SANDIA REPORT
SAND2020-4866

Printed April 2020

Sandia National Laboratories

Survey of Current State of the Art Entity-Relation Extraction Tools

Katrina Ward, Jonathan Bisila, Kelsey Cairns

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from

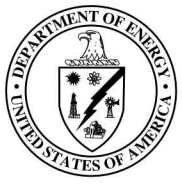
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods



ABSTRACT

In the area of information extraction from text data, there exists a number of tools with the capability of extracting entities, topics, and their relationships with one another from both structured and unstructured text sources. Such information has endless uses in a number of domains, however, the solutions to getting this information are still in early stages and has room for improvement. The topic has been explored from a research perspective by academic institutions, as well as formal tool creation from corporations but has not made much advancement since the early 2000's. Overall, entity extraction, and the related topic of entity linking, is common among these tools, though with varying degrees of accuracy, while relationship extraction is more difficult to find and seems limited to same sentence analysis. In this report, we take a look at the top state of the art tools currently available and identify their capabilities, strengths, and weaknesses. We explore the common algorithms in the successful approaches to entity extraction and their ability to efficiently handle both structured and unstructured text data. Finally, we highlight some of the common issues among these tools and summarize the current ability to extract relationship information.

CONTENTS

LIST OF FIGURES

PREFACE

The purpose of this research is to explore the current state of the art tools available to perform entity-relationship extraction from any text, both structured and unstructured, and analyze their capabilities. While we did research many tools, we recognize we did not research them all. However, based on the criteria we were looking for described in this document, we are of strong belief that we found the best tools at the time of writing this.

NOMENCLATURE

Abbreviation	Definition
DOE	Department of Energy
ERX	Entity-Relation Extraction
SVM	Support Vector Machine
MITIE	MIT Information Extraction Tool
CRF	Conditional Random Field
NER	Named Entity Recognizer
NLP	Natural Language Processing
TF-IDF	Term Frequency - Inverse Document Frequency

1. INTRODUCTION

In the area of information extraction from text data, there exists a number of tools with the capability of extracting entities, topics, and their relationships with one another, known as Entity-Relationship Extraction (ERX). Entities are important subjects in text such as people, dates, locations, and organizations. The extraction of entities can help in finding information of interest in large text files, news websites, social media, or a combination of all of the above. In addition to the entities themselves, there is a demand for understanding the relationship between these entities. For example, if we see the following sentence:

> "In 2009, Barack Obama was the youngest person to serve as President of the United States."

The entities extracted would be *2009*, *Barack Obama*, *President*, and *United States*. These entities may be enough information to tell whether a document contains enough relevant information to look into further, or if the text is from social media, tell the topics that are currently being discussed based on current events. From these entities, we could learn that Barack Obama was the President of the United States in 2009, thus understanding the context of the sentence and the relationships between them within it. While this may seem trivial in a single sentence, this information could be buried in millions of lines of text coming from several sources that is read over by an analyst. The information that can be potentially extracted could benefit multiple industries and research efforts.

Current ERX tools for entity extraction are typically grouped by the general approach of their algorithms. Some of the most common are database, dictionary, and SVM approaches. In database approaches, a user can specify certain terms they want the software to identify in text as entities. For example, if a user is a veterinarian looking for emerging diseases that could affect livestock, they could build a database looking for specific animals, feed brands, and virus or bacteria names. This approach is very accurate for the limited scope of terms, but poor in performance with general text data. Most of the tools we tested had the option to include custom entities and incorporated the database search with either dictionary or SVM based algorithms. However some, such as Catalyst, used database search almost exclusively because of the limited scope the tool was intended for.

Dictionary based algorithms collect data from common knowledgebases such as Wikipedia, DBpedia, Wikidata, and Freebase. The algorithms for these tools scan their databases for previous knowledge of an entity in order to classify it as a person, location, organization, date, etc. The benefits are that the results are accurate for known terms in the dictionary and the ability to understand context and meaning of the words is increased. However, the approach is extremely slow. It also has the weakness of not understanding abbreviations as well. For example, a human understands that *Barack Obama*, *President Barack Obama*, *B. Obama* are the same person. However, a dictionary based algorithm may not detect that all of these are a person, much less the *same person*. This issue was observed with

TextRazor in our tests. Like the database approaches, it is possible to supplement other algorithms with a dictionary search when looking for a small set of specific terms, however, this is seen less frequently.

The final large category of algorithms use support vector machines (SVM) in order to recognize patterns in text and identify entities as certain categories. This approach is much faster than the previous two approaches, however, requires very large training data sets in order to come close to the accuracy of database or dictionary approaches. An example of this approach is MITIE and SpaCy. Additional algorithm ideas have been attempted in academic research, but were never adopted due to not performing as well in either accuracy, speed, or both.

While most of these tools perform entity extraction with varying degrees of speed and accuracy, few actually perform relationship extraction as well. So far in previous work, the two problems are considered separate, though attempted in many of the same tools. The two pieces of information are heavily related and almost depend on one another to be truly useful. Using the example sentence above, it may be useful to know that a document contains the name of the President, however, knowing the context of the usage and how the usage relates to other entities in the text would have much more utility. Of the tools we present here, TextRazor has the biggest claim to perform relationship extraction, however, in our tests it does not do so very well. MITIE, CoreNLP, SpaCy, and ReVerb are also able to perform relationship extraction after extensive training, but again, not to any high standards.

In this report, we explore the current state of the art tools and explain their capabilities and limitations. We present a side-by-side comparison of these tools as well as dive deep into their functionality in order to provide a view of current capabilities in Entity-Relationship Extraction.

2. APPROACH TO UNDERSTANDING ENTITY-RELATIONSHIP EXTRACTION TOOLS

In this section, we explain how we evaluated the tools including what information we looked for, what features we looked at, what datasets we used to test the tools and how we evaluated their usefulness and effectiveness.

2.0.1. Goals for Tool Consideration

While there are a number of tools available for entity and relationship extraction, there were some major factors we looked for to determine which tools were the best available. We first evaluated accessibility. We wanted to be able to access not just the tool for use, but the source code as well. This way the tool could be tailored for multiple different uses and/or expanded upon. We also looked at the last time the tool was maintained to see of it was kept up to date and would be reasonable to assume the tool would be kept available. We also looked at cost, ease of installation, and ease of use.

We then looked at how well it performed entity extraction by seeing which entities were found, missed, and if the tool correctly identified which label they belonged to. With that, we considered the limitations of the labels that the entities were assigned. By that, we mean if the labels were given to the model or if the model could actually learn which labels to use. We considered if the tool was also able to extract the relationship between the entities and if the tool could perform as well on unstructured data as it could on structured text.

Throughout the tool analysis, we make sure to note features the tools have. For example, it's important to be able to supplement specific entities to focus on, such as bacteria names, restricted chemicals, specific people, and so forth, as well as an understanding of the context of the entities the tool found. We took a deeper look into the tools and discovered which programming languages they supported for use, if graph visualization was provided as an option, and how each tool evaluated its accuracy. Finally, since our world is becoming more and more connected, we looked for multi-language recognition.

In our final category, we evaluated the performance of the tools. Specifically, we considered how often we found errors, the accuracy of the tool using base models, and the ever important aspect of scalability.

