الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي

UNIVERSITÉ BADJI MOKHTAR - ANNABA
BADJI MOKHTAR – ANNABA UNIVERSITY

جامعة باجي مختار – عنابـــــــــــة

Faculty : Engineering Sciences
Département : Computer Science
Domaine : Mathematics and Computer Science
Filière : Computer Science
Spécialité : Système d'informations et écisions

Thesis
Presented in view of obtaining: Masters degree
Theme:

Towards Enriching Linked Open Data from unstructured data

Presented by : *Ms. Allouani Rayene*

Supervisor: *Hassina Seridi-Bouchelaghem Pr. University of Annaba*

Jury:

Farah Nadir	Pr.	University of Annaba	President
Azizi Nabiha	Pr.	University of Annaba	Examiner
Seridi-Bouchelaghem Hassina	Pr.	University of Annaba	Supervisor

University year : 2019/2020

Table of Content

Acknowledgments:

First and foremost I would like to thank Allah for granting me health, patience and enough composure to finish this work, especially during the current events.

I would like to thank my advisor Mrs. Seridi for guiding me and showing me the path when I got lost in the research world and couldn't find my way.

I also thank the jury for taking the time to read and evaluate this work.

A massive thank you to my parents, who not only encouraged and supported me beyond measures, but also put up with me when stress took over me and I became insufferable. Words can't express how thankful I am. I hope I make you proud.

I would also like to thank my little sister Alae Hibet Errahmen, who has been my comic relief throughout this journey, thank you for bringing me chocolate and making me tea at midnight when I asked, I love you.

The biggest of thanks to my uncle Bouabdallah Sebti, who believed in me day in and day out, even when I wasn't so sure myself. Thank you for your constant guidance and counseling.

My deepest thanks go to my aunts Hassiba and Djamila Bouaballah, your unconditional love and support means the world to me, I don't think I can ever express my gratitude enough, I love you both.

I would also like to thank my close friends, who gave me that extra push when I needed it, and never complained when they had to deal with my panic attacks in the middle of the night. I love you, all three of you.

Finally I would like to thank myself for finishing this work during a global pandemic without losing my mind.

Dedication:

This work is dedicated to:

My maternal grandparents, Mohamed Salah and Laakri, who I love immensely, I hope you are proud. May Allah bless and protect you.

My paternal grandparents, Abdallah and Yamina, may their souls rest in peace. I wish you were still here to see me graduate.

My cousins, Maroua and Afaf who are my family, friends and lifelong companions.

My big family, both maternal and paternal.

All of my friends, you guys are incredible.

My cats, without who this work would've been finished months earlier.

Each and every person that ever supported me, y'all are great.

List of Figures

List of Tables

Abstract:

To enrich Linked Open Data from text we must go through a lot of steps that allow us to represent the unstructured data (text) in a structured manner than can be processed by computers, a crucial one of those steps is Open Information Extraction. OpenIE is a field that still faces a lot of challenges, one of those is representing implicit data.

We utilize Paraphrasing with large language models that allows us to reformulate sentences, expressing the implicit information in them in a more explicit, direct way.

We use as well neural Coreference Resolution approaches, that permit us to replace pronouns with their original mentions, making it easier to link arguments in Knowledge Bases.

Our aim is to improve upon existing OpenIE approaches and provide more informative and accurate extracted tuples to then use to enrich Linked Open Data.

Keywords:

Natural Language Processing, Linked Open Data, Open Information Extraction, OpenIE, LOD, Knowledge base, semantic web

Résumé:

Pour enrichir les Linked Open Data à partir de texte, nous devons passer par de nombreuses étapes qui nous permettent de représenter les données non structurées (texte) d'une manière structurée qui peut être traitée par des ordinateurs, l'une de ces étapes cruciales est la Open Information Extraction. OpenIE est un domaine qui fait encore face à de nombreux défis, l'un d'entre eux étant la représentation de données implicites.

Nous utilisons la paraphrase avec de grands language models qui nous permet de reformuler des phrases, en exprimant les informations implicites qu'elles contiennent d'une manière plus explicite et directe.

Nous utilisons également des approches de résolution de coréférence neuronale, qui nous permettent de remplacer les pronoms par leurs mentions originales, ce qui facilite la liaison des arguments dans les bases de connaissances.

Notre objectif est d'améliorer les approches OpenIE existantes et de fournir des tuples extraits plus informatifs et plus précis à utiliser ensuite pour enrichir les Linked Open Data.

Introduction

With the spread of internet, the amount of accessible data is increasing exponentially. However, a large percentage of this data remains unexploited due to its unstructured nature. This has encouraged researchers to find new ways to structure and link this data in order to better use it (ie: to analyze it and extract new knowledge) One of the ways to structure data is Automatic Knowledge Base Construction which consists of representing the data found in Natural Language Text in the form of Knowledge Bases. One of this field's main challenges is extracting accurate triplets from text containing implicit information or n-ary relations, while keeping the semantic (meaning) of the sentence. Which lead to the emergence of multiple research fields such as Open Information Extraction, Named Entity Recognition...etc.

In this work we aim to get one step closer to enriching Linked Open Data from Natural Language Text by extracting knowledge from text in the form of triplets that could then be added to an LOD to further enrich it, using Automatic Knowledge Base construction tools.

Problematic:

The Semantic Web, a term coined by Tim Berners-Lee to describe a web of data that can be processed by machines. One of the most vital issues that's preventing us from creating a fully connected web (semantic web) is how to automatically populate and enrich Knowledge bases with newly extracted facts, as well as extracting said facts from unstructured data such as raw text.

This prompted us to develop a system for extracting information from text with the goal of eventually using it to populate Knowledge Bases, taking us one step closer to realizing the Semantic Web or Web of Data.

Structure of the thesis

Our work is organized as follows:

- Chapter 1 presents the state of the art in the field of Open Information Extraction.
- Chapter 2 details the conception of our system and all its components.
- Chapter 3 describes the implementation of the system as well as its results.
- We finish it off with a general conclusion.

Chapter 1:
State of the art

Chapter 1: State of the art

1.1.Introduction:

Information Extraction is the task of turning the unstructured information expressed in natural language text into a structured representation (Mintz, et al. 2009) in the form of relational tuples consisting of a set of arguments and a semantic relation between them (*arg1, rel, arg2*). Traditional approaches of Information Extraction tend to focus on answering narrow, limited requests over an already defined set of target relations on small, homogeneous data. Doing so relies on taking the target relations in the input, as well as hand-crafted extraction patterns, or patterns learned from hand-labeled training examples. Therefore, shifting to a new domain would require the user to both name the target relations and to manually define the new extraction rules, either that or to annotate the new training data by hand. Thus, these systems require a heavy human involvement.

In order to reduce the human effort needed by the classic IE approaches (Banko, et al. 2007) introduced a new extraction paradigm: Open Information Extraction which -unlike the classic approaches- isn't limited to a restricted, pre-defined set of target relations but rather it extracts all types of relations that could be found in the text. that way, it not only facilitates domain-independency discovery of relations in IE which allows it to scale to large , heterogeneous corpora such as the web, but also reduces the human effort considerably.

In their work, (Banko, et al. 2007) identified three major challenges OpenIE systems are likely to encounter:

- **Automation:** the extraction strategies used in OpenIE systems must be unsupervised. So instead of specifying the target relations in advance, the relations must be automatically found through a single pass over the corpus. In addition, the human effort in creating suitable data or extraction patterns must be reduced to a minimum.
- **Corpus Heterogeneity:** Heterogeneous datasets are one the biggest obstacles that face profound linguistic tools like syntactic or dependency parsers, since they work well when trained and applied to a specific domain. However the main point of OpenIE systems is domain-independency, so deep linguistic tools must be replaced by more shallow parsing methods such as part-of-speech tagging (POS tagging)
- **Efficiency:** OpenIE systems must be able to provide fast extraction over huge datasets in order to readily scale to large amounts of text (as is found in the web).

Alongside citing these major challenges, (Banko, et al. 2007) also released the first OpenIE system TEXTRUNNER, which was the first implemented OpenIE system to treat the challenges mentioned above, paving the way to many more systems to come.

1.2.Types of Open IE systems:

Since the introduction of the task of Open Information Extraction in 2007, a large body of work on it was described. Existing Open IE approaches generally use a set of patterns in order to extract relational tuples from a sentence, each consisting of two arguments and the phrase that expresses the semantic relation between them. These extraction patterns are either hand-crafted or learned from training data.

According to (Niklaus, Cetto, et al. 2018) the types of existing OpenIE approaches can be divided as follows;

1.2.1. Learning-based systems:

We cannot talk about OpenIE systems without starting with the work that kick started everything in this field, TEXTRUNNER (Banko, et al. 2007) which was a self-supervised learning approach that contained three modules. First, from a small sample of sentences from the Penn Treebank, the learner applies its dependency parser to heuristically identify and label a set of extractions as either positive of negative training examples. This labeled data is then used as input to a Naive Bayes classifier which through its learning creates a model of trustworthy relations using un-lexicalized POS and noun phrase (NP) features. The self-supervised nature of the system requires hand-labeling the training data, yet the un-lexicalized features help scale to the different relations that could be found on the web. Second, the extractor generates candidate tuples by identifying pairs of noun phrase arguments then heuristically designating each word between two NPs as a part of a relation phrase or not. Next, each candidate tuple is passed through the classifier, where only those labeled as trustworthy are kept. Finally, a redundancy-based assessor assigns a probability to each "trustworthy" extractions based on the number of sentences from which each extraction was found, thus using the redundancy of information of the web to boost the system, and assigning higher confidence scores to extractions that can be found multiple times.

Three years after the release of TEXTRUNNER came WOE (Wikipedia-based Open Extractor) (Wu et Weld 2010) which was also a self-supervised open information extractor. Its training data comes from Wikipedia by bootstrapping from entries in Wikipedia info-boxes; in other words, by matching the info-box attribute-value pairs to their corresponding sentences in the article. The data is then used to learn extraction patterns on POS tags (WOE^{pos}) and dependency parsers (WOE^{parse}), **Figure 1.** illustrates the system's architecture.

The former extractor uses a linear-chain Conditional Random Field (CRF) to train a model of relations on shallow features that outputs the text between two NPs when it presents a relation. The latter approach, however, uses dependency trees in order to build a classifier that decides whether the shortest path between two NPs is a semantic relation or not. This way, the system is able to capture even long-range dependencies.

In fact, when (Wu et Weld 2010) compared the two approaches, it became apparent that the use of dependency features results in an increase in both precision and recall compared to shallow linguistic features, however, that comes at the expense of the extraction speed, which affects the scalability of the system negatively.

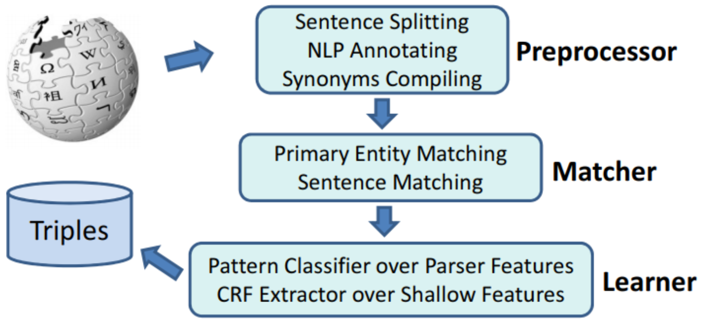


Figure 1. Architecture of WOE (Wu et Weld 2010)

In 2012, (Mausam, Schmitz, et al. 2012) introduced the world to OLLIE which follows the idea of bootstrap learning of patterns based on dependency parse paths. Unlike WOE whose training data came from Wikipedia-based bootstrapping, OLLIE applies a set of high precision seed tuples from its predecessor REVERB(section 2) to bootstrap a large training set, over which it learns extraction pattern templates using dependency parses. As opposed to previous systems that don't take into consideration the context of a tuple and thus can end up with extractions that aren't considered factual, OLLIE includes a context-analysis step, in which it analyzes the contextual information from the input sentence in order to expand the output representation by adding attribution and casual modifiers if necessary, which increases the system's precision, the architecture is illustrated in **Figure 2**.

OLLIE is considered the first OpenIE approach that not only extracts verbal relations but also nominal and adjective-based ones. With that, OLLIE expanded the syntactic scope of relational phrases to cover a wider range of relation expressions, resulting in a higher yield (at comparable precision) compared to previous systems.

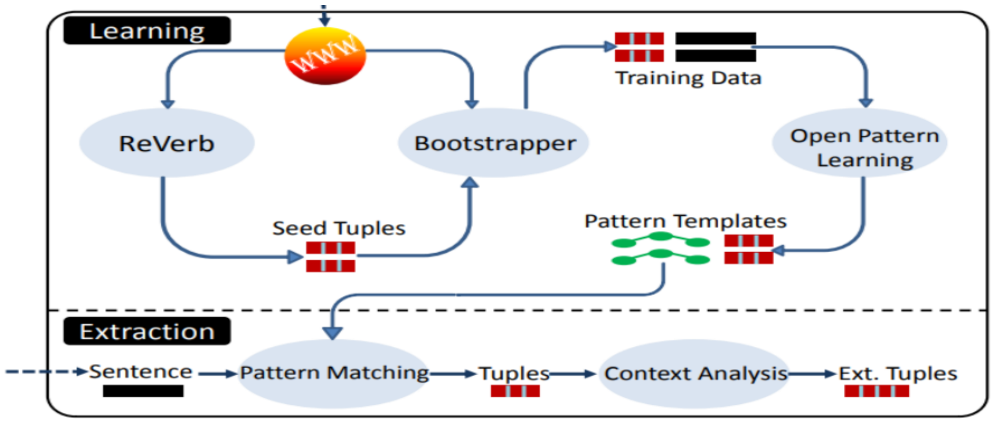


Figure 2. OLLIE's system architecture (Mausam, Schmitz, et al. 2012)

The most recent Learning-based OpenIE system dates back to 2014, when (Yahya, et al. 2014) proposed ReNoun, an OpenIE system that focuses entirely on nominal relations. It starts with a small set of high-precision seed facts that rely on manually specified lexical patterns which are specifically tailored for NPs, the system learns a set of dependency parse patterns for the extraction of nominal relations using distant supervision. These patterns are then utilized to generate a set of candidate extractions that are assigned a confidence score based on the frequency and coherence of the patterns producing them.

