ClausIE: Clause-Based Open Information Extraction

Luciano Del Corro
Max-Planck-Institute für Informatik
Saarbrücken, Germany
corogg@mpi-inf.mpg.de

Rainer Gemulla
Max-Planck-Institute für Informatik
Saarbrücken, Germany
rgemulla@mpi-inf.mpg.de

ABSTRACT

We propose ClausIE, a novel, clause-based approach to open information extraction, which extracts relations and their arguments from natural language text. ClausIE fundamentally differs from previous approaches in that it separates the detection of "useful" pieces of information expressed in a sentence from their representation in terms of extractions. In more detail, ClausIE exploits linguistic knowledge about the grammar of the English language to first detect clauses in an input sentence and to subsequently identify the type of each clause according to the grammatical function of its constituents. Based on this information, ClausIE is able to generate high-precision extractions; the representation of these extractions can be flexibly customized to the underlying application. ClausIE is based on dependency parsing and a small set of domain-independent lexica, operates sentence by sentence without any post-processing, and requires no training data (whether labeled or unlabeled). Our experimental study on various real-world datasets suggests that ClausIE obtains higher recall and higher precision than existing approaches, both on high-quality text as well as on noisy text as found in the web.

Categories and Subject Descriptors

I.2.7 [**Computing Methodologies**]: Artificial intelligence-Natural language processing

Keywords

Open information extraction; relation extraction

1 Introduction

Open information extraction (OIE) aims to obtain a shallow semantic representation of large amounts of natural-language text in the form of verbs (or verbal phrases) and their arguments [1, 2, 3, 5, 7, 9, 12, 10, 13, 19]. The key goals of OIE are (1) domain independence, (2) unsupervised extraction, and (3) scalability to large amounts of text. Most OIE techniques do not require any background knowledge or manually labeled training data; they are therefore not limited to a set of pre-specified relations or entities. For these reasons, OIE has gained significant traction in the recent past.

Consider for example the sentence "*A. Einstein, who was born in Ulm, has won the Nobel Prize*". OIE systems aim to extract triples ("*A. Einstein*", "*has won*", "*the Nobel Prize*") and ("*A. Einstein*", "*was born in*", "*Ulm*") from this sentence; no entity resolution or disambiguation of the verbal phrase is performed. We call each extraction a *proposition*; each proposition consists of a *subject* ("*A. Einstein*"), a relational phrase or simply *relation* ("*has won*"), and zero, one, or more *arguments* ("*the Nobel Prize*"). The extracted propositions can be used directly for shallow semantic querying (Who has won the Nobel Prize?), and—potentially combined with other techniques such as entity resolution—may serve as input for richer tasks such as targeted information extraction, semantic role labeling, coreference resolution, or ontology extension. Consider for example the task of extending a given ontology about persons and their prizes. Entity resolution techniques may identify both Albert Einstein and the Nobel Prize in the above sentence, OIE techniques establish the connection (Albert Einstein, "*has won*", Nobel Prize) between these entities, and ontology extension techniques such as [17] use statistical reasoning to try to obtain the fully disambiguated fact HasWon(Albert Einstein, Nobel Prize).

Virtually all of existing OIE techniques make use of hand-crafted extraction heuristics or automatically constructed training data to learn extractors or estimate the confidence of propositions. Some approaches—such as TextRunner [3], WOE$^{\text{pos}}$ [19], Reverb [9], and R2A2 [7]—focus on efficiency by restricting syntactic analysis to part-of-speech tagging and chunking. These fast extractors usually obtain high precision for high-confidence propositions, i.e., at low points of recall, but the restriction to shallow syntactic analysis limits maximum recall and/or may lead to a significant drop of precision at higher points of recall. Other approaches—such as Wanderlust [1], WOE$^{\text{parse}}$ [19], KrakeN [2], OLLIE [13], and [10]—additionally use dependency parsing. These extractors are generally more expensive than the extractors above; they trade efficiency for improved precision and recall. Each of these approaches makes use of various heuristics to obtain propositions from the dependency parses.

In this paper, we propose a novel approach to OIE called *ClausIE*[1] (for clause-based open information extraction), which falls into the second category, i.e., it also makes use of dependency parsing. ClausIE fundamentally differs from previous approaches in that it separates (i) the *detection*

[1]ClausIE's source code and other resources are available at http://www.mpi-inf.mpg.de/departments/d5/software/clausie.

of "useful" pieces of information expressed in a sentence from (ii) its *representation* in terms of one or more propositions. The main reasoning behind this separation is that (i) can be addressed accurately and in a principled way by exploiting properties of the English language. In particular, we identify the set of "clauses" of each sentence and, for each clause, the corresponding clause type according to the grammatical function of its constituent (e.g., subject-verb-object, SVO). Our detection of clauses is based on the dependency parse; to detect clause types, we additionally use a small set of domain-independent lexica (e.g., of copular verbs). In contrast to many previous approaches, ClausIE does not make use of any training data, whether labeled or automatically constructed, and does not require global post-processing (e.g., to filter out low-precision extractions), i.e., sentence processing in ClausIE is embarrassingly parallel. These properties allow ClausIE to process both individual sentences as well as large document collections automatically and in a scalable way.

Once clauses have been detected, we generate one or more propositions for each clause based on the type of the clause; the generation of propositions can be customized to the underlying application. For example, from the clause "*Anna passed the exam with ease*", we may want to generate one or more of the following propositions: ("*Anna*", "*passed the exam with*", "*ease*"), ("*Anna*", "*passed*", "*the exam with ease*"), ("*Anna*", "*passed*", "*the exam*"), or 4-tuple ("*Anna*", "*passed*", "*the exam*", "*with ease*"?). Moreover, ClausIE can (optionally) extract propositions in which the subject or one or more of the arguments do not constitute a noun phrase. For example, ClausIE may generate from "*A.E. from Ulm is vegetarian*" propositions ("*A.E.*", "*is*", "*vegetarian*") and/or ("*A.E. from Ulm*", "*is*", "*vegetarian*"), if so desired.

We conducted an experimental study on multiple real-world datasets of varying quality in order to compare ClausIE to alternative techniques. We found that ClausIE obtains significantly more propositions than previous approaches at similar or higher precision. In particular, ClausIE produced 2.5–3.5 times more correct propositions than OLLIE, which was the best-performing alternative extractor in our experiments. Most of the extraction errors performed by ClausIE were due to incorrect parse trees; here post-processing techniques—e.g., similar to the statistical techniques employed by Reverb [9] or OLLIE [13]—may help to further increase precision.

The remainder of this paper is structured as follows: In Sec. 2, we briefly summarize related work in the areas of OIE, semantic role labeling, and ontology construction. Sec. 3 establishes the connection between clauses and OIE. Our ClausIE extractor is described in Sec. 4. We present results of our experimental study in Sec. 5 and conclude in Sec. 6.

2 Related Work

The task of open information extraction was introduced by the seminal work of Banko et al. [3], who also proposed the TextRunner OIE system. A number of techniques have been developed to improve on TextRunner. At a high level, all of these approaches make use of a set of *patterns* in order to obtain propositions. Depending on the specific approach, these patterns are either hand-crafted (based on on various heuristics) or learned from automatically generated training data (e.g., in the form of a classifier); the patterns apply to part-of-speech (POS) tags, chunks, or dependency parses (DP). Most of the existing approaches aim to extract *triples*, i.e., propositions of form (subject, relation, argument); extraction of higher-arity propositions are handled by [2, 5].