2.0.2. Datasets Used for Evaluation

In order to test the tools, we used three datasets. The first was structured text about the Chernobyl disaster found at [15]. This dataset was a small, structured test to extract working tools from those that failed at the basic task. The second dataset is an unstructured dataset compiled of Tweets from Twitter on various nuclear disasters, which can be found at [3]. Finally, we created a text file with the entire story, $WarandPeace$, to test the scalability of the tools.

3. TOOL SUMMARY

In this section, we look at the individual tools and discuss our findings based on the criteria in the previous section. The order of the tools listed is the order in which we tested them and have no importance in terms of tool performance.

3.0.1. MIT Information Extraction (MITIE)

MITIE is a tool created by a research group at MIT CSAIL which focuses on natural language processing. Information about the tool has been presented in [6] with additional information and source code available at [11]. MITIE is currently maintained, which means new features and updated code are appearing regularly, including support for multiple programming languages. It currently supports Spanish and German in addition to English. It is available for free for all uses, both personal and commercial.

MITIE was created with C++ in mind and leans heavily on the Dlib library. It comes with mappings for other languages however, including R, Python, and Matlab. The source code is easily downloadable from its repository and installs with relative ease using the extensive documentation available. In this evaluation, we installed MITIE for R.

3.0.1.1. Technical Details

Behind the hood, MITIE performs text manipulation to extract nouns(entities) from sentences, and then uses the C++ Dlib library to perform a SVM analysis to classify them. The software comes with basic trainers to recognize people, locations, and organizations with everything else being labeled as miscellaneous. It does offer easy ways to train the data to recognize other labels however, such as dates, times, and nicknames, though additions are added as a dictionary-based search making it limited in scope when it comes to anything not considered a noun in a setence. Relationship extraction is also possible with MITIE with extensive training, which the tool does not come with. Even so, relationships are limited to within the same sentence and the type of relationship is unknown. The tool considered any two nouns within the same sentence to be related in some way.

3.0.1.2. Performance Observations

Entity extraction was completed with good accuracy, however, we noticed that the tool has no memory of previous labeling. For example, an entity recognized as a person was sometimes also recognized later as an organization. The tool comes with pre-trained models so that the software can be tested out of the

Sample output from MITIE on structured text (a) and unstructured text (b)

Sample Result
[1] "Chernobyl / LOCATION @ (24,24)"
[1] "Ukraine / LOCATION @ (26,26)"
[1] "Soviet Union / LOCATION @ (31,32)"
[1] "Chernobyl Nuclear Power Plant Accident Closed AreaEmergency / ORGANIZATION @ (53,59)"
[1] "Soviet / MISC @ (134,134)"
[1] "Chernobyl / LOCATION @ (161,161)"
[1] "Soviet / MISC @ (185,185)"
[1] "International Atomic Energy Agency / ORGANIZATION @ (196,199)"
[1] "Vienna / LOCATION @ (202,202)"
[1] "Austria / LOCATION @ (204,204)"
[1] "Soviet / MISC @ (257,257)"
[1] "Russian / MISC @ (263,263)"
[1] "UNSCEAR / ORGANIZATION @ (288,288)"
[1] "Chernobyl / LOCATION @ (301,301)"
[1] "U.S. / LOCATION @ (374,374)"
[1] "Chernobyl / ORGANIZATION @ (383,383)"
[1] "Chernobyl / ORGANIZATION @ (432,432)"
[1] "Belarus / LOCATION @ (438,438)"
[1] "Russian Federation / LOCATION @ (441,442)"
[1] "Ukraine / LOCATION @ (445,445)"
[1] "World Health Organization / ORGANIZATION @ (456,458)"
[1] "UNSCEAR / ORGANIZATION @ (504,504)"
[1] "UNSCEAR / ORGANIZATION @ (632,632)"
[1] "Kaschcheev / PERSON @ (710,710)"

Figure 3-1. (a)

[1] "Hello Japan / LOCATION @ (2,3)"
[1] "Renewable Energy Consumption Tops Nuclear for FirstTime {link} / ORGANIZATION @ (13,20)"
[1] "Fukushima Daiichi Nuclear Power Station / ORGANIZATION @ (49,53)"
[1] "Environment : Oxf ord Un iversity Pr / ORGANIZATION @ (77,83)"
[1] "ISBN : / MISC @ (88,89)"
[1] "Nuclear Power Safety Campaign Organizer / ORGANIZATION @ (101,105)"
[1] "Temporary / Union / ORGANIZATION @ (107,109)"
[1] "Cambridge / LOCATION @ (114,114)"
[1] "MA / LOCATION @ (116,116)"
[1] "Vermont / LOCATION @ (130,130)"
[1] "Environment : Oxf ord Un iversity Pr / ORGANIZATION @ (140,146)"
[1] "ISBN : / MISC @ (151,152)"
[1] "Masao Yoshida / PERSON @ (219,220)"
[1] "Fukushima Diichi Nuclear Station / ORGANIZATION @ (223,226)"
[1] "Mainichi / LOCATION @ (248,248)"
[1] "Mainichi Daily News {link} Coal / ORGANIZATION @ (251,255)"
[1] "EGAT / ORGANIZATION @ (264,264)"
[1] "Lithuanian / MISC @ (277,277)"
[1] "Star•_¨ Enters Nuclear Power Supply Market / ORGANIZATION @ (283,288)"
[1] "Star•_¨ Enters Nuclear Power Supply Market / ORGANIZATION @ (291,296)"

Figure 3-2. (b)

box, and therefore, without knowledge of the training dataset, some allowance was made for errors. When entities are identified, they are labeled and output with the location of the entity for additional reference later. An example of the output can be seen in 3-1.

Training for additional labels or custom entities is intuitive and easy to do, but tedious. It requires annotations of sentences with the exact location of entities and which category they belong to. A good training set requires thousands of these lines to perform well. We also tested MITIE on an unstructured dataset. An example of the output is presented in 3-2. The base installation does not contain a trained model for unstructured data, so as expected, there are many more errors. However, the results from the output are still more accurate than some other models and provide some meaningful information.

Scalability is an issue for MITIE. We tested a larger dataset (7mb)and found that MITIE took approximately 20 minutes to process. In the background, MITIE allows you to see the tool's progress through a command window on Windows 10, and we observed the entity extraction process being performed sequentially with no parallelism. We also noticed that MITIE loads the entire text into memory at once, making it cumbersome and inefficient. With the use of the C++ Dlib libraries, parallel processing is provided, therefore we believe MITIE could be made scalable without extensive alteration of the current algorithm. The tool itself appears to contain methods for multi-thread processing to increase speed, but we saw no actual use of these methods.

3.0.1.3. Challenges and Shortcomings

As mentioned above, the biggest shortcoming of MITIE is the lack of scalability. A single, small document is not difficult for a human to read and extract important information. The real power in an ERX tool is to be able to scan multiple large documents and/or websites and find interesting information. MITIE seems very limited in the types of entities it can extract, limiting itself to nouns and provided list of other interesting entities. The out-of-the-box base models are lacking and require a large amount of training. Additionally, the extensive effort needed to train for relationship extraction makes using the tool for that purpose very burdensome and not practical for users who want to use the tool but don't have a deep technical understanding of how it works. Also, the scope for relationship extraction is limited to single sentence, making it somewhat trivial and not necessarily true.