1.2.2. Rule-based systems:

The second type of OpenIE systems use Rule-based Reasoning, which consists of hand-crafted extraction rules. The first approach relying on RBR was REVERB (Fader, Soderland et Etzioni 2011), a shallow extractor that addressed three common errors of existing OpenIE systems: the output of these systems usually contains far too many uninformative extractions (ie: extractions that omit critical information), incoherent extractions (ie: extractions where the relational phrase makes no sense) and overly-specific relations that contain too much information which renders them useless in further downstream semantic tasks. REVERB tackled those issues by introducing a syntactic constraint in the form of simple POS-based regular expressions (**Figure 3.**), that covers approximately 85% of verbal relational phrases in English, as was revealed by a linguistic analysis. This way, the system produces less uninformative extractions. Moreover, in order to stray way from over-specified relational phrases, REVERB presents a lexical constraint that's based on the idea that a valid relational phrase should take many arguments in a large corpus. Besides, while previously introduced systems start by identifying the candidate argument pairs, REVERB's approach centers around the relational phrase, so it determines the relational phrase that follows the above mentioned constraints first, then finding the pairs of NP arguments of each relation.

$$
\begin{array}{c}
V \mid VP \mid VW^{*}P \\
V = \text{verb particle? adv?} \\
W = (\text{noun} \mid \text{adj} \mid \text{adv} \mid \text{pron} \mid \text{det}) \\
P = (\text{prep} \mid \text{particle} \mid \text{inf. marker})
\end{array}
$$

Figure 3. ReVerb’s POS-based regular expression for reducing incoherent and uninformative extractions. (Fader, Soderland et Etzioni 2011)

Unlike previously mentioned approaches that focused mainly on extracting binary relations, KRAKEN (Akbik et Löser 2012) was the first approach built to capture complete facts from sentences by gathering the full set of arguments for each relational phrase within it, thus producing tuples of arbitrary arity. The extraction of relational phrases is done by hand-written extraction rules over types dependency parses.

EXEMPLAR (Mesquita, Schmidek et Barbosa 2013) was another system focused on n-ary extraction, it uses a similar approach to that of KRAKEN (Akbik et Löser 2012), relying on hand-crafted patterns based on dependency parse trees in order to detect relation triggers and the arguments related to it. It then assigns each argument its corresponding role using the task of Semantic Role Labeling, that functions by the idea of classifying semantic constituents into different semantic roles (Christensen, et al. 2010)

(Falke, et al. 2016) suggested a more abstract approach with PROPS, they argued that reading out a complete structure of a sentence's propositions from a dependency parse is rather difficult, since, among others, different predications can be represented in a non-uniform manner, and the boundaries of propositions are not easy to detect. Therefore, their approach is a more semantically-oriented sentence representation that's generated by transforming a dependency parse tree into a directed graph tailored to represent the proposition structure of a given input sentence using a rule-based converter. As a result, extracting propositions from this novel approach is more straight-forward.

PredPatt (White, et al. 2016) follows a similar approach, by employing a set of non-lexicalized rules defined over Universal Dependency (UD) parses (Marneffe, et al. 2014) to extract predicate-argument structures. That way, PredPatt builds a directed graph, where a special dependency ARG is built between the head token of a predicate and the head tokens of its arguments, while the original UD relations are preserved within predicate and argument phrases. Since it uses language-agnostic patterns on UD structures, PredPatt is actually one of the few OpenIE systems that can be employed over different languages.

1.2.3. Clause-based systems:

In an effort to improve the accuracy of OpenIE approaches, more recent work is based on the idea of adding the sentence re-structuring stage with the goal of transforming complex sentences into a set of syntactically simplified independent clauses, which are easier to extract OpenIE tuples from. A prime example of such approaches is ClausIE (Del Corro et Gemulla 2013), which first employs linguistic knowledge about English grammar in order to map the dependency relations of input sentence to clause constituents. Second, a combination of knowledge of properties of verbs (with the help of domain-independent lexica) and knowledge about the structure of input clauses is used to determine the type of each clause. Finally, based on its type, one or multiple propositions are generated from each clause, each representing a different piece of information. **Table 1.** shows the basic set of patterns used in ClausIE.

	Pattern	**Clause type**	**Example**	**Derived clauses**
S_1:	SV_i	SV	AE died.	(AE, died)
S_2:	SV_eA	SVA	AE remained in Princeton.	(AE, remained, in Princeton)
S_3:	SV_eC	SVC	AE is smart.	(AE is smart)
S_4:	$SV_{mt}O$	SVO	AE has won the Nobel Prize.	(AE, has won, the Nobel Prize)
S_5:	$SV_{dt}OiO$	SVOO	RSAS gave AE the Nobel Prize.	(RSAS, gave, AE, the Nobel Prize)
S_6:	$SV_{et}OA$	SVOA	The doorman showed AE to his office.	(The doorman, showed, AE, to his office)
S_7:	$SV_{et}OC$	SVOC	AE declared the meeting open.	(AE, declared, the meeting, open)

Table 1. Basic patterns for proposition extraction in ClauseIE (Del Corro et Gemulla 2013)
S: Subject, V: Verb, C: Complement, O: Direct object, A: Adverbial, Vi : Intransitive verb, Vc: Copular verb, Ve: Extended-copular verb, Vmt: Monotransitive verb, Vdt: Ditransitive verb, Vct: Complex-transitive verb

Following a similar path (Schmidek et Barbosa 2014) proposed a strategy to break down structurally complex sentences into simpler ones. The strategy consists of decomposing the sentence into its basic building blocks through chunking, then determining the dependencies of each two chunks either using hand-crafted rules over dependency paths between two words in different chunks, or by using a Naive Bayes classifier trained on shallow features, two chunks can either be "connected", "disconnected" or "independent". Depending on what relation they have, chunks are then combined into simple sentences that are used for the extraction.

(Angeli, Premkumar et Manning 2015) introduced Stanford OpenIE, an approach relying on training a classifier to split a sentence into a set of logically entailed shorter utterances by recursively traversing its dependency tree and predicting at each step if an edge should yield an independent clause or not. With the aim of improving the usefulness of the extracted proposition for downstream applications, each self-contained clause is maximally shortened by running natural logic inference over it. Finally, a set of 14 hand-crafted patterns is used to extract a predicate-argument triple from each utterance. **Figure 4.** depicts the approach.

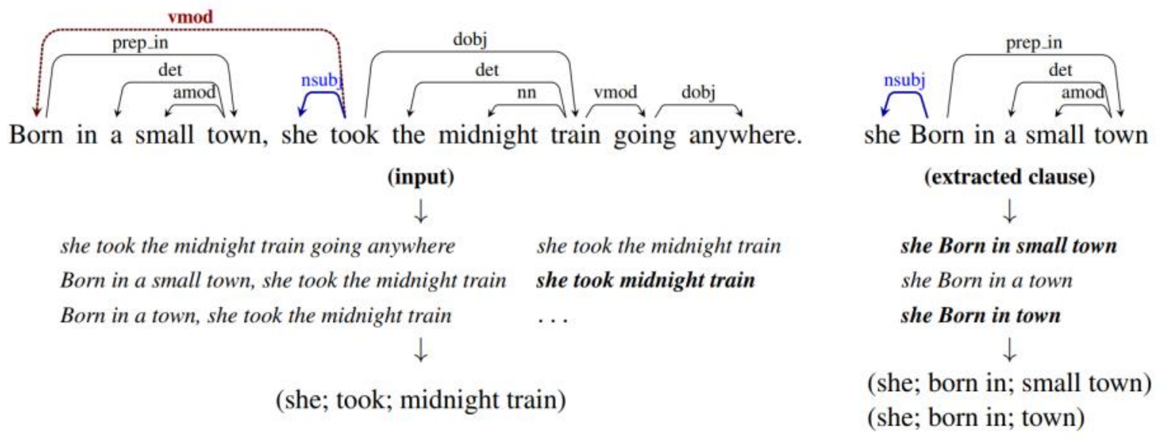

vmod
prep_in
det
amod
nsubj
dobj
det
nn
vmod
dobj
Born in a small town, she took the midnight train going anywhere.
(input)
she took the midnight train going anywhere
Born in a small town, she took the midnight train
Born in a town, she took the midnight train
she took the midnight train
she took midnight train
. . .
(she; took; midnight train)
nsubj
prep_in
det
amod
she Born in a small town
(extracted clause)
she Born in small town
she Born in a town
she Born in town
(she; born in; small town)
(she; born in; town)

Figure 4. Stanford OpenIE's approach
From left to right, a sentence yields a number of independent clauses. From top to bottom, each clause produces a set of entailed shorter utterances, and segments the ones which match an atomic pattern into a relational triple (Angeli, Premkumar et Manning 2015)

1.2.4. Systems capturing Inter-Proposition relationships:

If there is one thing the aforementioned OpenIE systems should be condemned for, it would be their lack of the expressiveness needed to form a proper interpretation of complex sentences, since most of them don't take context into consideration. For example, given the sentence "Early scientists believed that the earth is flat" most systems would extract the tuple (*The earth, is, flat*) from the sentence, except that wouldn't be correct since the input isn't asserting it but merely reporting what early scientists believed. An exception to those early systems was OLLIE (Mausam, Schmitz, et al. 2012) that attempted to treat this issue by extracting an attribution context, denoting a proposition that is reported or claimed by some entity:

(<the earth; is; flat> ;
`AttributedTo` *believe; Early Scientists)*
That way, it extended the default OpenIE representation of (*arg_1, rel, arg_2*) with an extra field. In addition to that, OLLIE also takes into consideration clausal modifiers such as:
(*<He; will be elected; President>;*
`ClausalModifier` *if; he wins five key states*)
Both modifiers are identified by matching patterns with the dependency parse of the sentence. Clausal modifiers are determined by an adverbial-clause edge and filtered lexically (the clause must begin with a term matching a list of cue terms, exp: if, but, when, although). Attribution modifiers however, are identified by a clausal-complement edge whose context verb has to match one of the terms in VerbNet's list of common verbs of communication and cognition.
OLLIE's successor OpenIE4 (Mausam 2016)produces a similar output by combining SRLIE (Christensen, et al. 2010) which converts the output of a Semantic Role Labeling system into an OpenIE extraction by treating the verb as a relational phrase , while taking its role-labeled arguments as the Open IE argument phrases related to the relation, and RelNoun (Pal et Mausam 2016), which is a rule-based OpenIE system that extracts nominal relations. OpenIE4 marks temporal and spatial arguments by assigning them a T or S label, respectively. Its successor OPENIE 5.0 was released afterwards . It utilized BONIE (Saha, Pal et Mausam 2017) and OpenIEListExtractor (swarnaHub 2017). While the former focuses on extracting tuples where one of the arguments is a number or a quantity-unit phrase, the latter targets the extraction of propositions from conjunctive sentences.
(Bhutani, Jagadish et Radev 2016)proposed an approach capturing the inter-proposition relationships, in a nested representation for OpenIE capable of capturing high-level dependencies, which leaves room for a more accurate representation of the meaning of the input sentence. The system, called NestIE uses bootstrapping over a dataset for textual entailment in order to learn binary triple (*arg_1; rel; arg_2*) and nested triple ((*arg_1; rel_1; arg_2*); *rel_2; arg_3*) representations of n-ary relations over dependency parse trees. The system uses manually defined rules in order to infer contextual links between extracted propositions from the dependency parse to generate an nested representation of assertions that are complete and closer in meaning to the original statement.
MinIE (Gashteovski, Gemulla et del Corro 2017) is an OpenIE system that can be considered the successor of ClausIE (Del Corro et Gemulla 2013) as it is built on top of it. ClausIE is often criticized for producing extremely specific extractions which hurt the performance of downstream semantic applications. Therefore, MinIE aims to minimize relational and argument phrases by identifying and removing the overly specific parts, producing maximally shortened, semantically enriched extractions.
To get even more expressive extracted propositions and sustain their interpretability in downstream applications, (Cetto, et al. 2018) propose Graphene, a framework for OpenIE that uses a set of hand-crafted simplification rules to transform complex natural language text into simple, compact structures by removing clauses and phrases

that offer no central information from the input sentence and converting them into stand-alone sentences. Transforming the source sentences into a hierarchical representation in the form of core facts and accompanying contexts (Niklaus, Bremeitinger, et al. 2016), inspired by the work on Rhetorical Structure Theory (Mann et Thompson 1988) a set of lexical and syntactic patterns is used to identify the rhetorical relations that rely the core sentences and their accompanying contexts, to further preserve their semantic relationships and return a set of semantically types and interconnected relational tuples. **Figure 5.** illustrates the extraction workflow.