As mentioned previously, there are two major categories of OIE systems: approaches that make use of only shallow syntactic parsing, and approaches that apply heavier NLP technology. TextRunner [3] belongs to the former class. It first trains a Bayes classifier based on DPs of 1000s of sentences in an offline phase; the classifier is then applied to efficiently extract propositions in an online phase. WOE$^{\text{pos}}$ [19] also uses a classifier, but the classifier is based on a high-quality training corpus obtained automatically from Wikipedia for improved precision and recall. Reverb [9] is the perhaps simplest (and thus very attractive) shallow extractor; it makes use of syntactical and lexical constraints that aim to reduce the amount of uninformative, incoherent, and over-specified extractions (see Sec. 3). Finally, R2A2 [7] uses a number of classifiers to identify the arguments of a verbal phrase (based on hand-labeled training data), and is able to extract propositions that contain arguments that are not noun phrases. R2A2 is the best-performing shallow OIE extractor to date. ClausIE is significantly slower than all of the above techniques, but produces high-quality extractions that can potentially be used as training data for systems such as R2A2.

The second category of OIE systems makes use of dependency parsing [1, 19, 2, 13, 10]. Some systems use either hand-labeled (Wanderlust [1]) or automatically generated (WOE$^{\text{parse}}$ [19], OLLIE [13]) training data to learn extraction patterns on the dependency tree. Other approaches (KrakeN [2] and [10, 20]) use a set of hand-crafted patterns on the dependency parse. In contrast to all existing approaches, ClausIE reasons about the set of clauses (and their types) that appear in an input sentence; this reasoning is based on the dependency parse and a small set of domain-independent lexica. Most of the patterns of [2, 10, 20] are naturally captured by ClausIE. Moreover, ClausIE can be customized to output triples or n-ary facts, to focus on either noun-phrase or more general subjects and arguments, or to flexibly adjust how much information is included in the relational phrase and how much in its arguments.

OIE is the perhaps simplest form of semantic analysis. A closely related and more general problem is semantic role labeling (SRL), which aims to identify arguments of verbs as well as their semantic roles. Christensen et al. [5] have shown that SRL can be used to increase the precision and recall of OIE; however, existing SRL systems heavily rely on manually labeled training data and are generally more compute-intensive than dependency parsing. Both SRL and OIE focus mainly on verb-mediated propositions. Although ClausIE does identify non-verb-mediated propositions to a very limited extent, specialized techniques—such as [18] for extracting the "is-a" relationship—go significantly further. OIE can also be seen as a first step towards richer semantic analysis. Patty [14], for example, aims to extract a set of typed lexical patterns that are indicative of a relation. Many techniques for automated ontology construction (e.g., [4, 17]) are also based on lexical patterns, but in contrast to Patty focus on a prespecified set of ontological relations. Since OIE readily identifies relations and their arguments in text, a combination of OIE with these techniques appears promising.

Table 1: **Patterns and clause types (based on [15]).**

	Pattern	Clause type	Example	Derived clauses
		Basic patterns		
S_1:	$\mathrm{SV_i}$	SV	AE died.	(AE, died)
S_2:	$\mathrm{SV_eA}$	SVA	AE remained in Princeton.	(AE, remained, in Princeton)
S_3:	$\mathrm{SV_cC}$	SVC	AE is smart.	(AE, is, smart)
S_4:	$\mathrm{SV_{mt}O}$	SVO	AE has won the Nobel Prize.	(AE, has won, the Nobel Prize)
S_5:	$\mathrm{SV_{dt}O_iO}$	SVOO	RSAS gave AE the Nobel Prize.	(RSAS, gave, AE, the Nobel Prize)
S_6:	$\mathrm{SV_{ct}OA}$	SVOA	The doorman showed AE to his office.	(The doorman, showed, AE, to his office)
S_7:	$\mathrm{SV_{ct}OC}$	SVOC	AE declared the meeting open.	(AE, declared, the meeting, open)
		Some extended patterns		
S_8:	$\mathrm{SV_i}AA$	SV	AE died in Princeton in 1955.	(AE, died)
				(AE, died, in Princeton)
				(AE, died, in 1955)
				(AE, died, in Princeton, in 1955)
S_9:	$\mathrm{SV_eA}A$	SVA	AE remained in Princeton until his death.	(AE, remained, in Princeton)
				(AE, remained, in Princeton, until his death)
S_{10}:	$\mathrm{SV_cC}A$	SVC	AE is a scientist of the 20th century.	(AE, is, a scientist)
				(AE, is, a scientist, of the 20th century)
S_{11}:	$\mathrm{SV_{mt}O}A$	SVO	AE has won the Nobel Prize in 1921.	(AE, has won, the Nobel Prize)
				(AE, has won, the Nobel Prize, in 1921)
S_{12}:	$A\mathrm{SV_{mt}O}$	SVO	In 1921, AE has won the Nobel Prize.	(AE, has won, the Nobel Prize)
				(AE, has won, the Nobel Prize, in 1921)

S: Subject, V: Verb, C: Complement, O: Direct object, $\mathrm{O_i}$: Indirect object, A: Adverbial, $\mathrm{V_i}$: Intransitive verb, $\mathrm{V_c}$: Copular verb, $\mathrm{V_c}$: Extended-copular verb, $\mathrm{V_{mt}}$: Monotransitive verb, V_{dt}: Ditransitive verb, $\mathrm{V_{ct}}$: Complex-transitive verb

3 The Seven Clauses

We first establish the connection between clauses, clause types, and OIE; a description of ClausIE can be found in Sec. 4.

A *clause* is a part of a sentence that expresses some coherent piece of information; it consists of one *subject* (S), one *verb* (V), and optionally of an *indirect object* (O), a *direct object* (O), a *complement* (C), and one or more *adverbials* (A). Not all combinations of these constituents appear in the English language. In fact, when clauses are classified according to the grammatical function of their constituents, we obtain only seven different clause types [15].[2] For example, the sentence "*AE has won the Nobel Prize*" is of type SVO; here "*AE*" is the subject, "*has won*" the verb, and "*the Nobel Prize*" the object. A complete list of all seven clause types is given in the upper part of Table 1.

Assume for the moment that the input sentence consists of only a single clause. ClausIE is based on the observation that the clause type conveys the minimal unit of coherent information in the clause. Intuitively, this means that if we remove a constituent of a clause that is also part of its type, the resulting clause does not carry semantically meaningful information (or the sense of the verb changes). For example, the sentence "*AE remained in Princeton*" consists of a subject, a verb, and an adverbial. The clause is of type SVA, i.e., the clause "*AE remained*" obtained by ignoring the adverbial is incoherent (and indeed semantically meaningless). In contrast, clause "*AE died in Princeton*"—which also consists of a subject, a verb, and an adverbial—is of type SV. Since here the adverbial does not appear in the clause type, the derived clause "AE died" is coherent. In what follows, we call constituents of a clause that are also part of the clause type *essential* (here "*AE*" and "*died*"); all other constituents are called *optional* ("*in Princeton*"). Note that subjects, verbs, (direct and indirect) objects, and complements are always essential; adverbials, however, may or may not be essential.