While MITIE contains mappings for multiple programming languages, making it easy to pick up for almost any programmer, the mappings seem to be outdated. For the R mapping, we had to downgrade our versions of R and R-tools in order to work with the tool. This creates a security concern and makes the tool unappealing for anyone who depends on the most current configuration for their environments.

3.0.2. TextRazor

TextRazor is an API based tool that can be used from TextRazor's website[9] or as a downloadable library. TextRazor is a London based start-up company that offers a free version of their API with limited number of uses(200 kb dataset), as well as paid tiers for more responses. TextRazor began in 2011 and was included in later works that benchmarked current ERX tools [13]. Like MITIE, TextRazor is currently being maintained and updated, though access to the tool is much more limited. The TextRazor documentation reveals that the tool can understand a long list of languages, with English being the primary.

3.0.2.1. Technical Details

While the software is optimized for C++, the company offers API interfaces for Java, Python, and PHP. With the free API, it can handle up to 200kb of data, though only given as a string or from a url. It does not support files as input arguments. This can be worked around in the downloadable API library, though the downloadable API requires a paid subscription. Documentation reveals built in parallelism to handle scalability and the Amazon AWS backend supports this capability. The source code itself is not accessible and heavily protected by the company, however many details can be extracted about the algorithms.

TextRazor is a dictionary based tool. It scrapes websites such as Wikipedia, DBpedia, Wikidata, and Freebase in order to understand which words are entities and their types. When entities are identified, the software offers the name of its source for the information in the output. When a term is detected that the software does not understand, it uses statistical analysis to guess which category the entity should belong to as well as regular expression analysis to detect special text such as websites and email

Sample category output for TextRazor on structured text(a) and unstructured text(b)

Figure 3-3. (a)

Figure 3-4. (b)

addresses. It does support dictionary augmentation to add specific entity information to assist in labeling.

TextRazor is not only able to extract entities and limited relationships, but also the context in which the entities were used and the words meaning based on that context with the rest of the data. Again, this information comes from the supplemental data where the algorithm gets the label for the word. This means that the labels entities are assigned to are not just given, but rather also generated based on the context of the words. This feature sets TextRazor apart from the other tools, but also limits it in some ways as well as the information must be found and even maintained elsewhere in a free location for the tool to find.

3.0.2.2. Performance Observations

While entity extraction was performed quickly for both the structured and unstructured datasets, we noticed more errors than MITIE. Common words such as "he", "it", and "her" were picked up as entities without any assessment of who these pronouns were referring to. From the structured dataset, we noticed clear indication of the learning capabilities for the algorithm in terms of word meaning and context. The software provides an initial score to rate its confidence in a dictionary meaning of the word and that score changes as the document is processed, often leaving the correct meaning with the highest confidence.

In order to understand meaning and context, and as a result relationships, TextRazor executes hierarchical and spanning trees to connect entities together. Due to this, TextRazor's performance drops significantly when presented with unstructured text such as Twitter feeds. In terms of relationships, we noticed a very weak performance. Like MITIE, TextRazor was limited to same sentence relationships and often relationships did not make sense. This is observable in the relationship graph visualization as well.

Sample output for TextRazor on structured text for entity extraction (a) and word definition discovery (b)

Words Phrases Relations **Entities** Meaning Dependency Parse

Entity	Confidence Score	Relevance Score	DBpedia Type	Freebase Type
Sand (/m/099tz) (Q34679)	4.832	0.1078		/visual_art/visual_art_medium /business/product_category /interests/collection_category
Fire (/m/02_41) (Q3196)	1.351	0.4793		/media_common/quotation_subject /people/cause_of_death /architecture/destruction_method /fictional_universe /fictional_object_destruction_method /event/disaster_type /aviation/accident_type /education/school_mascot /fictional_universe/character_powers /media_common/cause_of_loss
Nuclear and radiation accidents and incidents (/m/021yk8) (Q1620824)	1.008	1		

Figure 3-5. (a)

The **sand** was to stop the **fire** and additional **releases of radioactive material**; the **boron** was to prevent additional **nuclear reactions**.

Words Phrases Relations Entities Meaning Dependency Parse

Position	Token	Lemma	Stem	Part of Speech	Spelling Suggestions	Senses	Parent Position	Relation to Parent	Start Offset	End Offset
78	The	the	the	DT			79	det	447	450
79	sand	sand	sand	NN		*backbone.n.02* (0.0168) *sand.n.02* (0.1455) *sand.n.01* (0.6196)	80	nsubj	451	455
80	was	was	was	VBD					456	459
81	to	to	to	TO			82	aux	460	462
82	stop	stop	stop	VB		*stop.v.04* (0.1044) *intercept.v.01* (0.1366) *stop.v.05* (0.1512) *barricade.v.01* (0.2031) *stop.v.01* (0.2172) *hold_on.v.02* (0.219) *end.v.01* (0.2202) *break.v.10* (0.2569) *check.v.18* (0.2991) *discontinue.v.01* (0.3683) *stop.v.03* (0.4234)	80	xcomp	463	467
83	the	the	the	DT			84	det	468	471
84	fire	fire	fire	NN		*ardor.n.03* (0.005825) *fire.n.04* (0.09521) *fire.n.08* (0.0973) *fire.n.02* (0.2187) *fire.n.07* (0.2236) *fire.n.09* (0.2468) *fire.n.06* (0.2953)	82	dobj	472	476

Figure 3-6. (b)

Words Phrases **Relations** Entities Meaning Dependency Parse

Subject	Predicate	Object
The sand	was	to stop the fire and additional releases of radioactive material the boron was to prevent additional nuclear reactions
The sand	was to stop	the fire and additional releases of radioactive material the boron was to prevent additional nuclear reactions
the boron	was	to prevent additional nuclear reactions
the boron	was to prevent	additional nuclear reactions

Predicate	Property
releases of	additional
releases of	radioactive material
material	radioactive
reactions	additional
reactions	nuclear