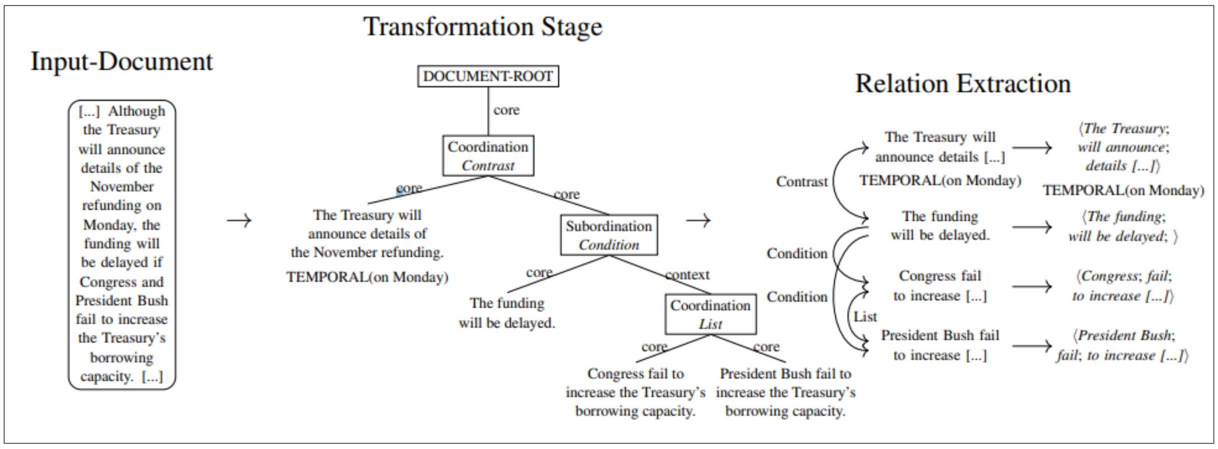

Input-Document
[...] Although the Treasury will announce details of the November refunding on Monday, the funding will be delayed if Congress and President Bush fail to increase the Treasury's borrowing capacity. [...]
Transformation Stage
DOCUMENT-ROOT
core
Coordination Contrast
core
core
The Treasury will announce details of the November refunding.
TEMPORAL(on Monday)
Subordination Condition
core
context
The funding will be delayed.
Coordination List
core
core
Congress fail to increase the Treasury's borrowing capacity.
President Bush fail to increase the Treasury's borrowing capacity.
Relation Extraction
Contrast
The Treasury will announce details [...]
TEMPORAL(on Monday)
⟨The Treasury; will announce; details [...]⟩
TEMPORAL(on Monday)
Condition
The funding will be delayed.
⟨The funding; will be delayed; ⟩
Condition
Congress fail to increase [...]
⟨Congress; fail; to increase [...]⟩
List
President Bush fail to increase [...]
⟨President Bush; fail; to increase [...]⟩

Figure 5. Graphene's extraction workflow for an example sentence (Cetto, et al. 2018)

In the recent years the OpenIE task has been treated using two more approaches that weren't talked about in (Niklaus, Cetto, et al. 2018)'s work;

1.2.5. Neural systems:

While the previous methods ended up giving good results, they still relied on too much human interference from hand-crafted patterns to man-made extraction rules, and they soon reached their limits, which led to the emergence of a new era of fully automatic OpenIE approaches, the neural ones.

(Zhang, Duh et Van Durme 2017) propose a neural sequence to sequence model that enables end-to-end cross-lingual Open Information Extraction.

The idea is to re-cast the problem as a structured translation problem, in which the model encodes natural language sentences and decodes predicate argument forms as is illustrated in **Figure 6**.

The goal is to learn a model which directly maps a sentence input A in the source language (eg: Chinese) into predicate-argument structures output B in the target language (eg: English). Formally, the input is regarded as a sequence: $A = x1,\cdot\cdot\cdot,x|A|$, and a linearized representation of the predicate-argument structure is used as the output sequence $B = y1,\cdot\cdot\cdot,y|B|$. The sequence-to-sequence (Seq2Seq) model consists of an encoder which encodes a sentence input A into a vector representation, and a decoder which learns to decode a sequence of linearized PredPatt (White, et al. 2016) output B conditioned on encoded vector. Results show that this approach outperforms classic approaches on various method, proving that the neural model is crucial for the OpenIE task, especially because of its capability of generating complex OpenIE patterns.

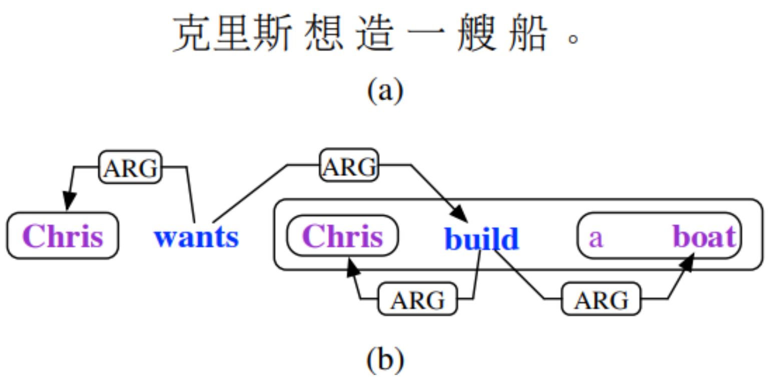


Figure 6. Example of input (a) and output (b) of cross-lingual OpenIE. (Zhang, Duh et Van Durme 2017)

(Stansovsky, Michael, et al. 2018) present a method that enables a supervised learning approach to OpenIE. The model, called RnnOIE, is a bi-LSTM transducer, inspired by the state of the art deep learning approach for SRL as was done by (Zhou et Xu 2015).

Given an input sentence of the form *(S, p)* in which *S* is the input sentence and *p* the word index of the predicate's syntactic head, the feature vector *feat* is extracted for every word $w_i \in S$:

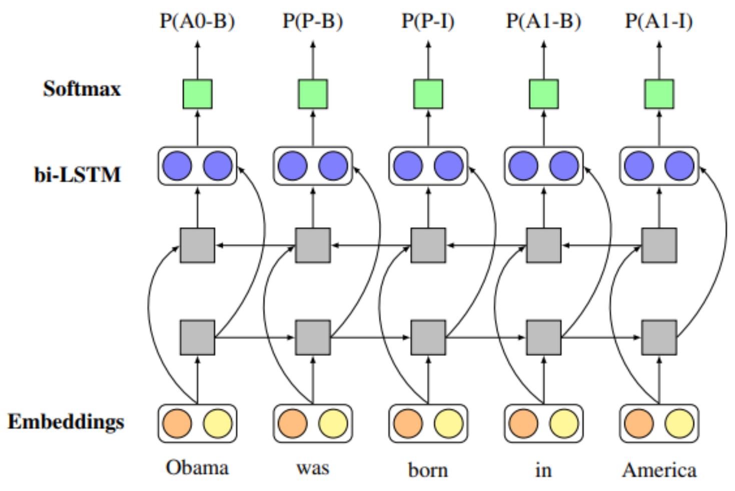


Figure 7. RnnOIE model architecture. (Stansovsky, Michael, et al. 2018)
Orange circles represent current word features: embedding for word and part of speech. Yellow circles represent predicate features, duplicated and concatenated to all other word features.

$feat(wi,p) = emb(w_i) \oplus emb(pos(w_i)) \oplus emb(w_p) \oplus emb(pos(w_p))$

Where

emb(w): a d-dimensional word embedding;

emb(pos(w)): a 5-dimensional embedding of *w*'s part of speech;

$\oplus$: concatenation.

The predicate head's features are duplicated on all words in order to allow the model to access this information more directly as it makes predicate-specific word label predictions. The features are then fed into a bi-directional deep learning LSTM transducer (Graves 2012) that calculates contextualized output embedding. Then, the outputs are used in softmaxes for each word producing independent probability distribution over possible BIO (Beginning, Inside, Outside) tags. **Figure 7.** depicts the system's architecture. RnnOIE uses a POS tagger to identify verbs and Catvar's sub-categorization for normalization in order to identify nouns that share the same frame with a verbal equivalent (e.g. *supposition* with *suppose*), it then generates an input sentence for each candidate predicate head. Each instance then gets each of its words tagged with its most likely BIO label under the model, and reconstruct OpenIE tuples from the resulting sequence.

RnnOIE also provides a confidence score accompanying each extraction that the result of multiplying the probabilities of the B and I labels participating in the extraction. This system overall outperformed the other systems across the datasets. On the larger test sets (OIE2016 and WEB) it provides the best performance in terms of AUC and F1, with a superior precision-recall curve. However it performs best on one metric and competitively on the others on the smaller test sets.

While RnnOIE falls under the *sequence labeling* category of Neural OpenIE methods (Cui, Wei et Zhou 2018) propose an approach that falls under the *sequence generation* category with CopyAttention. In principle, generation is more powerful because it can introduce auxiliary words or change word order. CopyAttention is an encoder-decoder based approach, in which they cast the OpenIE task as a sequence-to-sequence generation problem where the input sequence is the sentence and the output sequence is the tuples with special placeholders. So given the sentences *"Deep learning is a subfield of machine learning"* the output given by the encoder-decoder would be (*<arg1>Deep learning</arg1> <rel>is a subfield of</rel> <arg2>machine learning</arg2>*). The input and output sequence pairs used to train the model are obtained from highly confident tuples bootstrapped from a state-of-the-art OpenIE system. The encoder-decoder framework takes a variable length input sequence to a compressed representation vector that is used by the decoder to generate the output sequence. In this work, both the encoder and decoder are implemented using Recurrent Neural Networks (RNN). The encoder uses a 3-layer stacked LSTM network to convert the input sentences into a set of hidden representations. As for the decoder, it too uses a 3-layer LSTM network to accept the encoder's output and variable-length sequence. The system's architecture is shown in **Figure 8.** To train the model (Cui, Wei et Zhou 2018) used tuples extracted from Wikipedia using OpenIE4 (Mausam 2016) and only keeping tuples whose confidence score is equal or higher than 0.9. As for the test data, they used a large benchmark dataset (Stansovsky et Dagan 2016)that contains 3,200 sentences with 10,359 extractions.

This approach outperformed all tested systems such as OpenIE4 (Mausam 2016), ClausIE (Del Corro et Gemulla 2013), OLLIE (Mausam, Schmitz, et al. 2012), PropS (Falke, et al. 2016) and Stanford OpenIE (Angeli, Premkumar et Manning 2015) with an AUC (Area Under Precision-Recall) score of 0.473 which is significantly better than other systems.

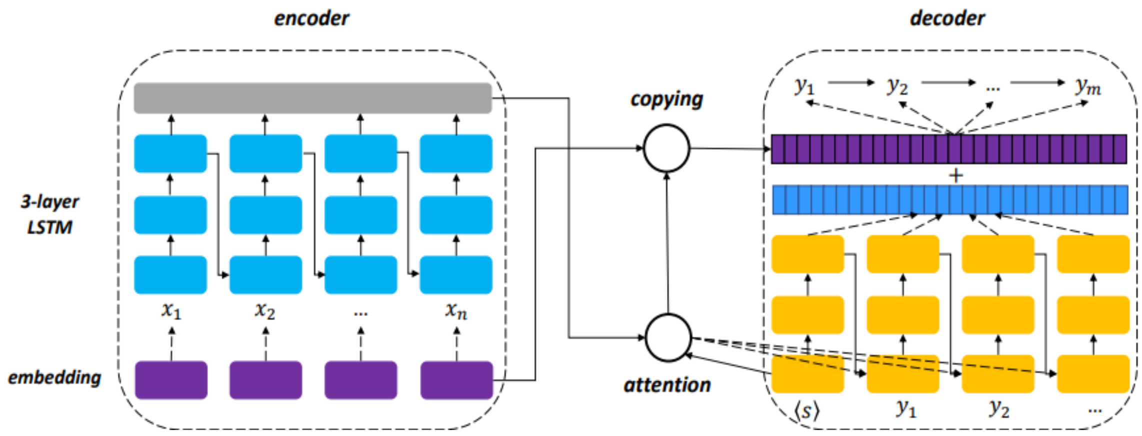


Figure 8. The encoder-decoder model architecture for the neural OpenIE system CopyAttention. (Cui, Wei et Zhou 2018)

And yet, upon further inspection, it turns out CopyAttention suffers from two main drawbacks. The first being its inability to naturally adapt the number of extractions to the length or complexity of the input sentence. The second drawback is that it's susceptible to stuttering: the extraction of multiple triples bearing redundant information. These drawback come from the fact that its decoder has no explicit mechanism to remember what parts of the sentence have already been 'consumed' or what triples have already been generated since it uses a fixed-size beam for inference.

1.2.6. Transformer-based systems:

After the publication of Large Bi-directional Transformer-based Language models such as BERT (Devlin, Chang, et al. 2019) and OpenAI's GPT-2 (Radford, et al. 2019), and more recently GPT-3 (Brown, et al. 2020) took the NLP world by storm, it was only a matter of time until it was utilized in the OpenIE domain. Its use comes in the form of IMOJIE, the first neural OpenIE system that uses sequential decoding of tuples conditioned on previous tuples, stimulating an iterative memory with the help BERT encoder.

IMOJIE (Kolluru, Aggarwal, et al. 2020), or **I**terative **M**em**O**ry **J**oint Open **I**nformation **E**xtraction is a generative approach that can output a variable number of diverse extractions per sentence. At a high level, the next extraction from a sentence is best determined in context of all tuples extracted from it so far. Thus, IMOJIE uses a decoding strategy that generates extractions sequentially (one after another) with each one being aware of all the previous ones. This sequential decoding is possible by the use of an *iterative memory*, where each of the generated extractions is added to the memory so that the next iteration of decoding can access all the ones that came before it. The iterative memory is simulated by using the BERT encoder, whose input includes the [*CLS*] token and original sentence appended with the decoded extractions so far, separated by the token [*SEP*] before each different extraction.

