 2002.

[6] Diego Esteves, Anisa Rula, Aniketh Reddy, and Jens Lehmann. Toward veracity assessment in rdf knowledge bases: An exploratory analysis. 9:1–26, 02 2018.

[7] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[8] Daniel Gerber, Diego Esteves, Jens Lehmann, Lorenz Bhmann, Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, and Ren Speck. Defacto - temporal and multilingual deep fact validation. 35, 08 2015.

[9] Daniel Gerber and Axel-Cyrille Ngonga Ngomo. Bootstrapping the linked data web. In *1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011*, 2011.

[10] Daniel Gerber and Axel-Cyrille Ngonga Ngomo. Extracting multilingual natural-language patterns for rdf predicates. 7603:87–96, 10 2012.

[11] Steven Grijzenhout and Maarten Marx. The quality of the xml web. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1719–1724, New York, NY, USA, 2011. ACM.

[12] Naeemul Hassan, Afroza Sultana, You Wu, Gensheng Zhang, Chengkai Li, Jun Yang, and Cong Yu. Data in, fact out: Automated monitoring of facts by factwatcher. In *Proceedings of the VLDB Endowment*, volume 7, 09 2014.

[13] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[14] N. Landwehr, M. Hall, and E. Frank. Logistic model trees 95. pages 161–205, 2005.

[15] Shuangyan Liu, Carlo Allocca, Mathieu d'Aquin, and Enrico Motta. TAA: A platform for triple accuracy measuring and evidence triples discovering. In *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017.*, 2017.

[16] Shuangyan Liu, Mathieu dAquin, and Enrico Motta. Measuring accuracy of triples in knowledge graphs. pages 343–357, 05 2017.

[17] Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. Knowledge-based trust: Estimating the trustworthiness of web sources. 8, 02 2015.

[18] Manuela Veloso Mehdi Samadi, Partha Talukdar and Manuel Blum. Claimeval: Integrated and flexible framework for claim evaluation using credibility of sources. In *In Proceedings of the 13th AAAI Conference on Artificial Intelligence (AAAI16)*, pages 222–228. AAAI Press, 2016.

[19] Pablo Mendes, Max Jakob, Andrs Garca-Silva, and Christian Bizer. Dbpedia spotlight: Shedding light on the web of documents. pages 1–8, 09 2011.

[20] T Mikolov. Distributed representations of words and phrases and their compositionality. pages 1–9, 01 2013.

[21] Satoshi Nakamura, Shinji Konishi, Adam Jatowt, Hiroaki Ohshima, Hiroyuki Kondo, Taro Tezuka, Satoshi Oyama, and Katsumi Tanaka. Trustworthiness analysis of web search results. pages 38–49, 09 2007.

[22] Jeff Pasternack and Dan Roth. Latent credibility analysis. In *WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web*, pages 1009–1020, 05 2013.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] MF Porter. An algorithm for suffix stripping. 14, 03 1980.

[25] Barna Saha and Divesh Srivastava. Data quality: The other face of big data. In *Proceedings - International Conference on Data Engineering*, pages 1294–1297, 03 2014.

[26] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. Adoption of the linked data best practices in different topical domains. 8796:245–260, 10 2014.

[27] John F. Sowa. Conceptual graphs as a universal knowledge representation. *Computers & Mathematics with Applications*, 23(2):75 – 93, 1992.

[28] Charles Sutton and Andrew Mccallum. An introduction to conditional random fields for relational learning. 01 2007.

[29] Charles Sutton and Andrew Mccallum. An introduction to conditional random fields. 4, 11 2010.