Figure 3-7. Example of TextRazor's relationship extraction

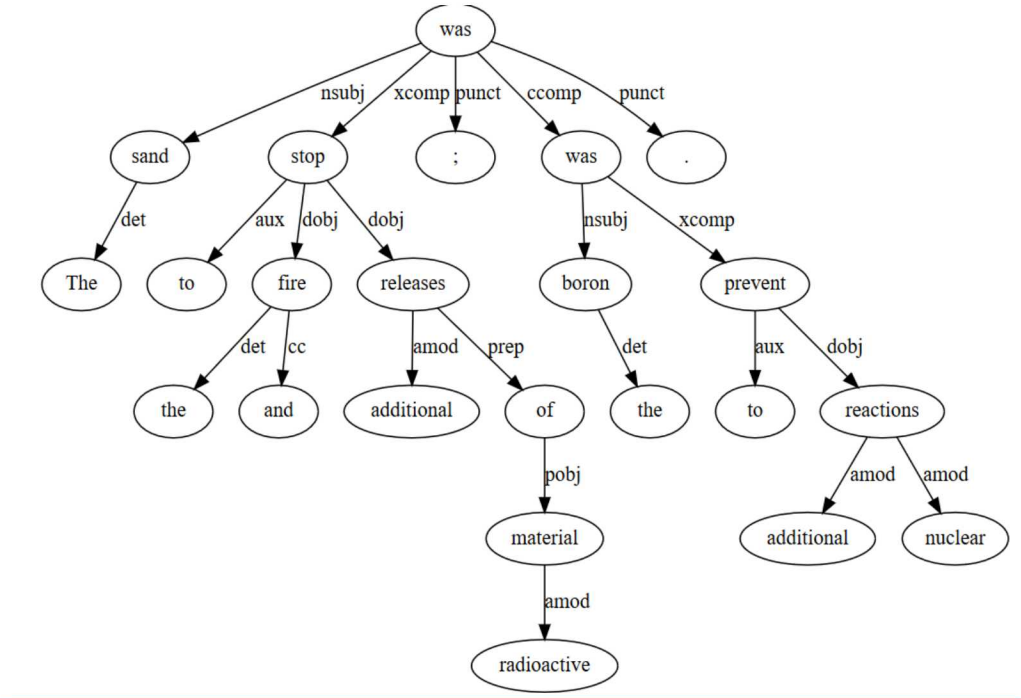


Figure 3-8. TextRazor depends on dependency trees to perform context and relationship analysis.

TextRazor also provides metrics for its extraction in the form of confidence and relevancy scores. The confidence score measures how sure the algorithm believes it is correct and the relevancy score rates each entity with how relevant it is with the rest of the article based on the number of times it is used and its context compared to the meaning of the other entities, almost like an extended TF-IDF analysis. The relevancy score is intuitive and seems to match the reality of the results. However, the documentation claims confidence is measured between one and ten, with ten being absolutely certain. In our tests, we noticed confidence scores much higher and much lower than the bounds given with no indication as to why. In addition, the scores given do not seem to make sense given the output results as a whole, therefore we assume it is in error.

TextRazor resides on Amazon AWS with the goal being to have access to multi-core processing in parallel. The documentation provides some description of parallel processing and hints at NoSQL databases and possibly map-reduce, however, in our use of the API we saw no evidence of parallel processing. When presented with data close to the 200kb limit, the API terminated without completing its task. When the data was reduced, we were able to measure a linear correlation between data size and processing time. With no access to the source code, we believe the algorithms are not using as much parallel processing as the documentation implies, or that the use of NoSQL databases actually limits the algorithm further on information it knows and new terms take much longer to understand. This would mean that the scope of the text analysis is narrow, being best used on a single topic of interest.

Finally, we tested the API's ability to handle other languages. When presented with Russian text, as indicated in the list of languages as a possibility, TextRazor performed very poorly. It was unable to identify any actual entities except those recognized in English such as Arabic numerals. When presented

text that was a mix of Russian and English, as a conversation may be, it ignored all Russian text and extracted information from the English portions only. This shows that the algorithm is able to detect the different languages, but information extraction is limited to Germainic languages as best.

Overall, for entity extraction, TextRazor performed well. It provides some interesting features, such as context and definition understanding. However, it has some major failings in terms of relationship extraction, scalability, and extra features such as multi-language recognition.

3.0.2.3. Challenges and Shortcomings

Without access to the source code, it is difficult to understand how the algorithms within TexRazor actually work. Given the observations in its performance and the information provided in documentation, we note a few issues that are critical to successful information extraction. As we noted with MITIE, scalability is a large issue. TextRazor provides the ability to copy text or url to the web API or feed the code based API strings of text or urls. However it is unable to process files without supplemental code to load the files into strings to give to the API. For entity extraction, relying on a large database to identify the entities and their categories is slow and can result in conflicts. This also means that the software will be unable to recognize different spellings and abbreviations of the same term, making it consider them separately.

As we observed, relationship extraction is poor. It is also a bottleneck for the algorithm and a source of its slow speed. To understand context and word meaning, and as a result relationships, it builds large dependency trees. Also, in order to understand context and update new meanings and understandings for entities at the beginning of the text, it must save information it learns on every entity, or at the very least, a sample of the entities. This will take a fair amount of memory and require multiple passes over some of the data, making it inefficient.

Multiple errors in the metrics calculations lowers confidence in the software's accuracy, as well as the software's inability to perform extra features such as multiple language recognition and processing. The tool has a lot of potential, particularly in the area of context resolution, however as it is now, it does not perform as the documentation claims.

3.0.3. ReVerb

ReVerb was developed by faculty at the University of Washington's Turing Center. It is an open information extraction tool, meaning that it attempts to find entities and relations with no prior training or context. Users can simply input raw text into the tool and it will output a tab separated value file with multiple fields, among which are the sentence the information was extracted from, the entities and relation that was extracted, a confidence score for the extraction, and others. Information about the tool and links to the source code can be found at [12]. Though ReVerb is no longer being maintained, with the most recent update in 2013, its capabilities in its current form make it worth exploring.

3.0.3.1. Technical Details

ReVerb accomplishes entity and relationship extraction through a regex approach in an attempt to overcome previous shortcomings of open information extractors, such as incoherent and uninformative extractions. It attempts to do this through a regular expression analysis designed to extract more meaningful relations within a sentence of text. The following regular expression is used:

V | VP | VW*P
V = (verb particle | adv)
W = (noun | adj | adv | pron | det)
P = (prep | particle | inf. marker)

For example, consider the sentence "Bob claimed responsibility for the broken window." Rather than the triple (Bob, claimed, responsibility) being extracted, the much more informative (Bob, claimed responsibility for, the broken window) is extracted.

Put into words, the constraint allows for just a simple verb phrase (e.g. jumped, jumped in, jumped happily), a verb phrase followed by a particle or preposition, or a verb phrase followed by a simple noun and ending with a preposition or particle (e.g. gave Tommy the). The convention used is that the longest expression matched is the relation extracted.

The entities are then extracted from the left and right hand sides of this relation. ReVerb attempts to overcome uninformative extractions through maintaining a dictionary of roughly 1.7 million normalized relations and excluding extracted relations that, after being normalized, are not included in the dictionary.

Put simply, ReVerb uses a combination of approaches discussed earlier. It builds its own database from Wikipedia and data sources from pre-trained models, such as Apache's OpenNLP [5] software. It uses Google's Guava [10] software for data organization and then uses Apache's OpenNLP to extract entities from the data. Finally, once entities are recognized and categorized, it uses clustering algorithms built into Weka [8] to understand relationships between entities.

3.0.3.2. Performance Observations

The tool itself is simple and painless to use. All code used has been compressed into a single Java jar file, with no building required, nor is training a model required. However, due to this, the tool performs only adequately across the board. In almost all instances, a specialized or custom model would be better suited to reduce the search space for terms. Additionally, attempting to follow the source code is difficult, as the decompiled source is severely bloated. Entire libraries are compiled into the jar with only portions of them being used or sometimes none at all. Though Java is an object oriented language, the paradigm is overused making the source code densely broken up and inefficient.

ReVerb performed faster than other tools, taking only 97 seconds on the 7mb dataset to complete. It accurately analyzed structured text, but performed very poorly on unstructured text, as expected without previous training. In addition, relationships were not necessarily extracted or inferred, but rather the entities and surrounding words were taken as the relationship, showing no context understanding and less than a sentence limitations.