Coherence plays an important role in OIE. For example, Reverb [9] employs heuristic rules in order to avoid (some) incoherent extractions. ClausIE ultimately aims to generate propositions from the constituents of the clause. Coherency tells us which constituents must be included into a proposition and which may be omitted. One option to ensure coherent extractions is to always construct propositions that include all constituents of a clause. Such an approach addresses coherency, but—as argued by [9]—may in turn lead to over-specified extractions. Consider, for example, sentence "*AE was awarded the NP in Sweden in 1921*" and suppose we limit attention to noun-phrase arguments; such an approach is followed by most OIE systems. We can then extract coherent propositions

P_1=("*AE*", "*was awarded*", "*the NP*"),
P_2=("*AE*", "*was awarded the NP in*", "*Sweden*"),
P_3=("*AE*", "*was awarded the NP in*", "*1921*"),
P_4=("*AE*", "*was awarded the NP in Sweden in*", "*1921*").

Here P_4 (and perhaps P_2 and P_3) is over-specified in that phrase "*was awarded the Nobel Prize in Sweden in*" is not relational. Since ClausIE detects essential and optional constituents of a clause, we can customize proposition generation as desired; coherency is always guaranteed. One potential customization—which we also used in our experimental study—is to extract all coherent propositions in combination

[2]There is also an existential clause (such as this one), which we treat similarly to SV.

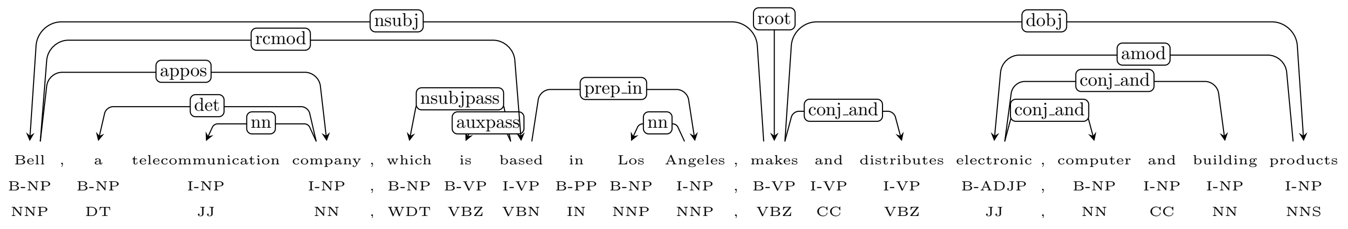

nsubj
rcmod
appos
det
nn
nsubjpass
auxpass
prep_in
nn
root
conj_and
dobj
amod
conj_and
conj_and
Bell , a telecommunication company , which is based in Los Angeles , makes and distributes electronic , computer and building products
B-NP , B-NP I-NP I-NP , B-NP B-VP I-VP B-PP B-NP I-NP , B-VP I-VP I-VP B-ADJP , B-NP I-NP I-NP I-NP
NNP , DT JJ NN , WDT VBZ VBN IN NNP NNP , VBZ CC VBZ JJ , NN CC NN NNS

Figure 1: An example sentence with dependency parse, chunks, and POS tags (chunks by Apache OpenNLP)

with zero or one optional adverbial. With this approach, we extract P_1, P_2, and P_3, but not P_4 from the sentence above.[3] Heuristic approaches such as Reverb do not allow for such flexibility (in our example, Reverb extracts P_2 only). As a final note, over-specificity can also arise in subjects, objects, and complements; here the dependency parse can be exploited to address over-specificity in a natural way, and ClausIE does so to a limited extent.

Given a clause, we can (in principle) determine its type. First observe that each occurrence of a verb in an English sentence is of exactly one of the following types: intransitive, (extended) copular, monotransitive, ditransitive, or complex transitive. For example, a verb is *intransitive* if it does not take an object argument, *monotransitive* if it takes a direct object, and *ditransitive* if it takes both a direct and an indirect object. As another example, *copular* verbs link the subject with a complement or predicative, while *extended-copular* verbs express a relation between the subject and an adverbial [15]. As can be seen in Table 1, which also gives an example sentence for each verb type, the verb type along with the presence of a direct object, indirect object, or complement *uniquely* identifies the type of a clause. Vice versa, the verb type is uniquely determined by the (type of the) constituents and the type of the clause. We exploit this observation directly in ClausIE, i.e., we exploit information about the clause obtained from the dependency parse, and information about verb types from a small set of-domain independent lexica. In many cases, this combined approach allows us to accurately determine the clause type; see Sec. 4.

If a sentence contains multiple (potentially nested) clauses, ClausIE considers each clause separately. Consider, for example, sentence "*AE was awarded the NP before Schrödinger devised his famous thought experiment.*". The sentence contains two clauses (one spanning the entire sentence, and one starting at "*Schrödinger*"); coherent propositions include ("*AE*", "*was awarded*", "*the NP*") and ("*Schrödinger*", "*devised*", "*his famous thought experiment*"). OIE does not aim to capture the "context" of each clause; this simplification allows for effective extraction but may also lead to non-factual extractions [13]. For example, the proposition ("*the only real valuable thing*", "*is*", "*intuition*") obtained from the second clause of sentence "*AE said the only real valuable thing is intuition*" is non-factual. We do not specifically avoid non-factual propositions in ClausIE; see [13] for techniques that can detect such propositions.

[3]ClausIE may also be customized to extract n-tuple ("*AE*", "*was awarded*", "*the NP*", "*in Sweden*"?, "*in 1921*"?), where "?" indicates optional arguments.

4 ClausIE

We now describe how we obtain and subsequently exploit clauses and clause types in ClausIE. For each input sentence, ClausIE conducts the following steps:

1. Compute the DP of the sentence (Sec. 4.1).
2. Determine the set of clauses using the DP (Sec. 4.2)
3. For each clause, determine the set of coherent derived clauses based on the DP and small, domain-independent lexica (Sec. 4.3).
4. Generate propositions from (a subset of) the coherent clauses (Sec. 4.4).

The overall runtime of ClausIE is dominated by dependency parsing in step 1; steps 2–4 are inexpensive.

4.1 Step 1: Dependency Parsing

ClausIE makes use of the unlexicalized Stanford dependency parser [11] to discover the syntactical structure of an input sentence. The DP consists of a set of directed syntactic relations between the words in the sentence. The root of the DP is either a non-copular verb or the subject complement of a copular verb. For instance, in sentence "*Messi plays football*", word "*plays*" forms the root of DP; it is connected to "*Messi*" via a subject relation (`nsubj`) and to "*football*" via the direct-object relation (`dobj`). A more complex example is shown in Fig. 1; a complete list of relations can be found in [6].

4.2 Step 2: From Dependencies to Clauses

We first identify the clauses in the input sentence, i.e., we aim to obtain the head word of all the constituents of each clause. For example, we obtain (S: *Bell*, V: *makes*, O: *products*) for the main clause of the sentence shown in Fig. 1. We use a simple mapping of dependency relations to clause constituents. First, we construct a clause for every subject dependency in the DP (e.g., `nsubj`); the dependant constitutes the subject (S) and the governor the verb (V).[4] All other constituents of the clause are dependants of the verb: objects (O) and complements (C) via `dobj`, `iobj`, `xcomp`, or `ccomp`; and adverbials (A) via dependency relations such as `advmod`, `advcl`, or `prep_in`.

To improve recall and informativeness of extractions, ClausIE additionally creates a number of "synthetic clauses", i.e., clauses that do not directly appear in the sentence. In subsequent steps, these synthetic clauses are treated in the same

[4]Except for the SVC clause type. Here the governor of the subject dependency is the complement (C), and both verb and adverbials are dependants of the complement.