IMOJIE uses an LSTM decoder, which is initialized with the [*CLS*] token. The contextualized-embedding of all word tokens are used for the Copy (Gu, et al. 2016) and Attention (Bahdanau, Cho et Bengio 2016) modules. The decoder generates the tuple one word at a time alongside the tokens <*obj*>, </*obj*> ,<*rel*> and </*rel*> indicating the start and finish of both the objects and relation of the tuple respectively. The extractions only stops once the *EndOfExtraction* token is generated, Figure 9. shows the model's architecture.

The entire extraction process of IMOJIE can be summed up as such:

1. Pass the input sentence through the Seq2Seq architecture to generate the first extraction
2. Concatenate the generated extraction with the existing input then passing it through the Seq2Seq architecture again to generate the next extraction.
3. Repeat step 2 until *EndOfExtraction* token is generated.

The model is trained using a cross-entropy loss between the generated output and the gold output, which is a measure of the difference between two probability distributions for a given variable or set of events, it calculates the number of bits required to represent an average event from one distribution compared to another.

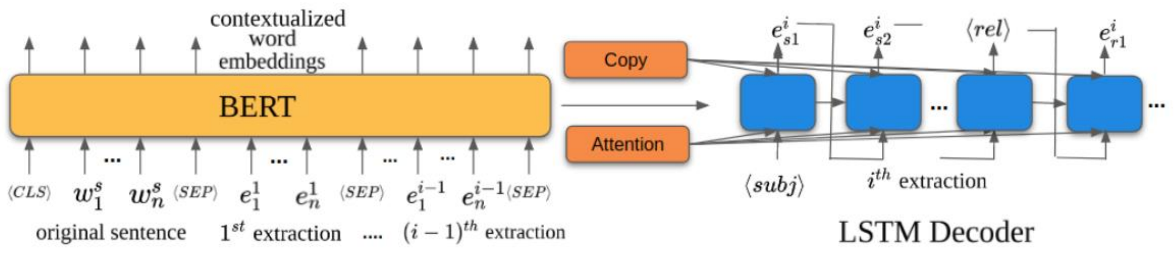


Figure 9. One step of the sequential decoding process, for generating the ith extraction, which takes the original sentence and all extractions numbered 1, . . . , i − 1, previously generated, as input. (Kolluru, Aggarwal, et al. 2020)

Following CopyAttention's example, IMOJIE also uses the bootstrapping method in the training process, using extractions from a pre-existing OpenIE system and labeling them as *'silver'* instances (to distinguish from the *'gold'* extractions that come from human annotation) to train the model. Though unlike CopyAttention, multiple pre-existing OpenIE systems were used in the bootstrapping step rather than just one.

To evaluate the system, the CaRB data and evaluation framework (Bhardwaj, Aggarwal et Mausam 2019) is used. After comparing IMOJIE to several non-neural approaches such as OpenIE4 (Mausam 2016), StanfordIE (Angeli, Premkumar et Manning 2015), OpenIE5, ClausIE (Del Corro et Gemulla 2013), PropS (Falke, et al. 2016), MinIE (Gashteovski, Gemulla et del Corro 2017) and OLLIE (Mausam, Schmitz, et al. 2012), it is apparent that it vastly outperforms them. The system is also compared to the sequence labeling baselines of RnnOIE (Stansovsky, Michael, et al. 2018) and SenseOIE (Roy, et al. 2019), which it also outperforms. Perhaps the most closely related OpenIE approach to IMOJIE is CopyAttention (Cui, Wei et Zhou 2018), which, to increase its diversity they compare against an English version of Logician, which adds coverage attention to a single decoder model that emits all extractions one after another. The system was compared as well to CopyAttention augmented with diverse beam search (Vijayakumar, et al. 2018) which adds diversity term to the loss function so that new beams have smaller redundancy with respect to all previous beams. Finally, since IMOJIE is based on BERT, it was also compared to a re-implemented CopyAttention with a BERT encoder. **Table 2.** shows the results of the comparison of different OpenIE systems against IMOJIE. **Table 3.** shows the results of the comparison between IMOJIE and different versions of CopyAttention, while **Table 4.** shows the results yielded from different combinations of various OpenIE systems in the bootstrapping step.

System	**Metrics**	**Metric**	
	Opt. F1	**AUC**	**Last F1**
StanfordIE	23	13.4	22.9
OLLIE	41.1	22.5	40.9
PropS	31.9	12.6	31.8
MinIE	41.9	- *	41.9
OpenIE-4	51.6	29.5	51.5
OpenIE-5	48.5	25.7	48.5
ClausIE	45.1	22.4	45.1
CopyAttention	35.4	20.4	32.8
RnnOIE	49.2	26.5	49.2
SenseOIE	17.2	- *	17.2
SpanOIE	47.9	- *	47.9
CopyAttention+BERT	51.6	32.8	49.6
IMOJIE	**53.5**	**33.3**	**53.3**

Table 2.Comparison of various OpenIE systems and IMOJIE on the CaRB benchmark (Bhardwaj, Aggarwal et Mausam 2019).

System		**Metric**	
	Opt. F1	**AUC**	**Last F1**
CopyAttention	35.4	20.4	32.8
CoverageAttention	41.8	22.1	41.8
CoverageAttention +BERT	47.9	27.9	47.9
IMOJIE (w/o BERT)	37.9	19.1	36.6
IMOJIE	**53.2**	**33.1**	**52.4**

Table 3. IMOJIE compared to different variations of CopyAttention. (Kolluru, Aggarwal, et al. 2020)

In comparison with existing neural and non-neural systems, IMOJIE trained on aggregated bootstrapped data performs the best. It outperforms OpenIE-4, the best existing OpenIE system, by 1.9 F1 pts, 3.8 pts of AUC, and 1.8 pts of Last-F1. Qualitatively, we find that it makes fewer mistakes than OpenIE-4, probably because OpenIE-4 accumulates errors from upstream parsing module, it also outperforms CopyAttention by large margins (about 18 Optimal F1 pts and 13 AUC pts) as well as its different variations.

Bootstrapping system	**Metric**		
	Opt. F1	**AUC**	**Last F1**
ClausIE	49.2	31.4	45.5
RnnOIE	51.3	31.1	50.8
OpenIE4	53.2	33.1	52.4
OpenIE4+ClauseIE	51.5	32.5	47.1
OpenIE4+RnnOIE	53.1	32.1	53.0
ClausIE+RnnOIE	50.9	32.2	49.8
All	**53.5**	**33.3**	**53.3**

Table 4. trained with different combinations of bootstrapping data from 3 systems - OpenIE-4, ClausIE and RnnOIE. (Kolluru, Aggarwal, et al. 2020)

1.3. Conclusion:

Open Information Extraction is a vast domain, many works have been done to improve its results, from learning-based systems to neural ones. They all aim to overcome the major challenges defined by (Banko, et al. 2007); automation, corpus heterogeneity and efficiency.

As explained in this chapter, every OpenIE method has its advantages and downsides, in fact each approach works better on a different kind of extractions. In many cases, neural systems aren't necessarily better than the rule-based ones and vice-versa. However, performance-wise, we found that the best existing system to this day is IMOJIE, the transformer-based system, as it outperforms all other systems when tested on the same benchmark.

This prompted us to develop a system that would allow us to improve on IMOJIE's state-of-the-art performances even further.

Chapter 2: Conception

Chapter 2: Conception

2.1. Introduction:

In this work we aim to get one step closer to enriching Linked Open Data (namely DBpedia) from natural language text. But to reach the enriching step we must first represent the information or knowledge found in text in a computer-friendly way, and that's where Open Information Extraction steps in. Open Information Extraction allows us to represent information found in text in the form of tuples (*arg1*, *rel*,*arg2*), allowing us to extract structured information from text that can be added to knowledge bases.

2.2.System's Architecture:

Our system *'ParaOpenIE'* takes as input a text and produces a tuple as an output, but to get from the input to the output we must go through a number of steps, **Figure 10**. shows our system's architecture.

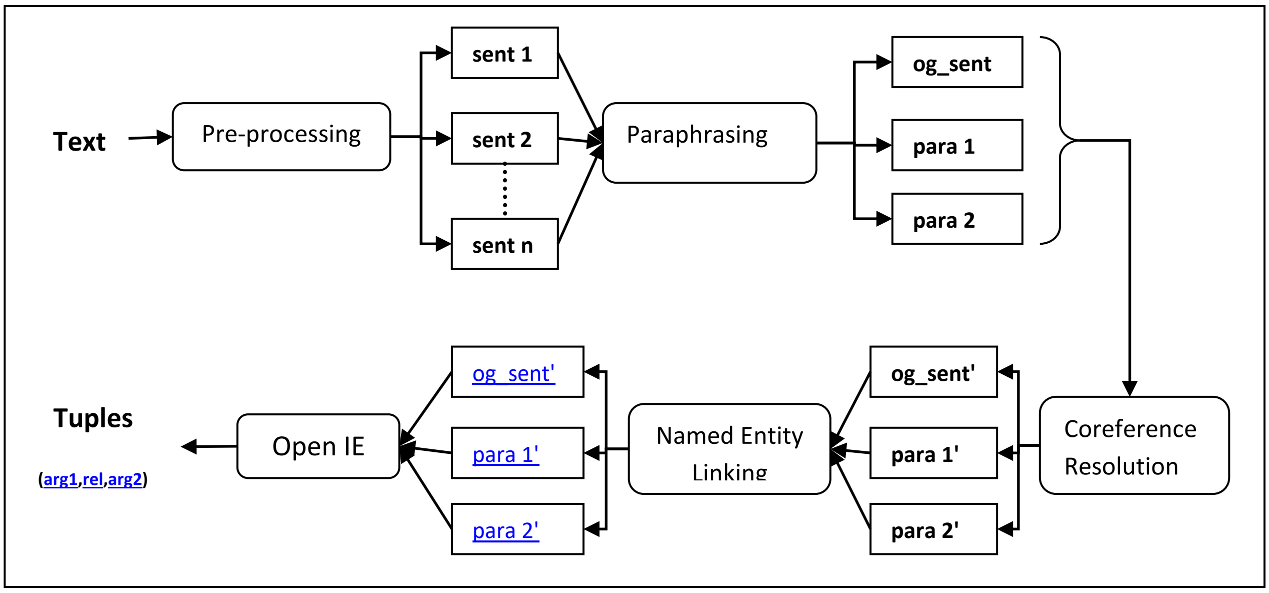


Figure 10. ParaOpenIE's architecture.

In the following portion, We break down the system's architecture step by step.

2.2.1. Text Pre-processing:

The system takes input in the form of text, however information is extracted from it one sentence at a time which makes the step of splitting the text into sentences an essential one.

We use the python library Nltk (or the Natural Language Toolkit) . According to (Wikipedia 2020), Nltk is a suite of libraries and programs for symbolic and statistical Natural Language Processing (NLP) for English. Nltk is written in Python and was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pernsylvania. It includes graphical demonstrations and sample data, and is intended to support research and teaching in the NLP field or closely related areas.

Splitting text into sentences cab be quite tricky especially if one is working with a specific kind of text (usually technical) that contain strange abbreviations or if one is working with a language that doesn't have a pre-trained model. Fortunately, we are using English so we don't have to worry about the lack of pre-trained models. However that leaves the issue of the abbreviations.

If we take the sentence *"I hold a Msc. in Computer Science"* and apply Nltk's `sent_tokenize` function the result would be *['I hold a Msc.' , 'in Computer Science']* since the tokenizer doesn't recognize Msc. as an abbreviation.

On a lower level `sent_tokenize` uses an instance of `PunktSentenceTokenizer` which is an unsupervised trainable model. That means it can be trained using unlabeled data, in other words non-split text. The process of training allows the model to learn abbreviations in the text, as it is the mechanism used by the tokenizer to "cut", training it prevents mistakes such as the one mentioned previously where it split the text because of an abbreviation.

We use the Gutenberg corpus available on nltk, which contains some 25,000 free electronic books such as Jane Austen's Emma, Pride and Prejudice, Shakespeare's works...etc.

To further enrich our tokenizer's collection of abbreviations, we fine-tune it by adding a list of abbreviations that aren't common enough to appear in those books, such as Corp. or Inc.

Once the fine-tuning step is over, we use the tokenizer to split the input text into sentences.

2.2.2. Paraphrasing:

A prominent issue in the OpenIE task is the extraction of implicit information which is the information that isn't explicitly expressed in the standard (*Subject, verb, Object*) format. In an effort to bypass this issue we propose using paraphrasing which is the task of reformulating a sentence while keeping its core meaning.

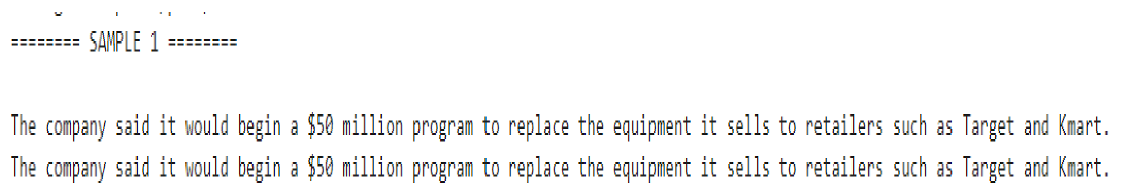
The idea of reformulating, or reconstructing a sentence in order to obtain a better extraction isn't a new one to the OpenIE task, ClausIE (Del Corro et Gemulla 2013) uses grammatical rules to reconstruct complex sentence into smaller, more manageable clauses, (Schmidek et Barbosa 2014) break down structurally complex sentences into simpler ones by decomposing the sentence into its building blocks then determining the dependencies of said blocks using hand-crafted rules. However, instead of following hand crafted patterns or even grammatical rules like previous works did, we opted instead to follow (Witteveen et Andrews 2019)'s steps and use OpenAI's GPT-2 (Radford, et al. 2019) for paraphrasing, which would eliminate the need for techniques such as top-k word or beam search.