Sample output for ReVerb for Entity Extraction for structured text (a) and unstructured text (b)

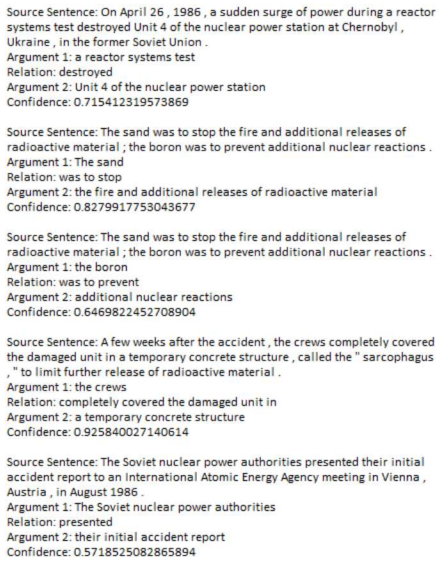


Figure 3-9. (a)

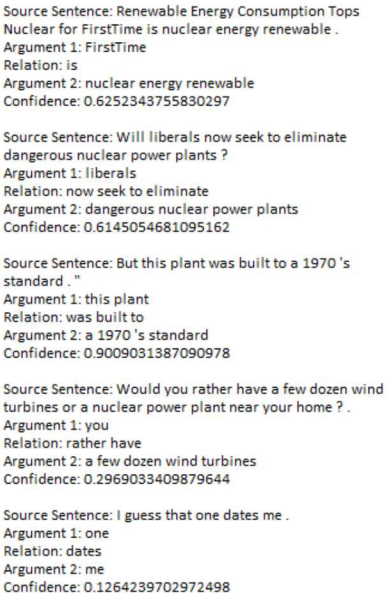


Figure 3-10. (b)

3.0.3.3. Challenges and Shortcomings

While ReVerb is faster than other tools, it is not considered scalable for large text and still has room for optimization. It also pays for its speed with sacrifices in adaptability. As mentioned above, it performed poorly for unstructured text and requires retraining for different datasets, which is time consuming both to create the training datasets and to process them. ReVerb makes using additional outside models difficult as the models are built in, requiring rebuilding of the software entirely as it is currently written each time a new model is introduced or updated. It relies very heavily on other current tools, software, and algorithms to train and process that the software is very bloated and difficult to use for custom processing.

3.0.4. Stanford CoreNLP

The Stanford CoreNLP software is a web based version of several Stanford NLP tools including part-of-speech(POS) tagger, named entity recognizer(NER), and sentiment analysis. It supports third party annotators, allowing multiple ways to train the software to user needs. The software source code and API can be downloaded at [7] with more information and use described in [4].

Stanford CoreNLP is still currently maintained and being updated with improved algorithms. It boasts the ability to recognize words, their parts of speech, and understand dependencies based on context. It supports several languages including English, Spanish, Chinese, and German.

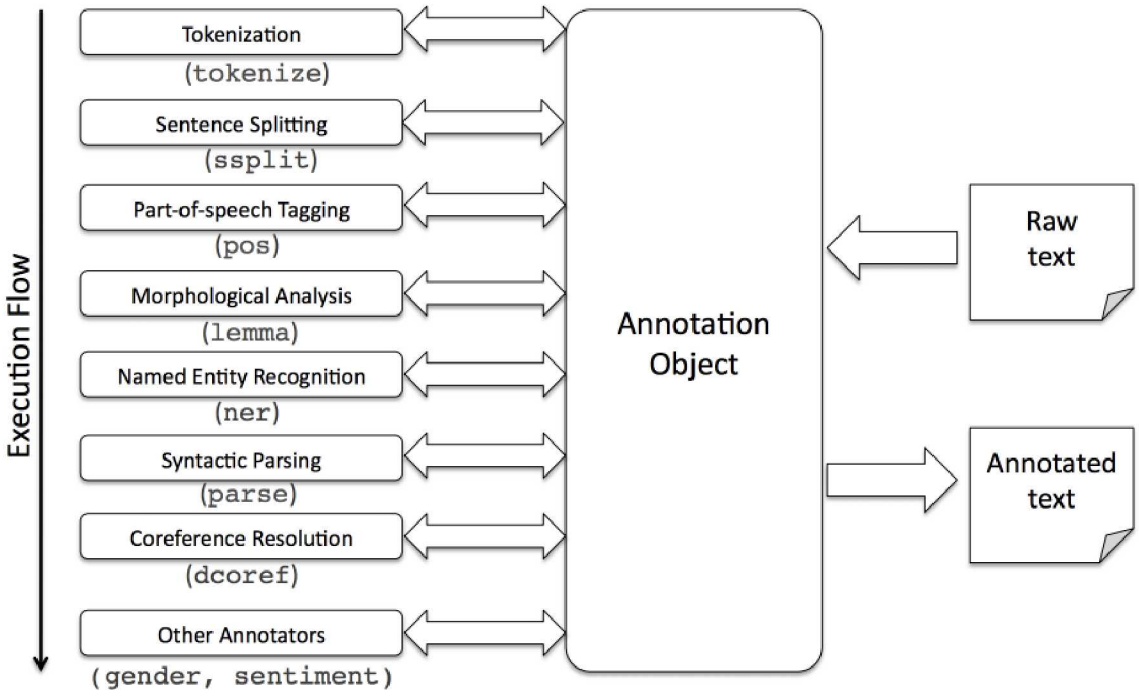


Figure 3-11. Stanford CoreNLP annotator map

3.0.4.1. Technical Details

CoreNLP is written in Java. When downloaded, it comes pre-built with a Java-applet in which you load a model and text and it will process without any extra build or installation needed. CoreNLP allows interaction through the applet or through a command line or web service. It provides an object oriented API for use with Java, Javascript, C++, and Python. Stanford offers support for Java, Javascript, and Python, though there are several third party APIs for other languages. All of this makes the tool very accessible and easy to use and install.

CoreNLP allows the user to choose what kind of annotator they wish for entity-relationship extraction. For example, there are several built in annotators to find dates, phone numbers, produce dependency trees, etc. Most of these perform well, even for unstructured text, however speed performance is heavily affected by the annotator(s) chosen. The software also allows the user to create their own annotator to train a model. And example of the annotator system can be seen in Figure 3-11.

CoreNLP is a strictly database approach. The software trains annotators to recognize parts of speech and entities and uses those annotators to process raw text.

3.0.4.2. Performance Observations

The software was accurate for both structured and unstructured text. The software produces the original text with entities highlighted and coded with a legend to determine category. And example of output can be seen in Figure 3-12 and Figure 3-13.