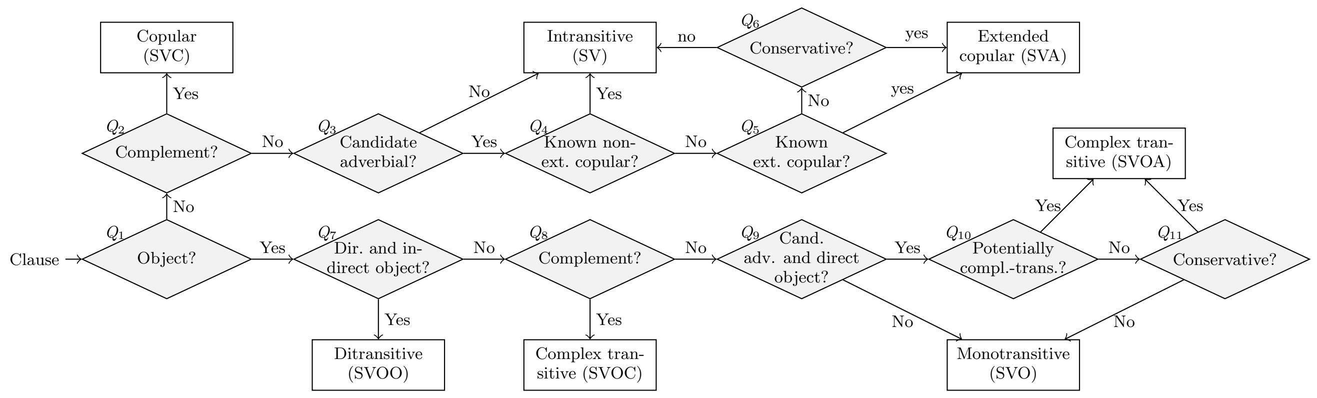


Figure 2: Flow chart for verb-type and clause-type detection

way as the actual clauses of the sentence. As discussed below, the constituents of a synthetic clause either refer to a word in the DP or correspond to an artificially created verb. In more detail, we replace the relative pronoun (e.g., *who* or *which*) of a relative clause by its antecedent, which is obtained via `rcmod` dependency to the governor of the relative pronoun. The replacement of relative pronouns aims to increase the informativeness of extractions; e.g., we obtain (S: *Bell*, V: *based*, A: *Angeles*) instead of (S: *which*, V: *based*, A: *Angeles*). ClausIE also handles non-verb-mediated extractions to a limited extent: We create synthetic clauses for appositions (`appos`) and possessives (`pos` or via the pronoun *whose*; see above). The so-obtained clauses use an artificial verb such as "*is*" (typed as copula) or "*has*" (typed as monotransitive), respectively. In our example, we obtain clause (S: *Bell*, V_c: "*is*", O: *company*) in this way, where words within quotation marks refer to an artificial verb and without quotation marks refer to a word in the (DP of the) original sentence. Finally, we generate a synthetic clause from participial modifiers (`partmod`), which indicate reduced relative clauses. The dependant of a participial modifier relation is a participial verb form, which we combine with an artificial verb such as "*are*" to obtain the "verb" of the synthetic clause (typed SVA). For example, we obtain from sentence "*Truffles picked during the spring are tasty*" the synthetic clause (S: *Truffles*, V: "*are picked*", A: [*during the*] *spring*).

In summary, we identify the following clauses for the sentence of Fig. 1:

(S: *Bell*, V: *makes*, O: *products*),
(S: *Bell*, V: *based*, A: *Angeles*),
(S: *Bell*, V_c: "*is*", C: *company*).

4.3 Step 3: Identifying Clause Types

Once clauses have been obtained, ClausIE tries to identify the type of each clause (recall Tab. 1). As argued in Sec. 3, we can combine knowledge of properties of verbs with knowledge about the structure of the input clause. Our approach to clause-type detection can be viewed as a decision tree, i.e., we ask a number of question whose answers ultimately determine the clause type. The decision tree is shown as Fig. 2; here questions Q_1–Q_3 and Q_7–Q_9 refer to the clause structure; questions Q_4, Q_5, and Q_{10} to verb properties, and questions Q_6 and Q_{11} deal with ambiguous cases. We describe each of these questions in detail below, and also discuss some techniques that help dealing with errors in the DP. After clause types have been identified, we mark all optional adverbials. In our example of Fig. 1, we obtain

(S: *Bell*, V: *makes*, O: *products*),
(S: *Bell*, V: *based*, A!: *Angeles*),
(S: *Bell*, V: "*is*", A!: *company*),

where "A!" indicates essential adverbials and "A?" indicates optional adverbials.

Clause types SVC, SVOO, and SVOC are identified solely by the structure of the clause; all adverbials are optional for these types. For example, if a clause does not contain an object (Q_1) but does contain a complement (Q_2), it must be of type SVC. For example, we identify S_{10} of Tab. 1 as SVC so that its adverbial "*of the 20th century*" is optional.

If the sentence contains neither object nor complement, we are left with distinguishing clause types SV (intransitive verb) and SVA (extended-copular verb), a more difficult task. In many cases, the distinction can be performed accurately. We say that an adverbial is a *candidate adverbial* (for an essential adverbial) if it (1) is a dependant of the verb and (2) appears to the right of the verb. If the clause does not contain a candidate adverbial (Q_3), it is of type SV; e.g., "*The year after, AE succeeded*". Otherwise, ClausIE makes use of two lexica of verb types: a lexicon of verbs that are known to be non-extended-copular (Q_4, implies SV) and a lexicon of verbs known to be extended-copular (Q_5, implies SVA).[5] E.g., the adverbial in "*AE remained in Princeton*" is identified as essential since *remain* is a copular verb. If both dictionaries fail, we cannot determine the clause type accurately. In its default configuration, ClausIE then proceeds conservatively (Q_6), i.e., it assumes SVA to avoid marking an essential adverbial as optional.

We proceed to distinguishing SVO and SVOA for clauses that neither contain a complement nor both a direct and an indirect object (SVOO). If the clause does not have a candidate adverbial (Q_9), we mark it as SVO. Similarly, if the clause has an indirect object (but not a direct object, Q_9),

[5]Note that these lexica can be learned automatically by observing which verbs appear (sufficiently frequently) without a candidate adverbial in the text collection. We did not yet employ such techniques; our current implementation makes use of only a small hand-crafted dictionary of 31 extended-copular verbs (e.g., "*be*" or "*remain*") and two non-extended copular verbs ("*die*" and "*walk*"). The dictionary is not hard-wired into ClausIE, though, and can be customized by applications.

it cannot be of type SVOA [15] so that we also mark it SVO; e.g., as in "*He taught his students passionately.*" Otherwise, the clause contains both a direct object (but no indirect object and no complement) and a candidate adverbial. The distinction between SVO and SVOA is difficult in this (quite common) case; e.g., S_{11} (SVO) and S_6 (SVOA) in Tab. 1. Here we proceed heuristically. First, ClausIE accepts a lexicon of verbs that are potentially complex-transitive (Q_{10}) and outputs SVOA if the verb appears in the lexicon.[6] Otherwise, in ClausIE's default configuration, we proceed greedily (Q_{11}) and choose SVO, i.e., we mark the adverbial as optional.

Dealing with DP ClausIE performs a number of additional steps in order to deal with design choices and errors of the Stanford parser.