GPT-2 is a large scale unsupervised language model that generates coherent paragraphs of text, to achieve its state-of-the-art performance, it was trained to simply predict the next word in 40GB of internet text, the equivalent of about 8 million web pages. It is a large bi-directional transformer-based language model that has 1.5 billion parameters, making it one of the most powerful language models. Its release in fact was so revolutionary that OpenAI didn't release the full version due to concerns about it being employed in malicious applications, they released much smaller versions for researchers to experiment with instead. They first released the 117M version, followed by the 345M version. They eventually released the 775M and the 1.5B versions.

Although GPT-2 was initially trained in a semi-supervised way to predict the next word in a word sequence, it can be fine-tuned and used in a variety of downstream NLP tasks such as sentiment analysis, classifications, tagging...etc.

In order to employ this model in the paraphrasing task, we take the pre-trained GPT-2 model and fine-tune it on a supervised dataset of pre-made paraphrase examples. The examples are fed into the models as sets of (original phrase / paraphrase pairs) separated by a specific identifying sequence. The training is done for a small number of epochs just to allow the model to grasp what the task it's supposed to do is, as to avoid over-fitting the model on the fine-tuning data so it can learn the general pattern of the task.

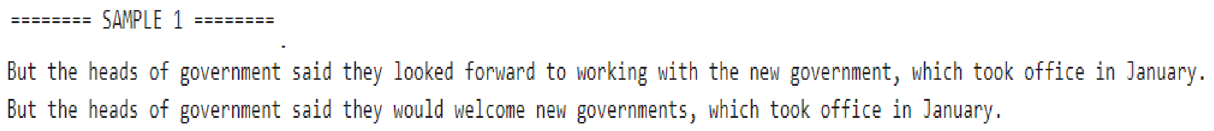
Once the model is fine-tuned, it can produce similar paraphrases to those found in the training dataset if sampled without any conditional input. We sample these paraphrases, that can be called 'naive' paraphrases, throughout the training to see the progression of the model's fine-tuning from one epoch to another, **Figure 11**. shows one of the samples generated in the first epoch, while **Figure 12.** shows a sample from the epoch before last.



```
======== SAMPLE 1 ========

The company said it would begin a $50 million program to replace the equipment it sells to retailers such as Target and Kmart.
The company said it would begin a $50 million program to replace the equipment it sells to retailers such as Target and Kmart.
```

Figure 11. Paraphrasing sample from epoch 1.



```
======== SAMPLE 1 ========

But the heads of government said they looked forward to working with the new government, which took office in January.
But the heads of government said they would welcome new governments, which took office in January.
```

Figure 12. Paraphrasing sample from epoch n-1.

After that, the model can generate paraphrases to any original input sentence it is given.

While the quality of the generated paraphrased phrase can vary from one generation to another, we overcome that by generating multiple outputs to the same input then selecting the best quality paraphrase based on a number criteria that allow to filter the output to a few satisfactory results.

The idea is to select the output that is the most semantically similar to the original, however uses the most varied set of words from those used in the original.

To obtain such a selection we first measure the similarity between the input text and every generated paraphrase by using SentBERT (Reimers et Iryna 2019) to create an embedding for each output sentence then compare each of them to the embedding of the input sentence by calculating the cosine measure between them. SentBERT is a model for sentence embedding based on Google's bi-directional transformer language model BERT (Devlin, Chang, et al. 2019).

SentBERT is a model based on Google's BERT (Devlin, Chang, et al. 2019)that derives semantically meaningful sentence embedding. It improves on BERT by using a siamese network (Bromley, et al. 1994) architecture that enables deriving fixed-sized vectors for input sentences.

Second, we measure the ROUGE-L (Dolan, Quirk et Brokett 2004) score of the outputs against the input sentence and eliminate the generated outputs that score above 0.7 in order to get rid of outputs that are too close to the original in terms of the words used. ROUGE-L helps us finding outputs that are more unique compared to the input.

Finally, we select the 2 outputs with the highest cosine similarities and a ROUGE-L score lower than 0.7, provided that the cosine score is equal or higher than 0.5. If none of the generated outputs meet this criteria, we generate more outputs from the same input sentence to get better quality paraphrases.

The paraphrasing step is very crucial in our OpenIE task as it not only allows us to generate reformulations of our original text that would express implicit information in a more obvious way leading to a richer and more informative extraction, it also automates the process of reformulation, instead of using rule-based and other statistical approaches which performed best on shorter spans of text and require a lot of time and effort to craft and perfect the patterns, it employs the state-of-the-art neural language model GPT-2 (Radford, et al. 2019) and the leader sentence embedding method, the BERT-based model SentBERT (Reimers et Iryna 2019).

2.2.3. Coreference Resolution:

Another prominent issue that may cause low quality extractions in OpenIE is the problem of figuring out what each pronoun used in a text refers to. This task is known as pronoun disambiguation or coreference resolution.

In natural language used by humans, endophoric awareness plays a key part in comprehension skills (decoding), writing skills (encoding) and general linguistic awareness. Endophora is the phenomenon of expressions that derive their reference from something within the surrounding text (Brendan s.d.). It consists of Anaphoric references which happen when a word refers to ideas previously expressed in the text , cataphoric references that occur when a word refers to an idea that is expressed later in the text, and self-references.

The task of coreference resolution in NLP is the equivalent of endophoric awareness in humans. According to Stanford NLP Group, coreference resolution is **"The task of finding all expressions that refer to the same entity in a text"**. We take for example the sentence *"Jake dropped the phone by accident and broke it"*, when asked what the pronoun *"it"* refers to, it is obvious to a human that *"it"* refers to *"the phone"* .

Algorithms that resolve coreferences usually look for the nearest preceding mention that is compatible with the referring expression. A typical coreference resolution algorithm goes as follows:

- A series of *mentions* (words that are potentially referring to real world entities) is extracted
- For each pair of mentions, a set of features is computed.
- The most likely antecedent for each mention (if existent) is selected. This step is called pairwise ranking.

Traditionally, the set of features was hand-engineered from linguistic features and it could be huge. Some high quality systems use upwards of 120 features as is the case with (De Marneffe, Recasens et Potts 2015) and (Moosavi et Strube 2016)'s works. However, a neural network can also be trained for coreference resolution.

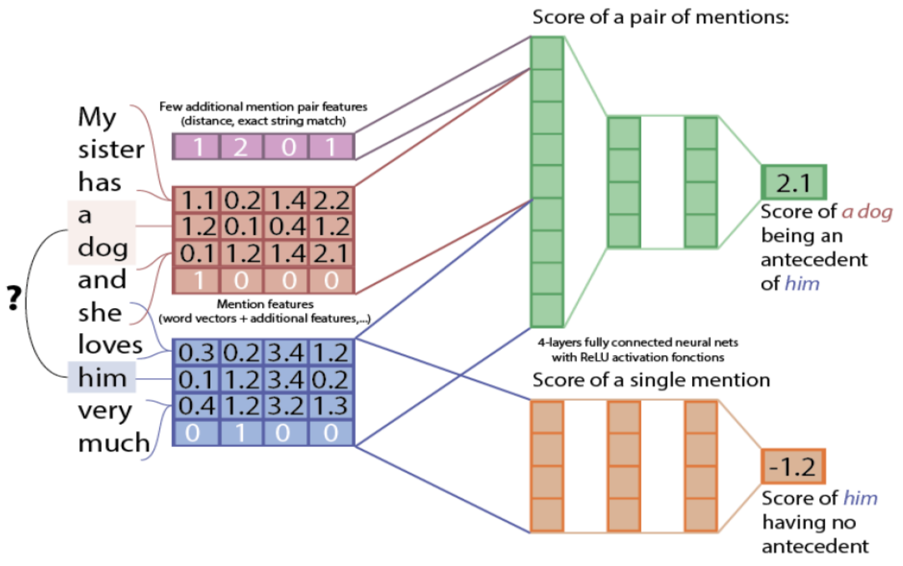

Score of a pair of mentions:
Few additional mention pair features (distance, exact string match)
1 2 0 1
My
sister
has
a
dog
and
she
loves
him
very
much
?
1.1 0.2 1.4 2.2
1.2 0.1 0.4 1.2
0.1 1.2 1.4 2.1
1 0 0 0
Mention features (word vectors + additional features,...)
0.3 0.2 3.4 1.2
0.1 1.2 3.4 0.2
0.4 1.2 3.2 1.3
0 1 0 0
4-layers fully connected neural nets with ReLU activation fonctions
Score of a single mention
2.1
Score of a dog being an antecedent of him
-1.2
Score of him having no antecedent

Figure 13. The coreference scoring model architecture. (Wolf 2017)

In this work we use HuggingFace's coreference resolution system NeuralCoref[1] that's based on neural nets and spaCy. First, the system's word embedding was trained on a large coreference annotated dataset in order to allow it to learn the distinction between gendered words. However, that isn't enough as the system needs to take the context of the mentions of said gendered words, and even external common knowledge for it to yield good results. Therefore, the system then takes word embedding for multiple words inside and around each mention, average them if needed and add some simple integer features to obtain a features representation for each mention and it surroundings. These

[1] https://github.com/huggingface/neuralcoref

representations are then plugged into two neural nets, the first produces a score for each pair of a mention and its possible antecedent and the second produces the score of the mention having no antecedent. The scores are then compared and the highest score is selected to determine if a mention or not and which one should it be. **Figure 13**. shows the scoring model's architecture.

We use NeuralCoref to first determine the antecedent of each mention in our text, and then to replace every mention with its antecedent in order to offer our OpenIE system even more clear and simple sentences resulting in higher quality extractions.

2.2.4. Named Entity Linking:

Since our end goal from this work is to be able to enrich Linked Open Data, DBpedia to be precise, from natural language text, we add the entity linking step to take us even closer to our goal.

Entity Linking, which can also be called Named Entity Linking (NEL), Named Entity Disambiguation (NED), Named Entity Recognition and disambiguation (NERD) or Named Entity Normalization (NEN), is the NLP task of assigning a unique identity (eg: company, city, brand) to entities mentioned in text, usually expressed in the form of a URI. As (Shen, Wang et Han 2014) put it, NEL is the task to link entity mentions in text with their corresponding entities in a knowledge base. The target knowledge base is dependent on the application. In our work we use DBpedia, which is a structured knowledge base extracted from Wikipedia.

There are many libraries available to implement NEL, in this work we used DBpedia Spotlight whose target knowledge base is DBpedia. Spotlight (Daiber, et al. 2013) is deployed as a web service however there is an available API that can be used in python clients.

Named Entity Linking is by no means a trivial task, in fact it is quite difficult due to the name variation and ambiguity problem. Name variation means one entity can be mentioned using different words, while the ambiguity problem means that one word can refer to multiple entities depending on the context it's used in.

Generally, a typical NEL system contains three modules;

- Candidate Entity Generation: in which the system retrieves a set of possible entities called candidates by filtering out the irrelevant ones in the KB. The entities left are the possible entities a mention can be referring to.
- Candidate Entity Ranking: in which the system leverages different kinds of evidence in order to rank the candidates from most likely to least likely entity for the mention.
- Unlinkable Mention Prediction: in which the system validates whether to entity that was ranked most likely candidate is the target entity for the mention at hand. If it isn't, it returns *nil* for the mention, meaning it is not referring an entity that exists in the KB. Basically, this module deals with unlinkable mentions.

In Spotlight's case, it uses Apache Open NLP[2] (Galitsky 2016), which is a machine learning based toolkit for NLP, to identify the entity mention. The disambiguation is done using the generative probabilistic model by (Han et Sun 2011), **Figure 14**. shows its architecture.

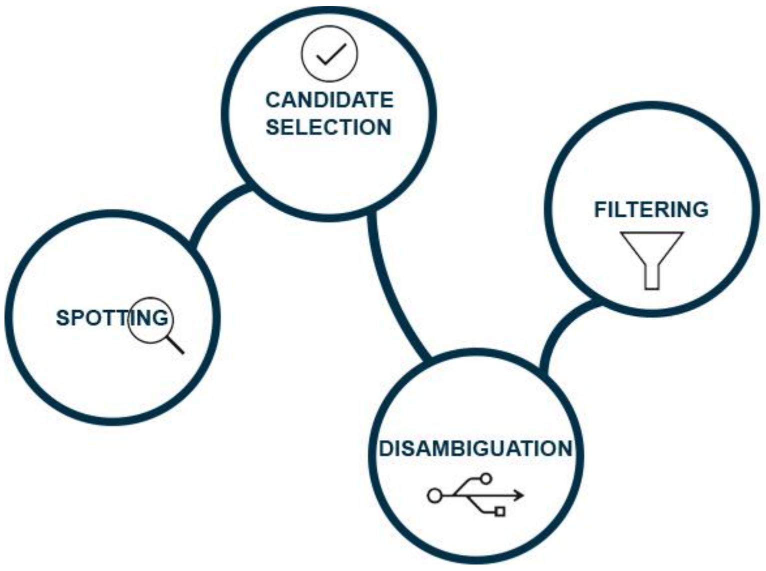


Figure 14. Spotlight's Entity Linking Architecture. (DBpedia 2018)

We use Spotlight to first get the entities, if existent, each mention could possibly refer to with a confidence score of 0.5 or higher, then linking it with the correct one's URI from DBpedia. We apply this on the original sentence as well as its paraphrases.