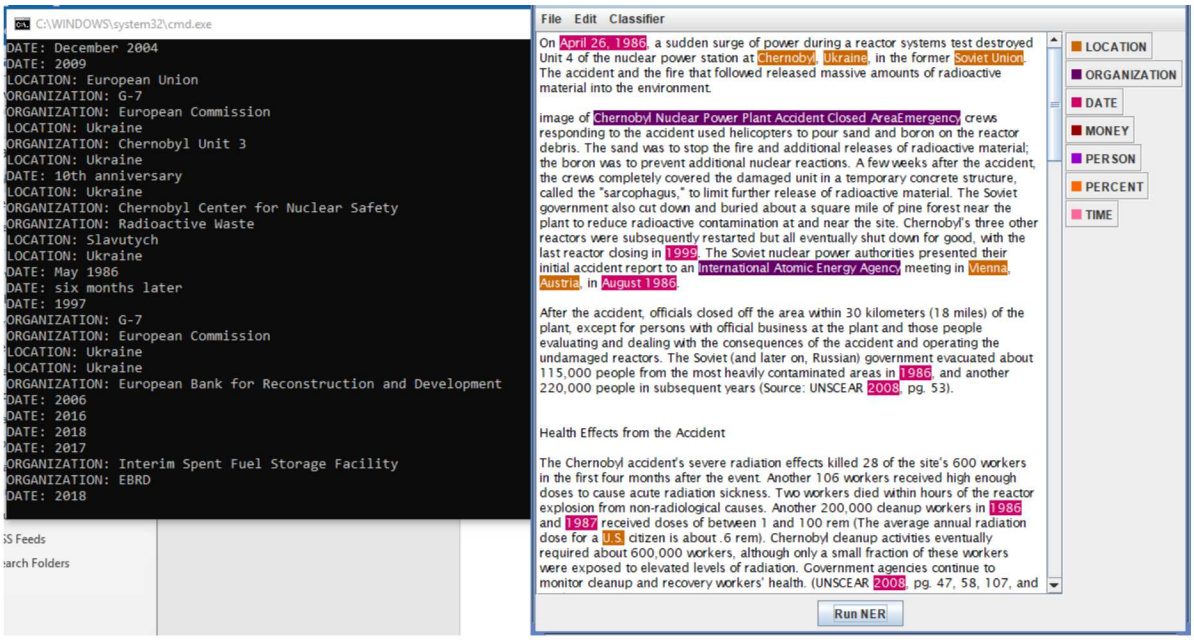


Figure 3-12. Stanford CoreNLP sample output for structured text

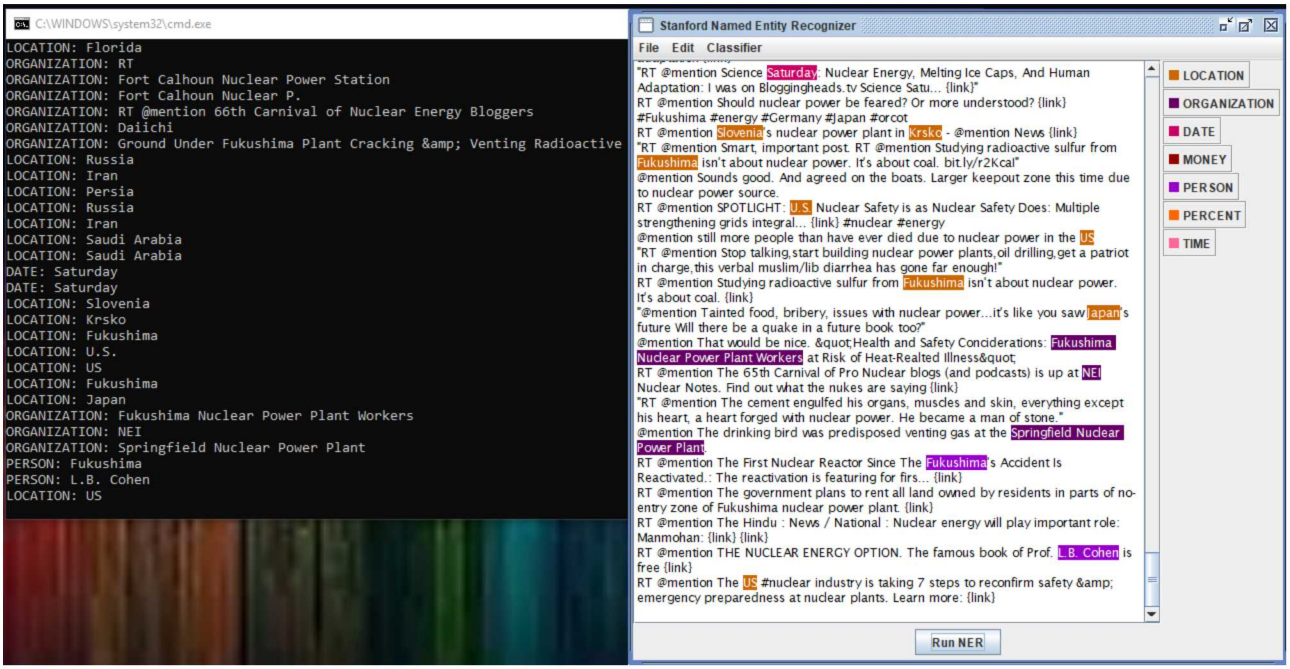


Figure 3-13. Stanford CoreNLP sample output for unstructured text

Installation was fast and ran instantly "out of the box". The software comes with several pre-trained models for different annotators. Examples of the training datasets and the annotators are also given. We tested the command line API using Java in order to get the fastest responses possible given the software is written in Java. We also used the provided Java applet GUI to compare performance and found they were very similar.

As with the other tools, there are some scalability issues, most of which is attributed to the database approach and sequential processing. The issue is common enough that the documentation specifically addresses it in the online FAQ. We were able to give the software our large dataset without the software terminating, but as with MITIE, the data was processed sequentially and was viewable in the command line window. In essence, the software is comparing each entity to the annotators to find the closest match, and then outputting the annotated text. The annotators are loaded into memory entirely and kept there during processing. As the documentation states, the larger the annotator and/or the more annotators used, the longer processing will take due to memory usage. Also, as with other tools, the entirety of the document being analyzed is also loaded into memory all at once. The documentation recommends input data be broken up into smaller documents before being given to CoreNLP in order to prevent slow processing. This shows room for parallel processing and more efficient memory usage.

The software is able to extract some linking information through its Coreference tool built in. For example, if a person is recognized in the text, it is also able to recognize that the word, "he" in the same sentence references that person. In addition, the software recognizes overall sentiment in text, meaning it can recognize some context information in both structured and unstructured text.

Overall, relationship extraction has been weak among the tools we looked into, often not existing at all. CoreNLP does not seem able to extract relationship information, however, it goes a step in the right direction with the parts of speech tagger, which could be a stepping stone towards greater context understanding.

3.0.4.3. Challenges and Shortcomings

The biggest shortcoming, like other tools, is scalability. The software struggled with a 7mb file and while this is common among all the tools, it is a small dataset. Even with the successful entity extraction and categorization and the advanced relationship extraction compared to other tools, the lack of ability to analyze large amounts of text gives it limited use. There are clear areas where the tool could be made faster however, and much more scalable.

The tools allow for custom training and custom annotators to be used. Aside from the reduced efficiency, creating these training datasets is time consuming. CoreNLP does provide extensive options for annotators for general text, however, any specialized use will require time to train.

The software, regardless of which API or language is used, requires Java to be installed. This presents the minor challenge of being more difficult to use on some operating systems and configurations.