We first discuss how we deal with clauses that have constituents of clausal form. The Stanford parser outputs dependency relations such as `xcomp` or `ccomp` for the object and the complement of a clause if they have a clausal form. We treat these dependencies as complements if the verb appears in our lexicon of copular verbs, and treat them as objects (or object complements) otherwise. If the clause additionally contains an indirect object, the parser outputs `dobj` instead of `iobj`. In this case, we cannot distinguish between SVOO and SVOC. Since we are ultimately interested in optional adverbials, and since all adverbials are optional for both SVOO and SVOC, we still obtain correct extractions. The Stanford parser sometimes places the object complement within the direct object. If this happens, we may determine clause types SVO or SVOA instead of SVOC. In both cases, extractions are coherent; if we detect SVOA, however, an optional adverbial is incorrectly flagged as essential. Finally, the parser outputs relation `dep` when it is unable to identify the type of a relation. ClausIE avoids processing the dependant of `dep` in verbal phrases to reduce potential extraction errors.

4.4 Step 4: From Clauses to Propositions

As a consequence of ClausIE's separation of clause and clause-type detection from proposition generation, the latter is flexible and can be customized to the application. There are two basic steps involved in proposition generation. The first step is to decide which (combinations of) constituents form a proposition; the second step then generates the proposition from the constituents.

Constituent selection. Recall that a proposition consists of a subject, a relation, and zero, one, or more arguments. A natural choice is to generate n-ary propositions that consist of all the constituents of the clause, potentially with some arguments being marked optional. ClausIE supports generation of such n-ary propositions, but in addition allows to generate triple propositions, i.e., propositions that consist of a subject, a relation, and a (potentially empty) argument.[7] In fact, the concept of a triple (or binary relation) is fundamental to the semantic web, most ontological knowledge bases, and most OIE systems. A key question is which constituents should be included into the generated triple. ClausIE takes a pragmatic approach: We do not only generate a single triple from each clause, but allow for the generation of multiple triples, each exposing different pieces of information. Consider for example the clause (S: *AE*, V: *died*, A?: [*in*] *Princeton*, A?: [*in*] *1955*) obtained from S_8 in Tab. 1. Since both adverbials are marked optional, we can select four coherent derived clauses:

(S: *AE*, V: *died*),
(S: *AE*, V: *died*, A: [*in*] *Princeton*),
(S: *AE*, V: *died*, A: [*in*] *1955*),
(S: *AE*, V: *died*, A: [*in*] *Princeton*, A: [*in*] *1955*).

In general, if there are n optional adverbials, there are 2^n coherent derived clauses. To avoid over-specified triples, our default choice in ClausIE—which we also used in our experiments—is to select at most one optional adverbial (and all essential constituents). ClausIE also makes use of a lexicon consisting of small set of adverbials to be always omitted (e.g., "*so*") or included (e.g., "*hardly*") when optional.

Coordinated conjunctions (CC). A *coordinated conjunction* is a conjunction that connects two or more parts of the sentence—called *conjoints*—via a coordinator such as "*and*" or "*or*". CCs are detected by the Stanford parser and indicated by dependency relations such as `conj`, `conj_and`, or `conj_or`, respectively. If a CC is present in a constituent of a clause, ClausIE optionally *processes* the CC, i.e., replaces the CC by each of its conjoints to avoid over-specified extractions. Consider the example sentence shown in Fig. 1. There is a CC in the verb constituent ("*makes and distributes*") and in the object constituent ("*electronic, computer, and building products*") of the main clause. By replacing CCs by conjoints, we obtain the following clauses:

(S: *Bell*, V: *makes*, O: [*electronic*] *products*),
(S: *Bell*, V: *makes*, O: [*computer*] *products*),
(S: *Bell*, V: *makes*, O: [*building*] *products*),
(S: *Bell*, V: *distributes*, O: [*electronic*] *products*),
(S: *Bell*, V: *distributes*, O: [*computer*] *products*),
(S: *Bell*, V: *distributes*, O: [*building*] *products*).

The processing of CCs is closely related to text simplification [8]; we can view the resulting clauses as simpler versions of the original clauses.

Note that in noun phrases, the replacement of a CC by one of its conjoints may lead to incorrect extractions when the CC is *combinatory* (as opposed to *segregatory*). For example, the CC in "*Anna and Bob married each other*" is combinatory; thus an extraction such as "*Anna married each other*" is incoherent. If the CC has an ampersand as coordinator, ClausIE treats it as combinatory and thus does not process it (e.g., "*Standard & Poor's*"). Similarly, CCs headed by words such as "*between*" are not processed (e.g., "*between Norway and Finland*"). In all other cases, the CC is treated as segregatory and thus processed. Combinatory CCs are rare in some domains [8], but may occur frequently in others. Since combinatory CCs are hard to detect (in some cases even for humans), ClausIE exposes an option to disable processing of CCs.

Finally, ClausIE treats CCs with preconjuncts (`preconj` dependency; e.g., "*both [red and blue]*") and (pre)determiners ((`pre`)`det`; e.g., "*both [the boys and the girls]*") specially. In particular, we omit all preconjuncts and some (pre)determiners when processing a CC. For example, we extract from

[6] The lexicon currently contains 15 verbs (e.g., "*put*" and "*get*").
[7] In OIE, the argument component of a triple is often called "object"; e.g., "*1921*" in ("*AE*", "*has won the NP in*", "*1921*"). Here we avoid the term object for the argument of a triple to avoid confusion with the object of the clause.

"*Anna likes both red and blue*" the propositions ("*Anna*", "*likes*", "*red*") and ("*Anna*", "*likes*", "*blue*").

Proposition generation. ClausIE generates one proposition for each selected subset of constituents. To generate a proposition, ClausIE needs to decide which part of each constituent to place into the subject, the relation, and the arguments. The perhaps simplest option is to first generate a textual representation of each constituent in its entirety and then use these representations to construct the proposition. ClausIE currently follows this approach but omits relative clauses (but ClausIE does generate propositions from each relative clause in separation). We map the subject (verb) of each clause to the subject (relation) of the proposition. When n-ary facts are extracted, we create an argument for each of the remaining constituents (first all constituents following the verb, then all constituents preceding the verb, in the order in which they appear). To extract triples, we concatenate all arguments. From the sentence of Fig. 1, ClausIE extracts the following triples:

("Bell", "is", "a telecommunication company"),
("Bell", "is based", "in Los Angeles"),
("Bell", "makes", "electronic products"),
("Bell", "makes", "computer products"),
("Bell", "makes", "building products"),
("Bell", "distributes", "electronic products"),
("Bell", "distributes", "computer products"),
("Bell", "distributes", "building products").

In future work, we plan to further customize the final step of proposition generation in multiple ways. For example, we can obtain extractions similar to Reverb [9] by appending all but the final argument into the relation; if the final argument is a prepositional phrase, we also include the preposition into the relation. Another natural direction is to analyze the composition of each constituent in order to generate alternative textual representations.

5 Experiments

We conducted an experimental study to compare ClausIE to alternative approaches. We found that ClausIE achieved significantly higher recall than the OIE extractors we compared to. Moreover, ClausIE consistently provided higher precision than alternative extractors over all levels of recall.

5.1 Experimental Setup

We first describe the datasets and the methodology used in our experiments.[8] We compared ClausIE to TextRunner [3], Reverb [9], WOE [19] (using DP), OLLIE [13] and KrakeN [2]; neither extractions nor source code of any other extractor were available to us. Since most of OIE techniques make use of machine-learning techniques, which require sensibly-chosen training data, or may need tweaking to provide good extractions, we did not compare ClausIE to these other OIE extractors. In all our experiments, we used the unlexicalized version of the Stanford DP (version 2.0.4). We configured ClausIE to generate triple propositions and ran it both with and without processing of coordinated conjunctions in subjects and arguments (denoted "ClausIE" and "ClausIE w/o CCs," respectively); coordinated conjunctions in verbal phrases were processed in both configurations.