2.2.5. Open Information Extraction:

Once our natural language text is split into sentences, paraphrased and has its pronouns disambiguated and replaced with their antecedents, we move on to the Open Information Extraction step in which we extract tuples from each original sentence and its accompanying paraphrases, we then keep the extraction whose relation has the highest confidence score.

In this work, rather than using a traditional rules-based or clause-based approach, we opted for a neural transformer-based system, so we follow (Kolluru, Aggarwal, et al. 2020)'s steps and use their generative approach used in the system IMOJIE (**I**terative **M**em**O**ry **J**oint **O**pen **I**nformation **E**xtraction) that can output a variable number of diverse extractions per sentence.

The system uses a decoding strategy which generates extractions sequentially, each extraction being aware of its predecessors. This is achieved by using an iterative memory with the help of BERT encoder. IMOJIE uses an LSTM decoder, which is initialized with the [*CLS*] token, it generates the extractions one word at a time in addition to the <*obj*>, </*obj*>, <*rel*> and </*rel*> tokens.

An overall high level look at how the system works would lead us to summarize it in the following three steps:

- Pass the sentence through the seq2seq module generating the extraction number *n*.

[2] https://opennlp.apache.org/

- Concatenate the generated extraction with the previous *n-1* extractions, then pass it through the seq2seq module again.
- Repeat the second step until the system generates the token *EndOfExtraction*.

At a lower level, IMOJIE uses bootstrapped data to train its LSTM decoder. In other words, instead of using a human-annotated training dataset, we use extractions from pre-existing OpenIE systems. Said extractions are labeled *'silver'* extractions to train the neural model. Since different OpenIE systems have different quality characteristics, we use the combination of extractions from multiple systems as the bootstrapping dataset, allowing us to capture all the qualities found in these extractions.

However simply pooling all the extractions in one dataset and using them doesn't work because of many reasons;

- The confidence scores set by different systems aren't calibrated and thus can't be compared.
- Using multiple systems to extract tuples would result in not only exact replicas, but also extremely similar extraction with a slight difference between them.
- Pooling would inevitably lead to the pollution of the 'silver' dataset with incorrect extractions, resulting in the downstream OpenIE system learning and eventually producing low quality extractions.

To bypass this issue we use the Score-and-Filter framework. **Figure 15.** illustrates its architecture. This method can be split into two steps;

- **Scoring:** Given an input sentence, all systems are applied to it, the extractions are then pooled then scored in way where the good (correct, accurate) extractions are given a high scores while the bad (incorrect, inaccurate) are given lower scores. This score can be set by IMOJIE trained on bootstrapped data from one system. However, such a system would likely be biased to give extractions by the system it was trained on a higher score despite their actual quality. To overcome this, the score is set by a *Random IMOJIE model* which is an IMOJIE model trained on random bootstrapping data. The random bootstrapping data is generated by randomly selecting an extraction for each sentence from any of the bootstrapping system. Each extraction in the pool is then given a confidence score assigned by *Random IMOJIE*.
- **Filtering:** Once the extractions are all given confidence scores, they are filtered for redundancy. The goal is to select the subset extractions with the highest confidence scores and the minimum similarities among them. This goal is modeled by an optimal sub-graph from a suitably designed weighed graph. Where each node in the graph represents one extraction in the pool, while every edge connecting a pair of nodes (u,v) has a weight $R(u,v)$ representing the similarity between the nodes or extractions. Each node u is given a score $f(u)$ representing its confidence score.

  Given the graph $G = (V,E)$ of all pooled extractions of a sentence, our goal is to select the sub-graph $G' = (V',E')$ where $V' \subseteq V$, where the most significant extractions are selected and the ones redundant with the already selected ones are discarded.

  To put it more formally, the objective can be expressed as:

$$max \sum_{i=\&}^{|V'|} f(u_i) - \sum_{j=1}^{|V'|-1} \sum_{k=j+1}^{|V'|} R(u_j, u_k) \qquad (1)$$

Where:

u_i : is the node $i \in V'$.

$R(u,v)$: the ROUGE2[3] (Ganesan 2015) score between the serialized triples represented by the nodes u and v.

Intuitively, the first term can be understood as the aggregated sum of significance of all selected triples, and the second term can be understood as the redundancy between these triples.

If G has n nodes, the objective can be posed as:

$$\max_{x \in \{0,1\}} x^T f - x^T R x \qquad (2)$$

Where:

$f \in R^{n}$: the node scores.

$R \in R^{n \times n}$: A symmetric matrix with entries $R_{j,k}$= ROUGE2(u_j, u_k).

x: the decision vector.

$x[i]$: indicator whether $u_i \in V'$ or not.

This problem is an instance of Quadratic Boolean Programming and is NP-hard, so in this work we employ the QPBO (Quadratic Pseudo Boolean Optimizer) solver[4] (Rother, et al. 2007) to find the optimal decision vector x^* and recover V'.

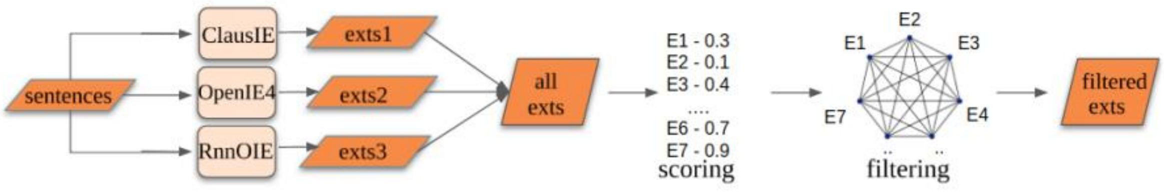


Figure 15. Scoring-Filtering subsystem architecture. ('Exts= extractions.) (Kolluru, Aggarwal, et al. 2020)

The training sentences used to create the bootstrapping data is obtained by scrapping Wikipedia. Due to the diverse nature of its content, using sentences scrapped from Wikipedia ensures that the model would not be biased toward or against a particular domain. However, if we were to work on a specific domain, we could train the model on data specific to that domain to make it more exclusive to its concepts and ideas.

OpenIE-4 (Pal et Mausam 2016), ClausIE (Del Corro et Gemulla 2013) and RnnOIE (Michael, et al. 2018) are run on these sentences to get the initial pooled extractions, which are then scored and filtered then further processed to generate IMOJIE's input-output training instances.

Each one of the sentences is used in multiple (input/output) pairs for the model, the first of which being the original sentence itself and its first tuple output, the rest being the sentence concatenated with all the previous iteration's tuples and that iteration's tuple. The last training instance generated from the sentence includes all the extractions

[3] https://github.com/kavgan/ROUGE-2.0

[4] https://pypi.org/project/thinqpbo/

generated concatenated with the original sentence as input and the token *EndOfExtraction* as output.
While forming these training instances, the tuples are considered in decreasing order of their confidence scores. If some OpenIE system does not provide confidence scores for extracted tuples, then the output order of the tuples may be used.

Once the model is trained, we use it to generate tuples from every sentence as well as its two paraphrases. In order to determine which of the three sentences (*original, paraphrase*1, *paraphrase*2) provides us with the best (most informative and accurate) extraction, we use the neural open relation extraction framework OpenNRE[5] (Han, et al. 2019) to get the confidence score of each of these sentences' relations and select the one with the highest score.
In their article, (Han, et al. 2019) describe OpenNRE as an open-source and extensible toolkit that provides a unified framework to implement neural models for relation extraction.
It doesn't just allow developers to create and train their own models, it also has multiple built-in pre-trained neural relation extraction models that provide real-time relation extraction.
We use the Tacred-BERT module, which is a module based on (Soares, et al. 2019) that uses a BERT encoder that was trained on the TACRED (Zhang, et al. 2017) dataset. TACRED is a large-scale relation extraction dataset with 106,264 examples built over newswire and web text, it contains over 41 relation types.
We pass our sentences through the OpenNRE model which provides us with the relation and its confidence score of each one, we then rank the sentences based on their confidence scores from highest to lowest, and take the tuple extracted from the one with the highest confidence score.
The reason we used OpenNRE rather than IMOJIE's own confidence score is to minimize the system's bias, as it is expected for a system to rank its own extractions relatively high, and since we used other OpenIE systems to bootstrap the training data, their scores would also be subject to a margin of bias. Thus, since the relation is a key component of the tuple, and the OpenNRE model was not used in obtaining the training data, its results would be more objective.

[5] https://github.com/thunlp/OpenNRE

2.3. Conclusion:

ParaOpenIE is a system that utilizes Paraphrasing and Coreference Resolution modules among others to provide better extracted triples from text.
In the next chapter we will detail the implementation of the system as well as its performance and results.

Chapter 3: Experimentation and results

Chapter 3: Experimentation and Results

3.1.Introduction:

After explaining the system's conception, we now move to its realization and performance.

The realization of this entire system consists of first implementing every step on its own and training its module if needed then combining everything together.

3.2. Tools used:

3.2.1. Programming language: Python:

In this work we use Python 3.6.

As (Kuhlman 2012) put it, Python is an interpreted, interactive, object-oriented, open-source programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface. Finally, Python is portable: it runs on many Unix variants, on the Mac, and on Windows 2000 and later.

It's also dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

3.2.2. Platform: Google Colaboratory:

Since our system requires different version of sometimes the same libraries as well as GPU powers, we use Colaboratory (colab for short).

Colab is a platform provided by Google that allows the user to write and execute python code in the browser, it was built on top of Jupyter Notebook. It provides its users with free access to GPUs and TPUs, 13GB of RAM and an access to Google Drive to use it as a storage disk.

3.3.Implementation:

3.3.1. Text Preprocessing:

Dataset:

To train this module we use the Gutenberg corpus which is a dataset made up of all English e-books available on the Gutenberg database. As of May 2020, it contains 62,108 books written by over 142 authors that have been manually cleaned to remove any metadata, license information and transcriber's notes.

Nltk module includes around 25,000 free books from the Gutenberg electronic text archive

Implementation:

To implement this step se use nltk[6] version 3.5, from which we import the `sent_tokenizer` which uses the unsupervised trainable model `PunktSentenceTokenizer`.

We train `PunktSentenceTokenizer` on the Gutenberg corpus in order for it to learn abbreviations so it doesn't split our input text where it shouldn't. Once it is trained we fine-tune it by adding abbreviations that are likely not present in the texts available in the corpus using `tokenizer._params.abbrev_types.add()`. Since we aren't going to use our system on a specific domain, we use Oxford English Dictionary (OED)'s common abbreviations list[7].

3.3.2. Paraphrasing:

Dataset:

In the paraphrasing module we use Microsoft's MSR Paraphrase Corpus[8], a dataset of 5801 pairs of sentences that were collected from thousands of news sources online, no more than one sentence has been extracted from every article to ensure a balanced dataset. In addition to the pairs of sentences, the corpus also includes information like the source of the sentence, its author and even the date it was published in. It also includes a binary human annotation judging whether the pair are semantically similar or not, in other words whether the pairing constitutes a paraphrase or not.

Implementation:

To implement our paraphrasing module, we use the 774M version of OpenAI's GPT-2 as it is the biggest version available for fine-tuning (the 1.5B version can only be used as is) thus has a better chance of producing higher quality outputs.

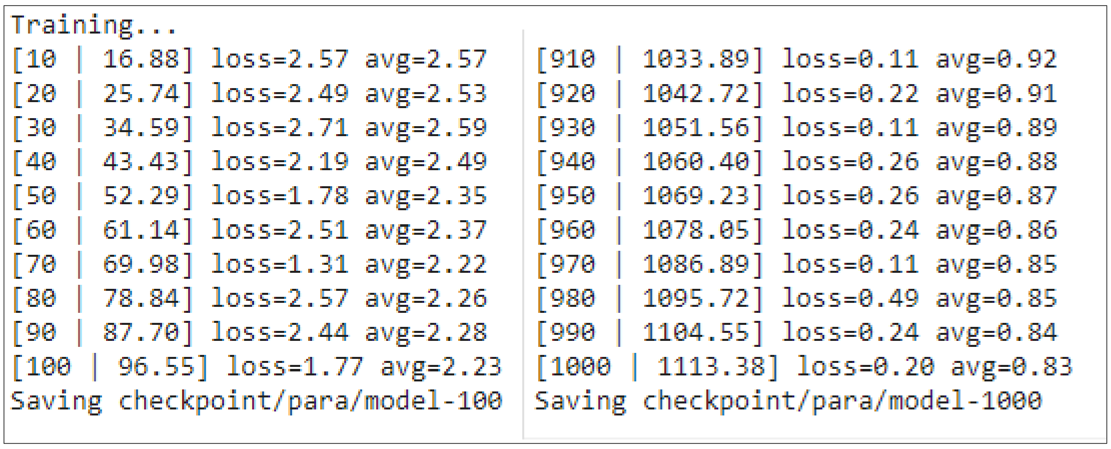
There are multiple python packages that allow us to use and retrain GPT-2 on new text, in this work we used gpt-2-simple[9], a package that wraps existing model fine-tuning and generation for GPT-2. Additionally, it provides an easier method for text generation, generating to a file for easy curation, allowing for prefixes to force the text to start with a given phrase. As for fine-tuning it allows for a faster training because it has the option of using GPU.

The fine-tuning of the model creates a persistent TensorFlow session that stores the training configuration then runs the training for a set number of steps. The checkpoints, which are by default saved in the repository **/checkpoint/run1**, **Figure 16.** shows the difference in the loss function between the first 100 iterations and the last 100 ones.