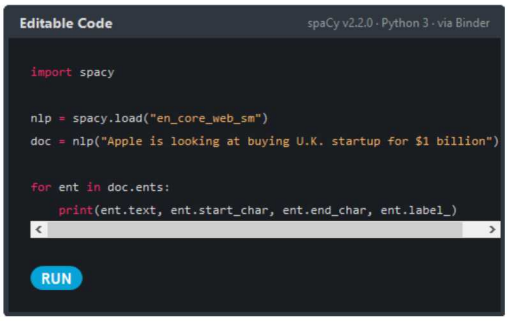


Figure 3-14. Example of code to use SpaCy within Python.

3.0.5. Spacy.io

SpaCy is an open source tool created by Explosion AI. Source code can be found at [1] with extensive documentation found at [2]. SpaCy is a Python based tool with language wrappers in C++, Juliam R, and Javascript. Despite its limitation in programming languages, it offers support for over 50 spoken and written languages and provides flexibility to provide models and annotated data. The source code is easy to view and is maintained regularly and constantly improved upon. Out of the boc, SpaCy offers several NLP tools such as the named entity recognizer, parts of speech tagger, regex matching, entity linking, and tokenization. SpaCY is both versatile in the algorithms it provides, and is easy to install and work with.

3.0.5.1. Technical Details

SpaCy is written in Python and is primarily for use in Python. It is easily available as both the source code can be found on Github as well as the software can be pip installed, making it the easiest tool to download and use. The base software comes with a few pre-trained models based on real structured data, as well as the datasets themselves so that the models can be updated if desired. The tool is accessed through library calls in Python that are very simple. Performing entity extraction on a file took at most five lines of code including setup and displaying the results. An example is presented in 3-14.

SpaCy allows the user to use pre-built models trained on news articles or structured websites. However, it is very adaptable to correcting and retraining the models and provides the datasets to do so. In addition, creating new models either based of the pre-existing ones or from scratch is simple, encouraged, and supported. The documentation to do most tasks with SpaCy is openly available on their website at [2].

The tool offers a suite of capabilities. For our purposes, it offers a named entity recognizer including entity linking, sentence dependency parsing, parts of speech tagging, and sentence segmentation. It also offers tokenization, regex matching, and pipeline management so multiple tasks can be completed at once making SpaCy the most versatile tool we tested.

When focusing on entity extraction, SpaCy uses a mix of dependency parsing, word vectors, and neural statistical modeling to recognize entities and supports the use of supplementary data in certain terms are desired as a specific label(s). Because of this unique approach, SpaCy also has the added benefit of

TEXT	START	END	LABEL	DESCRIPTION
Apple	0	5	ORG	Companies, agencies, institutions.
U.K.	27	31	GPE	Geopolitical entity, i.e. countries, cities, states.
$1 billion	44	54	MONEY	Monetary values, including unit.

Figure 3-15. An example of results from SpaCy entity extraction.

working seamlessly with deep learning tools such as TensorFlow and PyTorch for additional NLP analysis.

SpaCy is actively maintained and updated, more so than any of the other tools we found. As an open source project, user ideas and implementations are often considered and integrated into the software, making it grow faster than the more restrictive tools. The Explosion AI team takes an active role in any new features so the software remains stable and robust. Using SpaCy does require some knowledge and proficiency in Python and is not as simple as CoreNLP and TextRazor, but usage is much more simple than other options such ReVerb and MITIE.

3.0.5.2. Performance Observations

Entity extraction was performed quickly, though not the fastest of the tools. It gets significant bonuses from the neural approach, however it is still affected by the slower processing speed of Python that is dependent on the code written by the user. Its implementation makes it easy to use parallel processing however, though we did not use it to be consistent and fair in assessing against other tools. It does support the processing of smaller data chunks without the sacrifice of accuracy. In terms of memory, only the trained model is kept actively in memory throughout processing, making it memory efficient as well.

When using the base models to process text, SpaCy was reasonably accurate with structured text. SpaCy offers easy access to the data and the labels, as shown in 3-15. The results are stored in an object so that only information needed can be extracted and organized as the user wishes.

Unstructured text performed worse with low accuracy as expected. The pre-trained models were trained on structured reports and news articles. However, with about 500 lines of training data, the accuracy improved significantly. SpaCy does not provide its own confidence metrics, however, it does provide visualization tools to see the text it labeled within the sentence and paragraph the text was found and what labels it decided. The annotated text is easily accessible to help determine the accuracy of the results. Out of the tools tested, SpaCy required the least amount of training data and was the easiest for creating new or updated models.

Finally, we tested three of the languages SpaCy supports, Spanish, German, and Russian. While the accuracy was not as good as English, it clearly understood which language it was processing and could recognize which terms were entities, though not always the labels for them. It was able to tokenize the words in each language for additional analysis elsewhere.

3.0.5.3. Challenges and Shortcomings

While SpaCy could be used with other parallel libraries and is more efficient in terms of memory and processing time to other tools, it is still not very fast for large datasets on its own. The slow down of Python processing and the overhead of neural learning makes it difficult to process text in reasonable time. Also, while SpaCy offers several tools within its suite, relationship extraction is missing. The tool is able to understand parts of speech, but context understanding and relationships are beyond its capabilities.

SpaCy makes up for these shortcomings by being easily adaptable with models, training, and use. It's self contained, requiring little from outside libraries to use, though accuracy depends on additional models trained with fairly large amounts of data.

3.0.6. Other Tools

In our research, we recognize the potential of other tools. For example, we took a closer look at Deep Mind, Dandelion, Sintelix, Lexalytics, and Catalyst. These tools have potential, but were removed from deeper research for various reasons. Catalyst is a specialized NER tool for biological and medical purposes. At it's core however, it uses Spacy.io to perform all of the ERX functions. Since we already covered this tool, it was not worth analyzing again. Dandelion was interesting in that it provided an image search for entities extracted which could sometimes be useful. It is a web based only API that is still being maintained by a start-up company, but other than the image search, it did not stand out compared to other tools. Sintelix and Lexalytics are corporate based solutions with no downloadable API. Demos are available for a cost, however the tools are marketed as a one stop, one size fits all solution with no ability to customize, custom train, or otherwise choose parts to use. They are worth mentioning as they are commercial solutions with the claim they can perform ERX, however, with no ability to actually test or analyze along with the inability to customize, they are not the type of tool we are looking into here. Finally Deep Mind is a tool no longer maintained for several years. With no real documentation to use the code available and the requirement of outdated libraries already proven to be insufficient, we did not test the tool but recognize it as one of the first tools commonly referenced in literature.

4. FINAL OBSERVATIONS

In this chapter we summarize important information about ERX tools as well as other considerations of ERX that has been mentioned in the analysis of the tools but not explained.

4.0.1. Common Strengths of Current ERX Tools

Among the tools analyzed, all are able to perform entity extraction with a good amount of accuracy. In addition to structured text, such as books, articles, and blogs, many are also able to handle unstructured text such as text messages and Tweets on Twitter with varying amounts of degradation of accuracy. Accessibility to the tools is not difficult. All of the tools have APIs, many of which in multiple programming languages and even some with downloadable source code. The problem is still new enough that tools are still being maintained and created with imperfections being recognized.