We used three different datasets in our experiments. First, the Reverb dataset[9] consists of 500 sentences with manually-labeled extractions for TextRunner, TextRunner trained using Reverb, Reverb, OLLIE, and WOE. The sentences have been obtained via the random-link service of Yahoo and are generally very noisy. Second, we extracted 200 random sentences from Wikipedia pages. These sentences are shorter, simpler, and less noisy than those of the Reverb dataset. Since some Wikipedia articles are written by non-native speakers, however, the Wikipedia sentences do contain some incorrect grammatical constructions. Finally, we extracted 200 random sentences from the New York Times collection (NYT, [16]); these sentences are generally very clean but tend to be long and complex.

We manually labeled the extractions obtained from all extractors. To maintain consistency among the labels, the entire set of extractions of TextRunner, WOE, and Reverb for the Reverb dataset was relabeled; the precision numbers obtained using our labels closely agreed with those obtained using the original labels. For the Wikipedia and NYT datasets, we compare ClausIE with only Reverb and OLLIE, for which an extractor was publicly available. Each extraction was labeled by two independent labelers; an extraction was treated as correct only if it was labeled as correct by both labelers. Since we are primarily interested in the ability of OIE to capture verb-mediated propositions, labelers were instructed to ignore the *context* of the clause during labeling. For example, in the sentence "*But inexpensive point-and-shoot cameras can do the job if they have a telephoto setting or a zoom lens*", the proposition ("*inexpensive point-and-shoot cameras*", "*can do*", "*the job*") is treated as a correct extraction. We also asked labelers to be liberal w.r.t. coreference or entity resolution; e.g., a proposition such as ("*he*", "*has*", "*office*"), or any unlemmatized version thereof, is treated as correct. Finally, we instructed labelers to label as incorrect relations that were overly specific, i.e., that contained named entities or numbers, or were excessively long (e.g., "*has reported 1993 events in Moscow in*"). We measured the agreement between labelers in terms of Cohen's Kappa (Scott's Pi). The score was 0.57 (0.57) for the Reverb dataset, 0.68 (0.68) for the Wikipedia dataset, 0.63 (0.63) for the New York Times dataset. The lower agreement score for the Reverb data might be attributed to the high amount of noise in the input sentences, which made it hard to judge the correctness of some of the extractions.

We used the absolute number of extractions instead of recall since it is infeasible to obtain the set of "all correct" propositions. For ClausIE, we determined the total number of extractions but also the number of non-redundant extractions (marked "non-red."), i.e., extractions not "contained" in other extractions. For example, ClausIE extracts from sentence "*AE remained in Princeton until his death*" propositions ("*AE*", "*remained*", "*in Princeton*") and ("*AE*", "*remained*", "*in Princeton until his death*"); the former extraction is marked redundant. We ordered all extractions by decreasing confidence; for ClausIE, we took the confidence of the DP as obtained by the Stanford parser as the confidence of a proposition. For KrakeN, extractions were unavailable to us; we reproduce the information provided in [2] instead.

[8] All datasets, extractions, labels, as well as ClausIE's source code are available at http://www.mpi-inf.mpg.de/departments/d5/software/clausie.

[9] http://reverb.cs.washington.edu/

5.2 Example Extractions

We first illustrate the differences between the extractors for some manually-selected example sentences; Tab. 3 shows the extractions of each OIE extractor for a sentence of each of the datasets.

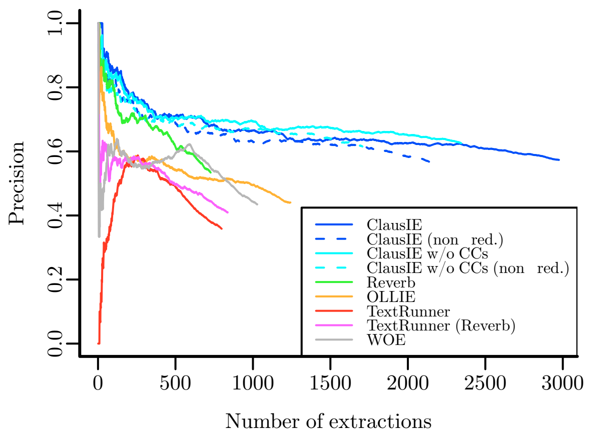
On the Reverb sentence, all OIE extractors agree on proposition R_1, which is correct. Reverb obtains a second proposition R_2, which is incorrect; it is obtained because Reverb restricts subjects to noun phrases without prepositions and thus incorrectly omits "*the only other name on*". In contrast, ClausIE identifies the subject correctly and hence extracts a correct proposition (R_{20}); it exploits access to the DP, which is (deliberately) not used by Reverb. WOE and OLLIE also make use of the DP, but still fail to identify the subject of the second clause correctly (R_5 and R_{11}, resp.), perhaps due to their use of automatically learned DP patterns (e.g., OLLIE learns from Reverb). For this reason, OLLIE also produces a number of additional incorrect extractions. Note that propositions R_{18} and R_{20} produced by ClausIE are labeled as redundant. As argued below, redundant extractions may be valuable by themselves due to their simpler structure.

On the Wikipedia dataset, almost all of the extractions are correct; ClausIE extracts the largest number of propositions, followed by OLLIE and Reverb. OLLIE misses the essential adverbial "*in Aberdeen*" in proposition N_5, but still produces a correct (although tautological) proposition. ClausIE produces incorrect proposition N_{11} due to an error in the dependency parse (which does not associate "*from Tuberculosis*" with "*death*"). Proposition N_{12} was labeled correct (although this is arguable); here "*his*" refers to "*he*", and "*has*" is our synthetic verb for a possessive. Finally, ClausIE produces propositions N_6–N_8 due to its processing of the coordinated conjunctions. In this particular case, the parser identified "*two children*", "*Edna*", and "*Donald*" incorrectly as conjoints; otherwise propositions N_7 and N_8 would not have been generated.

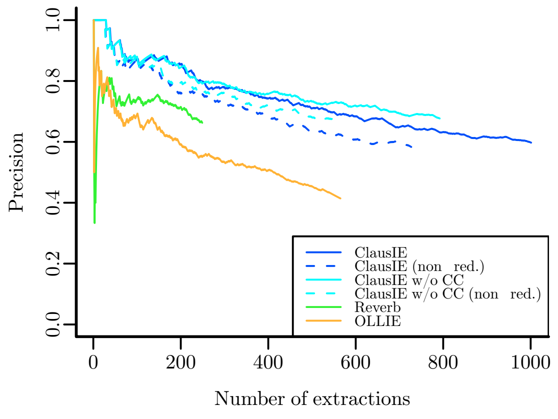
Finally, on the NYT dataset, Reverb produces incorrect proposition W_2 by incorrectly identifying the argument. Reverb is designed to extract at most one prepositional phrase following the verb and thus misses "*outside the United States*". It also misses "*in NATO*" due its use of a lexical constraint (i.e., the phrase "*includes the biggest standing army in*", which is over-specified, does not appear sufficiently frequent in the corpus). ClausIE creates a correct and an incorrect (but coherent) proposition (W_{13} and W_{12}, resp.) from this clause of the sentence; the latter proposition is incorrect due to an error in the DP parse (which does not correctly associate "*in NATO outside the United States*" with "*army*"). ClausIE also produces three additional incorrect propositions (W_{15}–W_{17}). Proposition W_{15} has an incorrect subject due to an incorrect DP, propositions W_{16} and W_{17} are non-informative and thus labeled as incorrect (here we labeled conservatively). The sentence also contains a possessive, which is processed correctly by ClausIE to obtain proposition W_{14}. Finally, OLLIE extracts three incorrect propositions with an over-specified relation (W_3–W_5), and incorrect proposition W_6 due to a noisy extraction pattern.