Some of the most important parameters for `gpt2.finetune()` are:

[6] https://pypi.org/project/nltk/
[7] https://public.oed.com/how-to-use-the-oed/abbreviations/
[8] https://www.microsoft.com/en-us/download/details.aspx?id=52398
[9] https://github.com/minimaxir/gpt-2-simple

- `dataset`: the name of the file containing the training dataset.
- `model_name`: the name of the GPT-2 version used in the fine-tuning (124M, 335M,774M).
- `steps`: the number of steps to run the fine-tuning for (could be set to -1 to have the fine-tuning run indefinitely).
- `restore_from`: set to `fresh` to start the training from the base GPT-2, or to `latest` to continue training from an existing checkpoint.
- `run_name`: the name of the `checkpoint` subfolder in which to save the fine-tuned model.
- `print_every`: the number of steps after which to print the training progress (loss and avg)
- `sample_every`: the number of steps after which to print a sample example output that showcases the training progress.
- `save_every`: number of steps after which to save a checkpoint.



```
Training...
[10 | 16.88] loss=2.57 avg=2.57
[20 | 25.74] loss=2.49 avg=2.53
[30 | 34.59] loss=2.71 avg=2.59
[40 | 43.43] loss=2.19 avg=2.49
[50 | 52.29] loss=1.78 avg=2.35
[60 | 61.14] loss=2.51 avg=2.37
[70 | 69.98] loss=1.31 avg=2.22
[80 | 78.84] loss=2.57 avg=2.26
[90 | 87.70] loss=2.44 avg=2.28
[100 | 96.55] loss=1.77 avg=2.23
Saving checkpoint/para/model-100
```

```
[910 | 1033.89] loss=0.11 avg=0.92
[920 | 1042.72] loss=0.22 avg=0.91
[930 | 1051.56] loss=0.11 avg=0.89
[940 | 1060.40] loss=0.26 avg=0.88
[950 | 1069.23] loss=0.26 avg=0.87
[960 | 1078.05] loss=0.24 avg=0.86
[970 | 1086.89] loss=0.11 avg=0.85
[980 | 1095.72] loss=0.49 avg=0.85
[990 | 1104.55] loss=0.24 avg=0.84
[1000 | 1113.38] loss=0.20 avg=0.83
Saving checkpoint/para/model-1000
```

Figure 16. The loss function of the first and last 100 iterations of fine-tuning GPT-2.

After the model is trained, we can use the saved checkpoint to generate paraphrases from a given input sentences.

To do so, we first load the trained model checkpoint as well as the metadata necessary for text generation using `gpt2.copy_checkpoint_from-gdrive(run_name='x')` 'x' being the name of our saved checkpoint, we then use `gpt2.generate()` to generate an output.

Some of the most important parameters of `gpt2.generate()` are:

- `length`: number of tokens to generate (default 1023, the maximum).
- `temperature`: affects the probability distribution of the next word to choose (best to keep between 0.7 and 1 if you want a coherent output).
- `prefix`: the given input sentence we want paraphrased.
- `nsamples`: the number of outputs to generate.
- `batch_size`: number of samples to generate in parallel.
- `top_k`: limits the generated outputs to the top k guesses.

- `top_p`: limits the generated outputs to a cumulative probability
- `include_prefix`: when set to `false`, the given prefix will not be included in the generated text.

Since we need the generated paraphrases to use later on, we add the `return_as_list` parameter and set it to `true` so that all the generated output is put in a list.

We set the `nsamples` parameter to 5 and the `top_k` to 10 so that our model generates 5 paraphrases of the top 10 samples for each given input.

Once our paraphrases are generated, we generate the embedding of every sentence (original and paraphrase) then compare them.

For sentence embedding we use Sentence Transformers[10] the python framework for dense vector representation for sentences and paragraphs. It offers a list of pre-trained models that are mostly based on transformer networks like BERT.

In this work we use the `distilbert-base-nli-stsb-mean-tokens` model as it is one of the best models on the Semantic Text Similarity task with performance of 85.16% and it has a good balance between speed and performance. `distilbert-base-nli-stsb-mean-tokens` is a sentence embedding model based on DistilBERT (Sanh, et al. 2019) and trained on both the NLI[11] and the STS[12] data.

We generate the embedding vector of each sentence, the original input as well as its 5 paraphrases, then calculate the cosine similarity between the original and each one of the paraphrases, we then eliminate the sentences with a similarity score less than 0.5. We then use the ROUGE-L-Tensorflow package[13], an implementation of the ROUGE-L metric using Tensorflow, to calculate the ROUGE-L metric between the original sentence and each paraphrase. We then only keep those with a score lower than 0.7 which we order by their cosine scores and then select the first and second sentences, if none of the sentences fulfils either the cosine or the ROUGE-L requirements, we generate another batch of paraphrases for the same input sentence.

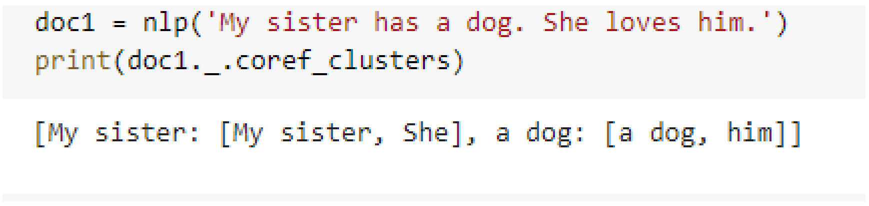
3.3.3. Coreference Resolution:

To solve the coreference issue we use HuggingFace's NeuralCoref[14] the pipeline extension for scpaCy for neural coreference resolution.

In addition to installing the neuralCoref library, we install spaCy and load the relevant coreference model, since we will be working with text from all kinds of sources we chose the large model. Then we add neuralCoref to spaCy's pipeline of annotations. Once that is done, we first use the `doc._.coref_clusters` method that returns a list of all

[10] https://github.com/UKPLab/sentence-transformers
[11] https://nlp.stanford.edu/projects/snli/
[12] http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark
[13] https://github.com/VD44/Rouge-L-Tensorflow
[14] https://github.com/huggingface/neuralcoref

corefering mentions in the given text and their main mention, **Figure 17.** shows an example of that list:



```
doc1 = nlp('My sister has a dog. She loves him.')
print(doc1._.coref_clusters)
```

```
[My sister: [My sister, She], a dog: [a dog, him]]
```

Figure 17. Example of the list of the corefering mentions and their main mention in the given text.

then we use the `doc._.coref_resolved` method that replaces each corefering mention with the main mention in the associated cluster.

We pass every triple of (*original sentence*, *paraphrase1*, *paraphrase2*) through the coreference resolution module and keep a trace of all the mentions that were replaced.

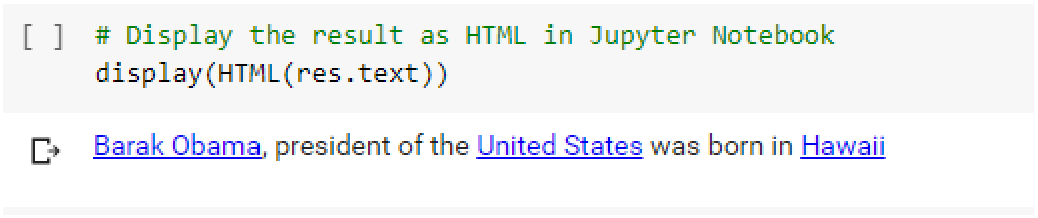
3.3.4. Named Entity Linking:

In the named entity linking step we use DBpedia's Spotlight REST API, since it's deployed as a web service we can use it directly with no need for a special python package.

We perform the NEL task through a GET request that needs three parameters:

- `base_url`: the link for spotlight API (http://api.dbpedia-spotlight.org/en/annotate).
- `params`: the parameters which are the text to link and the confidence score for linking.
- `headers`: the response content type.

We pass our sentences through the NEL module to get the DBpedia URI for every available mention in our sentences, **Figure 18.** shows an example of a sentences after being passed through the NEL module displayed as HTML.



```
[ ]  # Display the result as HTML in Jupyter Notebook
     display(HTML(res.text))
```

Barak Obama, president of the United States was born in Hawaii

Figure 18. Sentence after the NEL step, displayed in HTML.

3.3.5. Open Information Extraction:

Dataset:

The training data for this module is obtained by scraping Wikipedia, as it is a source of diverse informative text, then extracting tuples from this text using previous OpenIE systems and scoring and filtering them to select the best extraction which will then be used to train the model. The OpenIE systems used to create the pool of extraction are: OpenIE-4[15], ClausIE[16] and RnnOIE[17].

Implementation:

IMOJIE was implemented using the AllenNLP the deep semantic NLP framework[18] (Gardner, et al. 2018) using PyTorch 1.2 and "BERT-small" with a learning rate of $2 \times 10^{-5}$for faster training.

Since it would be counter-intuitive to re-implement the entire system, we use the IMOJIE package[19] to train the model on the bootstrapped, scrapped Wikipedia data. We train our model for 8 epochs as to allow it to learn the task at hand but not overfit it on the training data. **Table 5**. shows the loss, AUC and F1 scores across the training epochs.

	Epoch1	**Epoch2**	**Epoch3**	**Epoch4**	**Epoch5**	**Epoch6**	**Epoch7**	**Epoch8**
Loss	1.2379	0.4890	0.467	0.462	0.461	0.454	0.447	0.435
AUC	0.230	0.312	29.8	0.335	0.330	0.341	0.336	0.339
F1	0.431	0.527	51.5	0.550	0.554	0.548	0.546	0.548

Table 5. Progression of the system's loss, AUC and F1 scores throughout the training

Once the model is trained, we save its latest checkpoint to then use it to generate extractions.

However, before we get into the testing of the system, we first pass the testing dataset through all the previous steps of this system (coreference resolution, paraphrasing...etc), as a result we get a dataset that contains the sentences and its two paraphrases.

Next, we use the OpenNRE package[20] for neural relation extraction. We import one of its pre-trained models and use it to extract relations from our sentences.

We use the `tacred_bert_softmax` model which has a BERT encoder and was trained on the TACRED dataset.

To get our extraction, after uploading the model we simply use the instruction `model.infer()`that takes as parameter the sentence and returns its relation as well as its confidence score.

[15] https://github.com/knowitall/openie
[16] https://www.mpi-inf.mpg.de/clausie
[17] https://github.com/gabrielStanovsky/supervised-oie
[18] https://github.com/allenai/allennlp
[19] https://github.com/dair-iitd/imojie
[20] https://github.com/thunlp/OpenNRE

For every triplet (*sentence, paraphrase1, paraphrase2*) we pick the one with the highest confidence score and add it to our new dataset containing the best scoring sentence of each triplet. We then use this new dataset to extract triples from our OpenIE model.

3.4.Testing and Performance:

3.4.1. Dataset:

Comparing the performance of an OpenIE system against others can be very tricky especially since not only does each system work differently, some are rules-based others are neural others are clause-based, but also every one of these systems produces a different format of output. Which lead us to use the CaRB (**C**rowdsourced **a**utomatic open **R**elation extraction **B**enchmark) benchmark[21] (Bhardwaj, Aggarwal et Mausam 2019) which is a dataset cum evaluation framework for OpenIE systems, it can support different OpenIE output formats like ClausIE, OLLIE, OpenIE-4...etc or even simple tab format file where each line consists of (*sent, prob, pred, agr1,arg2,...*) making it easier to not only evaluate an OpenIE system, but also compare it against previous ones.

The benchmark provides a test sentences dataset that contains 641 sentences, we pass this dataset through our OpenIE system and store the output in a .txt file, then depending on our output's format, we can get an evaluation.

3.4.2. Implementation:

Before using the test sentences dataset to test our OpenIE system, we first pass it through our paraphrasing, coreference resolution and named entity linking modules, then we use both our IMOJIE model and OpenNRE one to generate extractions and select the best one every time. The generated extractions are then compared to a set of gold reference OpenIE extractions created and annotated by humans.

We calculate three important summary metrics from the precision-recall curve yielded by the benchmark, and use them as basis for our evaluation;

- Optimal F1: the point in the P-R curve that corresponds to the largest F1 value. It is the operating point for generating extractions with the best precision-recall trade-off.
- AUC: the area under the P-R curve, useful with downstream applications that use a confidence value of extraction
- Last F1: the last F1 score calculated at the point of zero confidence, it is important when we can't compute the optimal threshold because of a lack of any gold extractions for the domain.

[21] https://github.com/dair-iitd/CaRB

3.4.3. Results:

Testing our system on the CaRB benchmark provides promising results. Our system produces extractions that are closer to the gold extractions than the previous models and even IMOJIE without the paraphrasing and coreference resolution steps. **Table 6**. shows the system's performance compared to other OpenIE systems.

System	Metrics		
	Opt. F1	AUC	Last F1
StanfordIE	23	13.4	22.9
OLLIE	41.1	22.5	40.9
PropS	31.9	12.6	31.8
MinIE	41.9	- *	41.9
OpenIE-4	51.6	29.5	51.5
OpenIE-5	48.5	25.7	48.5
ClausIE	45.1	22.4	45.1
CopyAttention	35.4	20.4	32.8
RnnOIE	49.2	26.5	49.2
SenseOIE	17.2	- *	17.2
SpanOIE	47.9	- *	47.9
CopyAttention+BERT	51.6	32.8	49.6
IMOJIE	53.5	33.3	53.3
ParaOpenIE	**64.7**	**45.6**	**53.5**

Table 6. Performance of ParaOpenIE compared to other OpenIE systems.