4.0.2. Common Weaknesses of Current ERX Tools

Most commonly, scalability is a major issue among ERX tools. Nearly all the tools load entire datasets at once into memory for processing. During processing itself, it is performed sequentially and not in parallel. This leaves tools with the capability of processing small datasets only and/or consuming a large amount of time. All of the tools have clear ways of making them more efficient so that larger datasets can be analyzed. The largest dataset we tests was approximately 7mb, which in the world of social media and digital text, is quite small. Therefore, there is a demand for larger scale text analysis software.

Though entity extraction is performed with a fair amount of accuracy, relationship extraction remains an ongoing elementary stage issue. As of now, current tools can recognize two entities within the same sentence are related, which is both trivial and not necessarily true. Context information from the text is not understood, therefore meaning and relationships are lost. Relationship extraction remains in early stages and, based on literature, has been mostly abandoned for new ideas.

Finally, measurements of performance are rarely implemented and even when done, are ambiguous. Often there are errors in the calculations and other times, they are ignored entirely. There are currently no common performance metrics to determine the correctness and quality of ERX tools.

4.0.3. Other Aspects of ERX

In our research, we found other aspects of ERX tools that were either fascinating, but not common or done well yet. Or were issues that were common, but not from the tools themselves.

All of the tools require training to some degree in order to categorize entities. Though they all are intuitive and easy to train, the training datasets are quite large and time consuming to create. It requires creating thousands of lines of sample text with annotations either written into the training text, or programming the exact locations of the entities and the annotations associated with them. As of now, there is only one real tool to help with building training datasets, Prodigy[14]. Prodigy is an annotation tool with a GUI that allows a programmer to manually select entities and place them in categories. Aside from this, the job of creating training datasets is given to a room of data entry type workers to create the annotations to feed to the tools. A possible solution is a more intuitive annotation creation tool or to create tools with learning mechanisms to understand entity categorization automatically based off dictionaries or some other technique.

While all the tools provided output, a few provided more visualization of the data. Dandelion gave us an image search for each entity, but more helpful, TextRazor gave us a dependency graph to understand the relationships between the entities. To our knowledge, TextRazor was the only one that did this. CoreNLP and SpaCy showed us sentences with dependencies and context connections labeled which is a decent visualization as well. Still, we saw the benefits of the visualization and believe it is a feature current tools should provide. In addition, it would be a possible step towards more relationship extraction in ERX tools.

As mentioned above, performance metrics for ERX tools are ambiguous at best. Those that provided scores were not clear how they were calculated or had obvious errors in their calculation. In this strongly emerging are of natural language processing, a common set of metrics to measure accuracy, relevancy, and performance is needed.

Finally, we were surprised to see some work in context based understanding in TextRazor and CoreNLP. Also related, the ability to understand a word's exact meaning based on the context of the whole text. This is valuable information that could aid in relationship extraction and recognizing priority of entities. It was not common among the tools however and may be a newer branch of ERX software.

5. CONCLUSIONS

Entity-relationship extraction is still in early stages of research. While entity information extraction is largely solved, with varying degrees of success, relationship extraction alone is rare. Most tools available are easy to use, whether through code or an API, but they have some shortcomings making them fall short of current demands. Among these are poor scalability and no solid performance metrics to gauge their accuracy. There is ample room for improvement and additional research, however, for basic needs, there are tools currently available to assist in entity recognition.

REFERENCES

[1] Explosion AI. spacy.io: Industrial strength natural language processing. https://spacy.io. Published: 2016 Accessed: April 2020.

[2] CrowdFlower. Data for everyone - judge emotions about nuclear energy from twitter. https://www.figure-eight.com/data-for-everyone/. Published: August 2013 Accessed: June 2019.

[3] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. *Proceedings of the 43rd Annual Meeting of the Association for Computing Linguistics(ACL 2005)*, pages 363–370, 2005.

[4] Apache Software Foundation. Apache opennlp. http://opennlp.apache.org/. Published: 2017 Accessed: June 2019.

[5] Kelly Geyer, Kara Greenfield, Alyssa Mensch, and Olga Simek. Named entity recognition in 140 characters or less. *# Microposts*, pages 78–79, 2016.

[6] The Stanford Natural Language Processing Group. Stanford named entity recognizer(ner). https://nlp.stanford.edu/software/CRF-NER.html. Published: October 2018 Accessed: June 2019.

[7] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11, 2009.

[8] TextRazor Ltd. Textrazor: Extract meaning from your text. https://www.textrazor.com/. Published: 2019 Accessed: June 2019.

[9] Joshua Madadhain and Google. Goog guava project. https://github.com/google/guava/wiki. Published: October 2016 Accessed: June 2019.

[10] Massachusetts Institute of Technology. Mitie: Library and tools for information extraction. https://github.com/mit-nlp/MITIE. Published: August 2003 Accessed: June 2019.

[11] University of Washington Turing Center. Reverb: Open information extraction software. http://reverb.cs.washington.edu/. Published: Unknown Accessed: June 2019.

[12] Giuseppe Rizzo, Marieke van Erp, and Raphael Troncy. Benchmarking the extraction and disambiguation of named entities on the semantic web. *LREC*, pages 4593–4600, 2014.

[13] spaCY Explosion AI. Prodigy: Radically efficient machine teaching, powered by active learning. https://spacy.io/universe/project/prodigy. Published: 2019 Accessed: July 2019.

[14] USNRC. Backgrounder on chernobyl nuclear power plant accident. https://www.nrc.gov/reading-rm/doc-collections/fact-sheets/chernobyl-bg.html. Published: August 2018 Accessed: June 2019.

APPENDICES

A. TOOL SIDE-BY-SIDE COMPARISON

	MITIE	TextRazor	ReVerb	Stanford CoreNLP	SpaCy.io
Access					
API/Code	Code	API	Code	Both	Code
Cost	Free	Free	Free	Free	Free
Easy to Install	Yes	N/A	Yes	Yes	Yes
Easy to Use	Somewhat	Yes	Yes	Yes	Yes
Origin	Academic	Corporate	Academic	Academic	Corporate
Use Clause	None	May Use	No Commercial	GNU Public License	MIT License
Last Maintained	2018/2019	2019	2013	2018	2020
Extractions					
Entity Extraction	Good	Good	Fair	Great	Great
Relations Found	With Training	Yes	Yes	No	No
Categories given/generated	Given	Generated	Generated	Given	Both
Unstructured Data	Poor but trainable	Fair	Fair	Good	Good
Features					
Import Custom Entities	Yes	Yes	No	Yes	Yes
Multi-language recognition	Yes	No	No	Yes	Yes
Confidence Calculations	No	Yes	Yes	No	No
Relevancy Calculations	No	Yes	No	No	No
Programming Languages	C++, R, Python, C, Matlab	Python, Java, PHP	Java	Java/Python	Python, Cython
Graph Visualization	No	Yes	No	No	Yes
Context Data	No	Yes	No	No	Yes
Performance					
Errors Found	Yes	Yes	Yes	Yes	Yes
Scalable	No	No	No	No	No
Could be made scalable	Yes	No	Yes	Yes	Yes

Side-by-Side Summary of the top ERX Tools

DISTRIBUTION

Hardcopy—External

Number of Copies	Name(s)	Company Name and Company Mailing Address

Hardcopy—Internal

Number of Copies	Name	Org.	Mailstop

Email—Internal (encrypt for OUO)

Name	Org.	Sandia Email Address
Technical Library	01177	libref@sandia.gov

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.