5.3 Precision and Number of Extractions

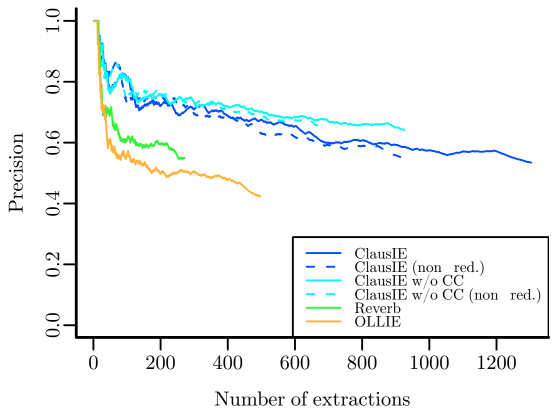
Our results are summarized in Tab. 2 and Fig. 3. Tab. 2 shows the total number of correct extractions as well as the



(a) Reverb dataset



(b) Wikipedia dataset



(c) New York Times dataset

Figure 3: Experimental results

Table 2: Number of correct extractions and total number of extractions

	ClausIE	ClausIE w/o CCs	ClausIE (non-redundant)	ClausIE w/o CCs (non-redundant)	OLLIE	Reverb	WOE	TextRunner (Reverb)	TextRunner
Reverb dataset	1706/2975	1466/2344	1221/2161	1050/1707	547/1242	388/727	447/1028	343/837	286/798
Wikipedia dataset	598/1001	536/792	424/727	381/569	234/565	165/249	-	-	-
NYT dataset	696/1303	594/926	508/926	444/685	211/497	149/271	-	-	-

total number of extractions for each method and dataset. Fig. 3 plots the precision of each OIE extractor as a function of the number of extractions (ordered by decreasing confidence).

We found that in its default configuration, ClausIE produced 2.5–3.5 times more correct extractions than OLLIE, the best-performing alternative method. This increase in recall is obtained because ClausIE considers all adverbials in a clause (instead of only the one following the verb), extracts non-verb-mediated propositions, detects non-consecutive constituents, processes coordinated conjunctions, and outputs triples with non-noun-phrase arguments. Roughly 27–29% of the extractions of ClausIE were redundant. We believe that redundant extractions can be valuable: Even though a non-redundant proposition expresses more information, the corresponding redundant propositions has a simpler structure and is easier to deal with. When redundant extractions are removed, ClausIE produces 1.8–2.4 times more correct extractions than OLLIE.

The precision of TextRunner was significantly lower than that of Reverb, WOE, and ClausIE. The latter three extractors obtain high precision on high-confidence propositions; the precision drops as we include more and more low-confidence propositions. In the case of ClausIE, the precision dropped quickly initially but then stabilized at between 53% and 60% (whether or not we include redundant propositions). Except for the Wikipedia dataset, the precision over all extractions obtained by ClausIE was higher than that of any other method, and ClausIE extracted significantly more propositions.

We also ran a configuration of ClausIE in which processing of coordinated conjunctions in subjects and arguments was disabled. This resulted in an increase of precision between 5% and 10.7% (on Wikipedia). Thus ClausIE's processing of CCs is somewhat error-prone, partly due to the presence of combinatory conjunctions and partly due to errors in the dependency parse. Nevertheless, when CCs are not processed, the number of extractions dropped significantly (between 11% and 27%), so that CC processing appears to be beneficial overall.

According to [2], KrakeN extracts 572 propositions from the Reverb data; 308 of these propositions were correct and complete, 81 were correct but not complete. Note that KrakeN extracts n-ary propositions, whereas our experiments focus on triples (which cannot be produced by KrakeN for $n > 3$). Note that KrakeN did not extract propositions from dependency parses that contained the `dep` relation (i.e., an unknown dependency); this was true for 155 out of the 500 sentences in the Reverb data. ClausIE handles such cases gracefully, e.g., by extracting propositions from clauses that appear unaffected by the unknown dependency.

The high recall and consistently good precision of ClausIE observed in our experiments indicates that reasoning over clauses and clause-types is a viable approach to OIE.

5.4 Extractions Errors of ClausIE

We did a preliminary analysis of the results obtained by ClausIE. We found that in most of the cases, ClausIE's extraction errors were due to incorrect dependency parses (see Sec. 5.2 for examples). In some cases, the incorrect DP resulted from noise in the input sentences, such as bad grammatical forms or spurious words. Our hope is that potential future improvements in dependency parsing will also lead to higher-precision extractions obtained by ClausIE. Another source of imprecision of ClausIE was due to our processing of coordinated conjunctions; see the discussion in Sec. 5.3. On the one hand, the Stanford DP parser tended to produce erroneous parses in the presence of CCs. On the other hand, when the coordinated conjunction was combinatory, the extractions obtained by ClausIE were incorrect. ClausIE also misclassified some SVOA clauses as SVO and thus omitted an essential adverbial. As mentioned previously, it is often hard to distinguish SVO from SVOA; an improved dictionary of potentially complex-transitive verbs may help to avoid some of these extraction errors. Moreover, Quirk [15] notes that adverbials in SVA and SVOA clauses are largely restricted to space adjuncts, which may also help in identifying such clauses. Finally, this problem is alleviated to some extent if ClausIE is configured to produce n-ary extractions; then essential adverbials will not be omitted, although they can potentially be flagged as optional.

6 Conclusion

We presented a novel, clause-based approach to open information extraction called ClausIE. In contrast to previous approaches, ClausIE separates the detection of clauses and clause types from the actual generation of propositions. This allows ClausIE to obtain more and higher-precision extractions than alternative methods, but also enables flexible generation of propositions. ClausIE can be seen as a first step towards clause-based open information extraction. Potential improvements include construction of richer lexica, improved processing of the constituents of each clause to avoid over-specification in subjects and arguments, as well as context analysis to detect relations between clauses.

7 Acknowledgements

We would like to thank Carla Miotto for her helpful comments.