As shown in **Table 6**. we notice that our system performs better than not only the non-neural OpenIE systems like OLLIE and ClausIE, but the neural and transformer-based ones on all three metrics.

It outperforms OpenIE-4, the best existing non-neural OpenIE model with an astounding 13.1 points of Optimal F1, 16.1 points of AUC and 2 points of Last F1.

As for the neural approaches, our system also outperforms CopyAttention with 29.3 points of Optimal F1, 25.4 points of AUC and 20.7 points of Last F1. We also compare our system to the CopyAttention model with the BERT encoder implemented in (Kolluru, Aggarwal, et al. 2020)'s work and it outperforms it with 13.2 points of Optimal F1, 12.8 points of AUC and 3.9 points of Last F1.

Lastly, we compare our system to IMOJIE, the state of the art OpenIE system, which it outperforms with 11.2 points of Optimal F1, 12.3 points of AUC and 0.2 points of Last F1.

3.5. Discussion:

After implementing and testing our system *'ParaOpenIE'* we find that adding the paraphrasing and the coreferences resolution steps takes us a long way into getting better quality extractions from natural language text.

These results can be attributed to the following arguments; First, paraphrasing allows reformulating sentences and thus expressing hidden information in text in a more direct way, it also reduces the number of nominal relations the system has to extract from allowing for more informative and accurate extractions.

Second, the coreference resolution helps solving disambiguaties that could be found in text due to the use of pronouns or even phrases that refer to things mentioned either way before it or that are still to come, which allows the OpenIE systems to extract tuples like (*Steve Jobs, created, Apple*) instead of (*He, created, Apple*). This is beneficial later on in enriching Linked Open Data because the arguments are more distinguishable than pronouns.

3.6. Conclusion

Our system, *'ParaOpenIE'*, achieved very promising results. It outperformed all existent state-of-the-art OpenIE systems with a high margin. Meaning that the exploration of this particular approach could take us very close to creating a Web of Data.

Conclusion and Perspectives

In this work we developed an Open Information Extraction system named '*ParaOpenIE*' with the goal of getting one step close to enriching Linked Open Data from natural language text.

Our system is based on a transformer-based OpenIE approach, with the addition of a Paraphrasing module using large language models, a neural coreference resolution module as well as a named entity linking module.

Testing the system has shown that the addition of these modules improves the quality of the extractions with a large margin, which would eventually lead to more informative and accurate tuples being used to enrich Knowledge Bases.

ParaOpenIE has scored 11.2 pts of Optimal F1, 12.3 pts of AUC and 0.2 pts of Last F1 better than the state-of-the-art OpenIE system, outperforming it on all metrics it was tested on.

We aim to improve this work by :

- Using an automatic way to choose the threshold of both the cosine similarity and the ROUGE-L score rather than having pre-defined thresholds.
- Trying different language models as well as the 1.5B version of GPT-2 to see if we can get better quality of the paraphrases.
- Improve the OpenIE system's performance.
- Use the system to enrich DBpedia from text scrapped from the web.

Bibliography

Akbik, Alan, et Alexander Löser. «KrakeN: N-ary Facts in Open Information Extraction.» *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, June 2012: 52-56.

Angeli, Gabor, Melvin Jose Johnson Premkumar, et Christopher D. Manning. «Leveraging linguistic structure for open domain information extraction.» *In Proceedings of the 53rd Annual Meeting of the Association for*, July 2015: 344-354.

Bahdanau, Dzmitty, Kyunghyun Cho, et Yoshua Bengio. «Neural Machine Translation by Jointly Learning to Align and Translate.» *Accepted at ICLR 2015 as oral presentation*, May 2016.

Banko, Michele, Alexander Yates, Matthew Broadhead, Michael Cafarella, Oren Etzioni, et Stephen Soderland. «TextRunner: Open Information Extraction on the web.» *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, April 2007: 25-26.

Bhardwaj, Sangnie, Samarth Aggarwal, et Mausam Mausam. «CaRB: A Crowdsourced Benchmark for Open IE.» *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, November 2019: 6262-6267.

Bhutani, Nikita, H. V. Jagadish, et Dragomir Radev. «Nested Propositions in Open Information Extraction.» *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, November 2016: 55-64.

Brendan, Gillon. *Grammatical Structure and its Interpretation: An Introduction to natural Language Semantics.* MIT Press.

Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger, et Roopak Shah. «Signature verification using a "Siamese" time delay neural network.» *Advances in Neural Information Processing Systems 6*, 1994: 737-744.

Brown, Tom B., et al. «Language Models are Few-Shot Learners.» *arXiv:2005.14165v4*, May 2020.

Cetto, Matthias, Christina Niklaus, André Freitas, et Seigfried Handschuh. «Graphene: Semantically-Linked Propositions in Open Information Extraction.» *Proceedings of the 27th International Conference on Computational Linguistics*, August 2018: 2300-2311.

Christensen, Janara, Mausam, Stephen Soderlan, et Oren Etzioni. «Semantic Role Labeling for Open Information Extraction.» *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, June 2010: 52-60.

Cui, Lei, Furu Wei, et Ming Zhou. «Neural Open Information Extraction.» *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, July 2018: 407-413.

Daiber, Joachim, Max Jakob, Chris Hokamp, et Pablo N. Mendes. «Improving Efficiency and Accuracy in Multilingual Entity Extraction.» *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*, September 2013: 121-124.

DBpedia. *DBpedia Spotlight.* 2018. https://www.dbpedia-spotlight.org/ (accès le 2020).

De Marneffe, Marie-Catherine, Marta Recasens, et Christopher Potts. «Modeling the Lifespan of Discourse Entities with Application to Coreference Resolution.» *Journal of Artificial Intelligence Research 52*, March 2015: 445-475.

Del Corro, Luciano, et Rainer Gemulla. «ClausIE: Clause-based Open Information Extraction.» *In Proceedings of the 22Nd International Conference on World Wide Web*, 2013: 355-366.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, et Kristina Toutanova. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.» *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, June 2019: 4171-4186.

Dolan, Bill, Chris Quirk, et Chris Brokett. «Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources.» *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, August 2004: 350-356.

Fader, Anthony, Stephen Soderland, et Oren Etzioni. «Identifying Relations for Open Information Extraction.» *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, July 2011: 1535-1545.

Falke, Tobias, Gabriel Stanovsky, Iryna Gurevych, et Ido Dagan. «Porting an Open Information Extraction System from English to German.» *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, November 2016: 892-898.

Galitsky, Boris. «A Deep Analysis System Based on OpenNLP.» *ApacheCon Europe*, November 2016.

Ganesan, Kavita. «ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks.» 2015.

Gardner, Matt, et al. «AllenNLP: A Deep Semantic Natural Language Processing Platform.» *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, July 2018: 1-6.

Gashteovski, Kiril, Rainer Gemulla, et Luciano del Corro. «MinIE: Minimizing Facts in Open Information Extraction.» *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, September 2017: 2630-2640.

Graves, Alex. «Sequence Transduction with Recurrent Neural Networks.» *arXiv preprint arXiv:1211.3711*, 2012.

Gu, Jiatao, Zhengdong Lu, Hang Li, et Victor O.K. Li. «Incorporating Copying Mechanism in Sequence-to-Sequence Learning.» *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, August 2016: 1631-1640.

Han, Xianpei, et Le Sun. «A generative entity-mention model for linking entities with knowledge base.» *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, June 2011: 945-954.

Han, Xu, Tinyu Gao, Yuan Yao, Deming Ye, Zhiyuan Liu, et Maosong Sun. «OpenNRE: An Open and Extensible Toolkit for Neural Relation Extraction.» *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, November 2019: 169-174.

Kolluru, Keshav, Samarth Aggarwal, Vipul Rathore, Mausam, et Soumen Chakrabarti. «IMoJIE: Iterative Memory-Based Joint Open Information Extraction.» *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, July 2020: 5871-5886.

Kuhlman, Dave. *A Python Book: Beginning Python, Advanced Python, and Python Exercises.* 2012.

Mann, William C., et Sandra A. Thompson. «Rhetorical structure theory: Toward a functional theory of text.» *Text-Interdisciplinary Journal for the Study of Discourse*, 1988: 243-281.

Marneffe, Marie-Catherine, et al. «Universal Stanford dependencies: A cross-linguistic typology.» *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, May 2014: 4585-4592.

Mausam. «Open Information Extraction Systems and Downstream Applications.» *In Proceedings of the TwentyFifth International Joint Conference on Artificial Intelligence, IJCAI 2016*, 9-15 July 2016: 4047-4077.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, et Oren Etzioni. «Open Language Learning for Information Extraction.» *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, July 2012: 523-534.

Mesquita, Filipe, Jordan Schmidek, et Denilson Barbosa. «Effectiveness and Efficiency of Open Relation Extraction.» *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, October 2013: 447-457.

Michael, Julian, Gabriel Stanovsky, Luheng He, Ido Dagan, et Luke Zettlemoyer. «Crowdsourcing Question-Answer Meaning Representations.» *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, June 2018: 560-568.

Mintz, Mike, Steven Bills, Rion Snow, et Daniel Jurafsky. «Distant supervision for relation extraction without labeled data.» *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, August 2009: 1003-1011.

Moosavi, Nafise Sadat, et Michael Strube. «Search Space Pruning: A Simple Solution for Better Coreference Resolvers.» *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2016: 1005-1011.

Niklaus, Christina, Bernhard Bremeitinger, Seigfried Handschuh, et André Freitas. «A Sentence Simplification System for Improving Relation Extraction.» *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, Dece:ber 2016: 170-174.

Niklaus, Christina, Matthias Cetto, André Freitas, et Seigfried Handschuh. «A Survey on Open Information Extraction.» *Proceedings of the 27th International Conference on Computational Linguistics*, August 2018: 3866-3878.

Pal, Harinder, et Mausam. «Demonyms and Compound Relational Nouns in Nominal Open IE.» *In Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, 2016: 35-39.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, et Ilya Sutskever. «Language Models are Unsupervised Multitask Learners.» 2019.

Radford, et al. «Language Models are Unsupervised Multitask Learners.» 2019.

Reimers, Nils, et Gurevych Iryna. «Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.» *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, November 2019: 3982-3992.

Rother, Carsten, Vladimir Kolmogorov, Victor S. Lempitsky, et Martin Szummer. «Optimizing Binary MRFs via Extended Roof Duality.» *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007: 1-8.

Roy, Arpita, Youngja Park, Taesung Lee, et Shimei Pan. «Supervising Unsupervised Open Information Extraction Models.» *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, November 2019: 728-737.

Saha, Swarnadeep, Harinder Pal, et Mausam. «Bootstrapping for Numerical OpenIE.» *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, July 2017: 317-323.

Sanh, Victor, Lysandre Debut, Julien Chaumond, et Thomas Wolf. «DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.» *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS 2019*, Febuary 2019.

Schmidek, Jordan, et Denilson Barbosa. «Improving Open Relation Extraction via Sentence Re-structuring.» *In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014).*, 2014.

Shen, Wei, Jiayong Wang, et Jiawei Han. « Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions.» *IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 2*, Febuary 2014: 443-460.

Soares, Livio Baldini, Nicholas FitzGerald, Jeffery Ling, et Tom Kwiatkowski. «Matching the Blanks: Distributional Similarity for Relation Learning.» *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, July 2019: 2895-2905.

Stansovsky, Gabriel, et Ido Dagan. «Creating a Large Benchmark for Open Information Extraction.» *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, November 2016: 2300-2305.

Stansovsky, Gabriel, Julian Michael, Luke Zettlemoyer, et Ido Dagan. «Supervised Open Information Extraction.» *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, June 2018: 885-895.

swarnaHub. *Github.* 10 Junuary 2017. https://github.com/dair-iitd/OpenIE-standalone (accessed in 2020).

Vijayakumar, Ashwin K., et al. «Diverse Beam Search for Improved Description of Complex Scenes.» *In AAAI Conference on Artificial Intelligence*, 2018: 7371-7379.

White, Aaron Steven, et al. «Universal Decompositional Semantics on Universal Dependencies.» *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, November 2016: 1713-1723.

Wikipedia. *Natural Language Tooklit.* 15 April 2020. https://en.wikipedia.org/wiki/Natural_Language_Toolkit (accessed in September 2020).

Witteveen, Sam, et Martin Andrews. «Paraphrasing with Large Language Models.» *Proceedings of the 3rd Workshop on Neural Generation and Translation*, November 2019: 215-220.

Wolf, Thomas. «State-of-the-art neural coreference resolution for chatbots.» *Medium.* 07 July 2017. https://medium.com/huggingface/state-of-the-art-neural-coreference-resolution-for-chatbots-3302365dcf30 (accessed in June 2020).

Wu, Fei, et Daniel S. Weld. «Open Information Extraction Using Wikipedia.» *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, July 2010: 118-127.

Yahya, Mohamed, Steven Whang, Rahul Gupta, et Alon Halevy. «ReNoun: Fact Extraction for Nominal Attributes.» *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, October 2014: 325-335.

Zhang, Sheng, Kevin Duh, et Benjamin Van Durme. «MT/IE: Cross-lingual Open Information Extraction with Neural Sequence-to-Sequence Models.» *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, April 2017: 64-70.

Zhang, Yuhao, Victor Zhong, Danqi Chen, Gabor Angeli, et Christopher D. Manning. «Position-aware Attention and Supervised Data Improve Slot Filling.» *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, September 2017: 35-45.

Zhou, Jie, et Wei Xu. «End-to-end learning of semantic role labeling using recurrent neural networks.» *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, July 2015: 1127-1137.