Table 3: Example extractions from a sentence of each dataset

System	**#**	**Proposition**	**Label**
		Reverb dataset	
		The principal opposition parties boycotted the polls after accusations of vote-rigging , and the only other name on the ballot was a little-known challenger from a marginal political party.	
Reverb	R_1:	(“*The principal opposition parties*”, “*boycotted*”, “*the polls*”)	Correct
	R_2:	(“*the ballot*”, “*was*”, “*a little-known challenger*”)	Incorrect
TextRunner	R_3:	(“*The principal opposition parties*”, “*boycotted*”, “*the polls*”)	Correct
WOE	R_4:	(“*The principal opposition parties*”, “*boycotted*”, “*the polls*”)	Correct
	R_5:	(“*the only other name*”, “*was*”, “*a little-known challenger*”)	Incorrect
OLLIE	R_6:	(“*The principal opposition parties*”, “*boycotted*”, “*the polls*”)	Correct
	R_7:	(“*The principal opposition parties*”, “*boycotted the polls after*”, “*accusations of vote-rigging*”)	Correct
	R_8:	(“*The principal opposition parties*”, “*was*”, “*a little-known challenger*”)	Incorrect
	R_9:	(“*The principal opposition parties*”, “*was a little-known challenger from*”, “*a marginal political party*”)	Incorrect
	R_{10}:	(“*the polls*”, “*be boycotted after*”, “*accusations of vote-rigging*”)	Correct
	R_{11}:	(“*the only other name*”, “*was*”, “*a little-known challenger*”)	Incorrect
	R_{12}:	(“*the only other name*”, “*was a little-known challenger from*”, “*a marginal political party*”)	Incorrect
	R_{13}:	(“*the only other name*”, “*boycotted*”, “*the polls*”)	Incorrect
	R_{14}:	(“*the only other name*”, “*boycotted the polls after*”, “*accusations of vote-rigging*”)	Incorrect
	R_{15}:	(“*a little-known challenger*”, “*be the only other name on*”, “*the ballot*”)	Correct
	R_{16}:	(“*only*”, “*be other name on*”, “*the ballot*”)	Incorrect
	R_{17}:	(“*other*”, “*be name on*”, “*the ballot*”)	Incorrect
ClausIE	R_{18}:	(“*The principal opposition parties*”, “*boycotted*”, “*the polls*”)	Correct (red.)
	R_{19}:	(“*The principal opposition parties*”, “*boycotted*”, “*the polls after accusations of vote-rigging*”)	Correct
	R_{20}:	(“*the only other name on the ballot*”, “*was*”, “*a little-known challenger*”)	Correct (red.)
	R_{21}:	(“*the only other name on the ballot*”, “*was*”, “*a little-known challenger from a marginal political party*”)	Correct
		Wikipedia dataset	
		He fathered two children, Edna and Donald, and lived in Aberdeen until his death from tuberculosis in 1942.	
Reverb	N_1:	(“*He*”, “*fathered*”, “*two children*”)	Correct
	N_2:	(“*two children*”, “*lived in*”, “*Aberdeen*”)	Correct
OLLIE	N_3:	(“*He*”, “*fathered*”, “*two children*”)	Correct
	N_4:	(“*He*”, “*lived in*”, “*Aberdeen*”)	Correct
	N_5:	(“*He*”, “*lived until*”, “*his death*”)	Correct
ClausIE	N_6:	(“*He*”, “*fathered*”, “*two children*”)	Correct
	N_7:	(“*He*”, “*fathered*”, “*Edna*”)	Correct
	N_8:	(“*He*”, “*fathered*”, “*Donald*”)	Correct
	N_9:	(“*He*”, “*lived*”, “*in Aberdeen*”)	Correct (red.)
	N_{10}:	(“*He*”, “*lived*”, “*in Aberdeen until his death*”)	Correct
	N_{11}:	(“*He*”, “*lived*”, “*in Aberdeen from tuberculosis in 1942*”)	Incorrect
	N_{12}:	(“*his*”, “*has*”, “*death*”)	Correct
		New York Times dataset	
		Taken for granted it sometimes may be, but this year the Defense Department sought $950 million in assistance from Congress (and secured half that amount) for Ankara’s huge military machine, which includes the biggest standing army in NATO outside the United States.	
Reverb	W_1:	(“*the Defense Department*”, “*sought*”, “*$ 950 million*”)	Correct
	W_2:	(“*Ankara’s huge military machine*”, “*includes*”, “*the biggest standing army*”)	Incorrect
OLLIE	W_3:	(“*the Defense Department*”, “*sought $ 950 million in assistance from Congress half*”, “*(and secured half)*”)	Incorrect
	W_4:	(“*the Defense Department*”, “*sought $ 950 million in assistance from Congress in*”, “*this year*”)	Incorrect
	W_5:	(“*Ankara ’s huge military machine*”, “*includes the biggest standing army in NATO outside*”, “*the United States*”)	Incorrect
	W_6:	(“*the biggest standing army*”, “*be includes by*”, “*Ankara ’s huge military machine*”)	Incorrect
ClausIE	W_7:	(“*the Defense Department*”, “*sought*”, “*$ 950 million*”)	Correct (red.)
	W_8:	(“*the Defense Department*”, “*sought*”, “*$ 950 million in assistance*”)	Correct
	W_9:	(“*the Defense Department*”, “*sought*”, “*$ 950 million this year*”)	Correct
	W_{10}:	(“*the Defense Department*”, “*sought*”, “*$ 950 million for Ankara’s huge military machine*”)	Correct
	W_{11}:	(“*the Defense Department*”, “*sought*”, “*$ 950 million from Congress*”)	Correct
	W_{12}:	(“*Ankara’s huge military machine*”, “*includes*”, “*the biggest standing army in NATO*”)	Incorrect
	W_{13}:	(“*Ankara’s huge military machine*”, “*includes*”, “*the biggest standing army in NATO outside the United States*”)	Correct
	W_{14}:	(“*Ankara*”, “*has*”, “*huge military machine*”)	Correct
	W_{15}:	(“*Taken for*”, “*granted*”, “*it sometimes may be*”)	Incorrect
	W_{16}:	(“*it*”, “*may be*”)	Incorrect
	W_{17}:	(“*it*”, “*may be*”, “*sometimes*”)	Incorrect

8 References

[1] Alan Akbik and Jürgen Broß. Wanderlust: Extracting Semantic Relations from Natural Language Text Using Dependency Grammar Patterns. In *1st Workshop on Semantic Search at 18th. WWWW Conference*, 2009.

[2] Alan Akbik and Alexander Löser. Kraken: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 52–56, 2012.

[3] Michele Banko, Michael J Cafarella, Stephen Soderl, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of Conference on Artificial Intelligence*, pages 2670–2676, 2007.

[4] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Conference on Artificial Intelligence*, 2010.

[5] Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 52–60, 2010.

[6] Marie-Catherine de Marnee and Christopher D. Manning. Stanford typed dependencies manual.

[7] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. Open information extraction: The second generation. In *Proceedings of the Conference on Artificial Intelligence*, pages 3–10, 2011.

[8] Richard J. Evans. Comparing methods for the syntactic simplification of sentences in information extraction. *Literary and Linguistic Computing*, 26(4):371–388, 2011.

[9] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing*, 2011.

[10] Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. Dependency-based open information extraction. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 10–18, 2012.

[11] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of Association of computational linguistics*, pages 423–430, 2003.

[12] Thomas Lin, Mausam, and Oren Etzioni. No noun phrase left behind: detecting and typing unlinkable entities. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 893–903, 2012.

[13] Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. Open language learning for information extraction. In *Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, 2012.

[14] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. PATTY: a taxonomy of relational patterns with semantic types. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2012.

[15] Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. *A Comprehensive Grammar of the English Language*. Longman, 1985.

[16] Evan Sandhaus. *The New York Times Annotated Corpus*, 2008.

[17] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. Sofie: a self-organizing framework for information extraction. In *Proceedings of WWW*, pages 631–640, 2009.

[18] Petros Venetis, Alon Y. Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. Recovering semantics of tables on the web. *PVLDB*, 4(9):528–538, 2011.

[19] Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127, 2010.

[20] Amal zouaq. An overview of shallow and deep natural language processing for ontology learning. In W. Wong, W. Liu, and M. Bennamoun, editors, *Ontology Learning and Knowledge Discovery Using the Web: Challenges and Recent Advances*. 2011.